คู่มือการติดตั้ง

1. ภาพรวมระบบ

ร้านค้าสำหรับแอพพลิเคชั่นการประมวลผลภาพซึ่งจัดการงานบนระบบ ประมวลผลแบบกลุ่ม พัฒนาด้วย Django, VueJS ในส่วนของ Web Application

ในส่วนของ API สำหรับการประมวลผลแสดงผลภาพ ใช้ Python Script โดยมีการเรียกใช้งานผ่าน FastAPI โดยจะมี API อยู่ 4 ส่วนหลักคือ

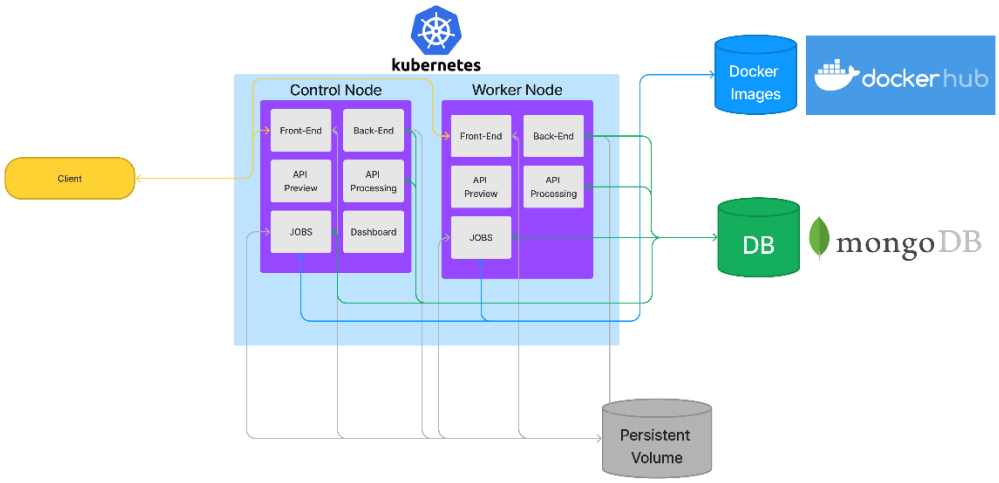
1) Basic API สำหรับแสดงผลตัวอย่างงานประมวลผลด้วย Application พื้นฐานที่ได้พัฒนาไว้

2) Yolo Model API สำหรับใช้งาน Yolo Detection Model ในการตรวจจับวัตถุภาพ

3) GAN Model API สำหรับใช้งาน GAN เพื่อสร้างภาพขึ้นมาใหม่

4) Zip API สำหรับทำการบีบอัดไฟล์ในโฟลเดอร์ให้เป็นไฟล์ Zip สำหรับดาวน์โหลด

โดยโครงสร้างการออกแบบระบบจะมีส่วนของการใช้งาน Web Application เพื่อให้ผู้ใช้ได้ใช้งาน และแบ่งย่อย Microservice ต่าง ๆ ออกเป็น API ดังที่กล่าวมาข้างตน และ มีการใช้งาน Docker Hub สำหรับเก็บ Docker Image ซึ่งใช้ในการ CICD ตัวระบบทั้งหมด, Persistent Volume ในการจัดเก็บข้อมูลไฟล์ภาพต่าง ๆ ของผู้ใช้ และ MongoDB ในการเป็นฐานข้อมูลของระบบ โดยจะมีโครงสร้างของทั้งระบบดังภาพ



