ใบเตรียมการสอน

แผนการสอนสัปดาห์ที่ 12	รหัสวิชา 405-47-07 บทเรียนที่ 4.3
หน่วยที่ 4 เทคโนโลยีบรรจุคอนเทนเนอร์ สำหรับการประมวลผลกลุ่มเมฆ	เวลาเรียน ทฤษฎี 2 ชั่วโมง ปฏิบัติ 2 ชั่วโมง

บทเรียนที่ 4.3 การบริหารจัดการอิมเมจและคอนเทนเนอร์

จุดประสงค์การสอน

4.3 เข้าใจการบริหารจัดการอิมเมจและคอนเทนเนอร์

4.3.1 อธิบายการจัดการอิมเมจหลายเวอร์ชัน

4.3.2 อธิบายการตรวจสอบการทำงานของคอนเทนเนอร์

4.3.3 อธิบายการเข้าถึงและการออกคอนโซลของคอนเทนเนอร์

4.3.4 อธิบายการตรวจสอบประวัติการทำงานของคอนเทนเนอร์

4.3.5 อธิบายการตรวจสอบทรัพยากรที่คอนเทนเนอร์ใช้

บทนำ

หลังจากที่ดาวน์โหลดอิมเมจและสร้างคอนเทนเนอร์ได้จากอิมเมจแล้ว คอนเทนเนอร์จะเริ่มต้นทำงานตามคำสั่งที่กำหนด โดยสามารถเชื่อมต่อกับไดเรกทอรีและพอร์ตเครือข่ายของด็อกเกอร์โฮสต์ได้ การเรียนรู้การจัดการอิมเมจและคอนเทนเนอร์ จะช่วยให้สามารถการใช้ประโยชน์ของเทคโนโลยีบรรจุคอนเทนเนอร์อย่างเต็มที่ และควบคุมการทำงานและตรวจสอบการทำงานของคอนเทนเนอร์ให้เป็นไปตามที่เราต้องการได้อย่างถูกต้องและมีประสิทธิภาพ

4.3.1 การจัดการอิมเมจหลายเวอร์ชัน

ลักษณะหนึ่งที่สำคัญของอิมเมจคือ อิมเมจหนึ่งอิมเมจประกอบด้วยแท็ก (Tag) ตั้งแต่ 1 แท็กเป็นต้นไป หากเวลาสร้างหรือเรียกใช้งานอิมเมจไม่ระบุชื่อแท็ก ชื่อแท็กเริ่มต้นจะเป็น latest เสมอ โดยหากอิมเมจมีชื่อแท็กที่แตกต่างกันจะมีข้อมูลที่บันทึกอยู่ในชั้นข้อมูลแตกต่างกันออกไป การตั้งชื่อแท็กมีประโยชน์สำหรับผู้ดูแลรักษาอิมเมจโดยเฉพาะกรณีที่ซอฟต์แวร์ที่เผยแพร่นั้นมีหลายเวอร์ชัน หรือหลายระบบปฏิบัติการ อิมเมจหนึ่งอิมเมจอาจมีแท็กหลายแท็กที่ชี้ไปยังอิมเมจที่มีชั้นข้อมูลเดียวกัน เช่น อิมเมจของ httpd มีแท็ก 2.4.43 แท็ก 2.4 แท็ก 2 และแท็ก latest หมายถึงอิมเมจที่มีชั้นข้อมูลเดียวกัน แต่มีวิธีการใช้งานต่างบริบทกัน เช่น ในกรณีที่ผู้ใช้งานต้องการเรียกเฉพาะเวอร์ชัน 2.4.43 สามารถอ้างอิงแท็ก 2.4.43 ได้ แต่เมื่อผู้พัฒนาปล่อยซอฟต์แวร์เวอร์ชัน 2.5.1 ผู้ใช้งานที่อ้างอิงแท็ก 2.4.43 จะไม่มีการอัพเดทอิมเมจ แต่ผู้ที่อ้างอิงแท็ก 2.5 แท็ก 2 และแท็ก latest จะได้รับการอัพเดทซอฟต์แวร์ ดังนั้น ประโยชน์ของแท็กจะทำให้ผู้ใช้งานสามารถควรคุมขอบเขตของเวอร์ชันที่ต้องการเรียกใช้ได้

เมื่อทราบขอบเขตของเวอร์ชันซอฟต์แวร์ที่ต้องการเรียกใช้งานแล้ว เราสามารถดาวน์โหลดอิมเมจ httpd ที่เป็นเวอร์ชัน 2 ที่ล่าสุดเสมอ มาเก็บไว้ในเครื่องได้ โดยใช้คำสั่ง

```
# docker pull httpd:2
```

แต่หากต้องการเวอร์ชันล่าสุดเสมอ สามารถใช้คำสั่ง

```
# docker pull httpd
```

เมื่อแสดงรายการของอิมเมจจะมีคอลัมน์ระบุชื่อแท็กของอิมเมจอยู่แล้ว โดยอิมเมจชื่อเดียวกันแต่มีแท็กต่างกันจะแสดงคนละรายการ

