CAR TALLY

Yolo Vehicle Detection Web

240-229 SOFTWARE DEFINED ARCHITECTURE ENGINEER MODULE

โดยกลุ่ม Dtect

การทำงานฝั่งเว็บไซด์

เทคนิคที่ใช้

- Multiusers (Category VI)

NEXT.JS

การทำงานฝั่งเซิร์ฟเวอร์

เทคนิคที่ใช้

- Concurrency (Category I)

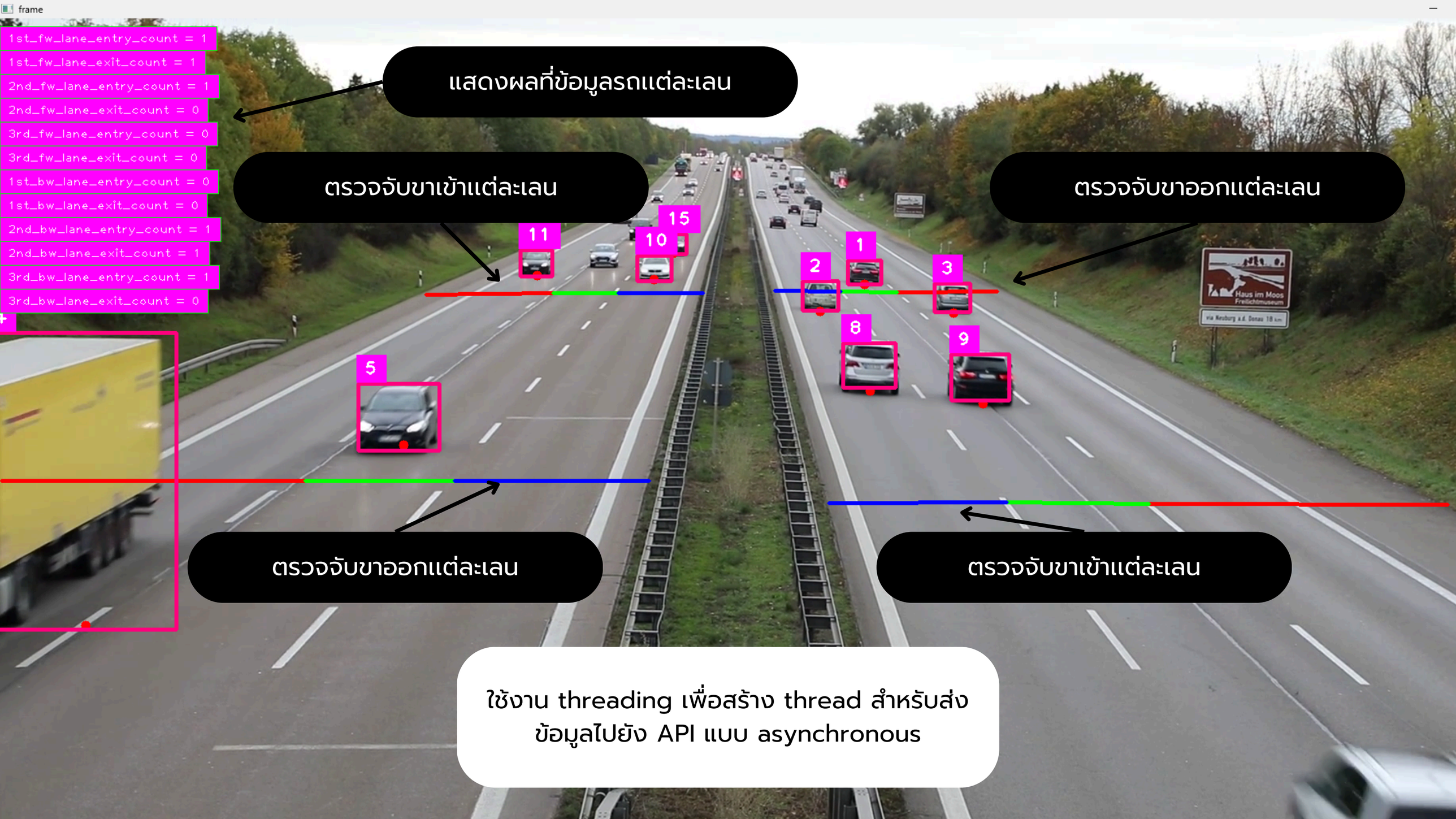
Express

การทำงานฝั่ง AI

เทคนิคที่ใช้

- AI (Other II)
- Multithreading (Category I)

ultralytics
YOLO


frame
1st_fw_lane_entry_count = 1
1st_fw_lane_exit_count = 1
2nd_fw_lane_entry_count = 1
2nd_fw_lane_exit_count = 0
3rd_fw_lane_entry_count = 0
3rd_fw_lane_exit_count = 0
1st_bw_lane_entry_count = 0
1st_bw_lane_exit_count = 0
2nd_bw_lane_entry_count = 1
2nd_bw_lane_exit_count = 1
3rd_bw_lane_entry_count = 1
3rd_bw_lane_exit_count = 0
แสดงผลที่ข้อมูลรถแต่ละเลน
ตรวจจับขาเข้าแต่ละเลน
ตรวจจับขาออกแต่ละเลน
ตรวจจับขาออกแต่ละเลน
ตรวจจับขาเข้าแต่ละเลน
ใช้งาน threading เพื่อสร้าง thread สำหรับส่ง
ข้อมูลไปยัง API แบบ asynchronous

การใช้งาน Cloud เพื่อ Deploy เว็บไซด์

เทคนิคที่ใช้

- VM (Category II)
- Terraform (Category III)
- Loadbalancer (Category IV)
- Snapshot (Category IV)
- Security (Category IV)

ใช้งาน Terraform ในการสร้าง VM

```
resource "google_service_account" "default" {
    account_id   = "dtect-web-sa"
    display_name = "Custom SA for DTECT Web"
}

resource "google_compute_firewall" "allow_http_https_instance" {
    name    = "allow-http-https-instance"
    network = "default"

    allow {
        protocol = "tcp"
        ports    = ["80", "443"]
    }

    source_ranges = ["0.0.0.0/0"]
    target_service_accounts = [google_service_account.default.email]
}

resource "google_compute_firewall" "allow_lb_instance" {
    name    = "allow-loadbalancer-instance"
    network = "default"

    allow {
        protocol = "tcp"
        ports    = ["80", "443"]
    }

    source_ranges = ["130.211.0.0/22", "35.191.0.0/16"]
    target_service_accounts = [google_service_account.default.email]
}
```

```
resource "google_compute_instance" "frontend-yolo-detection" {
    name             = "frontend-yolo-detection"
    zone             = "asia-southeast1-a"
    machine_type     = "n2d-standard-2"
    min_cpu_platform = "AMD Milan"

    confidential_instance_config {
        enable_confidential_compute = true
        confidential_instance_type  = "SEV"
    }

    boot_disk {
        initialize_params {
            image = "ubuntu-os-cloud/ubuntu-2004-lts"
            labels = {
                my_label = "value"
            }
            size = 25
        }
    }

    network_interface {
        network = "default"

        access_config {
        }
    }

    tags = ["web-server"]

    service_account {
        email  = google_service_account.default.email
        scopes = ["cloud-platform"]
    }
}
```

รายละเอียดไฟล์ที่ใช้สร้าง VM ด้วย Terraform

Deploy เว็บไซด์ลงใน VM

- ลงแพ็คเกจต่างๆเบื้องต้นใน VM ผ่านคำสั่ง sudo apt install nginx certbot python3-certbot-nginx และลง npm ,nodejs, pm2 เพื่อใช้สำหรับรัน Client

- ตั้งค่า config ใน /etc/nginx/sites-available

```
server {
    server_name dtect-sda-project.xyz;

    location /socket.io/ {
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Proto $scheme;

        add_header Content-Security-Policy "upgrade-insecure-requests";
        add_header Access-Control-Allow-Origin *;
        add_header Access-Control-Allow-Methods "GET, POST, OPTIONS";
        add_header Access-Control-Allow-Headers "Content-Type, Authorization, X-Requested-With";
    }

    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Host $host;
    }

    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/dtect-sda-project.xyz/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/dtect-sda-project.xyz/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
```

- นำไฟล์โปรเจกต์ไปไว้ใน var/www/client และทำการ build ผ่านคำสั่ง npm run build

- รันโปรเจกต์ด้วยคำสั่ง pm2 start npm --name name_of_app –- start, pm2 save และ pm2 startup

