ชื่อ-นามสกุล นายนิพัทธ์ ชาภักดี รหัสนักศึกษา 653380332-7 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

```
nipatchapakdee@Macintosh-2 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 q
uickview busybox
nipatchapakdee@Macintosh-2 Lab8_1 % docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
busybox      latest    fc0179a204e2   3 months ago   4.04MB
nipatchapakdee@Macintosh-2 Lab8_1 %
```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร_____ชื่อ images ในข้อนี้ชื่อ busybox _____

(2) Tag ที่ใช้บ่งบอกถึงอะไร_____version ของ images นั้น ๆ ในข้อนี้จะเป็น latest _____

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

```
nipatchapakdee@Macintosh-2 Lab8_1 % docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
busybox      latest    fc0179a204e2   3 months ago   4.04MB
nipatchapakdee@Macintosh-2 Lab8_1 % docker run busybox
nipatchapakdee@Macintosh-2 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 23 02:47 .
drwxr-xr-x    1 root     root          4096 Jan 23 02:47 ..
-rwxr-xr-x    1 root     root             0 Jan 23 02:47 .dockerenv
drwxr-xr-x    2 root     root         12288 Sep 26 21:31 bin
drwxr-xr-x    5 root     root           360 Jan 23 02:47 dev
drwxr-xr-x    1 root     root          4096 Jan 23 02:47 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  218 root     root             0 Jan 23 02:47 proc
drwx------    1 root     root          4096 Jan 23 02:47 root
dr-xr-xr-x   11 root     root             0 Jan 23 02:47 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
nipatchapakdee@Macintosh-2 Lab8_1 % docker run busybox echo "Nipat Chapakdee 653380332-7 from busybox"
Nipat Chapakdee 653380332-7 from busybox
nipatchapakdee@Macintosh-2 Lab8_1 %
nipatchapakdee@Macintosh-2 Lab8_1 % docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS                          PORTS     NAMES
48534d7c1dad   busybox   "echo 'Nipat Chapakd…"   About a minute ago   Exited (0) About a minute ago             gracious_khorana
23f4f497ceb9   busybox   "sh"                     3 minutes ago        Exited (0) 2 minutes ago                  eloquent_lamport
7fb27d36b7f9   busybox   "sh"                     3 minutes ago        Exited (0) 3 minutes ago                  agitated_dhawan
nipatchapakdee@Macintosh-2 Lab8_1 %
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

___หลังจากใช้ option -it จะเป็นการเข้าไปใน images นั้น เข้าไปใช้งาน service ต่าง ๆ ใน images หากไม่ใช้ option -it images นั้นจะรันอยู่พื้นหลังไม่มีการตอบโต้กันกับผู้ใช้__

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

_____คอลัมน์ STATUS แสดงถึงข้อมูลสถานะปัจจุบัน container ว่าทำงานอยู่อยู่ หรือหยุดทำงานไปแล้ว____

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

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

```
nipatchapakdee@Macintosh-2 Lab8_1 % docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS                          PORTS     NAMES
48534d7c1dad   busybox   "echo 'Nipat Chapakd…"   About a minute ago   Exited (0) About a minute ago             gracious_khorana
23f4f497ceb9   busybox   "sh"                     3 minutes ago        Exited (0) 2 minutes ago                  eloquent_lamport
7fb27d36b7f9   busybox   "sh"                     3 minutes ago        Exited (0) 3 minutes ago                  agitated_dhawan
nipatchapakdee@Macintosh-2 Lab8_1 % docker rm 7fb27d36b7f9
7fb27d36b7f9
nipatchapakdee@Macintosh-2 Lab8_1 % docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                          PORTS     NAMES
48534d7c1dad   busybox   "echo 'Nipat Chapakd…"   11 minutes ago   Exited (0) 11 minutes ago                 gracious_khorana
23f4f497ceb9   busybox   "sh"                     13 minutes ago   Exited (0) 12 minutes ago                 eloquent_lamport
```

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 << EOFdocker

FROM busybox

CMD echo “Hi there. This is my first docker image.”

CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น”

EOF

หรือใช้คำสั่ง

$ touch Dockerfile

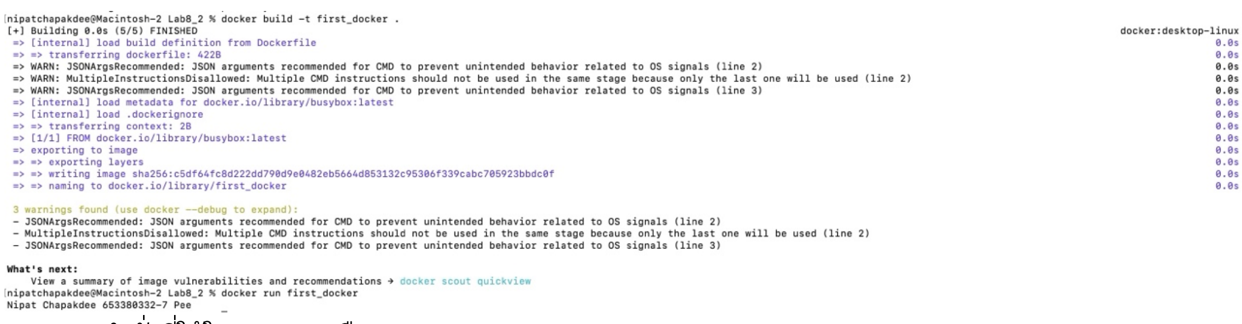
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

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



```
nipatchapakdee@Macintosh-2 Lab8_2 % docker build -t first_docker .