4.3.2 การตรวจสอบการทำงานของคอนเทนเนอร์

เมื่อคอนเทนเนอร์ถูกสร้างคอนเทนเนอร์จะมีสถานการณ์ทำงานเป็นเปิดการทำงาน (Active) อย่างไรก็ตาม หากการตั้งค่าจากคำสั่ง docker run ผิดพลาดไม่สอดคล้องกับที่ผู้พัฒนาอิมเมจกำหนด คอนเทนเนอร์อาจออกการทำงาน (Exited) ได้ ในกรณีที่คำสั่งสร้างคอนเทนเนอร์ระบุให้รีสตาร์ทคอนเทนเนอร์ (พารามิเตอร์ --restart always) ด็อกเกอร์โฮสต์จะหยุดรอเป็นระยะเวลาแบบสุ่มเพื่อทดลองเปิดการทำงานคอนเทนเนอร์

ใหม่ (Restarting) อีกครั้ง โดยระยะเวลาสุ่มนี้จะค่อย ๆ ห่างขึ้นเรื่อย ๆ อย่างไรก็ตาม เราสามารถตรวจสอบสถานะการทำงานของคอนเทนเนอร์ว่ายังทำงานอยู่หรือไม่ ด้วยคำสั่ง

```
# docker ps -a
```

พารามิเตอร์ -a จะแสดงรายการของคอนเทนเนอร์ทั้งหมดที่เคยถูกสร้าง แต่ยังไม่ถูกลบออกจากเครื่อง ทำให้เราสามารถตรวจสอบสถานะของคอนเทนเนอร์ แต่หากต้องการตรวจสอบว่าภายในคอนโซลของคอนเทนเนอร์นั้นมีการแสดงผลว่าอย่างไร เพื่อตรวจสอบการทำงานโดยละเอียด และหากคอนเทนเนอร์ทำงานได้ไม่ถูกต้องหน้าจอคอนซอลอาจแสดงข้อผิดพลาดว่าทำไมคอนเทนเนอร์ทำงานผิดพลาด เพื่อความสะดวกแก่ผู้ใช้การอ้างอิงชื่อของคอนเทนเนอร์ในด็อกเกอร์จะต้องใช้ข้อมูลเพียงอย่างใดอย่างหนึ่ง คือ รหัสของคอนเทนเนอร์ หรือชื่อของคอนเทนเนอร์ จากตัวอย่างคอนเทนเนอร์ชื่อ frosty_tereshkova เราสามารถใช้คำสั่ง ดังนี้

```
# docker logs frosty_tereshkova
```

ในกรณีที่คอนเทนเนอร์ยังคงทำงานอยู่ และคอนโซลอาจแสดงผลข้อความที่ต้องการตรวจสอบเพิ่มเติม เราสามารถเพิ่มพารามิเตอร์ -f เพื่อให้คำสั่งยังไม่กลับมาสู่เชลล์แต่ยังรอผลลัพธ์ของคอนโซล เช่น

```
# docker logs -f frosty_tereshkova
```

หรือหากคอนเทนเนอร์ถูกต้องการตรวจสอบทำงานได้ถูกเปิดการทำงานมานานแล้ว เราสามารถตรวจสอบผลลัพธ์ของคอนโซลตามช่วงระยะเวลาได้ด้วย 2 พารามิเตอร์ คือ

1) พารามิเตอร์ --since ใช้ระบุช่วงเวลาที่ต้องการแสดงรายการหลังจากเวลานี้ โดยอาจระบุว่า 40m หมายถึง รายการที่แสดงต้องเกิดขึ้นภายใน 40 นาที หรือ 1h หมายถึง รายการที่แสดงต้องเกิดขึ้นภายใน 1 ชั่วโมงที่แล้ว หรือระบุวันที่ในรูปแบบ YYYY-MM-DDTHH:mm:ss เช่น 2020-02-02T15:00:00 หมายถึงรายการที่แสดงต้องเกิดขึ้นหลังวันที่ 2 กุมภาพันธ์ พ.ศ. 2563 เวลา 15.00น.
2) พารามิเตอร์ –until ใช้ระบุช่วงเวลาที่ต้องการแสดงรายการก่อนเวลานี้ มีรูปแบบการระบุเวลาเหมือนพารามิเตอร์ --since แต่ใช้ในความหมายตรงข้าม เช่น 40m หมายถึง รายการที่แสดงต้องเกิดขึ้นนานกว่า 40 นาที หากต้องการแสดงข้อมูลถึงปัจจุบันให้ระบุ 0m หมายถึง รายการที่แสดงต้องเกิดขึ้นต้องนานกว่า 0 นาที หรือระบุวันที่ในรูปแบบ YYYY-MM-DDTHH:mm:ss เช่น 2020-02-02T15:00:00 หมายถึงรายการที่แสดงต้องเกิดขึ้นก่อนวันที่ 2 กุมภาพันธ์ พ.ศ. 2563 เวลา 15.00น.