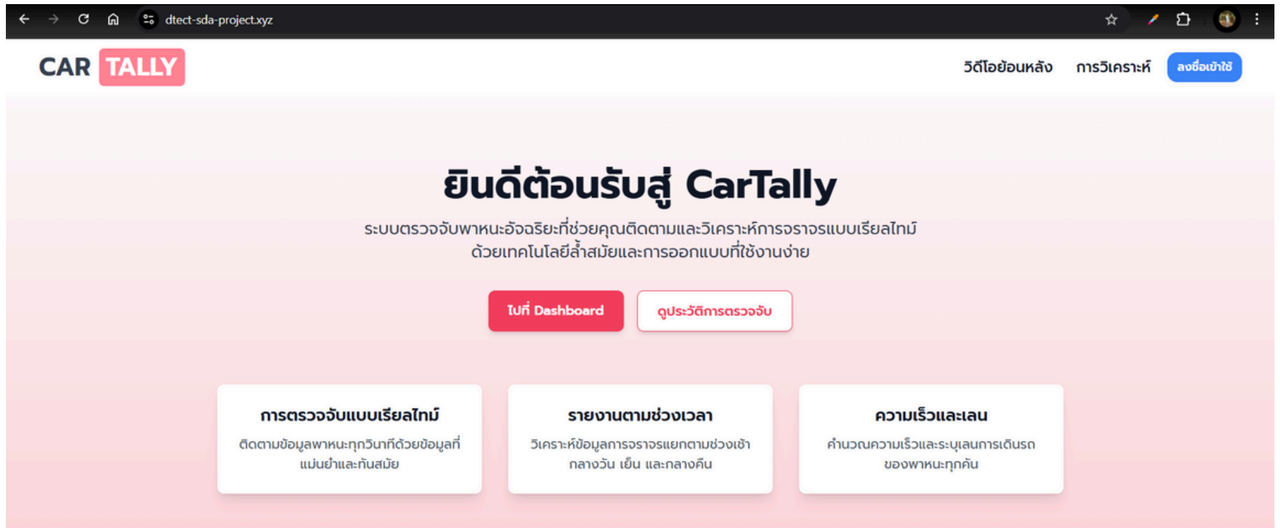
- ลง certbot เพื่อทำให้เว็บเป็น https ด้วยคำสั่ง sudo certbot --nginx -d dtect-sda-project.xyz -d www.dtect-sda-project.xyz

```
Use --update-env to update environment variables
[PM2] Applying action restartProcessId on app [dtect](ids: [ 0 ])
[PM2] [dtect](0) ✓
```

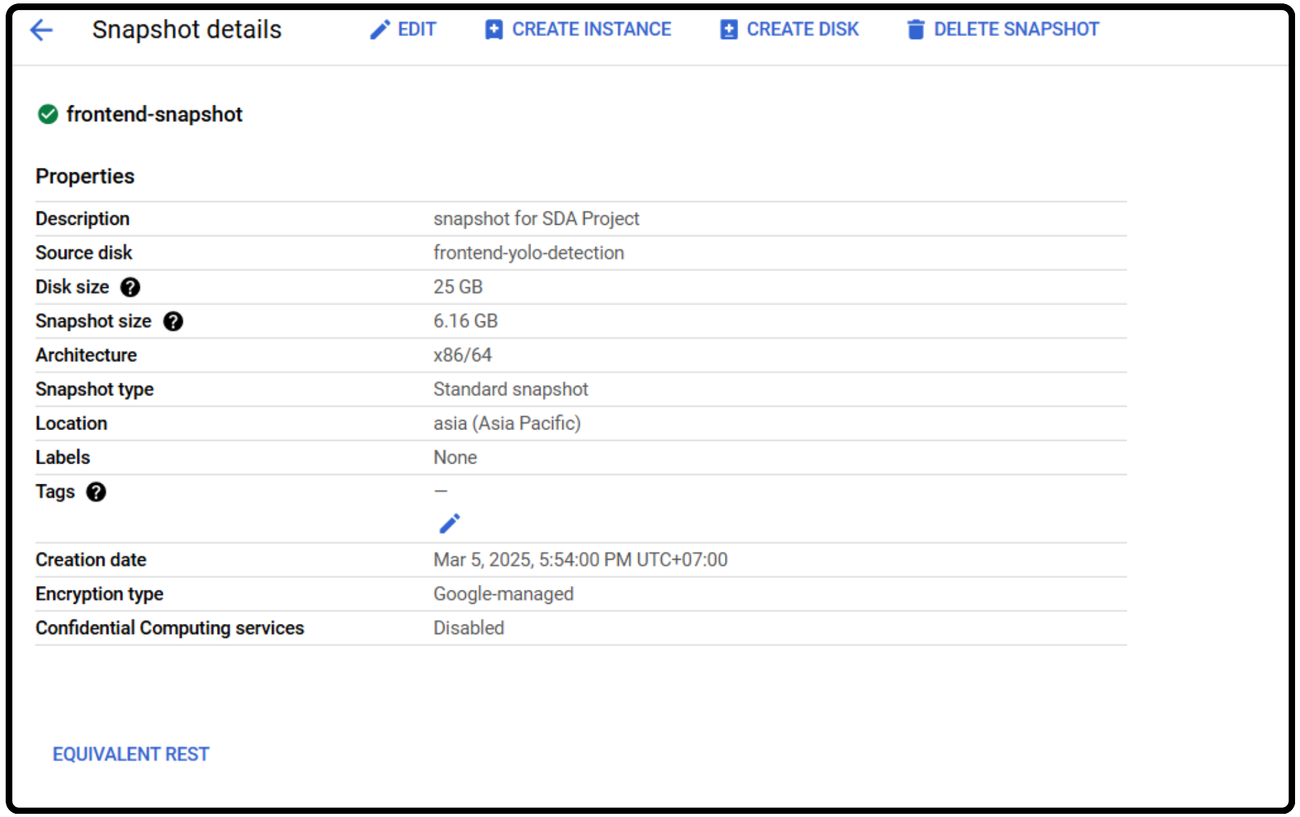
id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	dtect	default	0.39.1	fork	2115	0s	2	online	0%	22.1mb	root	disabled

```
root@frontend-yolo-detection:/var/www/client#
```

- นำไฟล์โปรเจกต์ไปไว้ใน var/www/client และทำการ build ผ่านคำสั่ง npm run build



Snapshot VM หลังจาก Deploy เสร็จสิ้น



เมื่อทำการ Deploy เว็บไซต์เสร็จสิ้นและสามารถใช้งานได้แล้ว ต่อมาจะสร้าง Snapshot ของ VM ซึ่งเป็นขั้นตอนสำคัญเพื่อสำรองข้อมูลและป้องกันความเสียหายที่อาจเกิดขึ้นในอนาคต Snapshot ช่วยให้สามารถกู้คืนระบบกลับมาได้อย่างรวดเร็วหากเกิดข้อผิดพลาด

การใช้งาน Cloud เพื่อ Deploy เซิร์ฟเวอร์

เทคนิคที่ใช้

- Docker (Category II)
- Makefile (Category III)
- K8S (Category IV)
- Backup (Category V)

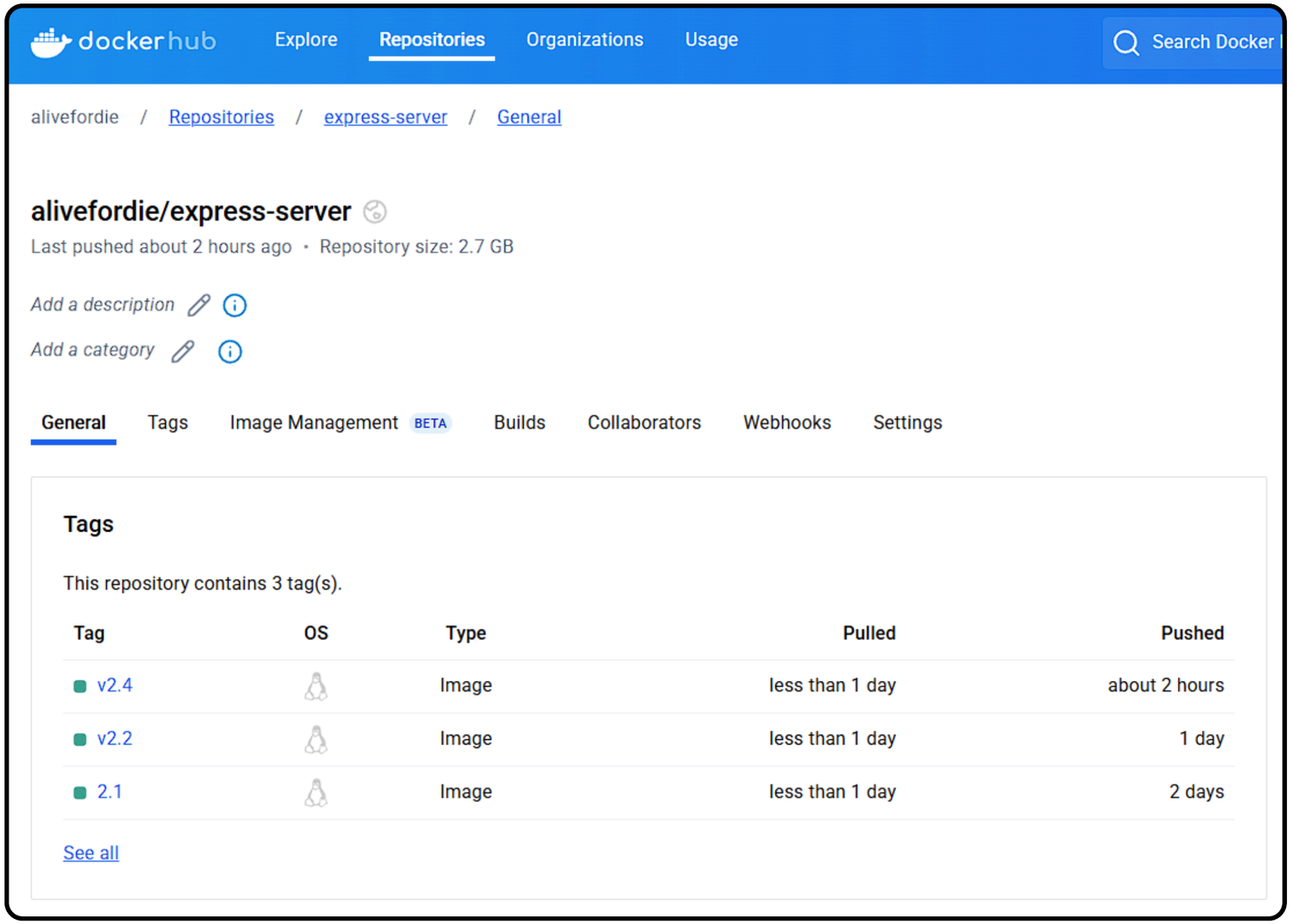
Makefile

```
all: cdl delete apply
crush:
  gcloud container clusters create --machine-type=e2-medium --disk-size 50GB --region=asia-southeast1-a crush-on-k8s

cdl:
  gcloud container clusters get-credentials crush-on-k8s --zone=asia-southeast1-a --project=learn-vm-442206

appsel:
  kubectl apply -f $(sel)

delsel:
  kubectl delete -f $(sel)

apply:
  kubectl apply -f all.yaml

delete:
  kubectl delete -f all.yaml

logs:
  kubectl logs -f -l app=express-app -c express-app
```

