Python System Monitoring Lab

Introduction

In this lab, students learn to monitor server performance and metrics (CPU, RAM, Disk Usage) using simple python and ***psutil*** (process and system utilities) library. They also learn how to define metric thresholds and send automated email alerts to System Administrators when metrics are exceeded.

Lab Objectives

At the end of this Lab, Learners will know how to:

- Understand how to use Python's **`psutil`** library to monitor system metrics such as CPU usage, RAM usage, and disk space.
- Learn how to trigger email alerts when system performance metrics exceed defined thresholds.
- Implement a basic alerting mechanism using Mailjet for email notifications.
- Practice integrating third-party libraries in Python to build monitoring tools.
- Create a simple monitoring script that can be extended for more complex use cases.

Task 1: Signup to Mail Jet Email API

Sign up for **MAIL JET** free email smtp service and create API Key & Secret Key via Mailjet Developer Portal.

Mail Jet documentation

Taks 2: Python Environment Setup

- Setup a python environment using **venv**
  - cli command: *python3 –m venv <environment name>*
- Install required libraries
  - cli command: *pip install psutil mailjet_rest*

Task 3: Creating Application file:

- Cli command: touch monitor.py and write your application code within the file.
  - Import required libraries
    - `import time`
    - `from mailjet_rest import Client`
    - `import psutil`

- Define mailjet credentials (import using **os library** as environment variables or define directly)

    -
      ```
      # Define mail credentials
      api_key="enter your api key from mailjet"
      api_secret=" enter your secret key from mailjet "
      ```
- Define System time (to show time of capturing metrics)

  ```
  # Define system time
  current_time = time.localtime()
  formatted_time = time.strftime("%Y-%m-%d %H:%M:%S",current_time)
  ```

- Define thresholds:

  ```
      # Define System thresholds ( 10% RAM, 50% free disk space, 10%
  CPU )
      CPU_THRESHOLD = 2
      RAM_THRESHOLD = 10
      DISK_THRESHOLD = 50
  ```

- Create function to send email alert:

```
def send_alert(subject, message):

    # instantiate mailjet client
    mailjet = Client(auth=(api_key, api_secret), version='v3.1')

    data = {
        'Messages': [
            {
            "From": {
                "Email": "your email address",
                "Name": "24/7 SysMon"
            },

            "To": [
                {
                "Email": "recipient email address”,
                "Name": "Admin"
                }
            ],
            "Subject": subject,
```

```
            "HTMLPart": f"<h3>{message}</h3>"
            }
        ]
    }

    try:
        result = mailjet.send.create(data=data)
        print(f"Email sent: {result.status_code}")
    except Exception as e:
        print(f"Failed to send email: {str(e)}")
```

- Collect system metrics

```
# Check system metrics
cpu_usage = psutil.cpu_percent(interval=1)
# print(cpu_usage)

ram_usage = psutil.virtual_memory().percent
# print(ram_usage)

disk_usage = psutil.disk_usage('/').percent
# print(disk_usage)
```

- Create a store for email message

```
# Create alert message based on threshold breaches
alert_message = ""
```

- Compose email alert message based on metric collection:

```
if cpu_usage > CPU_THRESHOLD:
    alert_message += f"CPU usage is high: {cpu_usage}% (Threshold:
{CPU_THRESHOLD}%)\n"

if ram_usage > RAM_THRESHOLD:
    alert_message += f"RAM usage is high: {ram_usage}% (Threshold:
{RAM_THRESHOLD}%)\n"

if disk_usage > DISK_THRESHOLD:
    alert_message += f"Disk space is low: {100 - disk_usage}% free (Threshold:
{DISK_THRESHOLD}% free)\n"
```

- Trigger email alert if threshold breached

```
# If any threshold is breached, send an email alert
if alert_message:
    send_alert(f"Python Monitoring Alert Alert-{formatted_time}", alert_message)
else:
    print("All system metrics are within normal limits.")
```

- Run monitor.py
- Verify email notification:
    - Check your inbox or spam to verify receipt of system metrics.

END OF LAB:

Submission: Take a screenshot of the email with system metrics and include in your github repo. Submit github repo link.