รูป 1 ภาพรวมของระบบ

2. ความต้องการของระบบ

กรณีเครื่อง Server ความต้องการของระบบดังนี้

1) เครื่อง Server ต้องติดตั้ง Docker

2) เครื่อง Server สำหรับ Deploy ต้องมีการติดตั้ง Kubernetes

3) เครื่อง Server ต้องมีการติดตั้ง Storage แยกในระบบ Network

4) clone repo github จาก https://github.com/SuteeSaraphan/IPAuTSoNS

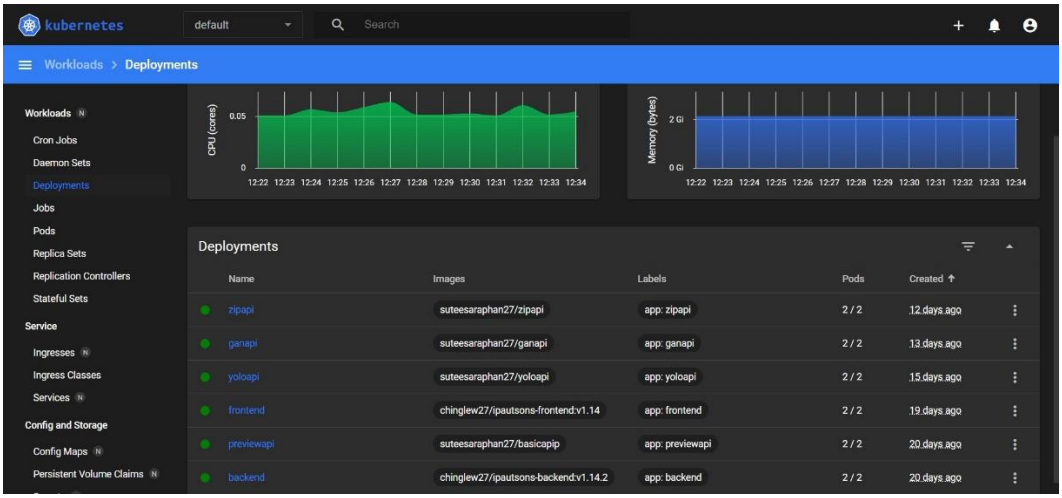
กรณีเครื่อง Local ความต้องการของระบบดังนี้

1) เครื่องต้องติดตั้ง Docker

3. ขั้นตอนการติดตั้งบน Kubernetes ระบบ Cluster

3.1 การติดตั้งส่วนของเว็บแอพพลิเคชั่นบน Kubernetes

1) ในส่วนของ Front-End ให้ใช้ไฟล์ frontend.yaml ที่อยู่ในโฟลเดอร์ IPAuTSoNS/webb/frontend.yaml ในการสั่งงานด้วยคำสั่ง `kubectl apply -f  frontend.yaml`
2) ในส่วนของ Back-End ให้ใช้ไฟล์ backend.yaml ที่อยู่ในโฟลเดอร์ IPAuTSoNS/webb/backend.yaml ในการสั่งงานด้วยคำสั่ง `kubectl apply -f  backend.yaml`
3) เมื่อสั่งงานด้วยคำสั่งดังข้อที่ 1 และ 2 แล้ว สามารถทำการตรวจเช็คจาก Kubernetes ได้ว่าได้ทำการติดตั้งเสร็จสิ้นแล้ว หรือ ไม่ โดยจะแสดงผลดังภาพ



รูป 2 Dashboard Kubernetes เพื่อเช็คว่าทำการ Deployment ในส่วนของ Back-End และ Front-End

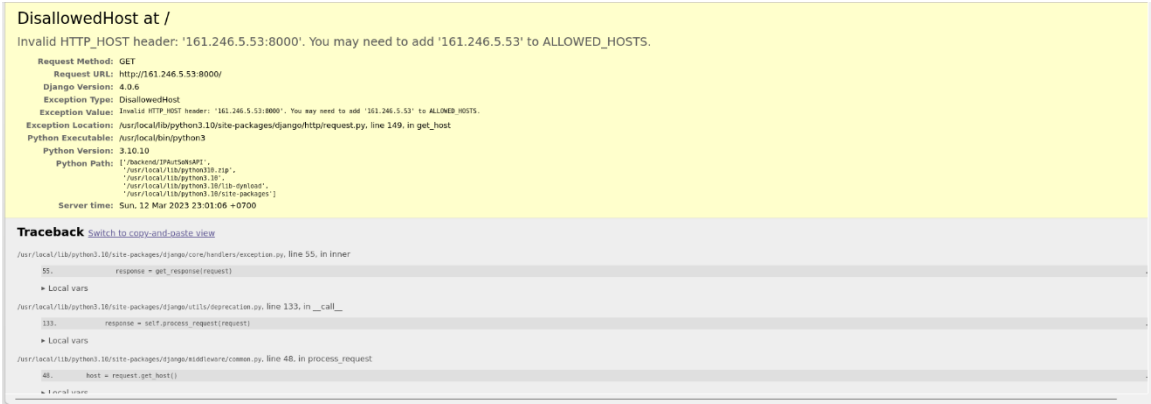
4) ทำการสั่งสร้าง Service เพื่อเข้าใช้งานในส่วนของ Front-End และ Back-End ด้วยคำสั่งในไฟล์ frontend-serive.yaml และ backend-service.yaml ในโฟลเดอร์ IPAuTSoNS/webb/ โดยมีคำสั่งดังนี้

```
kubectl apply -f frontend-service.yaml  backend-service.yaml
```