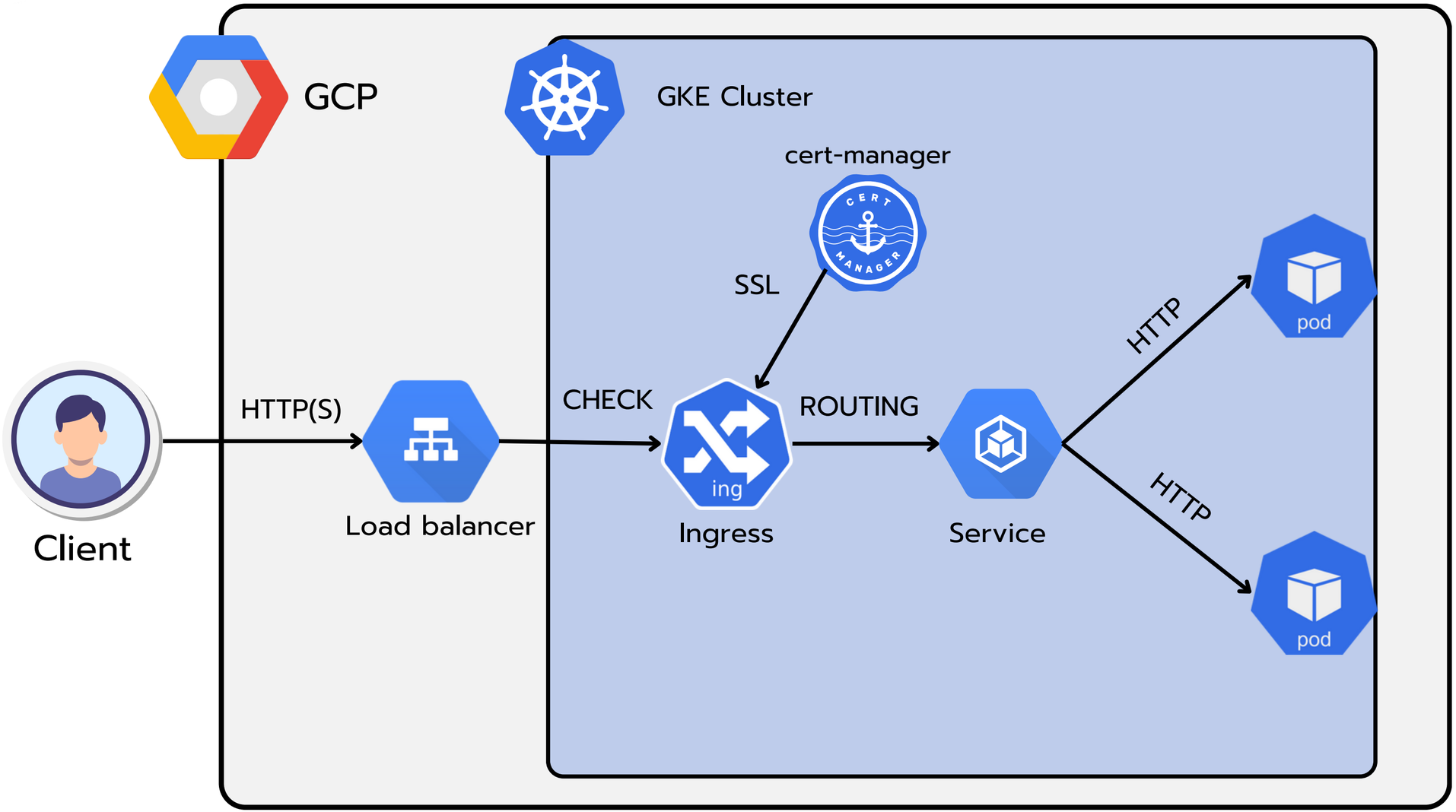
ตัวอย่าง Makefile ที่ใช้สำหรับจัดการ k8s

