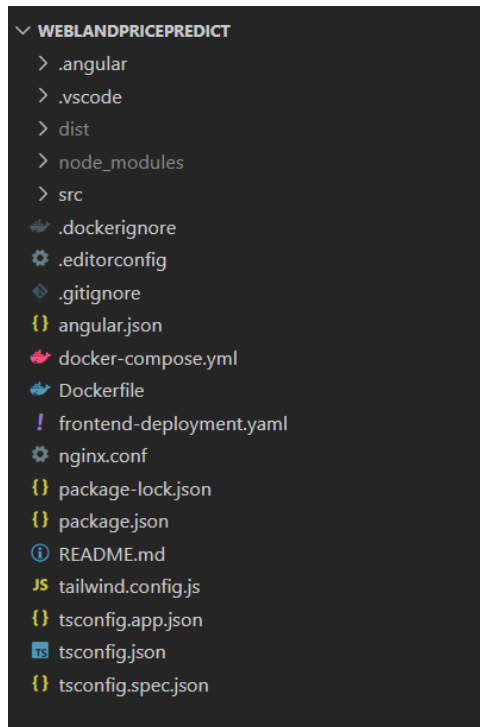


## คู่มือการติดตั้งโปรแกรม

### 1. การติดตั้งในฝั่ง Website LPPM

#### 1.1. ขั้นตอนการติดตั้ง Frontend ของโปรเจค

- 1) เปิดโปรแกรม Visual Studio Code และใช้คำสั่ง git clone  
<https://github.com/LPPACEGroup/WebLandPricePredict.git>
- 2) เข้าสู่ folder ของโปรเจคจากนั้นทำการสร้าง docker image โดย image นั้นจะถูกเก็บบน docker hub



- 3) โดยก่อนที่จะทำการสร้าง docker image นั้นจำเป็นต้องเข้ามาแก้ env ที่เป็น path ของ backend และ API ก่อนโดยจะเข้ามาแก้ไขที่ไฟล์ src\environments\environment.prod.ts

```
export const environment = {  
  production: true,  
  API_URL: 'http://:30600',  
  BE_URL: 'http://:30080/api',  
  LE_URL: 'http://:30500',  
  DB_URL: 'http://:30036',  
  PD_URL: 'http://:30070'  
};
```

4) ให้แก้ไขส่วน path ต่างๆ ตาม ip ของ server ซึ่งในแต่ละ path นั้นเราจะอธิบายในส่วนของการติดตั้งฝั่ง backend

5) รันคำสั่งต่อไปนี้เพื่อสร้าง docker image

```
docker build -t <Your docker hub>:tag (เช่น v0.1) .
```

6) รันคำสั่งนี้เพื่อเก็บ docker image ไว้บน docker hub

```
docker push <Your docker hub>:tag
```

#### 1.1.1. ติดตั้งโดยใช้ Kubernetes

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
  labels:
    app: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
          image: <your docker image> # ใช้ Image บน Docker Hub
          ports:
```

1) ใช้ docker image จาก docker hub พิมพ์ใส่ลงในไฟล์ frontend-deployment.yaml ลงที่ image ส่วนของ containers

2) นำไฟล์ frontend-deployment.yaml คัดลอกลงใน server ที่ใช้ Kubernetes จากนั้นรันคำสั่ง

```
kubectl apply -f frontend-deployment.yaml เพื่อทำการ deploy
```

## 2. การติดตั้งในฝั่ง Backend และ API ต่างๆ

### 2.1. ทำการติดตั้ง Backend ของโปรเจค

- 1) เปิดโปรแกรม Visual Studio Code และใช้คำสั่ง git clone  
`https://github.com/LPPACEGroup/Api1.git`
- 2) ทำการแก้ไข path ของ frontend เพื่อกำหนด allow origin ตาม path ที่คุณได้ทำการ deploy ไปแล้ว