พารามิเตอร์ --since และพารามิเตอร์ --until เป็นเพียงพารามิเตอร์เสริม สามารถไม่ระบุ ระบุเพียงหนึ่งพารามิเตอร์ หรือระบุทั้งสองพารามิเตอร์ร่วมกัน เช่น

```
# docker logs --since 1h frosty_tereshkova
```

เมื่อพิมพ์คำสั่ง ด็อกเกอร์โฮสต์จะแสดงข้อความที่แสดงในคอนโซลของคอนเทนเนอร์ frosty_tereshkova ภายใน 1 ชั่วโมง

4.3.3 การเข้าถึงและการออกคอนโซลของคอนเทนเนอร์

หลังจากที่ตรวจสอบการทำงานของคอนเทนเนอร์แล้ว ในบางครั้ง เราอาจต้องการเข้าไปในคอนเทนเนอร์เพื่อพิมพ์คำสั่ง หรือแก้ไขการตั้งค่าบางประการ ด็อกเกอร์โฮสต์มีวิธีการที่ทำให้สามารถเข้าถึงคอนโซลของคอนเทนเนอร์ได้ 2 วิธี คือ

1) เข้าถึงคอนโซลของคอนเทนเนอร์โดยตรง วิธีการนี้นิยมใช้กับคอนเทนเนอร์ที่ทำงานอยู่เบื้องหลังและมีเชลล์ให้เราเข้าถึงได้ หรือเมื่อตอนสร้างคอนเทนเนอร์ได้มีการระบุเชลล์ลงไป เช่น การสั่งให้คอนเทนเนอร์ของระบบปฏิบัติการ CentOS 7 เวอร์ชันก่อนหน้าทำงาน โดยใช้คำสั่ง

```
# docker run -dit centos:7 bash
```

เมื่อพิมพ์คำสั่ง ด็อกเกอร์โฮสต์จะสร้างคอนเทนเนอร์ของระบบปฏิบัติการ CentOS เวอร์ชัน 7 ที่มีเชลล์ bash ให้เราพิมพ์คำสั่งได้ ในกรณีนี้ หากคอนเทนเนอร์มีชื่อว่า frosty_tereshkova เราสามารถเข้าถึงคอนโซลของคอนเทนเนอร์นี้ได้ด้วยคำสั่ง

```
# docker attach frosty_tereshkova
```

เมื่อพิมพ์คำสั่งแล้ว จะปรากฏเชลล์ให้เราสามารถเข้าถึงภายในคอนเทนเนอร์ได้
วิธีการออกคอนโซลของคอนเทนเนอร์จากวิธีการนี้จะต้องออกด้วยการกดปุ่มคีย์ลัด CTRL + P + Q เท่านั้น หากกด CTRL + C จะเป็นการส่งสัญญา SIGHUP ไปให้คอนเทนเนอร์นี้ปิดตัวลง

2) เข้าถึงคอนโซลของคอนเทนเนอร์ด้วยการสร้างเชลล์ใหม่ วิธีการนี้นิยมใช้กับคอนเทนเนอร์ที่มีคำสั่งเริ่มต้นการทำงานเฉพาะ และไม่ได้สร้างเชลล์ไว้ให้เราเข้าถึงได้ เมื่อพิมพ์คำสั่ง docker attach กลับขึ้นผลลัพธ์ของการทำงานของซอฟต์แวร์ภายใน ไม่มีเชลล์ให้เราสามารถพิมพ์คำสั่งได้ วิธีการนี้เราจะสั่งให้เชลล์ทำงานเพิ่มอีกโพรเซสหนึ่งในคอนเทนเนอร์นั้น ในกรณีนี้ หากคอนเทนเนอร์มีชื่อว่า frosty_tereshkova เราสามารถสร้างเชลล์เพื่อเข้าถึงคอนโซลของคอนเทนเนอร์นี้ได้ด้วยคำสั่ง

```
# docker exec -it frosty_tereshkova bash
```

เมื่อพิมพ์คำสั่งแล้ว ด็อกเกอร์โฮสต์จะสร้างโพรเซสของ Bash shell และเราให้เข้าถึงเชลล์นั้น Bash shell เป็นเชลล์ที่สนับสนุนคำสั่งในการโปรแกรมสมบูรณ์ที่สุด แต่ไม่ใช่ทุกอิมเมจจะมี Bash shell ในกรณีที่ผู้พัฒนาอิมเมจไม่ได้ติดตั้ง Bash shell มาด้วย เราสามารถใช้อีกทางเลือกหนึ่ง คือ

```
# docker exec -it frosty_tereshkova sh
```