Docker

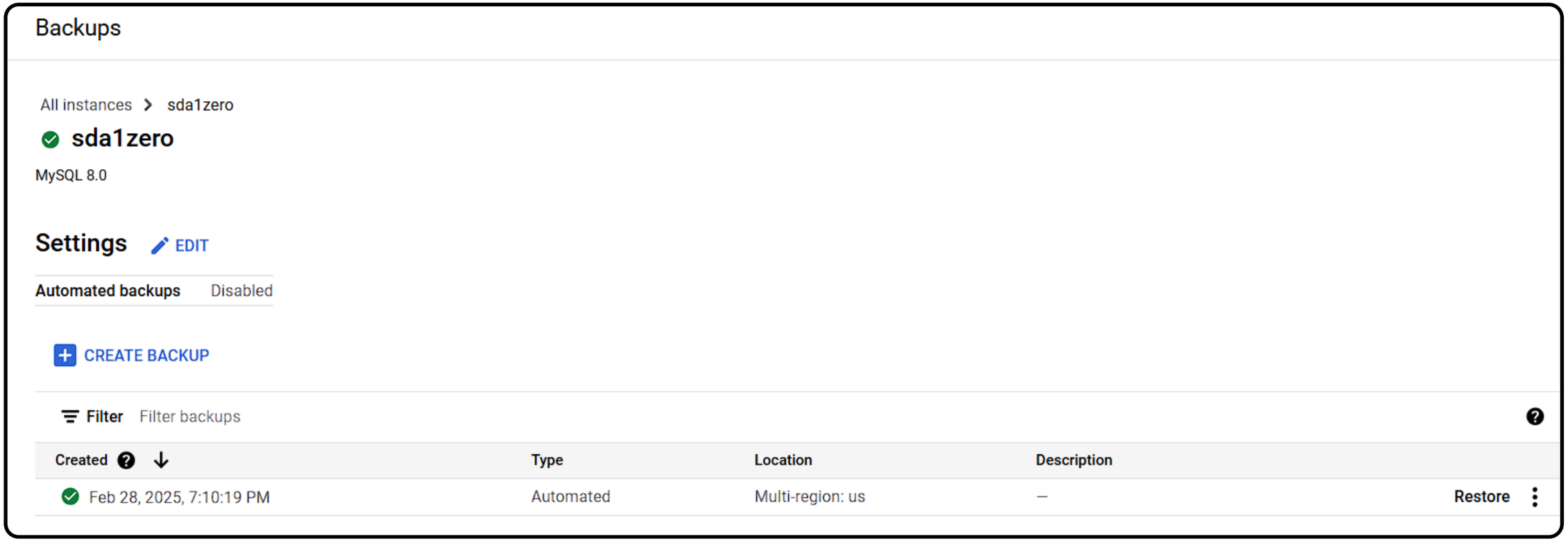
Docker Image สำหรับ
การสร้าง container



แผนผังการใช้งาน K8S ในโปรเจกต์



Backup โดย Cloud SQL



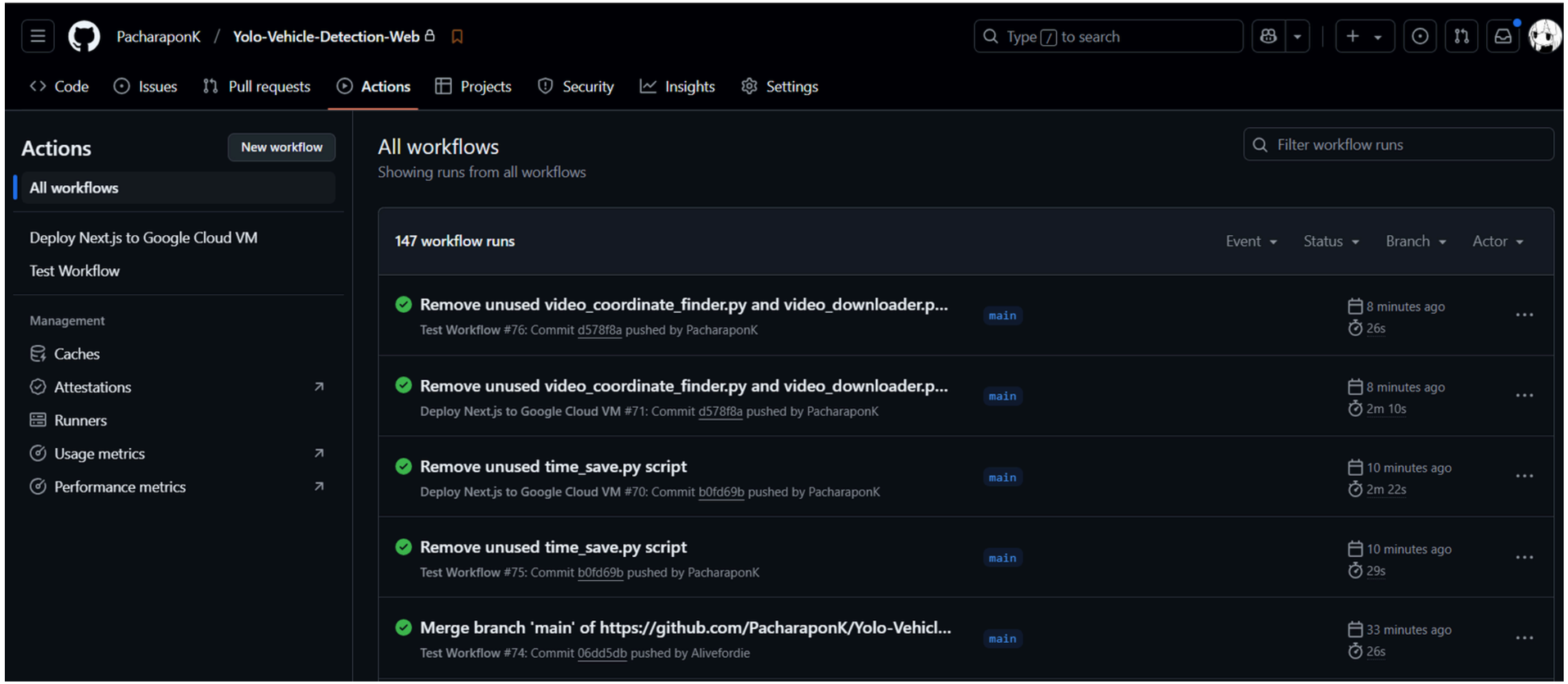
ใช้ Google Cloud SQL (MySQL) ซึ่งสามารถเปิดใช้งาน Automatic Backups และ Point-in-time Recovery เพื่อป้องกันข้อมูลสูญหาย

การใช้งาน Git Actions เพื่อทำ CI/CD

เทคนิคที่ใช้

- Git (Category III)

Workflows ใน Git Actions



GitHub Actions มี 2 Workflows สำหรับ CI/CD:

- Deploy Next.js to Google Cloud VM
- Test Workflow

Deploy Next.js to Google Cloud VM

main Yolo-Vehicle-Detection-Web / .github / workflows / test.yml

PacharaponK fix node version on CI/CD workflow

Code Blame 30 lines (24 loc) · 630 Bytes

```yaml
name: Test Workflow

# เมื่อไหร่ที่ workflow จะทำงาน
on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

# งานที่ต้องการรัน
jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '20'

      - name: Install dependencies
        run: npm install
        working-directory: ./client

      - name: Run tests
        run: npm run lint
        working-directory: ./client
```

Test Workflow

Remove unused video_coordinate_finder.py and video_downloader.py scripts #76 Re-run all jobs

Summary

Jobs

test

Run details

Usage

Workflow file

test
succeeded 3 minutes ago in 21s

Search logs

Set up job	1s
Checkout repository	2s
Setup Node.js	1s
Install dependencies	13s
Run tests	2s
Post Setup Node.js	0s
Post Checkout repository	0s
Complete job	0s

- ใช้ deploy.yml สำหรับ deploy แอป Next.js ไปยัง Google Cloud VM เมื่อ push ไปที่ main
- ใช้ secrets ต่างๆเพื่อเชื่อมต่อ VM ผ่าน SSH
- Checkout โค้ด → ติดตั้ง Node.js & dependencies → run lint & build
- คัดลอกไฟล์ไปยังเซิร์ฟเวอร์ด้วย rsync → ย้าย build ไปที่ /var/www/client → ตั้งค่าความเป็นเจ้าของไฟล์ → รีสตาร์ทแอปผ่าน pm2

Test Workflow

main Yolo-Vehicle-Detection-Web / .github / workflows / deploy.yml

PacharaponK exclude .next directory from rsync during deployment

Code Blame 102 lines (83 loc) · 3.17 KB

```yaml
name: Deploy Next.js to Google Cloud VM

on:
  push:
    branches:
      - main

env:
  VM_IP: ${{ secrets.VM_IP }}
  SSH_USERNAME: ${{ secrets.SSH_USERNAME }}
  SSH_KEY: ${{ secrets.SSH_KEY }}
  PROJECT_DIR: "/home/${{ secrets.SSH_USERNAME }}/my-next-app"
  WORKING_DIR: "client"

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '20'

      - name: Install dependencies
        run: npm install
        working-directory: ${{ env.WORKING_DIR }}

      - name: Run lint
        run: npm run lint
        working-directory: ${{ env.WORKING_DIR }}
```

Deploy Next.js to Google Cloud VM

Remove unused video_coordinate_finder.py and video_downloader.py scripts #71 Re-run all jobs

Summary

Jobs

deploy

Run details

Usage

Workflow file

deploy
succeeded 1 minute ago in 2m 5s

Search logs

Set up job	1s
Checkout repository	3s
Setup Node.js	0s
Install dependencies	14s
Run lint	2s
Build Next.js app	31s
Verify Next.js build output	0s
Setup SSH	0s
Debug SSH Connection	3s
Deploy Next.js build to VM using rsync	25s
Move Next.js build to /var/www/client	3s
Restart Next.js app on VM	39s

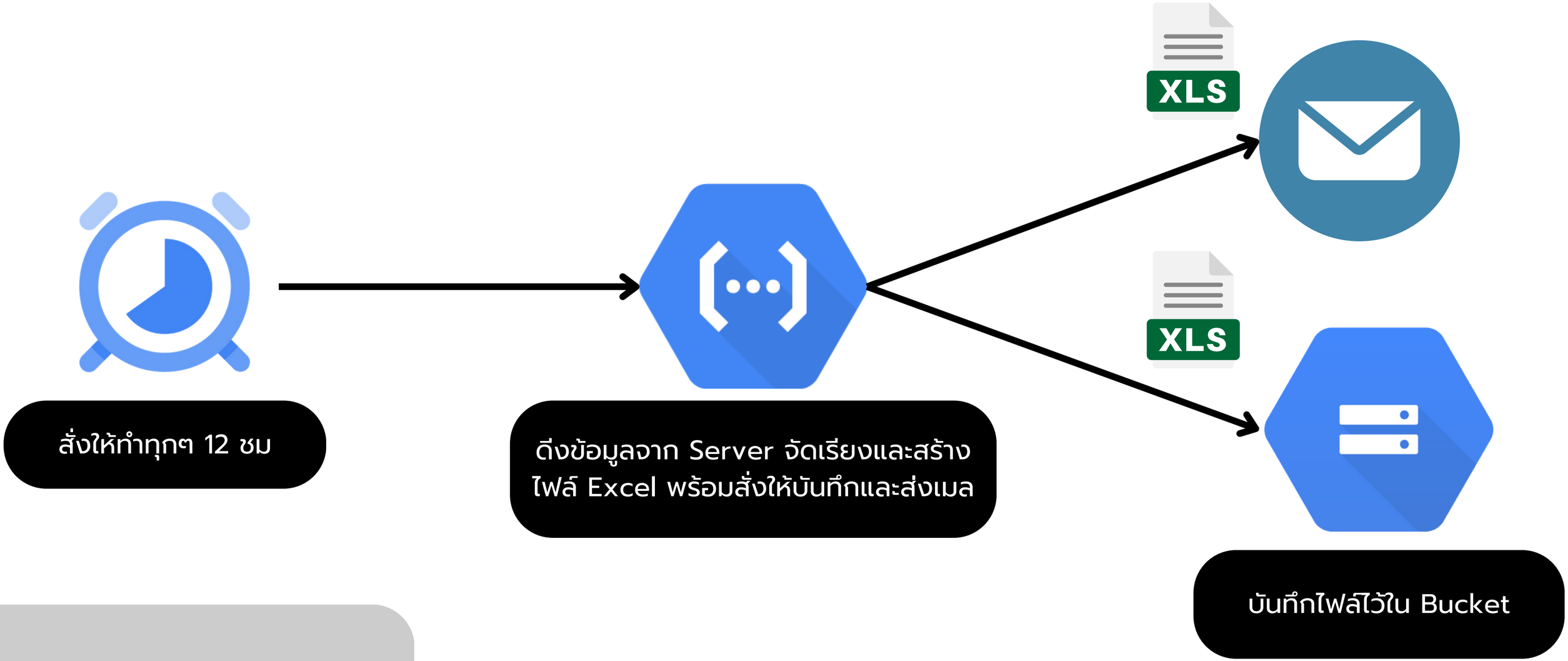
- ใช้ test.yml สำหรับทดสอบโค้ดเมื่อ push หรือ pull request ไปที่ main
- รันบน ubuntu-latest → Checkout โค้ด → ติดตั้ง Node.js 20 & dependencies → รัน npm run lint เพื่อตรวจสอบคุณภาพโค้ด