5) เมื่อทำการสร้าง Service เสร็จสิ้นสามารถเข้าทดสอบ Front-End โดยการเข้าไปที่เว็บไซต์ IP ของเครื่อง Server และ ด้วย Port 80 ดั่งภาพตัวอย่างนี้

รูป 3 หน้าเว็บไซต์ Front-End ใน Port 80

6) เมื่อทำการสร้าง Service เสร็จสิ้นสามารถเข้าทดสอบ Back-End โดยการเข้าไปที่เว็บไซต์ IP ของเครื่อง Server และ ด้วย Port 8000 ดั่งภาพตัวอย่างนี้

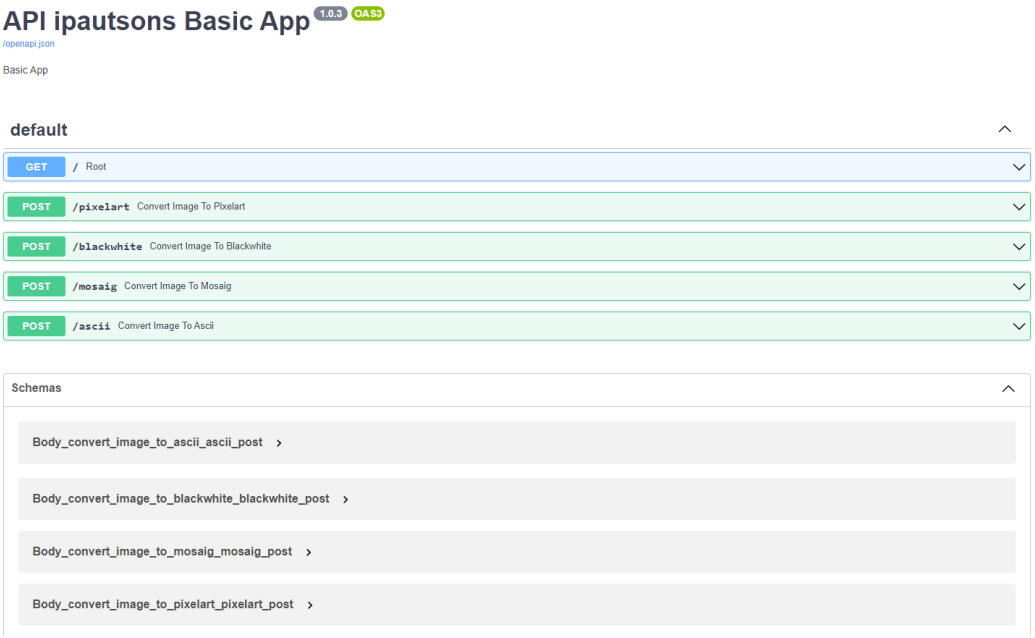


รูป 4 หน้าเว็บไซต์ Back-End ใน Port 8000

3.2 การติดตั้งส่วนของ API แอพพลิเคชั่นบน Kubernetes

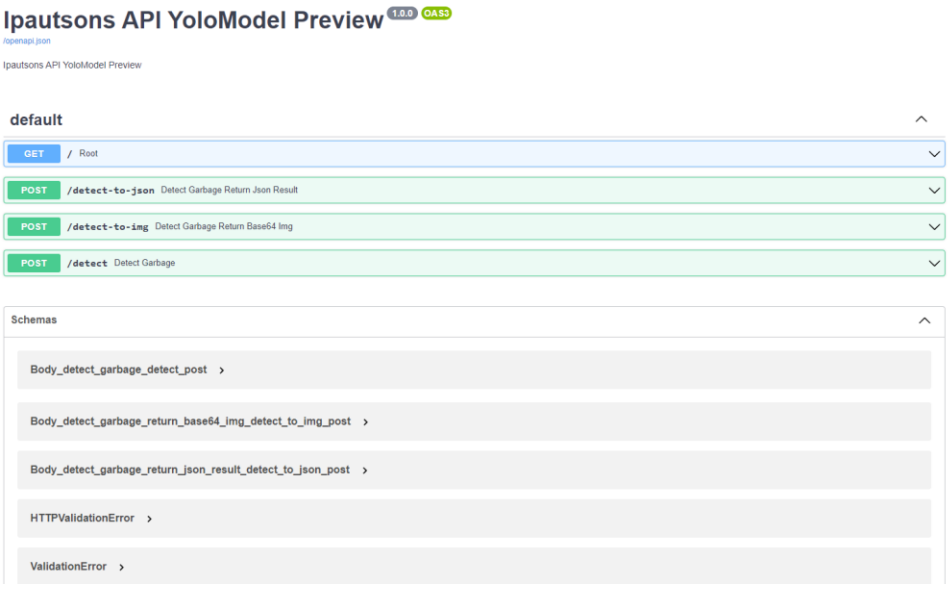
1) ในส่วนของ API การแสดงผลตัวอย่างแอพพลิเคชั่นประมวลผลภาพเบื้องต้น ให้ใช้ไฟล์ basicapi.yaml ที่อยู่ในโฟลเดอร์ IPAuTSoNS/webb/basicapi.yaml ในการสั่งงานด้วยคำสั่ง `kubectl apply -f basicapi.yaml`
2) ในส่วนของ API การแสดงผลตัวอย่างแอพพลิเคชั่นประมวลผลภาพด้วย Model YoloV5 ให้ใช้ไฟล์ yoloapi.yaml ที่อยู่ในโฟลเดอร์ IPAuTSoNS/webb/yoloapi.yaml ในการสั่งงานด้วยคำสั่ง `kubectl apply -f yoloapi.yaml`
