ใบเตรียมการสอน

แผนการสอนสัปดาห์ที่ 11	รหัสวิชา 405-47-07 บทเรียนที่ 4.2
หน่วยที่ 4 เทคโนโลยีบรรจุคอนเทนเนอร์ สำหรับการประมวลผลกลุ่มเมฆ	เวลาเรียน ทฤษฎี 2 ชั่วโมง ปฏิบัติ 2 ชั่วโมง

บทเรียนที่ 4.2 การใช้งานเบื้องต้นแอปพลิเคชันเทคโนโลยีบรรจุคอนเทนเนอร์

จุดประสงค์การสอน

4.2 เข้าใจการใช้งานเบื้องต้นแอปพลิเคชันเทคโนโลยีบรรจุคอนเทนเนอร์

4.2.1 อธิบายการค้นหาและเรียกใช้งานอิมเมจมาตรฐาน

4.2.2 อธิบายการสร้างและเริ่มต้นคอนเทนเนอร์

4.2.3 อธิบายการกำหนดการทำงานของคอนเทนเนอร์

4.2.4 อธิบายการหยุดและบังคับหยุดการทำงานของคอนเทนเนอร์

4.2.5 อธิบายการหยุดการทำงานของคอนเทนเนอร์ชั่วคราว

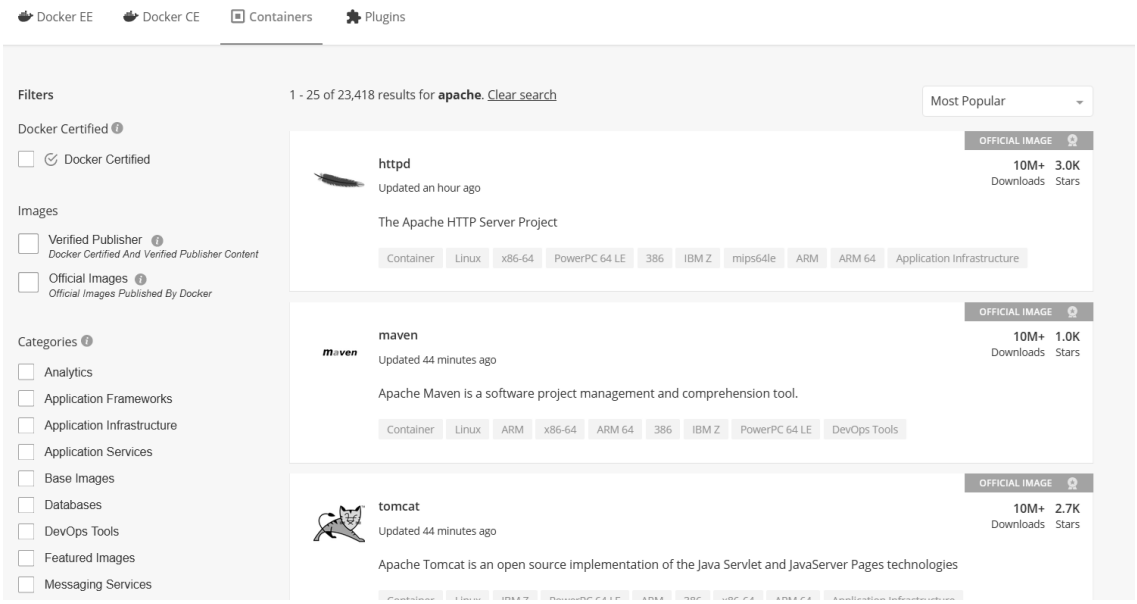
4.2.6 อธิบายการลบคอนเทนเนอร์และอิมเมจ

บทนำ

โปรแกรมด็อกเกอร์เมื่อเริ่มต้นพัฒนาถูกออกแบบให้ทำงานเฉพาะบนระบบปฏิบัติการตระกูลยูนิกซ์ หรือลีนุกซ์เท่านั้น แต่หลังจากที่ได้รับความนิยม บริษัทไมโครซอฟต์จึงเข้ามาทำความร่วมมือด้วยจึงทำให้ในปัจจุบันโปรแกรมด็อกเกอร์สนับสนุนการทำงานของระบบปฏิบัติการวินโดว์สและระบบปฏิบัติการลีนุกซ์ด้วยเช่นกัน อย่างไรก็ตาม อิมเมจส่วนใหญ่ที่ถูกเก็บไว้ที่รีจีสตรีสาธารณะยังคงเป็นอิมเมจที่ถูกสร้างมาเพื่อระบบปฏิบัติการลีนุกซ์มากกว่า