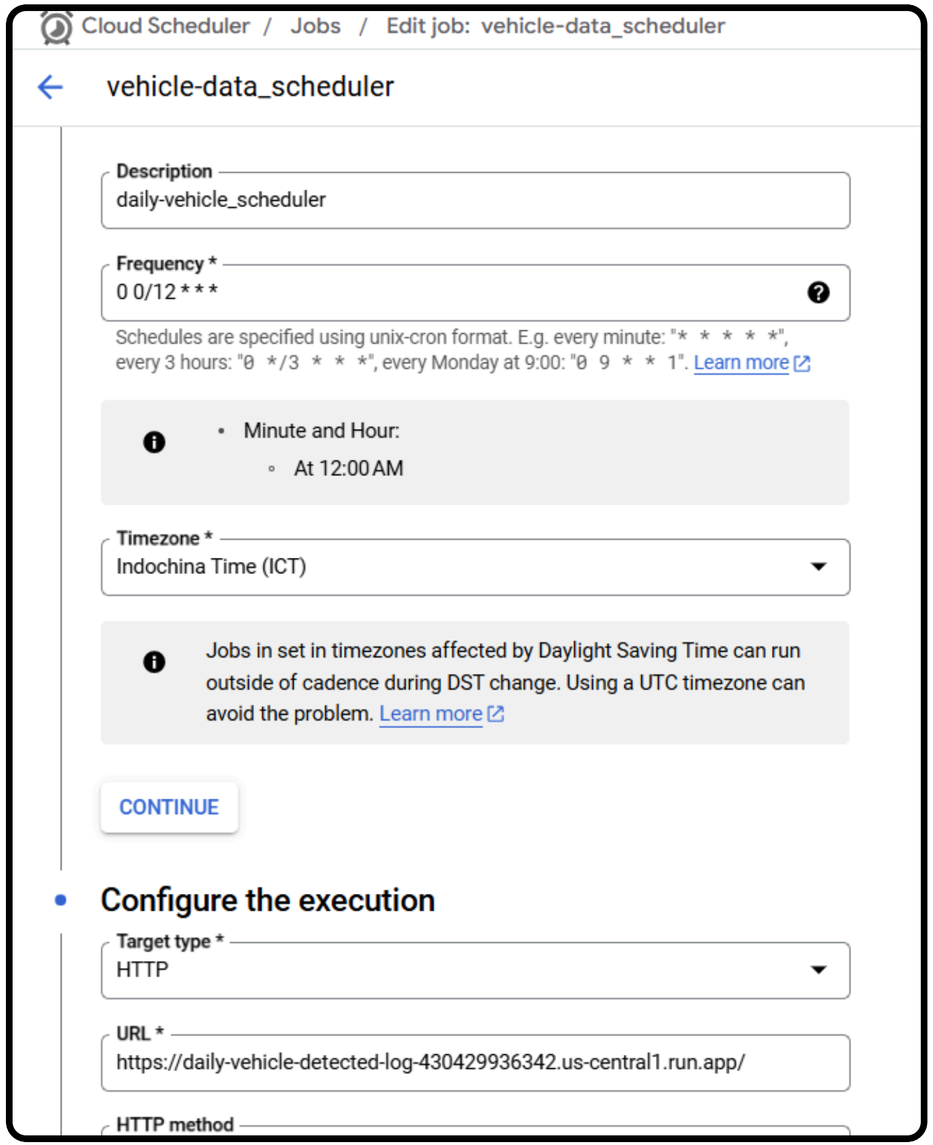
การใช้งาน Cloud สำหรับส่งข้อมูลพาหนะรายวัน

เทคนิคที่ใช้

- Cloud Functions (Category II)
- Scheduler (Category IV)
- Cloud Storage

แผนภาพการใช้งาน Cloud สำหรับส่งข้อมูลพาหนะรายวัน





Cloud Scheduler

ตั้งค่าสำหรับงาน "vehicle-data_scheduler" โดยกำหนดให้รันทุกวันเวลา 12:00 น. (ตามเวลา Indochina Time - ICT) ด้วยความถี่ "0 0/12 * * *" (ทุก 12 ชั่วโมง) และใช้เป้าหมายการรันเป็น HTTP โดยระบุ URL *"https://daily-vehicle-detected-log-430429936342-us-central1.run.app"* พร้อมระบุ method HTTP ที่จะใช้ในการรันงานนี้ ซึ่งเป็นฟังก์ชั่นที่ถูกสร้างด้วย Cloud Functions

Cloud Funtions

Cloud Run | Service details | EDIT & DEPLOY NEW REVISION | SET UP CONTINUOUS DEPLOYMENT | TEST

daily-vehicle-detected-log Region: us-central1 URL: https://daily-vehicle-detected-log-430429936342.us-central1.run.app Scaling: Auto (Min: 0)

METRICS | SLOS | LOGS | REVISIONS | SOURCE | TRIGGERS | NETWORKING | SECURITY | YAML

Source Base image: Python 3.12 (Ubuntu 22) Function entry point: vehicle_data_to_gcs_and_email EDIT SOURCE

main.py

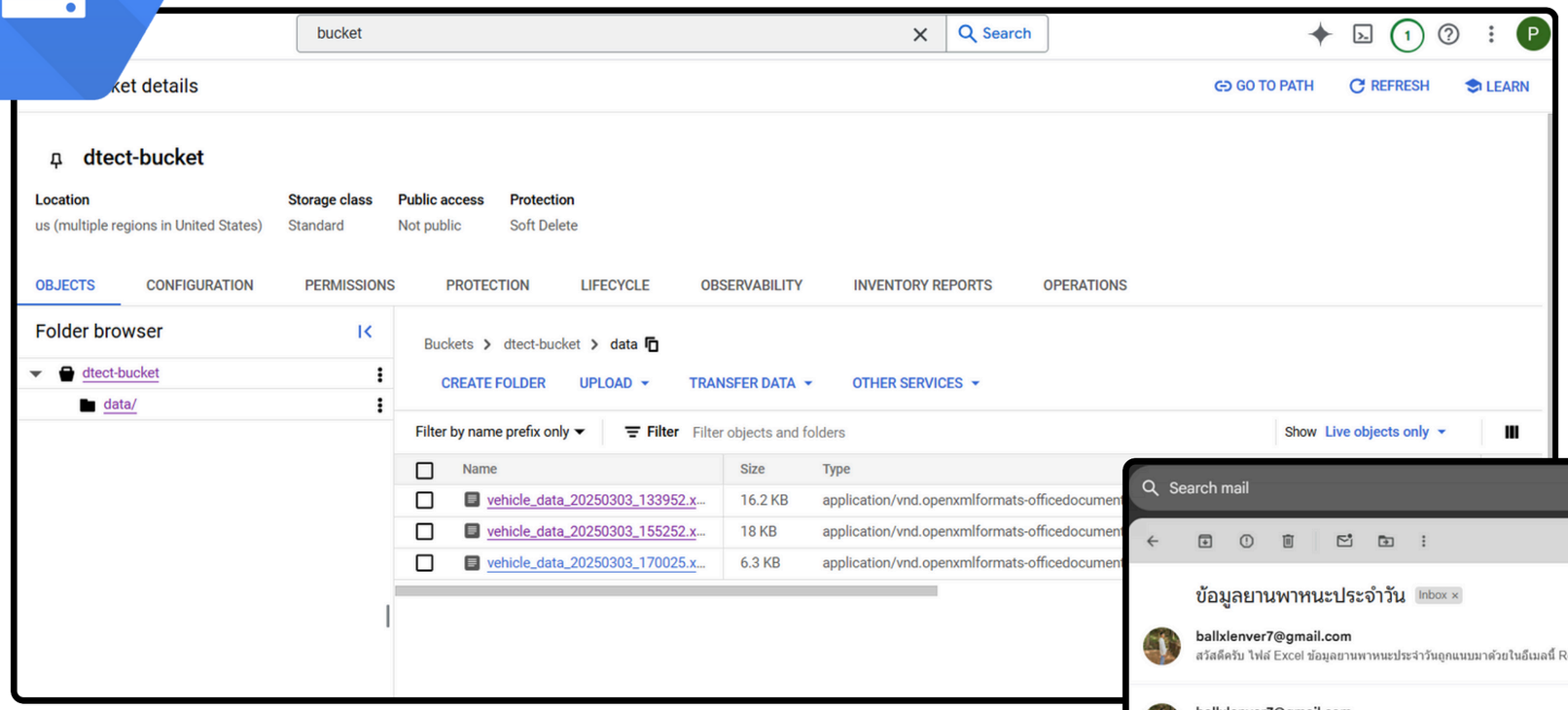
requirements.txt

```python
def fetch_data():
    url = "http://alivefordie.life/api/vehicle/today?GMT=7"
    retries = 3
    for i in range(retries):
        try:
            response = requests.get(url)
            response.raise_for_status()
            return response.json()
        except requests.exceptions.HTTPError as http_err:
            logger.error(f"HTTP error occurred: {http_err} - Status Code: {response.status_code}")
            logger.error(f"Response Text: {response.text}")
            if response.status_code == 500 and i < retries - 1:
                logger.info(f"Retrying... ({i+1}/{retries})")
                time.sleep(2)
                continue
            break
        except requests.exceptions.RequestException as req_err:
            logger.error(f"Request error occurred: {req_err}")
            break
    return {"error": "Failed to fetch data", "status_code": 500}

def upload_to_gcs(bucket_name, destination_blob_name, file_bytes):
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)
    blob.upload_from_file(file_bytes, content_type="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet")
    logger.info(f"Uploaded to {bucket_name}/{destination_blob_name} successfully")

def export_to_gcs_and_get_buffer(forward_lane_list, backward_lane_list):
    df_forward = pd.DataFrame(forward_lane_list)
    df_backward = pd.DataFrame(backward_lane_list)
```