3) ในส่วนของ API การแสดงผลตัวอย่างแอพพลิเคชั่นประมวลผลภาพด้วย Model GAN ให้ใช้ไฟล์ ganapi.yaml ที่อยู่ในโฟลเดอร์ IPAuTSoNS/webb/ganapi.yaml ในการสั่งงานด้วยคำสั่ง `kubectl apply -f ganapi.yaml`
4) ในส่วนของ API ในการบีบอัดไฟล์ในรูปแบบ zip ไฟล์ ให้ใช้ไฟล์ zipapi.yaml ที่อยู่ในโฟลเดอร์ IPAuTSoNS/webb/zipapi.yaml ในการสั่งงานด้วยคำสั่ง `kubectl apply -f zipapi.yaml`
5) ทำการสั่งสร้าง Service เพื่อเข้าใช้งานในส่วนของ BasicAPI, YoloAPI, GANAPI และ ZipAPI ด้วยคำสั่งในไฟล์ basicapi-serive.yaml, yoloapi-service.yaml, ganapi-service.yaml และ zipapi-service.yaml ในโฟลเดอร์ IPAuTSoNS/webb/ โดยมีคำสั่งดังนี้ `kubectl apply -f basicapi-serive.yaml, yoloapi-service.yaml, ganapi-service.yaml zipapi-service.yaml`

6) เมื่อทำการสร้าง Service เสร็จสิ้นสามารถเข้าทดสอบ BasicAPIโดยการเข้าไปที่เว็บไซต์ IP ของเครื่อง Server และ ด้วย Port 4020 ดั่งภาพตัวอย่างนี้



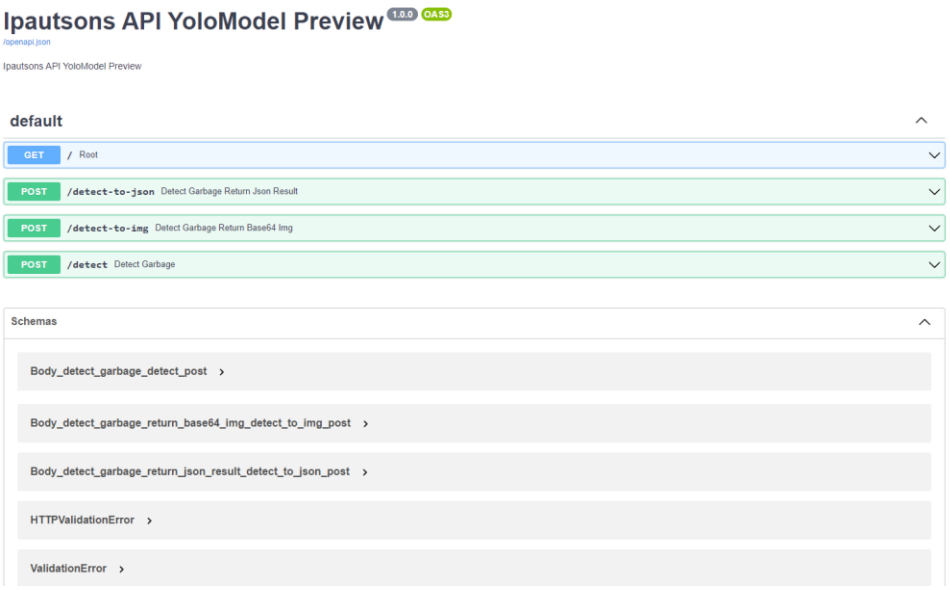
รูป 5 BasicAPI เข้าใช้งานด้วย Port 4020

7) เมื่อทำการสร้าง Service เสร็จสิ้นสามารถเข้าทดสอบ Yolo Model APIโดยการเข้าไปที่เว็บไซต์ IP ของเครื่อง Server และ ด้วย Port 4050 ดั่งภาพตัวอย่างนี้