4.2.1 การค้นหาและเรียกใช้งานอิมเมจมาตรฐาน

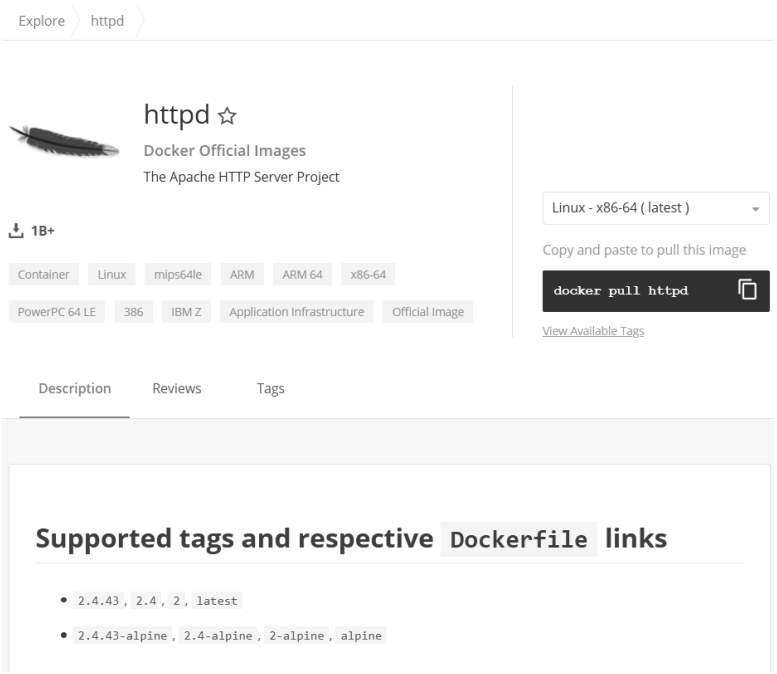
อิมเมจมาตรฐานคือ อิมเมจของซอฟต์แวร์ที่ผู้พัฒนาซอฟต์แวร์เป็นผู้จัดทำขึ้นมาและดูแลรักษาด้วยตัวเอง ข้อดีของอิมเมจมาตรฐานคืออิมเมจที่ให้ใช้งานจะเป็นอิมเมจที่ทันสมัย เพราะเมื่อผู้พัฒนาซอฟต์แวร์ปล่อยซอฟต์แวร์ในรูปแบบปกติหรือแฟ้มข้อมูลไบนารีแล้ว ผู้พัฒนาซอฟต์แวร์จะสามารถปล่อยซอฟต์แวร์ในรูปแบบอิมเมจไปพร้อมกันด้วย เมื่อการต้องการใช้งานอิมเมจมาตรฐาน เราสามารถค้นหาอิมเมจมาตรฐานได้จากรีจีสตรีสาธารณะของด็อกเกอร์ที่เว็บไซต์ https://hub.docker.com (Docker Inc. 2020a) โดยมีช่องที่สามารถค้นหาอิมเมจมาตรฐานได้ ยกตัวอย่างเช่น หากต้องการค้นหาโปรแกรมเว็บเซิร์ฟเวอร์ apache ก็พิมพ์คำค้นว่า apache เป็นต้น



ภาพที่ 4.4 ผลลัพธ์การค้นหาอิมเมจ

จากภาพที่ 4.4 แสดงผลลัพธ์อิมเมจที่เกี่ยวข้องกับคำค้น apache โดยอิมเมจมาตรฐานของโปรแกรม Apache (The Apache Software Foundation 2020) มีชื่อ httpd ปัจจุบันมียอดดาวน์โหลดมากกว่า 10 ล้านครั้ง เมื่อคลิกเข้าไปจะแสดงรายละเอียดของอิมเมจมาตรฐาน httpd ดังภาพที่ 4.5 เราสามารถเลือก

แพลตฟอร์มทางฝั่งขวามือสำหรับเครื่องคอมพิวเตอร์ส่วนบุคคลที่ใช้งานกันทั่วไปจะเลือกเป็น Linux – x86-64 (latest) จะปรากฏชื่ออิมเมจมาตรฐานที่เหมาสมกับแพลตฟอร์มที่เลือก