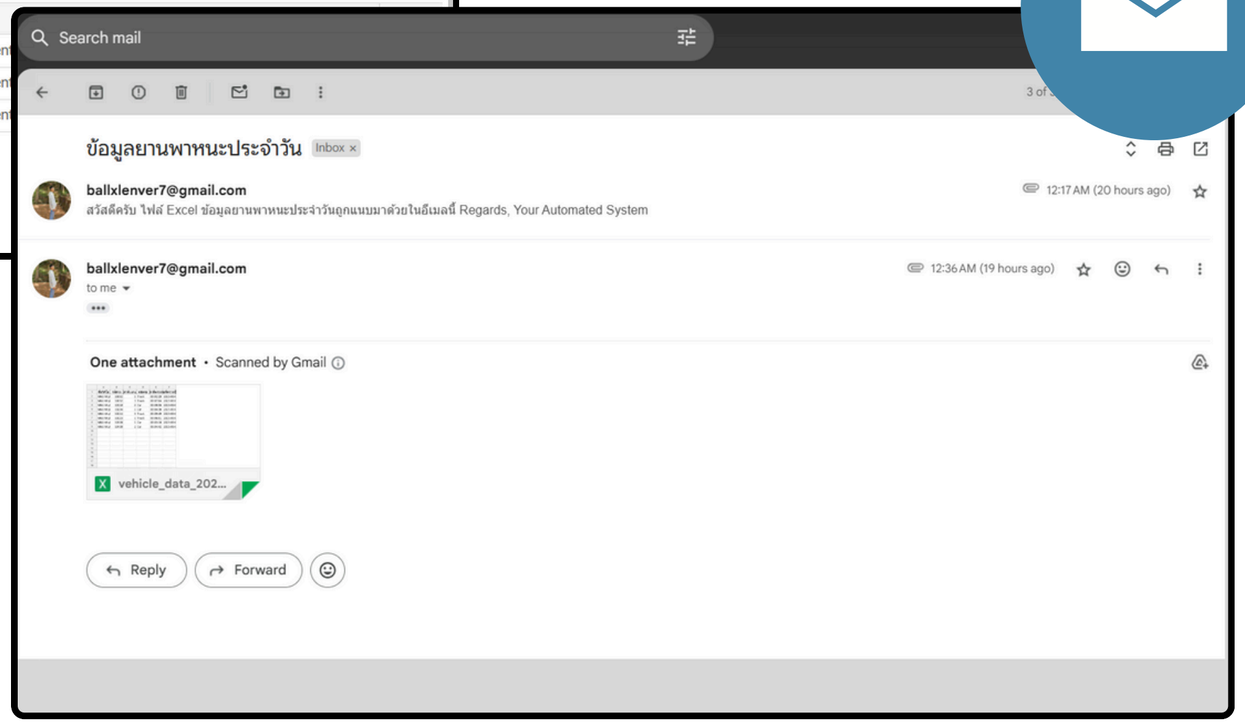
DOWNLOAD ZIP

สร้างฟังก์ชั่นชื่อ “daily-vehicle-detected-log”ที่ดึงข้อมูลยานพาหนะจาก API ประมวลผลด้วย Pandas สร้างไฟล์ Excel จากนั้นอัปโหลดไปยัง Cloud Storage และส่งทางอีเมลเป็นไฟล์แนบ

Cloud Bucket



Email



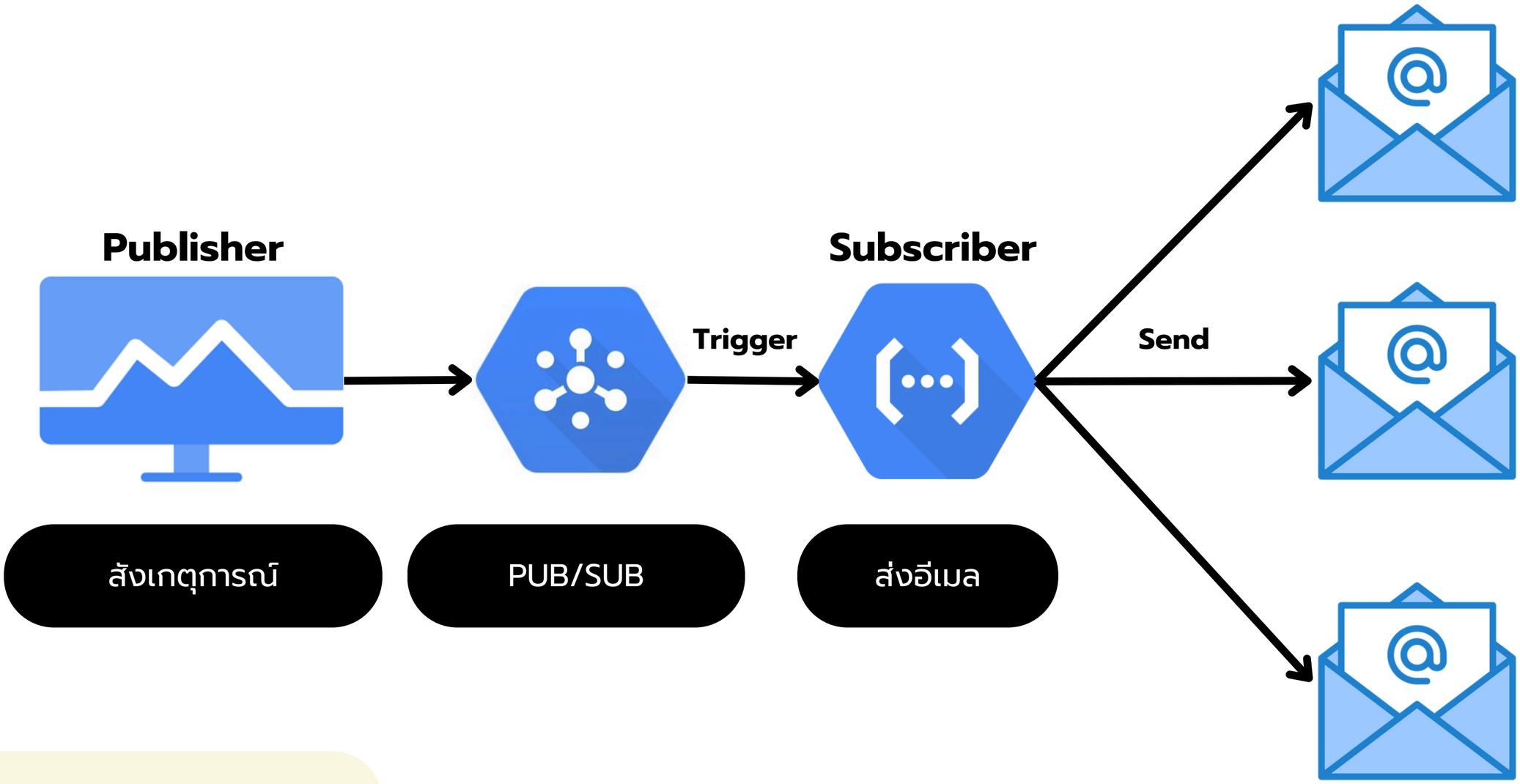
ผลลัพธ์หลังจากที่ฟังชั่น daily-vehicle-detected-log ใน Cloud Functions ทำงาน