```
app = FastAPI(docs_url=None, redoc_url=None)
app.add_middleware(
    CORSMiddleware,
    allow_origins=[
        "http://:30008", # Add this origin
    ],
    allow_credentials=True,
    allow_methods=["GET", "POST", "PUT", "DELETE"],
    allow_headers=["*"],
)
```

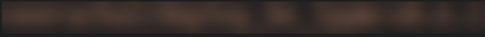
หลังจากที่กำหนด allow origin แล้ว เราจะทำการสร้าง docker image และเก็บไว้บน docker hub โดยใช้คำสั่งนี้ในการสร้าง image `docker build -t <Your docker hub>:tag` (เช่น v0.1) .

- 3) จากนั้นทำการเก็บไว้บน docker hub ด้วยคำสั่งนี้ `docker push <Your docker hub>:tag`

#### 2.1.1. การติดตั้งโดยใช้ Kubernetes

- 1) ก่อนที่จะทำการ deploy นั้นเราจะต้องทำการสร้าง secret บน server ก่อนโดยเราจะมี 3 secret ด้วยกันที่จะต้องสร้างในขั้นตอนนี้ ซึ่งสามารถนำ secret จากโปรเจคที่โฟลเดอร์ Secret ไปใช้ได้เลย
  - email-secret เพื่อใช้ในการส่ง email ถึงผู้ใช้ เช่น การรีเซตรหัสผ่าน
  - app-secret เพื่อใช้ในการตั้งค่าการเข้ารหัสและตรวจสอบความถูกต้องของ JSON Web Tokens (JWT)
  - backend-secret สำหรับการเชื่อมต่อกับฐานข้อมูล
- 2) ทำการคัดลอกไฟล์ secret ทั้ง 3 ตัวลงบน server จากนั้นใช้คำสั่งทั้ง 3 นี้
  - `Kubectl apply -f email-secret.yaml`
  - `Kubectl apply -f app-secret.yaml`
  - `Kubectl apply -f backend-secret.yaml`

- 3) จากนั้นในโฟลเดอร์โปรเจกต์จะมีไฟล์ backend-deployment.yaml ให้ทำการเปลี่ยน docker image ตามที่ได้สร้างไป

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: backend
          image:  # ใช้ Image บน Docker Hub
          ports:
            - containerPort: 8000
```

- 4) จากนั้นคัดลอกไฟล์ backend-deployment.yaml ลงบน server และรันคำสั่งนี้

```
kubectl apply -f backend-deployment.yaml
```

## 2.2. ทำการติดตั้ง API หลักของโปรเจค

- 1) ทำการเปิด Visual Studio Code และใช้คำสั่ง git clone  
<https://github.com/LPPACEGroup/APIServerLPPM.git>
- 2) เข้าไปที่ไฟล์เตอร์ API\_server และทำการแก้ allow origin เพื่อให้ตรงกับ path ของ frontend

```
CORS(app, resources={
    r"/*": {
        "origins": ["http://[redacted]:30008"],
        "methods": ["GET", "POST", "PUT", "DELETE", "OPTIONS"],
        "allow_headers": ["Content-Type", "Authorization"],
        "supports_credentials": True
    }
})
```

- 3) ทำการสร้าง docker image และเก็บไว้บน docker hub ด้วยคำสั่งนี้ตามลำดับ

docker build -t <Your docker hub>:tag (เช่น v0.1) .

docker push <Your docker hub>:tag (เช่น v0.1)

### 2.2.1. การติดตั้งโดยใช้ Kubernetes

- 1) ในไฟล์เตอร์ API\_server นั้นจะมีไฟล์ชื่อ APIserver-deployment.yaml ให้ทำการแก้ไข image ตาม image ที่เก็บไว้บน docker hub ที่ได้ทำการสร้างไป