เมื่อพิมพ์คำสั่งแล้ว ด็อกเกอร์โฮสต์จะสร้างโพรเซสของ Bourne shell แทน

วิธีการออกคอนโซลของคอนเทนเนอร์จากวิธีการนี้จะต้องออกด้วยการพิมพ์ exit หรือกด CTRL + C เพื่อปิดการทำงานของโพรเซสเชลล์ที่เราสร้างขึ้นมาใหม่ มิเช่นนั้น คอนเทนเนอร์จะมีโพรเซสเชลล์ที่สร้างขึ้นใหม่ทำงานค้างไว้อยู่เพิ่มทุกครั้งที่เราเข้าถึงคอนโซลด้วยวิธีการนี้

การเข้าถึงคอนโซลจะมีประโยชน์มากในกรณีที่เราต้องการตรวจสอบแฟ้มการตั้งค่า หรือคัดลอกแฟ้มข้อมูลจากคอนเทนเนอร์ออกมาทั้งนี้ หากแฟ้มข้อมูลนั้นไม่ได้เชื่อมต่อกับไดเรกทอรีของด็อกเกอร์โฮสต์ การเข้าถึงแฟ้มข้อมูลจากภายนอกคอนเทนเนอร์จึงเป็นเรื่องยาก เมื่อเข้าถึงได้แล้ววิธีการที่สะดวกที่สุดคือคัดลอกแฟ้มข้อมูลนั้นไปยังไดเรกทอรีที่เชื่อมต่อกับด็อกเกอร์โฮสต์

4.3.4 การตรวจสอบประวัติการทำงานของคอนเทนเนอร์

เมื่ออิมเมจถูกใช้ในการสร้างคอนเทนเนอร์ ในบางกรณีคอนเทนเนอร์อาจทำงานผิดปกติ เช่น ออกการทำงานเอง หรือรีสตาร์ทคอนเทนเนอร์เอง หรืออิมเมจที่ใช้มีการอัพเดทแท็กใหม่ การตรวจสอบรายการของคอนเทนเนอร์และอิมเมจด้วยคำสั่ง docker image ls และ docker ps จะแสดงเฉพาะสถานะสุดท้ายของอิมเมจหรือคอนเทนเนอร์นั้น อย่างไรก็ตาม โปรแกรมด็อกเกอร์ลูกข่ายมีคำสั่งที่สามารถตรวจสอบเหตุการณ์ที่เกิดขึ้น พร้อมทั้งแสดงวันที่ และเวลาที่เกิดเหตุการณ์นั้น ในกรณีนี้ หากเราสร้างคอนเทนเนอร์มีชื่อว่า frosty_tereshkova จากอิมเมจ httpd เราสามารถตรวจสอบเหตุการณ์ด้วยคำสั่ง docker events แต่หากพิมพ์คำสั่งนี้โดยที่ไม่มีพารามิเตอร์ใด ๆ แล้วจะไม่ปรากฏผลลัพธ์ใด ๆ แต่จะเป็นการรอการแสดงผล และจะแสดงรายการของเหตุการณ์ก็ต่อเมื่อเกิดเหตุการณ์ขึ้นหลังจากที่พิมพ์คำสั่งแล้วเท่านั้น ซึ่งโดยปกติเรานิยมตรวจสอบเหตุการณ์ย้อนหลังมากกว่าตรวจสอบเหตุการณ์ที่กำลังจะเกิด ดังนั้นคำสั่งนี้ ต้องมีการระบุพารามิเตอร์ ดังนี้

1) พารามิเตอร์ --filter ใช้ระบุเงื่อนไขการแสดงผล โดยทั่วไปนิยมระบุเงื่อนไขตามอิมเมจหรือคอนเทนเนอร์ หากต้องการระบุเงื่อนไขการแสดงผลตามอิมเมจ โครงสร้างของพารามิเตอร์นี้จะเป็น --filter image=<ชื่ออิมเมจ> เช่น --filter image=httpd แต่หากต้องการระบุเงื่อนไขการแสดงผลตามคอนเทนเนอร์ โครงสร้างของพารามิเตอร์นี้จะเป็น --filter container=<รหัสคอนเทนเนอร์> หรือ –filter container=<ชื่อคอนเทนเนอร์> ในกรณีที่เป็นรหัสคอนเทนเนอร์เราสามารถพิมพ์แค่บางส่วนที่ไม่ซ้ำกับรหัสคอนเทนเนอร์ของคอนเทนเนอร์อื่นได้ เช่น หากคอนเทนเนอร์มีรหัสคอนเทนเนอร์เป็น 1bba17d1cb3c และมีชื่อว่า frosty_tereshkova ตัวอย่างการใช้พารามิเตอร์นี้จะเป็น --filter container=5 หรือ --filter container=5d หรือ --filter container=1bba17d1cb3c หรือ --filter container=frosty_tereshkova อย่างใดอย่างหนึ่ง
2) พารามิเตอร์ --since ใช้ระบุช่วงเวลาที่ต้องการแสดงรายการหลังจากเวลานี้ โดยอาจระบุว่า 40m หมายถึง รายการที่แสดงต้องเกิดขึ้นภายใน 40 นาที หรือ 1h หมายถึง รายการที่แสดงต้องเกิดขึ้นภายใน 1 ชั่วโมงที่แล้ว หรือระบุวันที่ในรูปแบบ YYYY-MM-DDTHH:mm:ss เช่น 2020-02-02T15:00:00 หมายถึงรายการที่แสดงต้องเกิดขึ้นหลังวันที่ 2 กุมภาพันธ์ พ.ศ. 2563 เวลา 15.00 น.