ภาพที่ 4.5 ส่วนหนึ่งหน้าเว็บแสดงรายละเอียดอิมเมจ httpd

เราสามารถเริ่มดาวน์โหลดอิมเมจ httpd ที่เป็นเวอร์ชันล่าสุด มาเก็บไว้ในเครื่องได้ โดยใช้คำสั่ง

```
# docker pull httpd
```

```
[root@localhost ~]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
54fec2fa59d0: Pull complete
8219e18ac429: Pull complete
3ae1b816f5e1: Pull complete
a5aa59ad8b5e: Pull complete
4f6febfae8db: Pull complete
Digest: sha256:c9e4386ebcdf0583204e7a54d7a827577b5ff98b932c498e9ee603f7050db1c1
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
```

ภาพที่ 4.6 หน้าจอผลลัพธ์คำสั่งดาวน์โหลดอิมเมจ

เมื่อดาวน์โหลดเสร็จสิ้น จะปรากฏข้อความดังภาพที่ 4.6 ว่าอิมเมจได้ถูกดาวน์โหลดสำเร็จแล้ว เราสามารถตรวจสอบรายการอิมเมจที่ถูกดาวน์โหลดมาไว้ในเครื่อง โดยใช้คำสั่ง

```
# docker image ls
```

```
[root@localhost ~]# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
httpd               latest              b2c2ab6dcf2e        8 days ago          166MB
```

ภาพที่ 4.7 หน้าจอผลลัพธ์คำสั่งแสดงรายการอิมเมจ

4.2.2 การสร้างและเริ่มต้นคอนเทนเนอร์

สำหรับการสร้างคอนเทนเนอร์โปรแกรมด็อกเกอร์สนับสนุนพารามิเตอร์ที่ควรกำหนดให้เสร็จก่อนการสร้าง เนื่องจากคอนเทนเนอร์นั้นถูกทำงานผ่านเครือข่ายเสมือนของด็อกเกอร์ไม่สามารถติดต่อกับด็อกเกอร์โฮสต์ได้โดยตรง โดยเฉพาะใช้จองพอร์ตไว้เชื่อมต่อกับด็อกเกอร์โฮสต์ ดังนั้น หากต้องการกำหนดการเชื่อมต่อคอนเทนเนอร์กับด็อกเกอร์โฮสต์นั้น ดังนี้