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: api-server
  template:
    metadata:
      labels:
        app: api-server
    spec:
      containers:
        - name: api-server
          image: 
```

- 2) จากนั้นคัดลอกไฟล์ APIserver-deployment.yaml และรันคำสั่งต่อไปนี้ kubectl apply -f APIserver-deployment.yaml

- 3) หลังจากนั้นในไฟล์เตอร์ API\_server จะมีไฟล์ที่ชื่อ pvc.yaml อยู่ให้ทำการคัดลอกลงบน server จากนั้นรันคำสั่งดังต่อไปนี้

Kubectl apply -f pvc.yaml

- 4) จากนั้นในไฟล์เตอร์ API\_server จะมีไฟล์ที่ชื่อ temp-copy-pod.yaml ให้ทำการคัดลอกลงบน server อีกครั้งจากนั้นทำการรันคำสั่งดังต่อไปนี้ kubectl apply -f temp-copy-pod.yaml

- 5) จากนั้นเราจะทำการเอารูปของที่ดินต่างๆเข้าสู่ pvc เพื่อให้สามารถนำไปแสดงบนระบบได้ ซึ่งรูปที่ดินต่างๆ นั้นถูกเก็บใน NAS เรียบร้อยแล้ว เราจึงจะรันคำสั่งนี้

kubectl cp /nfs/public/lppmlImage/land\_images/050368/. temp-copy-pod:/app/images/land\_images

- 6) ตรวจสอบดูให้แน่ใจว่ารูปถูกเก็บแล้วด้วยคำสั่งนี้ kubectl exec temp-copy-pod -- ls -l /app/images/land\_images

## 2.3. ทำการติดตั้ง API สำหรับการประเมินความน่าสนใจของที่ดิน

- 1) จากขั้นตอนในหัวข้อที่ 2.2 นั้นได้ทำการ git clone  
`https://github.com/LPPACEGroup/APIServerLPPM.git` ให้เข้าไปที่โฟลเดอร์ที่ชื่อ InterestLand
- 2) ทำการแก้ไขไฟล์ชื่อ `api_estimate.py` ในส่วนของ `allow origin` ให้แก้ไขเป็น `path` ของ `frontend` ที่ได้ทำการตั้งไว้

```
CORS(app, resources={
    r"/*": {
        "origins": ["http://[REDACTED]:30008"],
        "methods": ["GET", "POST", "PUT", "DELETE", "OPTIONS"],
        "allow_headers": ["Content-Type", "Authorization"],
        "supports_credentials": True
    }
})
```

- 3) ทำการสร้าง docker image แล้วเก็บไว้บน docker hub ด้วยคำสั่งนี้ตามลำดับ  
  
`docker build -t <Your docker hub>:tag (เช่น v0.1) .`  
  
`docker push <Your docker hub>:tag (เช่น v0.1)`
- 4) จากนั้นเข้าไปแก้ไขไฟล์ที่ชื่อ `LandEstimate_deployment.yaml` เพื่อแก้ไข `image` ให้ตรงกับ `image` ที่ได้ทำการสร้างไปในขั้นตอนก่อนหน้านี้

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: land-estimate
  template:
    metadata:
      labels:
        app: land-estimate
    spec:
      containers:
        - name: land-estimate
          image: [REDACTED]
```

### 2.3.1. การติดตั้งโดยใช้ Kubernetes

- 1) จากนั้นนำไฟล์ `LandEstimate_deployment.yaml` คัดลอกลงบน server

- 2) จากนั้นรันคำสั่งต่อไปนี้ `kubectl apply -f LandEstimate_deployment.yaml`

## 2.4. ทำการติดตั้ง API สำหรับโมเดลการทำนายราคาที่ดินเฉลี่ย

- 1) ทำการเปิด Visual Studio Code และใช้คำสั่ง `git clone https://github.com/LPPACEGroup/DeployML.git`
- 2) เข้าสู่โปรเจกต์และเข้าสู่โฟลเดอร์ชื่อ Deploy จากนั้นจะมีไฟล์ชื่อ `model_service.py` ในไฟล์นี้จะต้องทำการแก้ไข `allow origin` เช่นกัน
- 3) ให้แก้ไข `allow origins` ให้เป็น path เดียวกับ frontend ที่ได้ทำการตั้งเอาไว้

```
CORS(app, resources={
    r"/*": {
        "origins": ["http://[redacted]:30008"],
        "methods": ["GET", "POST", "PUT", "DELETE", "OPTIONS"],
        "allow_headers": ["Content-Type", "Authorization"],
        "supports_credentials": True
    }
})
```