3) พารามิเตอร์ –until ใช้ระบุช่วงเวลาที่ต้องการแสดงรายการก่อนเวลานี้ มีรูปแบบการระบุเวลาเหมือนพารามิเตอร์ --since แต่ใช้ในความหมายตรงข้าม เช่น 40m หมายถึง รายการที่แสดงต้องเกิดขึ้นนานกว่า 40 นาที หากต้องการแสดงข้อมูลถึงปัจจุบันให้ระบุ 0m หมายถึง รายการที่แสดงต้องเกิดขึ้นต้องนานกว่า 0 นาที หรือระบุวันที่ในรูปแบบ YYYY-MM-DDTHH:mm:ss เช่น 2020-02-02T15:00:00 หมายถึงรายการที่แสดงต้องเกิดขึ้นก่อนวันที่ 2 กุมภาพันธ์ พ.ศ. 2563 เวลา 15.00น.

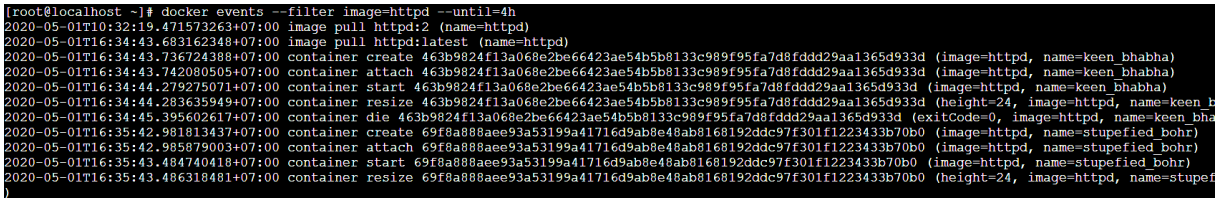
พารามิเตอร์ --since และพารามิเตอร์ --until เป็นเพียงพารามิเตอร์เสริม สามารถไม่ระบุ ระบุเพียงหนึ่งพารามิเตอร์ หรือระบุทั้งสองพารามิเตอร์ร่วมกัน แต่พารามิเตอร์ --filter เป็นพารามิเตอร์ที่จำเป็นต้องระบุ ยกตัวอย่างเช่น หากต้องการตรวจสอบเหตุการณ์ของคอนเทนเนอร์รหัส 1bba17d1cb3c ภายในช่วงระยะเวลา 1 ชั่วโมง สามารถใช้คำสั่ง

```
# docker events --filter container=1bba17d1cb3c –since=1h --until=0
```

หากต้องการตรวจสอบเหตุการณ์ของอิมเมจชื่อ httpd ที่เกิดขึ้นนานกว่า 4 ชั่วโมง สามารถใช้คำสั่ง

```
# docker events --filter image=httpd --until=4h
```

ภาพที่ 4.12 แสดงบางส่วนของรายการการตรวจสอบเหตุการณ์ของอิมเมจ httpd ที่เกิดขึ้นนานกว่า 4 ชั่วโมง



ภาพที่ 4.12 หน้าจอผลลัพธ์คำสั่งแสดงรายการการตวรจสอบเหตุการณ์ของอิมเมจ

4.3.5 การตรวจสอบทรัพยากรที่คอนเทนเนอร์ใช้

เนื่องจากคอนเทนเนอร์เป็นเสมือนหนึ่งเครื่องคอมพิวเตอร์เสมือนอีกเครื่อง แต่ต่างกันที่ไม่มีไฮเปอร์ไวเซอร์จำลองฮาร์ดแวร์ มีข้อดีคือ การใช้ทรัพยากรของโพรเซสในคอนเทนเนอร์จะมีปริมาณที่เทียบเท่าการสั่งให้โพรเซสทำงานนอกคอนเทนเนอร์จึงทำให้เราสามารถใช้คำสั่งทั่วไปสำหรับตรวจสอบทรัพยากรที่โพรเซสในคอนเทนเนอร์ใช้ได้โดยตรง อย่างไรก็ตาม โปรแกรมด็อกเกอร์มีเครื่องมือที่สามารถสอบถามการใช้ทรัพยากรองค์รวมของทั้งคอนเทนเนอร์นั้นไปยังด็อกเกอร์โฮสต์ โดยทรัพยากรหลักที่สามารถตรวจสอบได้ คือ การใช้งานหน่วยประมวลผลกลาง (CPU time) หน่วยความจำหลักที่ใช้ (RAM used) และพื้นที่เก็บข้อมูลที่คอนเทนเนอร์ใช้ยกเว้นไดเรกทอรีที่เชื่อมต่อกับไดเรกทอรีของด็อกเกอร์โฮสต์