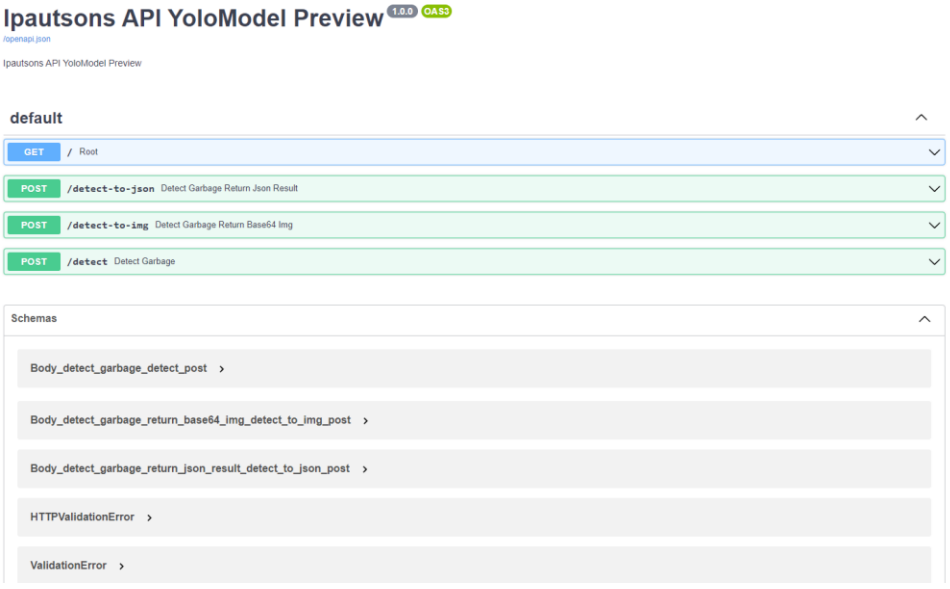
รูป 6 Yolo Model API เข้าใช้งานด้วย Port 4050

8) เมื่อทำการสร้าง Service เสร็จสิ้นสามารถเข้าทดสอบ GANAPI โดยการเข้าไปที่เว็บไซต์ IP ของเครื่อง Server และ ด้วย Port 4070 ดั่งภาพตัวอย่างนี้



รูป 7 GAN API เข้าใช้งานด้วย Port 4070

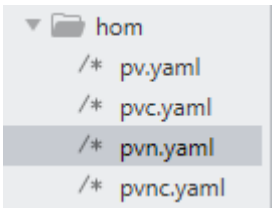
9) เมื่อทำการสร้าง Service เสร็จสิ้นสามารถเข้าทดสอบ ZipAPI โดยการเข้าไปที่เว็บไซต์ IP ของเครื่อง Server และ ด้วย Port 4090 ดั่งภาพตัวอย่างนี้



รูป 8 Zip API เข้าใช้งานด้วย Port 4090

3.3 การติดตั้งส่วนของ Persistent Volume บน Kubernetes

1) ให้ใช้คำไฟล์คำสั่ง Yaml ในโฟลเดอร์ hom โดยรันคำสั่งด้วยคำสั่ง kubctl apply -f pvn.yaml เพื่อทำการกำหนดขนาดของพื้นที่จัดเก็บไฟล์, ความสามารถในการ ReadWrite ของไฟล์ภายใน Volume และ กำหนดที่ตั้งของ Volume นั้น ๆ โดยกำหนดไว้เป็นรูปแบบ NFS ด้วย IP Address ของ Storage ที่จัดเก็บไฟล์ โดยตัวไฟล์คำสั่งจะอยู่ในโฟลเดอร์ดังรูป



รูป 9 โฟลเดอร์ไฟล์คำสั่งในการสร้าง Persistent Volume

2) เมื่อสร้าง Persistent Volume เรียบร้อยแล้วจำเป็นต้องมีการ Claim Volume เพื่อเข้าใช้งานด้วยไฟล์คำสั่งในโฟลเดอร์ hom โดยใช้คำสั่ง kubctl apply -f pvc.yaml เพื่อทำการ Claim ในส่วนของ Persistent Volume ที่ได้สร้างไว้โดยจะมีการกำหนดเรื่อง accessModes และ Storage Requests