- 4) จากนั้นทำการสร้าง docker image และเก็บไว้บน docker hub ด้วยคำสั่งนี้ตามลำดับ  
`docker build -t <Your docker hub>:tag .`  
`docker push <Your docker hub>:tag`
- 5) จากนั้นในโฟลเดอร์ Deploy มีโฟลเดอร์ Kubernetes อยู่ให้เข้าไปแก้ไข image ที่ไฟล์ `model-deployment.yaml` ให้เป็น image ที่ได้ทำการสร้างขึ้น

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: price-prediction-model
  template:
    metadata:
      labels:
        app: price-prediction-model
    spec:
      containers:
        - name: price-prediction-model
          image: [redacted]
```



6) ก่อนที่จะทำการ deploy API นั้นเราจำเป็นต้อง deploy 3 ไฟล์นี้ก่อนคือ PVC, service เนื่องจากแยกออกมาจากตัว deployment, pod ชั่วคราวสำหรับโอนย้าย model และ scaler ซึ่ง 3 ไฟล์นี้อยู่ในโฟลเดอร์ Kubernetes

- model-pvc.yaml
- model-service.yaml
- temp-pod.yaml

7) ให้ทำการคัดลอกทั้ง 3 ไฟล์นี้ลง server และรัน 3 คำสั่งนี้

```
kubectl apply -f model-pvc.yaml
```

```
kubectl apply -f model-service.yaml
```

```
kubectl apply -f temp-pod.yaml
```

8) จากนั้นรันคำสั่งนี้เพื่อนำ model และ scaler ที่ถูกเก็บไว้บน NAS นำมาเก็บไว้ใน pvc

```
kubectl cp /nfs/public/lppmModel/price_prediction_model_AddN.h5 temp-copy-pod:/models/
```

```
kubectl cp /nfs/public/lppmModel/scaler_XAddN.save temp-copy-pod:/models/
```

```
kubectl cp /nfs/public/lppmModel/scaler_yAddN.save temp-copy-pod:/models/
```

9) ตรวจสอบว่าไฟล์ต่างๆถูกเก็บไปแล้วด้วยคำสั่งนี้

```
kubectl exec temp-copy-pod -- ls -l /models
```

10) จากนั้นทำการลบ pod ชั่วคราวนี้ทั้งด้วยคำสั่งนี้

```
kubectl delete pod temp-copy-pod
```

#### 2.4.1. การติดตั้งโดยใช้ Kubernetes

1) เมื่อเตรียมทุกอย่างเสร็จแล้วให้นำไฟล์ model-deployment.yaml คัดลอกลงบน server

2) รันคำสั่งนี้เป็นอันเสร็จสิ้น

```
kubectl apply -f model-deployment.yaml
```

### 3. การติดตั้งและสร้างฐานข้อมูลของโปรเจค

#### 3.1. การติดตั้งฐานข้อมูล MySQL

3.1.1. จากขั้นตอนที่ 2.1 เราได้ทำการ git clone backend ของโปรเจคมาแล้วจะเห็นว่ามีไฟล์เดอร์ชื่อ MySQL\_DB ในไฟล์เดอร์นี้จะมีไฟล์ทั้งหมด 3 ไฟล์ด้วยกัน ทำการคัดลอกทั้ง 3 ไฟล์ลง server

3.1.2. จากนั้นทำการจองพื้นที่สำหรับฐานข้อมูลโปรเจคนี้ด้วยไฟล์ mysql-storage.yaml ทำการรันคำสั่งนี้

```
kubectl apply -f mysql-storage.yaml
```

3.1.3. จากนั้นทำการสร้างรหัสสำหรับเข้าถึงฐานข้อมูลซึ่งเป็นรหัส root สามารถแก้ไขได้ในไฟล์ mysql-secret.yaml ในส่วน password และรันคำสั่งต่อไปนี้ kubectl apply -f mysql-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
type: kubernetes.io/basic-auth
stringData:
  password: "XXXXXX"
```