การใช้งานหน่วยประมวลผลกลาง โดยปกติในระบบปฏิบัติการลีนุกซ์เราสามารถใช้คำสั่ง ps เพื่อตรวจสอบการใช้งานหน่วยประมวลผลกลางตามโพรเซสที่ทำงานอยู่ สำหรับการตรวจสอบเฉพาะในคอนเทน

เนอร์ สามารถใช้พารามิเตอร์เป็นรหัสคอนเทนเนอร์หรือชื่อคอนเทนเนอร์อย่างใดอย่างหนึ่ง ในกรณีที่เป็นรหัสคอนเทนเนอร์เราสามารถพิมพ์แค่บางส่วนที่ไม่ซ้ำกับรหัสคอนเทนเนอร์ของคอนเทนเนอร์อื่นได้ เช่น หากคอนเทนเนอร์มีรหัสคอนเทนเนอร์เป็น 1bba17d1cb3c และมีชื่อว่า frosty_tereshkova การตรวจสอบการใช้งานหน่วยประมวลผลกลางสามารถใช้คำสั่ง

```
# docker top frosty_tereshkova
```

โดยผลลัพธ์ที่แสดงจะมีทั้งหมด 8 คอลัมน์ แต่ละคอลัมน์มีความหมายดังนี้

1) UID หมายถึงรหัสผู้ใช้ที่เป็นเจ้าของโพรเซสนั้น เช่น หากเป็น 0 คือบัญชีผู้ใช้ เป็นต้น
2) PID หมายถึง รหัสของโพสเซส
3) PPID หมายถึง รหัสของโพสเซสที่สร้างโพรเซสนี้มาอีกทีหนึ่ง
4) C หมายถึง เปอร์เซ็นของหน่วยประมวลผลกลางที่ใช้ ณ ขณะนั้น
5) STIME หมายถึง เวลาที่เริ่มต้นโพรเซสนี้ เช่น 9:00 หมายถึงเริ่มต้นโพรเซสนี้ตอนเก้านาฬิกา คอลัมน์นี้ไม่ได้ระบุวันที่
6) TTY หมายถึง หมายเลขหน้าจอคอนโซลที่กำลังใช้งานอยู่ หากทำงานอยู่เบื้องหลังจากแสดงเป็น ?
7) TIME หมายถึง เวลาของหน่วยประมวลผลกลางที่ถูกใช้ไป เช่น หากโพสเซสนี้ใช้เปอร์เซ็นต์ของหน่วยประมวลผลกลาง 50% เป็นเวลา 10 นาที คอลัมน์ TIME จะแสดงผลเป็น 50% x 10 นาที = 5 นาที
8) CMD หมายถึง คำสั่งที่สร้างโพสเซส

เราสามารถเลือกแสดงบางคอลัมน์ได้ เช่น หากต้องการแสดงเฉพาะคอลัมน์ UID PIDTIME และ CMD สามารถใช้คำสั่ง

```
# docker top frosty_tereshkova o uid,pid,time,cmd
```

เมื่อพิมพ์คำสั่งจะปรากฏผลลัพธ์ดังภาพที่ 4.13 ว่ามีโพรเซส httpd กำลังทำงานอยู่

```
[root@localhost ~]# docker top frosty_tereshkova o uid,pid,time,cmd
UID                 PID                 TIME                CMD
0                   13091               00:00:00            httpd -DFOREGROUND
1                   13132               00:00:00            httpd -DFOREGROUND
1                   13133               00:00:00            httpd -DFOREGROUND
1                   13134               00:00:00            httpd -DFOREGROUND
```

ภาพที่ 4.13 หน้าจอผลลัพธ์คำสั่งตรวจสอบการใช้งานหน่วยประมวลผลกลาง

สำหรับการตรวจสอบการใช้งานทั้งหน่วยประมวลผลกลางและหน่วยความจำที่ใช้งานอยู่ในปัจจุบัน โดยปกติในระบบปฏิบัติการลีนุกซ์สามารถใช้คำสั่ง free เพื่อตรวจสอบหน่วยความจำที่ใช้และคงเหลือ แต่จะเป็นการตรวจสอบของทั้งเครื่อง ไม่ใช่รายคอนเทนเนอร์ สำหรับโปรแกรมด็อกเกอร์สามารถใช้คำสั่ง

```
# docker stats
```

เมื่อพิมพ์คำสั่งจะปรากฏหน้าจอแสดงข้อมูลสถานะการใช้ทรัพยากรและอัพเดททุก 2 วินาที แต่หากต้องการให้แสดงผลข้อมูล ณ เวลานั้น แล้วกลับมาที่เชลล์สามารถใช้คำสั่ง

```
# docker stats --no-stream
```