1) คอนเทนเนอร์นี้ทำงานด้วยโหมดปฏิสัมพันธ์หรือโหมดพื้นหลัง หากเป็นโหมดปฏิสัมพันธ์ เราจะสามารถเข้าถึงเชลล์และพิมพ์คำสั่งได้ทันที แต่หากเป็นโหมดพื้นหลัง คอนเทนเนอร์จะทำงานพื้นหลังโดยที่ไม่มีการปฏิสัมพันธ์กับเรา
   พารามิเตอร์ โหมดปฏิสัมพันธ์ -it โหมดพื้นหลัง คือ -d และโหมดปฏิสัมพันธ์แต่สั่งให้ทำงานอยู่พื้นหลัง คือ -dit
2) คอนเทนเนอร์นี้มีการส่งตัวแปรสภาพแวดล้อมหรือไม่ ตัวแปรสภาพแวดล้อมที่ส่งไปยังคอนเทนเนอร์มักเป็นเงื่อนไขในการตัดสินใจการทำงานของอิมเมจ เราสามารถศึกษาตัวแปรสภาพแวดล้อมได้จากรีจีสตรีอิมเมจมาตรฐาน ซึ่งผู้พัฒนาอิมเมจจะกำหนดตัวแปรที่สำคัญ เช่น รหัสผู้ใช้ รหัสผ่าน ฐานข้อมูลเริ่มต้น เป็นต้น
   พารามิเตอร์ของตัวแปรสภาพแวดล้อม คือ -e <ชื่อตัวแปร>=<ค่าตัวแปร>
3) คอนเทนเนอร์นี้มีการเชื่อมต่อที่เก็บข้อมูลใดกับไดเรกทอรีของด็อกเกอร์โฮสต์หรือไม่ การเชื่อมต่อที่เก็บข้อมูลกับไดเรกทอรีบนโฮสต์จะช่วยให้เมื่อลบคอนเทนเนอร์ข้อมูลจะไม่สูญหายไป การเชื่อมต่อที่เก็บข้อมูลของด็อกเกอร์จะสนับสนุนการเชื่อมต่อแบบรายไดเรกทอรี (Per directory) เท่านั้น ไม่สามารถรายแฟ้มข้อมูล (Per file) ได้
   พารามิเตอร์ของการเชื่อมที่เก็บข้อมูล คือ -v <ไดเรกทอรีบนโฮสต์>:<ไดเรกทอรีบนคอนเทนเนอร์>
4) คอนเทนเนอร์นี้มีเปิดพอร์ตและการเชื่อมต่อกับพอร์ตบนเครื่องด็อกเกอร์โฮสต์หรือไม่ การเชื่อมต่อพอร์ตจะช่วยให้ซอฟต์แวร์ที่ทำงานอยู่ในคอนเทนเนอร์สามารถเชื่อมต่อไปยังเครือข่ายภายนอกได้ เช่น คอนเทนเนอร์ของเว็บเซิร์ฟเวอร์เปิดพอร์ต 80 เพื่อให้เข้าถึงเว็บเซิร์ฟเวอร์ในคอนเทนเนอร์ได้ เราจำเป็นต้องเชื่อมต่อพอร์ตของคอนเทนเนอร์กับพอร์ตของด็อกเกอร์โฮสต์
   พารามิเตอร์ของการเชื่อมต่อพอร์ต คือ -p <ที่อยู่ไอพีของโฮสต์:พอร์ตของโฮสต์>:<พอร์ตของคอนเทนเนอร์> ในกรณีที่ไม่ระบุที่อยู่ไอพีของโฮสต์ ค่าที่อยู่ไอพีของโฮสต์ที่จะเชื่อมต่อคือ 0.0.0.0 หมายความว่าเครื่องคอมพิวเตอร์ใดที่เชื่อมต่อกับด็อกเกอร์โฮสต์จะสามารถเข้าถึงคอนเทนเนอร์ตามพอร์ตที่กำหนด ในบางกรณีเพื่อความมั่นคงปลอดภัย เราอาจระบุให้ที่อยู่ไอพีของโฮสต์เป็น 127.0.0.1 ทำให้บริการที่พอร์ตของคอนเทนเนอร์นั้นไม่สามารถเข้าถึงจากเครื่องที่อยู่ในเครือข่ายเดียวกับด็อกเกอร์โฮสต์ได้

5) คอนเทนเนอร์นี้ใช้อิมเมจใดเพื่อสร้างชั้นข้อมูลฐาน ในกรณีที่เป็นอิมเมจมาตรฐาน ชื่อของอิมเมจมักเป็นคำเดียว เช่น httpd แต่หากเราสร้างบัญชีที่รีจีสตรีและส่งออกอิมเมจของเราเองชื่อจะเป็น <รหัสผู้ใช้>/<ชื่ออิมเมจ> เช่น utarn/httpd เป็นต้น
6) ต้องการกำหนดคำสั่งพิเศษที่ทำงานเมื่อเริ่มต้นคอนเทนเนอร์หรือไม่ หากไม่กำหนดคอนเทนเนอร์จะเริ่มต้นการทำงานตามคำสั่งที่ระบุไว้ในอิมเมจ คำสั่งพิเศษอาจไม่จำเป็นหากเราใช้อิมเมจมาตรฐานหรืออิมเมจที่มีผู้พัฒนาดูแลและเป็นที่นิยมอยู่แล้ว ส่วนใหญ่นิยมใช้กำหนดคำสั่งเข้าถึงเชลล์เพื่อตรวจสอบการทำงานหรือตรวจสอบข้อผิดพลาดของอิมเมจที่สร้างขึ้นมาว่าทำงานได้ถูกต้องหรือไม่ ตัวอย่างคำสั่งพิเศษ เช่น sh หรือ bash สำหรับการเข้าถึงเชลล์ในลีนุกซ์

จากพารามิเตอร์สำหรับการสร้างคอนเทนเนอร์ เราสามารถสร้างคอนเทนเนอร์ของเว็บเซิร์ฟเวอร์ Apache ที่ตรงกับพารามิเตอร์ดังนี้