3.1.4. จากนั้นรันคำสั่งนี้ kubectl apply -f mysql-deployment.yaml เพื่อ deploy pod ของฐานข้อมูล และ service การเข้าถึงฐานข้อมูลด้วยของ Backend และ API อีกด้วย

3.1.5. เมื่อทำการ deploy pod mysql ไปแล้วจะต้องสร้างผู้ใช้สำหรับ backend โดยให้รันคำสั่งนี้

```
kubectl exec -it <pod-name> -- /bin/bash
```

3.1.6. จากนั้นรันคำสั่งนี้ใน pod mysql -h 192.168.1.7 -u root -p -P 3306 จากนั้นจะต้องใส่รหัสผ่าน 044467 หรือตามที่ได้ deploy ตามข้อ 3.1.3 หลังจากนั้นจะเข้าสู่ฐานข้อมูล mysql แบบ command line

3.1.7. จากนั้นรันคำสั่ง CREATE USER 'backend\_user'@'%' IDENTIFIED BY '024767'; สามารถเปลี่ยนชื่อผู้ใช้และรหัสผ่านตามต้องการได้แต่ต้องแก้ไข secret ในข้อที่ 2.1.1 หัวข้อที่ 2 โดยแก้ไขในส่วน DB\_USER และ DB\_PASSWORD และจะต้องอยู่ในรูปแบบเลขฐาน 64 ด้วย

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-secret
type: Opaque
data:
  DB_USER: [REDACTED]
  DB_PASSWORD: [REDACTED]
  DB_HOST: [REDACTED]
  DB_PORT: [REDACTED]
  DB_NAME: [REDACTED]
```

3.1.8. และให้รันคำสั่งดังต่อไปนี้ตามลำดับ

```
GRANT ALL PRIVILEGES ON LPPM.* TO 'backend_user'@'%';
```

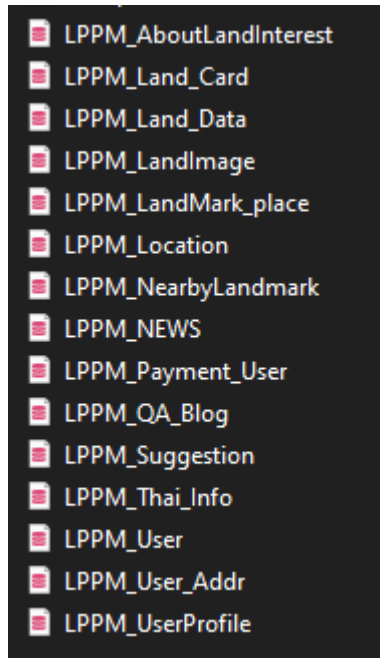
```
GRANT SELECT, INSERT, UPDATE, DELETE ON LPPM.* TO 'backend_user'@'%';
```

```
FLUSH PRIVILEGES;
```

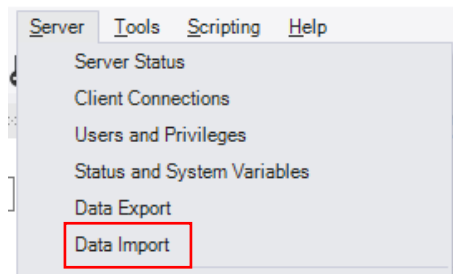
## 3.2. การเตรียมฐานข้อมูลให้พร้อมใช้งาน

### 3.2.1. นำไฟล์ข้อมูลเข้าสู่ฐานข้อมูล

3.2.1.1. ทำการแตกไฟล์ Database Dump ด้วยโปรแกรมแตกไฟล์ .zip จะได้ไฟล์ .sql ดังรูป



3.2.1.2. เข้าสู่โปรแกรม MySQL Workbench แล้วหาเมนู Server จากนั้นเลือก Data Import



3.2.1.3. ในหน้านี้ให้เลือก Import from Self-Contained File แล้วคลิกที่ปุ่ม ...



3.2.1.4. หลังจากนั้นให้เลือกไฟล์ที่อยู่ในข้อ 3.2.1.1 จากนั้นคลิก Open

3.2.1.5. ทำซ้ำจนครบทุกไฟล์