จะปรากฏผลลัพธ์ดังภาพที่ 4.14 ที่แสดงข้อมูลต่อไปนี้ รหัสคอนเทนเนอร์ (Container ID) ชื่อคอนเทนเนอร์ (Name) เปอร์เซ็นต์ที่หน่วยประมวลผลกลางใช้ (CPU %) หน่วยความจำหลักที่ใช้และที่จำกัด (Mem usage / limit) เปอร์เซ็นต์ของหน่วยความจำหลักที่ใช้ (Mem %) ปริมาณการเขียนและอ่านดิสก์ (Net I/O) จำนวนบล็อกของดิสก์ที่เขียนและอ่าน (Block I/O) รหัสโพรเซสที่ควบคุมคอนเทนเนอร์นี้อยู่ (PIDs)

```
[root@localhost ~]# docker stats --no-stream
CONTAINER ID        NAME                CPU %   MEM USAGE / LIMIT     MEM %   NET I/O     BLOCK I/O   PIDS
1bba17d1cb3c        frosty_tereshkova   0.00%   26.69MiB / 3.691GiB   0.71%   2.1kB / 0B  0B / 0B     82
```

ภาพที่ 4.14 หน้าจอผลลัพธ์คำสั่งแสดงข้อมูลสถานะการใช้ทรัพยากรของคอนเทนเนอร์

สำหรับการตรวจสอบพื้นที่เก็บข้อมูลที่คอนเทนเนอร์ใช้ยกเว้นไดเรกทอรีที่เชื่อมต่อกับไดเรกทอรีของด็อกเกอร์โฮสต์นั้น สามารถตรวจสอบได้จากชั้นข้อมูลที่คอนเทนเนอร์นั้นใช้งานอยู่ เพราะเมื่อสร้างคอนเทนเนอร์ใหม่ โปแกรมด็อกเกอร์จะสร้างชั้นข้อมูลของคอนเทนเนอร์ที่อ้างอิงอิมเมจและจะเก็บเฉพาะข้อมูลที่แตกต่างไปจากเดิมเท่านั้น โดยทั่วไปชั้นข้อมูลที่สร้างขึ้นจะถูกเก็บไว้เป็นไดเรกทอรีหนึ่งบนเครื่องของด็อกเกอร์โฮสต์ซึ่งไดเรกทอรีถูกตั้งชื่อตามรหัสคอนเทนเนอร์เต็ม การตรวจสอบสามารถใช้พารามิเตอร์เป็นรหัสคอนเทนเนอร์หรือชื่อคอนเทนเนอร์อย่างใดอย่างหนึ่ง ในกรณีที่เป็นรหัสคอนเทนเนอร์เราสามารถพิมพ์แค่บางส่วนที่ไม่ซ้ำกับรหัสคอนเทนเนอร์ของคอนเทนเนอร์อื่นได้ เช่น หากคอนเทนเนอร์มีรหัสคอนเทนเนอร์เป็น 1bba17d1cb3c และมีชื่อว่า frosty_tereshkova การตรวจสอบรายละเอียดของคอนเทนเนอร์สามารถใช้คำสั่ง

```
# docker inspect frosty_tereshkova
```

เมื่อพิมพ์คำสั่ง จะปรากฏข้อมูลภายในของคอนเทนเนอร์ frosty_tereshkova ในรูปแบบ JavaScript Object Notation (JSON) ซึ่งประกอบด้วยข้อมูลที่ยาวมาก เราสามารถตั้งค่ารูปแบบการแสดงผลด้วยพารามิเตอร์ -f โดยพารามิเตอร์นี้จะสามารถสอบถามแค่บางส่วนของข้อมูล JSON ได้ในที่นี่เราต้องการเพียงที่อยู่ที่ของชั้นข้อมูลที่ถูกสร้างใหม่ จึงใช้คำสั่ง

```
# docker inspect frosty_tereshkova -f '{{.GraphDriver.Data.UpperDir}}'
```

เมื่อพิมพ์คำสั่งเสร็จ จะได้ที่อยู่ของไดเรกทอรีที่เก็บข้อมูลไว้เพิ่มเติม โดยเราสามารถเข้าถึงไดเรกทอรีนั้นได้โดยใช้คำสั่ง cd แต่ว่าหากพิมพ์ในคอนโซลที่ไม่สามารถคัดลอกข้อความจะเกิดความลำบากเนื่องจากที่อยู่ไดเรกทอรีค่อนข้างยาว

เราสามารถแก้ไขปัญหาโดยเก็บที่อยู่ไดเรกทอรีไว้ในชื่อตัวแปร จากนั้นใช้คำสั่ง cd กับชื่อตัวแปร ดังนี้