[+] Building 0.0s (5/5) FINISHED                                              docker:desktop-linux
 => [internal] load build definition from Dockerfile                                         0.0s
 => => transferring dockerfile: 422B                                                         0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)   0.0s
 => WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)   0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)   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:c5df64fc8d222dd790d9e0482eb5664d853132c95306f339cabc705923bbdc0f 0.0s
 => => naming to docker.io/library/first_docker                                              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
nipatchapakdee@Macintosh-2 Lab8_2 % docker run first_docker
Nipat Chapakdee 653380332-7 Pee
```

(1) คำสั่งที่ใช้ในการ run คือ

___รัน container จาก image first_docker __

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

Option -t เมื่อใช้จะเป็นการตั้งชื่อให้ image และสามารถเรียกใช้และลบผ่านชื่อ image ได้เลย หากไม่ใช้ docker จะสร้าง image ขึ้นมาและเมื่อเราจะใช้งานจะต้องเรียกใช้ผ่าน image ID

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

```
nipatchapakdee@Macintosh-2 Lab8_3 % docker build -t peenipat/lab8 .
[+] Building 0.1s (5/5) FINISHED                                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                           0.0s
 => => transferring dockerfile: 172B                                                                           0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)   0.0s
 => WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)   0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)   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 sha256:77315ae57f142df3994048d1ba703a6dd0872d943f5aaa0fe943cf1a45fd9ad0                   0.0s