4. ขั้นตอนการติดตั้งในระบบ Local

4.1 การติดตั้งส่วนของเว็บแอพพลิเคชั่นด้วย Docker

1) ในส่วนของ Front-End ให้ใช้คำสั่ง `docker pull chinglew27/ipautsons-frontend` เพื่อทำการ pull Docker Image ของ Front-End และ ใช้คำสั่ง `docker run -d –name frontend -p 80:80 chinglew27/ipautsons-frontend` เพื่อทำการเปิดใช้งาน Front-End ที่ Port 80
2) ในส่วนของ Back-End ให้ใช้คำสั่ง `docker pull chinglew27/ipautsons-backend` เพื่อทำการ pull Docker Image ของ Front-End และ ใช้คำสั่ง `docker run -d –name backend -p 8000:8000 chinglew27/ipautsons-backend` เพื่อทำการเปิดใช้งาน Front-End ที่ Port 8000
3) เมื่อสั่งงานด้วยคำสั่งดังข้อที่ 1 และ 2 แล้ว สามารถทำการตรวจเช็คจาก Docker ด้วยคำสั่ง `Docker container ls` ได้ว่าได้ทำการติดตั้งเสร็จสิ้น และ ทำงานอยู่แล้ว หรือ ไม่ โดยจะแสดงผลดังภาพ

```
> docker container ls
CONTAINER ID   IMAGE                                    COMMAND                  CREATED          STATUS          PORTS                    NAMES
745c487daadf   chinglew27/ipautsons-backend:v1.14.2     "python3 manage.py r…"   7 seconds ago    Up 3 seconds    0.0.0.0:8000->8000/tcp   backend
97d5f08ff014   chinglew27/ipautsons-frontend:v1.14      "yarn run serve"         2 minutes ago    Up 2 minutes    0.0.0.0:80->80/tcp       frontend
```

รูป 10 คำสั่งในการตรวจเช็คการทำงานของ Docker Image

4.2 การติดตั้งส่วนของ API ด้วย Docker

1) ในส่วนของ Basic API ในการแสดงผลลัพธ์ตัวอย่างงานประมวลผลภาพที่ได้พัฒนาขึ้นเบื้องต้นให้ใช้คำสั่ง `docker pull suteesaraphan27/basicapip` เพื่อทำการ pull Docker Image ของ Front-End และ ใช้คำสั่ง `docker run -d –name basicapi -p 4020:4020 suteesaraphan27/basicapip` เพื่อทำการเปิดใช้งาน Basic API ที่ Port 4020
2) ในส่วนของ Yolo Model API ในการแสดงผลลัพธ์ตัวอย่างงานประมวลผลภาพด้วย Yolo Model ในการตรวจจับวัตถุในภาพให้ใช้คำสั่ง `docker pull suteesaraphan27/yoloapi` เพื่อทำการ pull Docker Image ของ Front-End และ ใช้คำสั่ง `docker run -d –name yoloapi -p 4050:4050 suteesaraphan27/yoloapi` เพื่อทำการเปิดใช้งาน Yolo Model API ที่ Port 4050
3) ในส่วนของ GAN Model API ในการแสดงผลลัพธ์ตัวอย่างงานประมวลผลภาพด้วย GAN Model ในการนำภาพเดิมมาประมวลผลสร้างขึ้นมาใหม่ให้ใช้คำสั่ง `docker pull suteesaraphan27/ganapi` เพื่อทำการ pull Docker Image ของ GAN Model API และ ใช้คำสั่ง `docker run -d –name ganapi -p 4070:4070 suteesaraphan27/ganapi` เพื่อทำการเปิดใช้งาน GAN Model API ที่ Port 4070
4) ในส่วนของ Zip API ในการบีบอัดโฟลเดอร์ไฟล์จาก Directory ที่จัดเก็บไฟล์ที่ส่งไปให้ API ในการบีบอัดไฟล์ และ ส่งให้ผู้ใช้ดาวน์โหลดให้ใช้คำสั่ง `docker pull suteesaraphan27/zipapi` เพื่อทำการ pull Docker Image ของ Zip API และ ใช้คำสั่ง `docker run -d –name zipapi -p 4090:4090 suteesaraphan27/zipapi` เพื่อทำการเปิดใช้งาน Zip API ที่ Port 4090