1) ทำงานอยู่ในโหมดเบื้องหลัง ใช้พารามิเตอร์ -d
2) เชื่อมต่อไดเรกทอรี /usr/local/apache2/htdocs/ ของคอนเทนเนอร์กับไดเรกทอรี /root/website ของโฮสต์
3) เชื่อมต่อต่อพอร์ต 80 ของคอนเทนเนอร์ กับพอร์ต 80 ของโฮสต์
4) ใช้อิมเมจมาตรฐาน ชื่อ httpd

จากเงื่อนไขดังกล่าวจะสามารถสร้างเป็นคำสั่ง docker run ได้ดังนี้

```
# docker run -d -v /root/website:/usr/local/apache2/htdocs -p 80:80
     httpd
```

เมื่อพิมพ์คำสั่งแล้วจะปรากฏผลลัพธ์ดังภาพที่ 4.8 แสดงรหัสของคอนเทนเนอร์ที่สร้างใหม่ หลังจากที่สั่งให้คอนเทนเนอร์เว็บเซิร์ฟเวอร์ทำงานแล้ว เราสามารถตรวจสอบการทำงานของคอนเทนเนอร์ด้วยคำสั่ง

```
# docker ps
```

```
[root@localhost ~]# docker run -d -v /root/website:/usr/local/apache2/htdocs -p 80:80 httpd
1bba17d1cb3cb6e906a2cfa35540cc4fe8d946945a3d49f8dd961fa1da2670e2
```

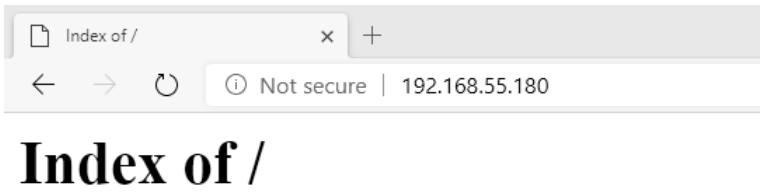
ภาพที่ 4.8 หน้าจอผลลัพธ์คำสั่งสร้างคอนเทนเนอร์ใหม่

เมื่อพิมพ์คำสั่ง จะปรากฏผลลัพธ์ดังภาพที่ 4.9 แสดงว่าคอนเทนเนอร์ที่สร้างขึ้นมามีรหัสเป็น 1bba17d1cb3c และมีชื่อว่า frosty_tereshkova

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
1bba17d1cb3c        httpd               "httpd-foreground"       2 minutes ago       Up 2 minutes
```

ภาพที่ 4.9 หน้าจอผลลัพธ์คำสั่งแสดงรายการคอนเทนเนอร์

เนื่องจากคอนเทนเนอร์ได้เชื่อมต่อพอร์ต 80 ของด็อกเกอร์โฮสต์ เราสามารถทดลองเข้าเว็บของคอนเทนเนอร์เว็บเซิร์ฟเวอร์นี้ได้จากเบราเซอร์ที่ http://<ที่อยู่ไอพีของโฮสต์> จะปรากฏผลลัพธ์ดังภาพที่ 4.10

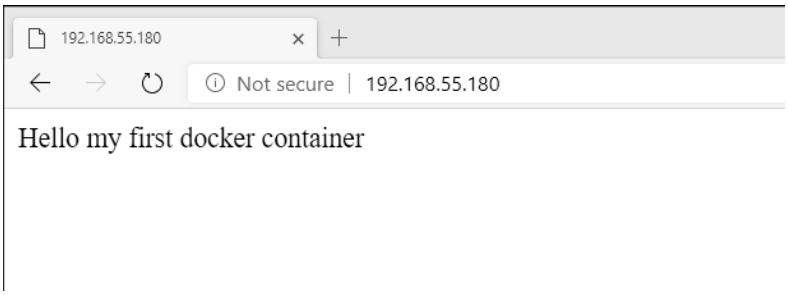


ภาพที่ 4.10 หน้าจอผลลัพธ์ของเว็บเบราเซอร์เมื่อเข้าถึงเว็บของคอนเทนเนอร์

เนื่องจากคอนเทนเนอร์ได้เชื่อมต่อกับไดเรกทอรีของ /root/website เราสามารถทดลองสร้างหน้าแรกของเว็บไซต์อย่างง่ายๆ ด้วยคำสั่ง