 => => naming to docker.io/peenipat/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
nipatchapakdee@Macintosh-2 Lab8_3 % docker run peenipat/lab8
Nipat Chapakdee 653380332-7
```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

   $ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8

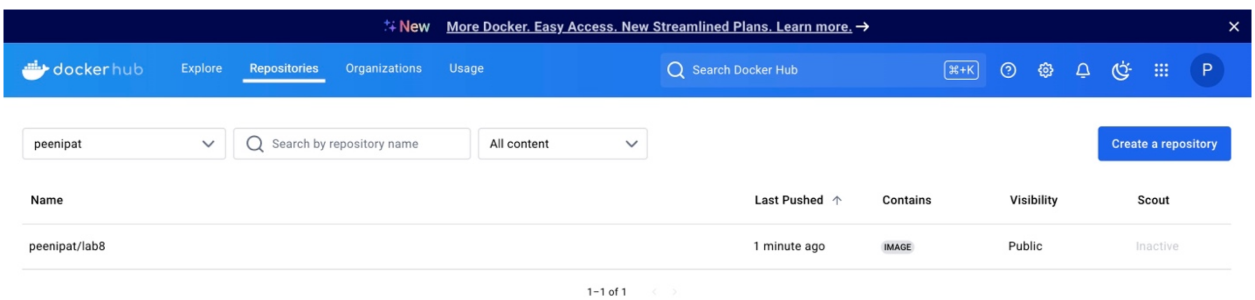
   ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

   $ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

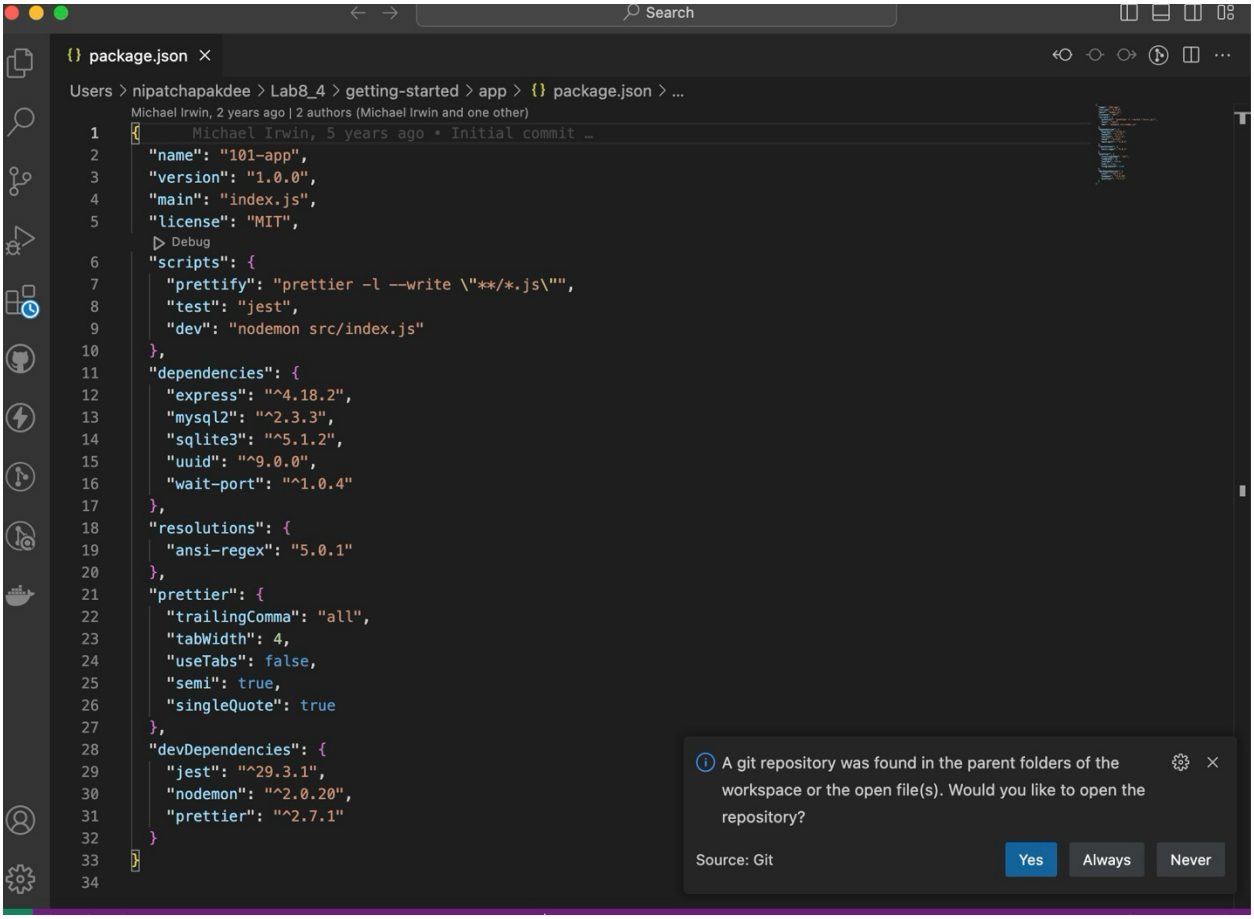
7. ไปที่ Docker Hub กด Tab ชื่อ Tags หรือไปที่ Repository ก็ได้

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




```
Login Succeeded
nipatchapakdee@Macintosh-2 Lab8_3 %
nipatchapakdee@Macintosh-2 Lab8_3 % docker push peenipat/lab8
Using default tag: latest
The push refers to repository [docker.io/peenipat/lab8]
613e5fc506b9: Mounted from library/busybox
latest: digest: sha256:0d9ade47c1781aaa5ef10db5fccbdb4218b7f452ed6f3b6a3428175ba0cb5061 size: 527
nipatchapakdee@Macintosh-2 Lab8_3 %
```

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
4. 


```
nipatchapakdee@Macintosh-2 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 | 6.44 MiB/s, done.
Resolving deltas: 100% (523/523), done.
nipatchapakdee@Macintosh-2 Lab8_4 %
```

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

5. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปในไฟล์

   FROM node:18-alpine

   WORKDIR /app

   COPY . .

   RUN yarn install --production

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

   EXPOSE 3000

6. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp_รหัสนศ. ไม่มีขีด

   $ docker build -t <myapp_รหัสนศ. ไม่มีขีด> .

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

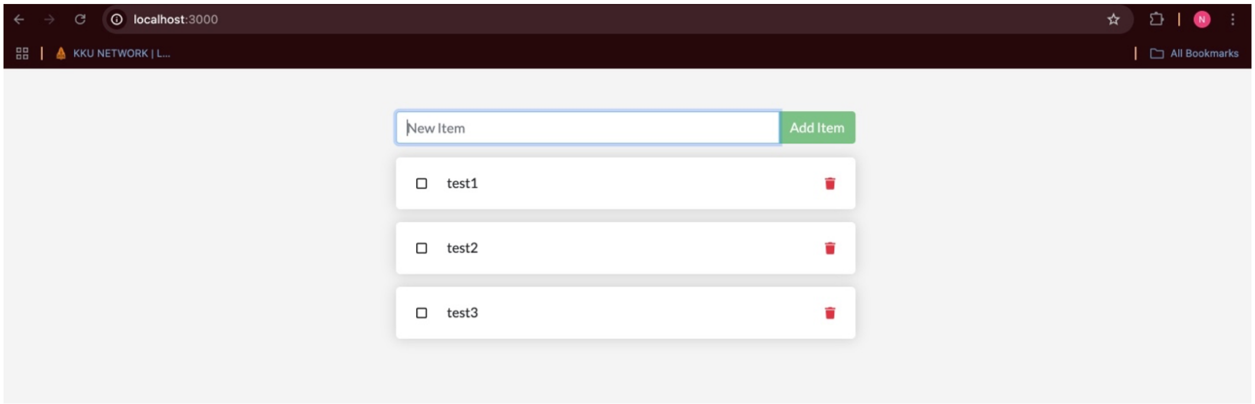
```
nipatchapakdee@Macintosh-2 app % docker build -t myapp_6533803327 .