การใช้งาน Cloud เพื่อสังเกตการณ์ค่าใช้จ่าย แจ้งเตือนไปยังอีเมล

เทคนิคที่ใช้

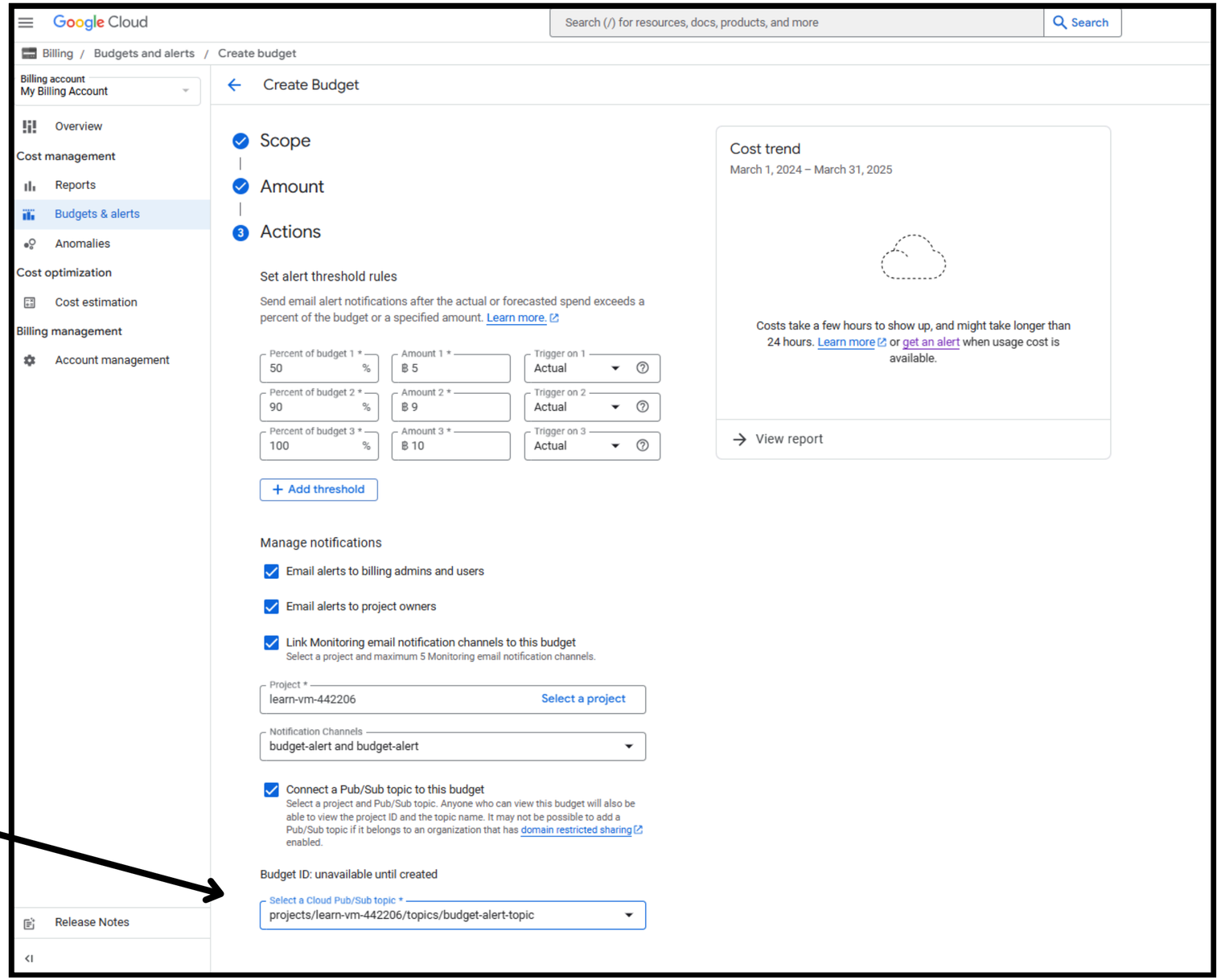
- Monitoring and Alerting (Other II)
- Pub/Sub (Category IV)