```
# echo "Hello my first docker container" > /root/website/index.html
```

จากนั้นเมื่อกด F5 เพื่อรีเฟรชเบราเซอร์ใหม่ เราจะได้ผลลัพธ์ดังภาพที่ 4.11 ว่าหน้าแรกที่ทดลองสร้างปรากฏบนเว็บเซิร์ฟเวอร์แล้ว



ภาพที่ 4.11 หน้าจอผลลัพธ์ของเว็บเบราเซอร์เมื่อเพิ่มหน้าเว็บหน้าแรก

4.2.3 อธิบายการกำหนดการทำงานของคอนเทนเนอร์

หนึ่งในพารามิเตอร์ที่นิยมแก้ไขหลังการสร้างคอนเทนเนอร์คือ การกำหนดการเปิดการทำงานของคอนเทนเนอร์เมื่อเริ่มต้นบริการของด็อกเกอร์ ซึ่งบริการของด็อกเกอร์นั้นสามารถตั้งค่าให้เริ่มการทำงานเมื่อรีบูตเครื่องโฮสต์ได้ แต่โดยเริ่มต้นหากคอนเทนเนอร์อยู่ในสถานะเปิดการทำงานก่อนการปิดหรือรีสตาร์ทบริการด็อกเกอร์ เมื่อบริการของด็อกเกอร์ในด็อกเกอร์โฮสต์เริ่มต้นการทำงานอีกครั้ง คอนเทนเนอร์จะอยู่ในสถานะออกการทำงาน (Exited)

เพื่อความสะดวกแก่ผู้ใช้ การอ้างอิงชื่อของคอนเทนเนอร์ในด็อกเกอร์จะต้องใช้ข้อมูลเพียงอย่างใดอย่างหนึ่ง คือ รหัสของคอนเทนเนอร์หรือชื่อของคอนเทนเนอร์ ในกรณีที่เป็นรหัสคอนเทนเนอร์เราสามารถพิมพ์แค่

บางส่วนที่ไม่ซ้ำกับรหัสคอนเทนเนอร์ของคอนเทนเนอร์อื่น เช่น หากคอนเทนเนอร์มีรหัสคอนเทนเนอร์เป็น 1bba17d1cb3c และมีชื่อว่า frosty_tereshkova เราสามารถใช้คำว่า 1 หรือ 1b หรือ 1bba17d1cb3c โดยที่ส่วนที่พิมพ์นั้นต้องไม่มีคอนเทนเนอร์ที่มีรหัสซ้ำกันในส่วนนั้นได้ ดังนั้น เพื่อกำหนดให้คอนเทนเนอร์อยู่ในสถานะเริ่มการทำงานพร้อมกับบริการของด็อกเกอร์ได้ สามารถใช้คำสั่งอย่างใดอย่างหนึ่ง ดังนี้

```
# docker update --restart always 1b

# docker update --restart always 1bba17d1cb3c

# docker update --restart always frosty_tereshkova
```

4.2.4 การหยุดและบังคับหยุดการทำงานของคอนเทนเนอร์

วิธีการหยุดการทำงานของคอนเทนเนอร์นั้นขึ้นอยู่กับโหมดการทำงานของคอนเทนเนอร์ว่าทำงานในโหมดการทำงานใด ในกรณีที่ทำงานด้วยโหมดปฏิสัมพันธ์ หน้าเราปิดหน้าจอการเข้าถึงระยะไกล เช่น โปรแกรม Putty หรือกด CTRL + C ก็จะหยุดการทำงานของคอนเทนเนอร์นั้นทันที เราสามารถใช้คำสั่งต่อไปนี้หยุดการทำงานของคอนเทนเนอร์ได้

```
# docker stop frosty_tereshkova
```

เราสามารถใช้คำสั่งต่อไปนี้เพื่อบังคับหยุดการทำงานของคอนเทนเนอร์ได้

```
# docker kill frosty_tereshkova
```

ในกรณีที่คอนเทนเนอร์หยุดการทำงานไปแต่ยังไม่ถูกลบ เราสามารถตรวจสอบสถานะการทำงานของคอนเทนเนอร์ที่หยุดการทำงานไปด้วยคำสั่ง