[+] Building 13.6s (10/10) FINISHED                                        docker:desktop-linux
 => [internal] load build definition from Dockerfile                                        0.0s
 => => transferring dockerfile: 446B                                                        0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                           3.2s
 => [auth] library/node:pull token for registry-1.docker.io                                 0.0s
 => [internal] load .dockerignore                                                           0.0s
 => => transferring context: 2B                                                             0.0s
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4   0.0s
 => [internal] load build context                                                           0.1s
 => => transferring context: 4.60MB                                                         0.0s
 => CACHED [2/4] WORKDIR /app                                                               0.0s
 => [3/4] COPY . .                                                                          0.0s
 => [4/4] RUN yarn install --production                                                     9.7s
 => exporting to image                                                                      0.6s
 => => exporting layers                                                                     0.6s
 => => writing image sha256:17cd394eef48d78691c03f012ca864da261b152c2d173b895627977e54fcee23  0.0s
 => => naming to docker.io/library/myapp_6533803327                                         0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
nipatchapakdee@Macintosh-2 app % docker run -dp 3000:3000 myapp_6533803327
f477abb83c4d71cee4abab70b8cfc3dade5b151eb670119e587d86cbb8aa6331
```

7. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

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

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



หมายเหตุ: นศ.สามารถทดลองเล่น Web application ที่ทำงานอยู่ได้

9. 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 <u>ชื่อและนามสกุลของนักศึกษา</u></p>
   b. Save ไฟล์ให้เรียบร้อย
10. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5
11. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

```
nipatchapakdee@Macintosh-2 app % docker run -dp 3000:3000 myapp_6533803327
f477abb83c4d71cee4abab70b8cfc3dade5b151eb670119e587d86cbb8aa6331
nipatchapakdee@Macintosh-2 app % docker build -t myapp_6533803327 .
[+] Building 14.3s (10/10) FINISHED                                   docker:desktop-linux
 => [internal] load build definition from Dockerfile                                  0.0s
 => => transferring dockerfile: 446B                                                  0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                     2.5s
 => [auth] library/node:pull token for registry-1.docker.io                           0.0s
 => [internal] load .dockerignore                                                     0.0s