แผนภาพการใช้งาน Cloud เพื่อสังเกตการณ์ค่าใช้จ่ายจากนั้นแจ้งเตือนไปยังอีเมล

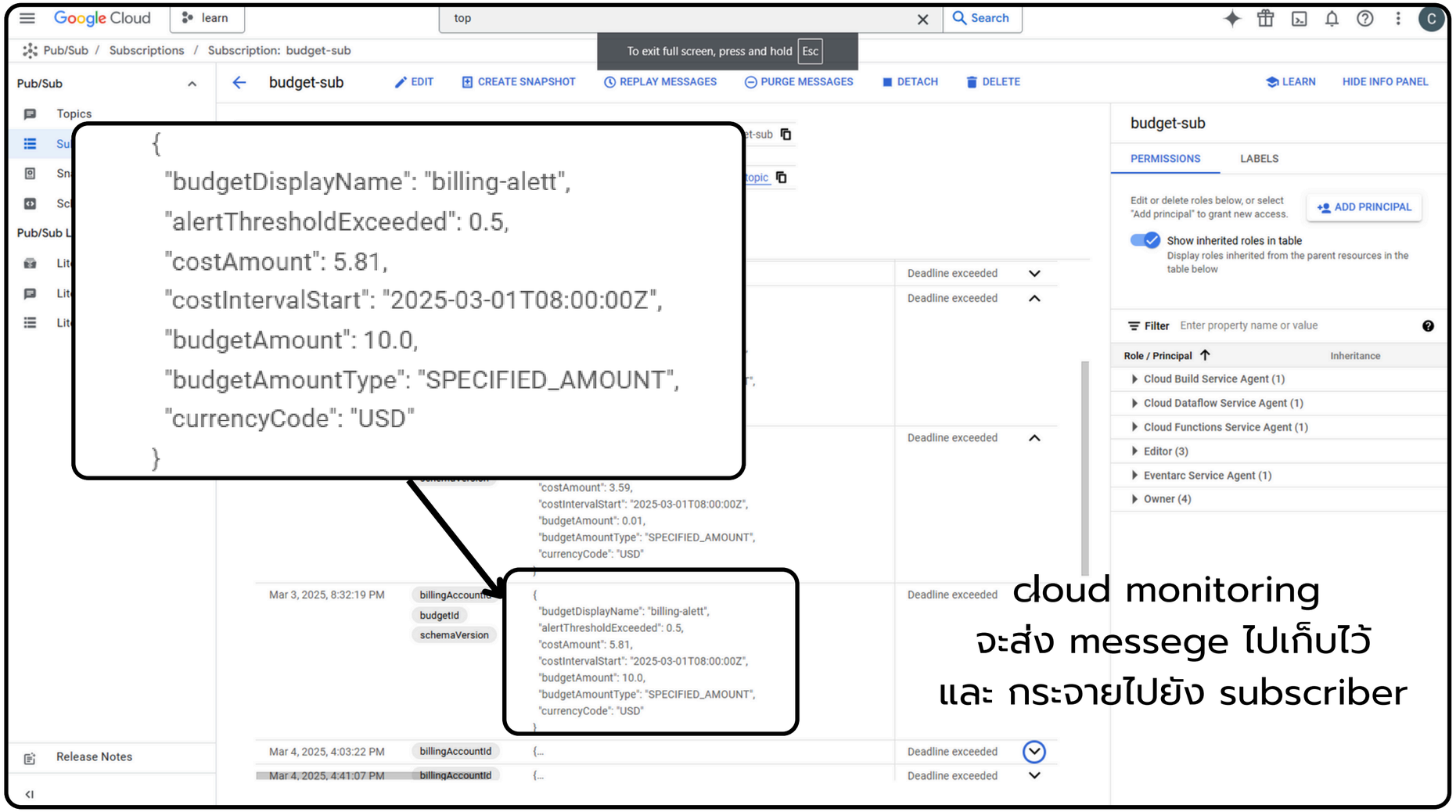


Cloud Monitoring

สร้าง topics
และ กำหนดให้ส่งข้อมูลไปเก็บ

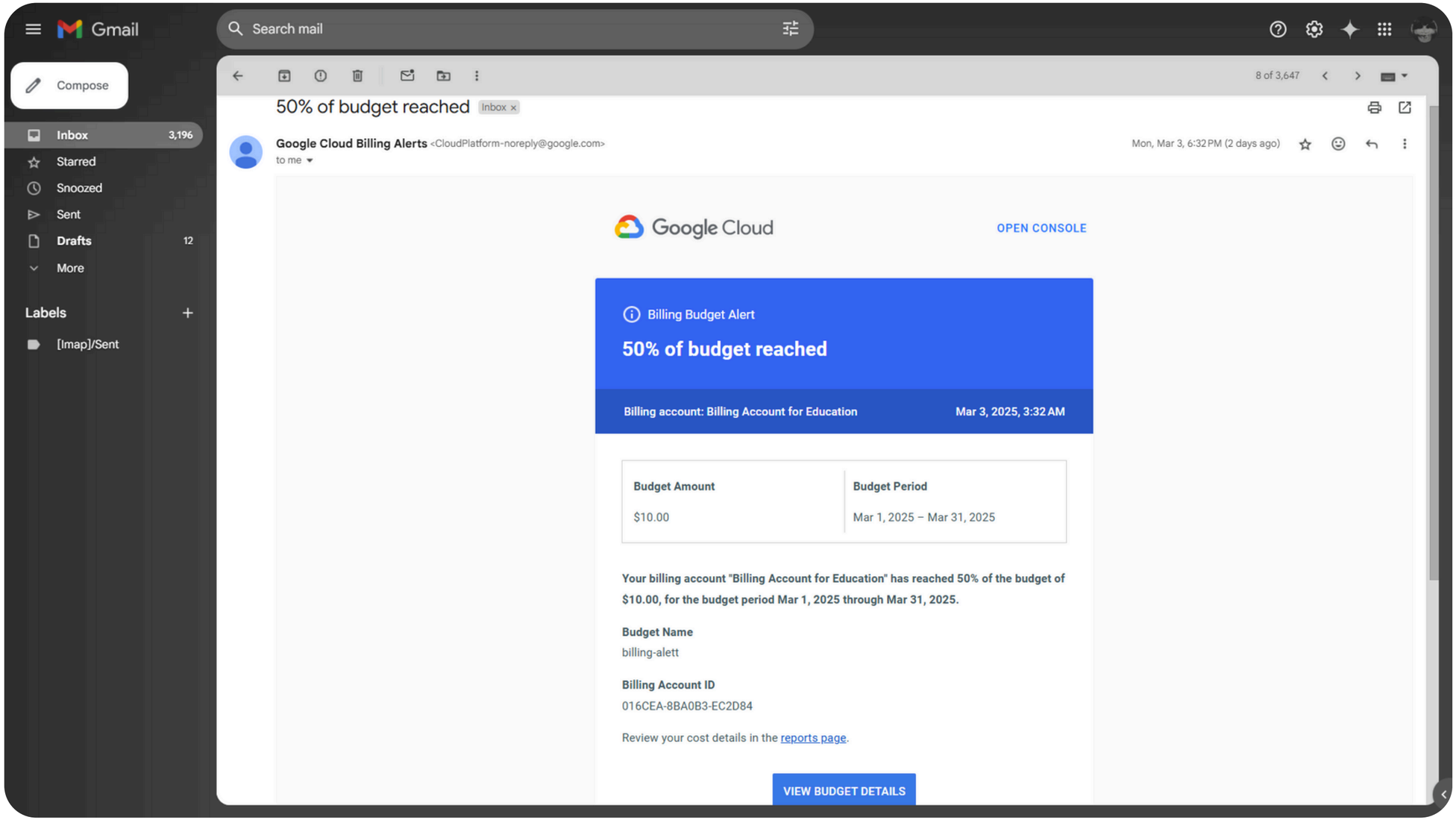


Pub/Sub

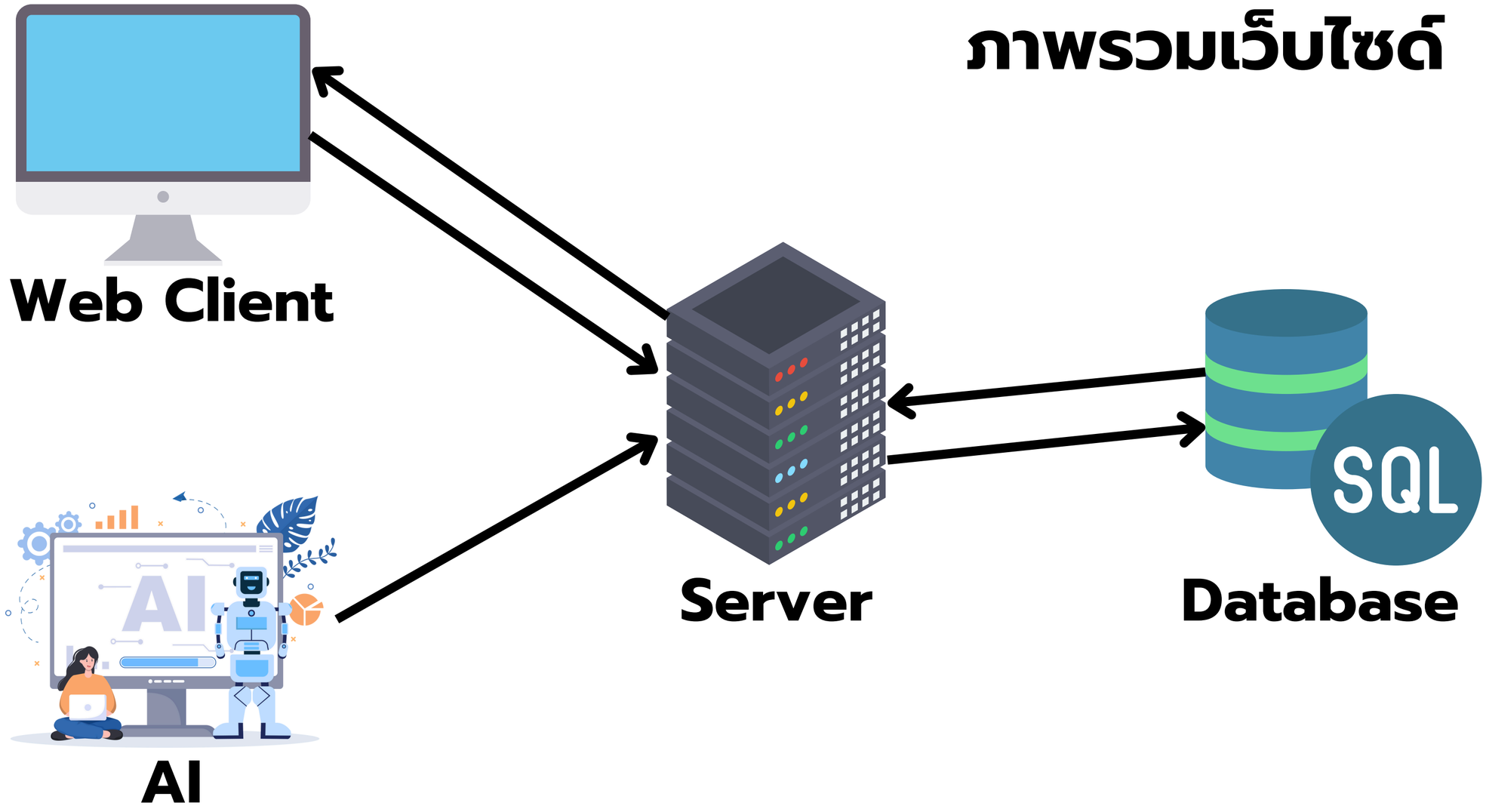


cloud monitoring
จะส่ง messege ไปเก็บไว้
และ กระจายไปยัง subscriber

ส่งอีเมลแจ้งเตือน

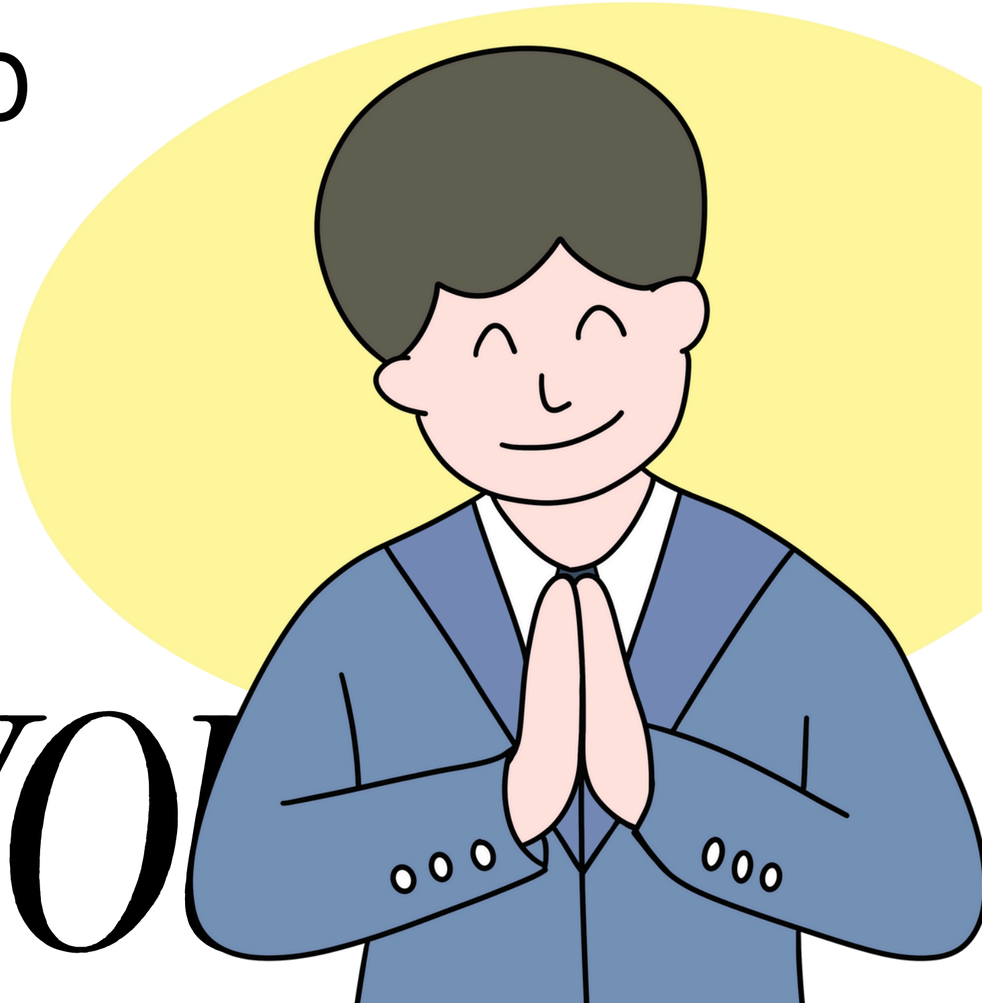


ภาพรวมเว็บไซต์



สมาชิกกลุ่ม Dtect

- นายโชติมนัส ชนะกุล 6610110074
- นายพชรพล เกตุแก้ว 6610110190
- นายคมสัน คงผล 6610110424



THANK YOU