```
# docker ps -a
```

หากต้องการเริ่มการทำงานใหม่ เราสามารถใช้คำสั่งต่อไปนี้เพื่อเริ่มการทำงานของคอนเทนเนอร์อีกครั้งได้

```
# docker start frosty_tereshkova
```

4.2.5 การหยุดการทำงานของคอนเทนเนอร์ชั่วคราว

การหยุดการทำงานของคอนเทนเนอร์ชั่วคราวจะเป็นการหยุดการทำงานของโพรเซสในคอนเทนเนอร์ ทำให้โพสเซสในคอนเทนเนอร์ไม่ใช้ทรัพยากรจากหน่วยประมวลผลกลาง แต่คอนเทนเนอร์ยังคงอยู่ในสถานะที่ทำงานและยังจองพื้นที่หน่วยความจำหลักอยู่ เราสามารถใช้คำสั่งต่อไปนี้เพื่อหยุดการทำงานของคอนเทนเนอร์ชั่วคราวได้

```
# docker pause frosty_tereshkova
```

เมื่อพร้อมให้คอนเทนเนอร์ที่หยุดการทำงานช่วยคราวสามารถทำงานได้ต่อ เราสามารถใช้คำสั่งต่อไปนี้เพื่อดำเนินการทำงานของคอนเทนเนอร์ต่อได้

```
# docker unpause frosty_tereshkova
```

4.2.6 การลบคอนเทนเนอร์และอิมเมจ

เมื่อสร้างและสั่งให้คอนเทนเนอร์ทำงานแล้ว โปรแกรมด็อกเกอร์จะสร้างชั้นข้อมูลสำหรับคอนเทนเนอร์ขึ้นมาอีกชั้น ในชั้นข้อมูลนั้นบางส่วนอาจเชื่อมต่อไปยังไดเรกทอรีของด็อกเกอร์โฮสต์ทำให้คอนเทนเนอร์นั้นใช้พื้นที่เก็บข้อมูลบนดิสก์ เมื่อหยุดการทำงานแล้ว โปรแกรมด็อกเกอร์จะไม่ลบคอนเทนเนอร์นั้นออก เพราะคอนเทนเนอร์เมื่อหยุดการทำงานไปแล้ว ยังสามารถเปิดการทำงานใหม่อีกครั้งได้ ดังนั้น ผู้ใช้จึงควรลบคอนเทนเนอร์ที่ไม่ได้ใช้งานออกอยู่เสมอเมื่อไม่ได้ใช้ เราสามารถใช้คำสั่งต่อไปนี้เพื่อลบของคอนเทนเนอร์ต่อได้

```
# docker rm frosty_tereshkova
```

ในกรณีที่มีคอนเทนเนอร์จำนวนมากที่ต้องการลบ เราสามารถลบคอนเทนเนอร์ที่ไม่ได้ทำงานแล้ว โดยใช้คำสั่ง

```
# docker container prune -f
```

พารามิเตอร์ -f ของคำสั่งมีไว้เพื่อลบคอนเทนเนอร์ทันทีทุกคอนเทนเนอร์

สำหรับอิมเมจ เมื่อดาวน์โหลดมาแล้วหรือถูกสร้างขึ้นมา แม้โปรแกรมด็อกเกอร์จะดาวน์โหลดหรือเก็บข้อมูลเฉพาะชั้นข้อมูลที่ไม่ซ้ำกัน แต่เมื่อเวลาผ่านไป เราอาจมีอิมเมจที่ไม่ได้ใช้งานหรือจะไม่ได้ใช้สร้างคอนเทนเนอร์อีก ซึ่งอิมเมจเหล่านี้ใช้พื้นที่เก็บข้อมูลบนดิสก์ การลบอิมเมจออกจำเป็นต้องใช้รหัสของอิมเมจเท่านั้น เช่น หากอิมเมจ httpd มีรหัสอิมเมจดังภาพที่ x เป็น b2c2ab6dcf2e สามารถใช้คำสั่ง

```
# docker image rm b2c2ab6dcf2e
```

ในกรณีที่ต้องการลบอิมเมจที่ไม่ได้ใช้งานแล้วทั้งหมด เราสามารถใช้คำสั่ง