 => => transferring context: 2B                                                       0.0s
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4  0.0s
 => [internal] load build context                                                     0.0s
 => => transferring context: 10.06kB                                                  0.0s
 => CACHED [2/4] WORKDIR /app                                                         0.0s
 => [3/4] COPY . .                                                                    0.0s
 => [4/4] RUN yarn install --production                                              11.1s
 => exporting to image                                                                0.6s
 => => exporting layers                                                               0.6s
 => => writing image sha256:c85702bbadc1ca23692915d6531bda7059f39709e85cb07b144c673d893ddd22  0.0s
 => => naming to docker.io/library/myapp_6533803327                                   0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
nipatchapakdee@Macintosh-2 app % docker run -dp 3000:3000 myapp_6533803327
6254dbb0ba3bf0e96cac0fbb9a498cfe1c14654b01351fd25b35a0f8a08ff4ca
docker: Error response from daemon: driver failed programming external connectivity on endpoint sharp_ptolemy (304288b9546a750cdbe943de66f94b6108bf0e48c34332b5382fdf631ca69fa0): Bind for 0.0.0.0:3000 failed:
port is already allocated.
nipatchapakdee@Macintosh-2 app % docker run -dp 3000:3000 myapp_6533803327
e35cc47e3d7ab339aec078da60c87b211fe1f5704fb07c25ed334d01ecb8e095
docker: Error response from daemon: driver failed programming external connectivity on endpoint gracious_haslett (3b635e7761770287f7a7d6ab3482450ce2573eb5573dec84db5e25c1eca347bc): Bind for 0.0.0.0:3000 faile
d: port is already allocated.
nipatchapakdee@Macintosh-2 app % docker run -dp 3000:3000 myapp_6533803327
f1eecfd468a46231fffef56de3109e97d094335e7b3697b62e861f9c02b72da3
```

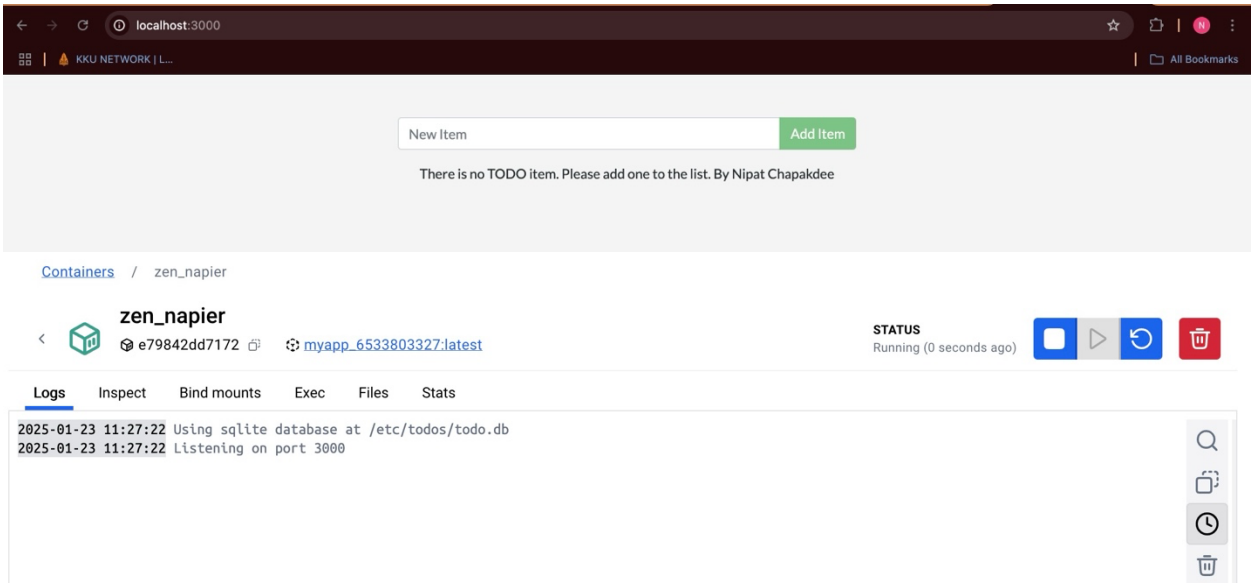
(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

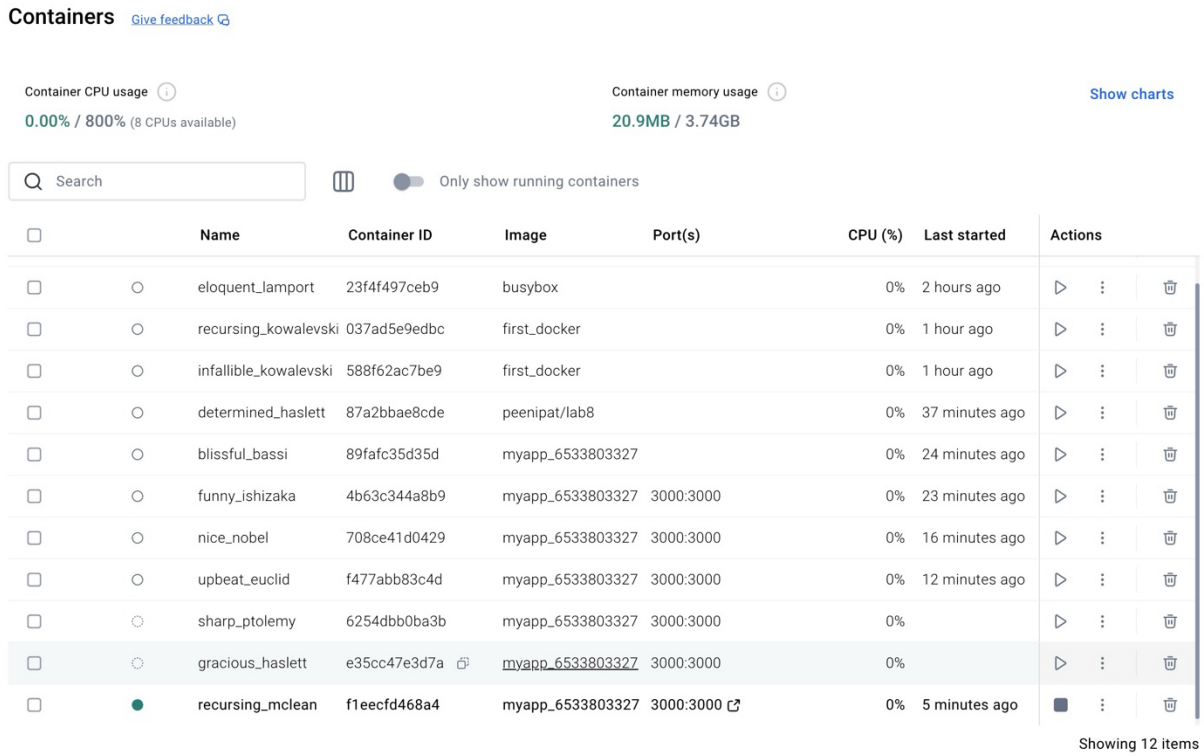
ไม่สามารถเชื่อมกับ port 3000 ในเครื่องได้เพราะว่าในเครื่องมีการใช้งาน port 3000 อยู่

12. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้
    a. ผ่าน Command line interface
        i. ใช้คำสั่ง $ docker ps เพื่อดู Container ID ที่ต้องการจะลบ
        ii. Copy หรือบันทึก Container ID ไว้
        iii. ใช้คำสั่ง $ docker stop <Container ID ที่ต้องการจะลบ> เพื่อหยุดการทำงานของ Container ดังกล่าว
        iv. ใช้คำสั่ง $ docker rm <Container ID ที่ต้องการจะลบ> เพื่อทำการลบ

    b. ผ่าน Docker desktop
        i. ไปที่หน้าต่าง Containers
        ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
        iii. ยืนยันโดยการกด Delete forever
13. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6
14. เปิด Browser ไปที่ URL = http://localhost:3000

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





แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

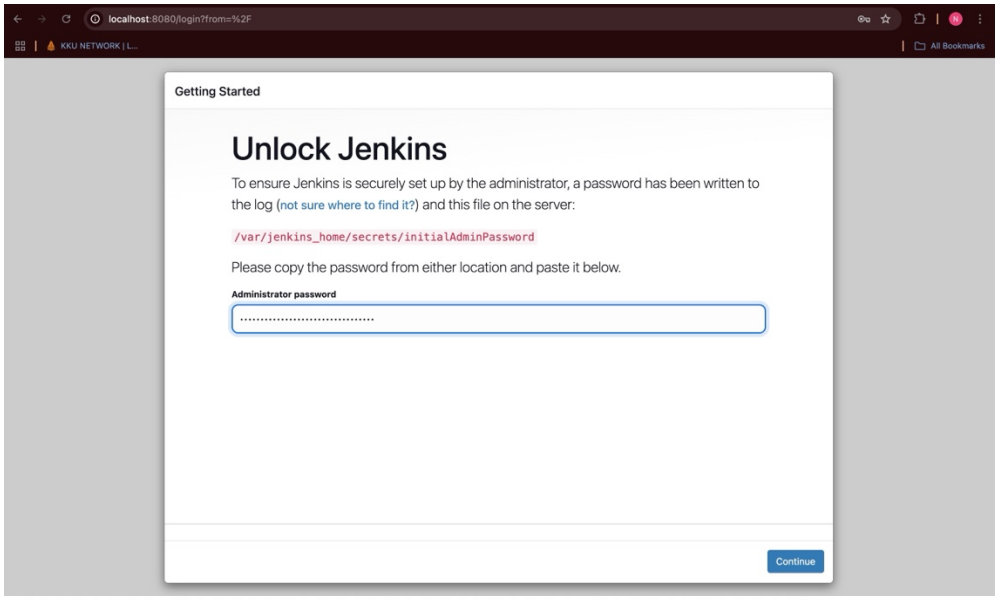
1. เปิด Command line หรือ Terminal บน Docker Desktop