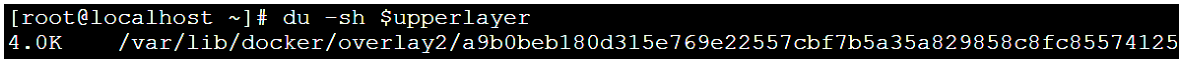
```
# upperlayer=$( docker inspect frosty_tereshkova -f
     '{{.GraphDriver.Data.UpperDir}}' )

# cd $upperlayer
```

เมื่อเข้าถึงไดเรกทอรีดังกล่าวแล้ว เราจะสามารถตรวจทุกแฟ้มข้อมูลและทุกไดเรกทอรีที่เกิดการแก้ไขในคอนเทนเนอร์และเป็นเฉพาะรายการแฟ้มข้อมูลที่ต่างจากที่มีอยู่ในอิมเมจเท่านั้น การตรวจสอบนี้มีประโยชน์เพราะสามารถใช้ค้นหาแฟ้มข้อมูลที่ต้องการได้ และตรวจสอบได้ว่าคอนเทนเนอร์ได้สร้างความเปลี่ยนแปลงข้อมูลอะไรบ้างหลังการทำงาน โดยเราสามารถตรวจสอบพื้นที่ทั้งหมดที่ใช้งานได้โดยใช้คำสั่ง

```
# du -sh $upperlayer
```

เมื่อพิมพ์คำสั่งเสร็จ จะปรากฏผลลัพธ์ดังภาพที่ 4.15 แสดงว่าคอนเทนเนอร์ frosty_tereshkova ใช้พื้นที่ชั้นข้อมูลที่ไม่รวมไดเรกทอรีที่เชื่อมต่อกับด็อกเกอร์โฮสต์จำนวน 4 กิโลไบต์



ภาพที่ 4.15 หน้าจอผลลัพธ์คำสั่งแสดงพื้นที่ที่คอนเทนเนอร์ใช้งาน

บทสรุป

เนื้อหา	คำสั่งหรือชื่อคำสั่งที่เกี่ยวข้อง
ดาวน์โหลดอิมเมจ	# docker pull httpd:2 # docker pull httpd
ตรวจสอบหน้าจอคอนโซลของคอนเทนเนอร์	# docker logs frosty_tereshkova
ข้อความที่แสดงในคอนโซลของคอนเทนเนอร์ frosty_ tereshkova ภายใน 1 ชั่วโมง	# docker logs --since 1h frosty_tereshkova
เข้าถึงคอนโซลของคอนเทนเนอร์	# docker attach frosty_tereshkova
สร้างเชลล์เพื่อเข้าถึงคอนโซลของคอนเทนเนอร์	# docker exec -it frosty_tereshkova bash # docker exec -it frosty_tereshkova sh
ตรวจสอบเหตุการณ์ของคอนเทนเนอร์	# docker events --filter container= frosty_tereshkova --since=1h --until=0 # docker events --filter image=httpd --until=4h
ตรวจสอบการใช้งานหน่วยประมวลผลกลาง	# docker top frosty_tereshkova
ตรวจสอบการใช้งานทั้งหน่วยประมวลผลกลางและหน่วยความจำที่ใช้งานอยู่ในปัจจุบัน	# docker stats
ตรวจสอบรายละเอียดของคอนเทนเนอร์	# docker inspect frosty_tereshkova
ที่อยู่ไดเรกทอรีเก็บข้อมูลของคอนเทนเนอร์	# docker inspect frosty_tereshkova -f '{{.GraphDriver.Data.UpperDir}}'

แบบฝึกหัดที่ 4.3

1. อธิบายความสัมพันธ์ระหว่างอิมเมจและแท็ก

2. บอกคำสั่งที่ใช้แสดงผลคอนโซลของคอนเทนเนอร์ชื่อ mycontainer ภายใน 1 ชั่วโมงล่าสุด

3. อธิบายความแตกต่างระหว่างคำสั่ง docker attach และ docker exec

4. บอกคำสั่งที่ใช้ตรวจสอบเหตุการณ์ทั้งหมดของอิมเมจ httpd

5. อธิบายวิธีการหาชั้นข้อมูลของคอนเทนเนอร์ และพื้นที่ที่คอนเทนเนอร์ใช้งาน

ใบงานฝึกปฏิบัติที่ 4.3

1. เข้าถึงเชลล์ Bash shell ของคอนเทนเนอร์ และตรวจสอบโพรเซสการทำงานภายในคอนเทนเนอร์ด้วยคำสั่ง ps
2. ตรวจสอบโพรเซสการทำงานภายในคอนเทนเนอร์ด้วยคำสั่ง docker top
3. เปรียบเทียบผมลัพธ์ที่ได้จากการพิมพ์คำสั่งข้อที่ 1 และข้อที่ 2
4. คำนวณปริมาณพื้นที่ที่คอนเทนเนอร์นี้ใช้จัดเก็บข้อมูล