```
# docker image prune
```

แต่อิมเมจที่จะลบได้นั้นจะต้องไม่ถูกอ้างอิงจากคอนเทนเนอร์ใด ๆ เลยแม้คอนเทนเนอร์นั้นจะหยุดการทำงานไปแล้วก็ตาม หากต้องการลบอิมเมจที่ไม่ถูกอ้างอิงจากคอนเทนเนอร์ที่กำลังทำงานอยู่เท่านั้น สามารถเพิ่มพารามิเตอร์ -a ได้เป็นคำสั่ง

```
# docker image prune -a
```

บทสรุป

เนื้อหา	คำสั่งหรือชื่อคำสั่งที่เกี่ยวข้อง
รีจีสตรีสาธารณะของด็อกเกอร์	https://hub.docker.com
คำสั่งดาวน์โหลดอิมเมจ	# docker pull
ตรวจสอบรายการอิมเมจที่ถูกดาวน์โหลดมาไว้ในเครื่อง	# docker image ls
ตัวอย่างคำสั่งสร้างคอนเทนเนอร์จากอิมเมจ	# docker run -d -v /root/website:/usr/local/apache2/htdocs -p 80:80 httpd
ตรวจสอบการทำงานของคอนเทนเนอร์	# docker ps
หลักการอ้างอิงชื่อคอนเทนเนอร์	1. รหัสของคอนเทนเนอร์โดยพิมพ์เฉพาะบางส่วนที่ไม่ซ้ำกับรหัสคอนเทนเนอร์อื่น 2. ชื่อคอนเทนเนอร์เต็ม
กำหนดให้คอนเทนเนอร์อยู่ในสถานะเริ่มการทำงานพร้อมกับบริการของด็อกเกอร์	# docker update --restart always frosty_tereshkova
หยุดการทำงานของคอนเทนเนอร์	# docker stop frosty_tereshkova
บังคับหยุดการทำงานของคอนเทนเนอร์	# docker kill frosty_tereshkova
เริ่มการทำงานของคอนเทนเนอร์	# docker start frosty_tereshkova
หยุดการทำงานของคอนเทนเนอร์ชั่วคราว	# docker pause frosty_tereshkova
ดำเนินการทำงานของคอนเทนเนอร์ต่อ	# docker unpause frosty_tereshkova
ลบของคอนเทนเนอร์	# docker rm frosty_tereshkova
ลบคอนเทนเนอร์ที่ไม่ได้ทำงานแล้ว	# docker container prune -f
ลบอิมเมจ	# docker image rm รหัสอิมเมจ
ลบอิมเมจที่ไม่ได้ใช้งานแล้วทั้งหมด	# docker image prune
ลบอิมเมจที่ไม่ถูกอ้างอิงจากคอนเทนเนอร์ที่กำลังทำงานอยู่เท่านั้น	# docker image prune -a

แบบฝึกหัดที่ 4.2

1. บอกที่อยู่ของรีจีสตรีมาตรฐานที่สามารถค้นหาอิมเมจมาตรฐานได้

2. บอกพารามิเตอร์ และความหมายของพารามิเตอร์สำหรับใช้สร้างคอนเทนเนอร์มาอย่างน้อย 5 พารามิเตอร์

3. คำสั่ง docker stop แตกต่างจากคำสั่ง docker kill อย่างไร

4. บอกหลักการการอ้างอิงชื่อและรหัสคอนเทนเนอร์เพื่อใช้ในคำสั่งเครื่องมือด็อกเกอร์ลูกข่าย

5. บอกคำสั่งที่ใช้ลบอิมเมจทั้งหมดที่ไม่ถูกใช้ในเครื่องด็อกเกอร์โฮสต์

ใบงานฝึกปฏิบัติที่ 4.2

1. สร้างแฟ้มข้อมูล index.html โดยใส่เนื้อหาเป็นชื่อ และรหัสนักศึกษา แล้วบันทึกไว้ที่ /root/website
2. สร้างคอนเทนเนอร์เว็บเซิร์ฟเวอร์ ให้มีคุณสมบัติดังนี้
    2.1. เปิดการทำงานพร้อมด็อกเกอร์โฮสต์เมื่อรีบูตเครื่อง
    2.2. อ่านข้อมูลจาก /root/website เป็นที่อยู่สำหรับเว็บไซต์
    2.3. สามารถเปิดเว็บไซต์โดยตรงได้จากที่อยู่ไอพีของด็อกเกอร์โฮสต์