2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

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

   หรือ

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

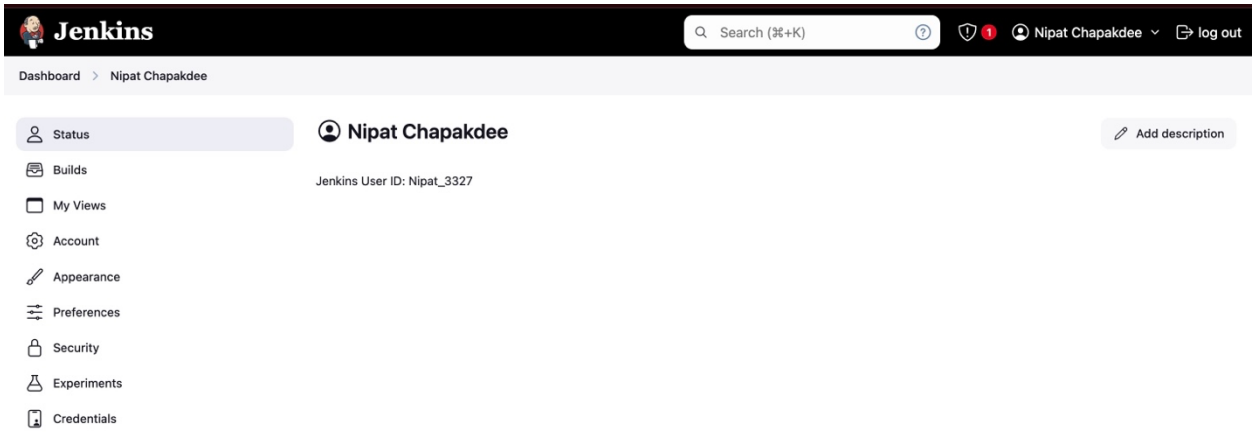
[Check point#12] Capture หน้าจอที่แสดงผล Admin password




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

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

735f5d71130247f7949d23d9514c0025

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

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

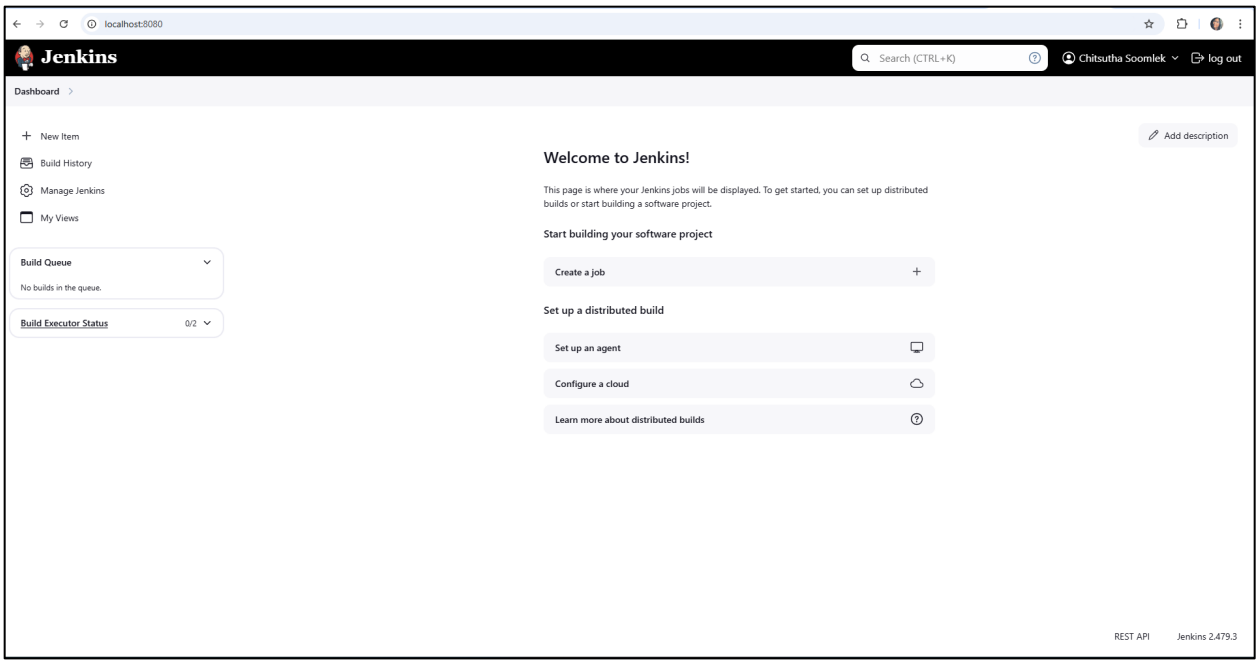
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

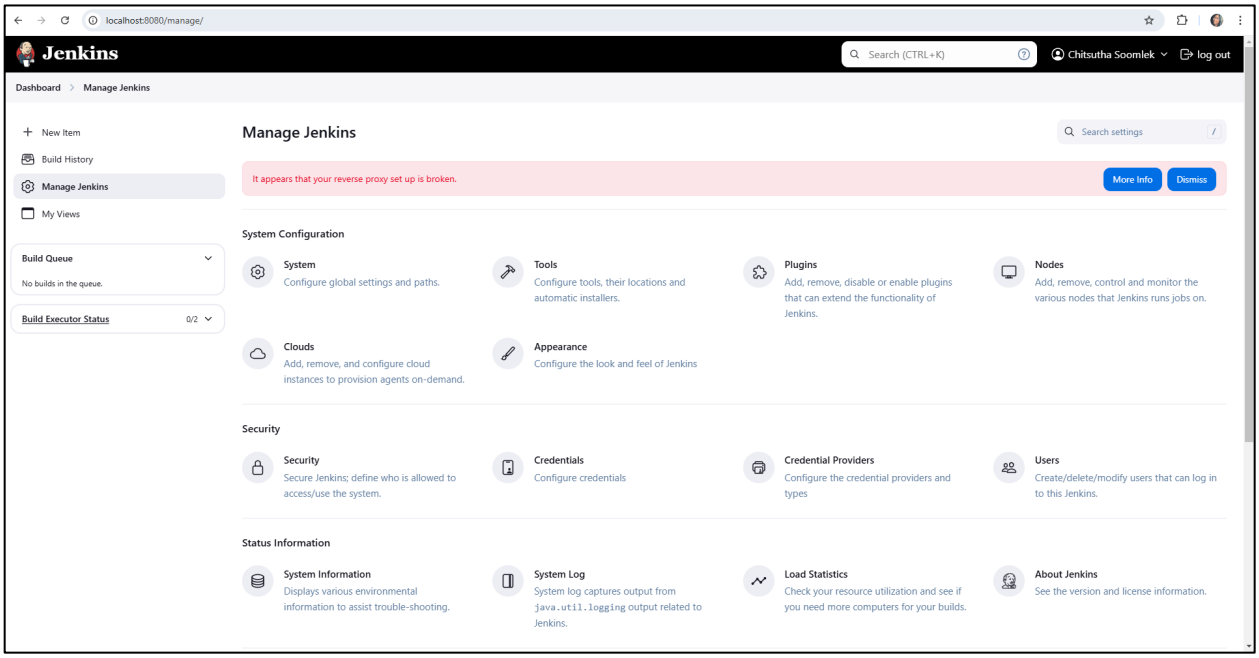


7. กำหนด Jenkins URL เป็น http://localhost:8080/lab8

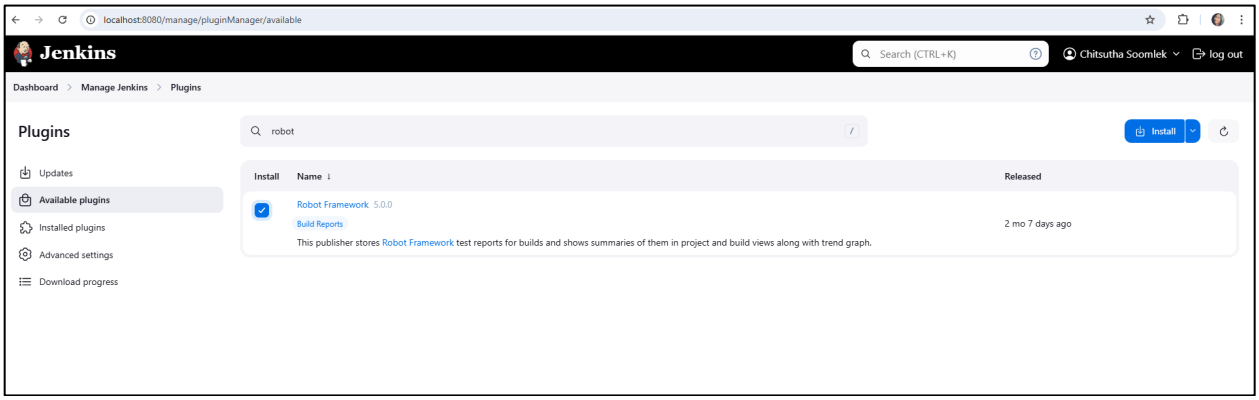
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



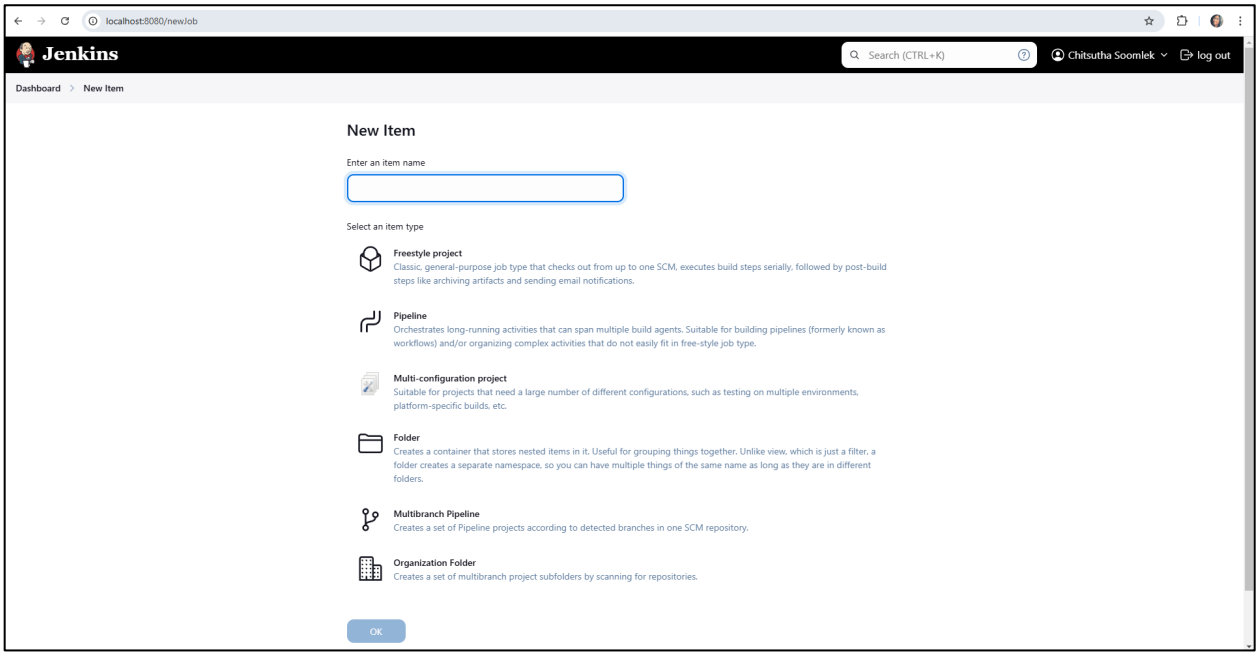
9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



12. นำไฟล์ .robot ที่ทำให้แบบฝึกปฏิบัติที่ 7 (Lab#7) ไปไว้บน Repository ของนักศึกษา จากนั้นตั้งค่าที่จำเป็นในหน้านี้ทั้งหมด ดังนี้

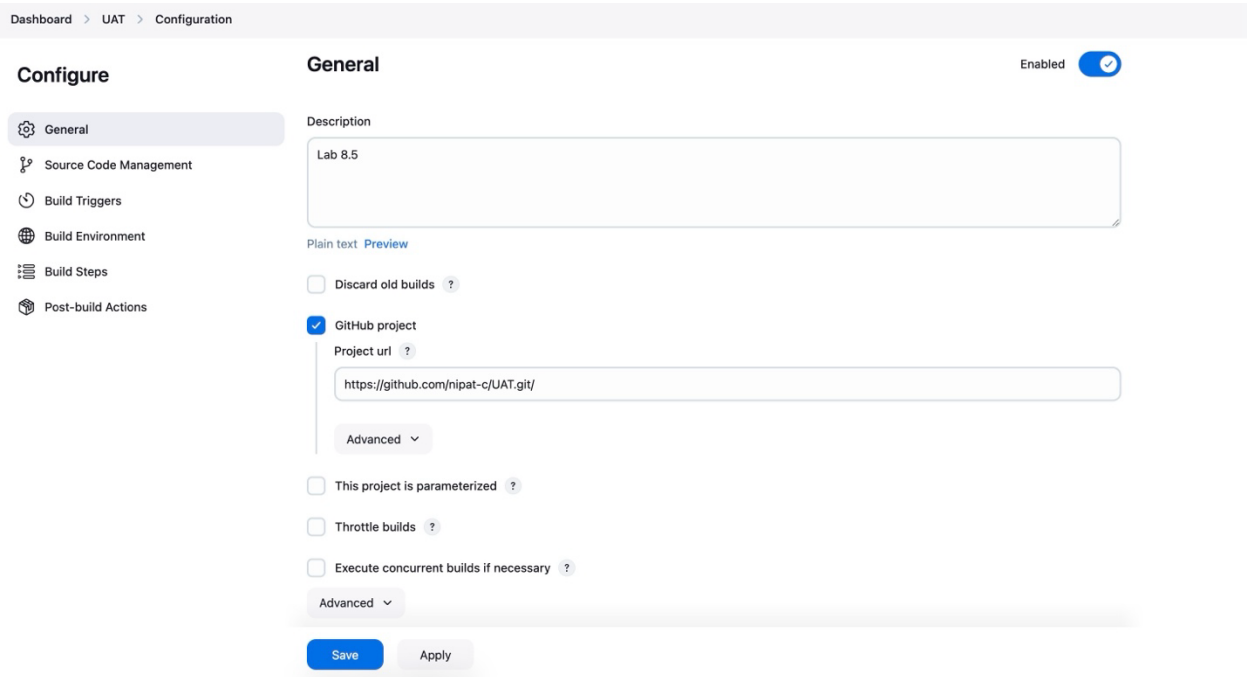
Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้



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

```
. /var/jenkins_home/robot_env/bin/activate
export PATH=$PATH:/usr/bin
mkdir -p results
robot --outputdir results test.robot
```

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

13. กด Apply และ Save
14. สั่ง Build Now

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