St. JOSEPH'S GROUP OF INSTITUTIONS
OMR, CHENNAI – 119

aws

Placement Empowerment Program

Cloud Computing and DevOps Centre

WRITE A PYTHON SCRIPT TO MONITOR AN APPLICATION

(Create a Python script that sends periodic HTTP requests to your application and alerts you if it's down)

NAME: NIDHISHA A DHAS

DEPARTMENT: AML

St. JOSEPH'S COLLEGE OF ENGINEERING | St. JOSEPH'S INSTITUTE OF TECHNOLOGY

AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

INTRODUCTION:

Ensuring high availability and reliability of an application is crucial for maintaining a seamless user experience. This Proof of Concept (PoC) focuses on developing a lightweight Python script to monitor the health status of an application by sending periodic HTTP requests.

IMPORTANCE:

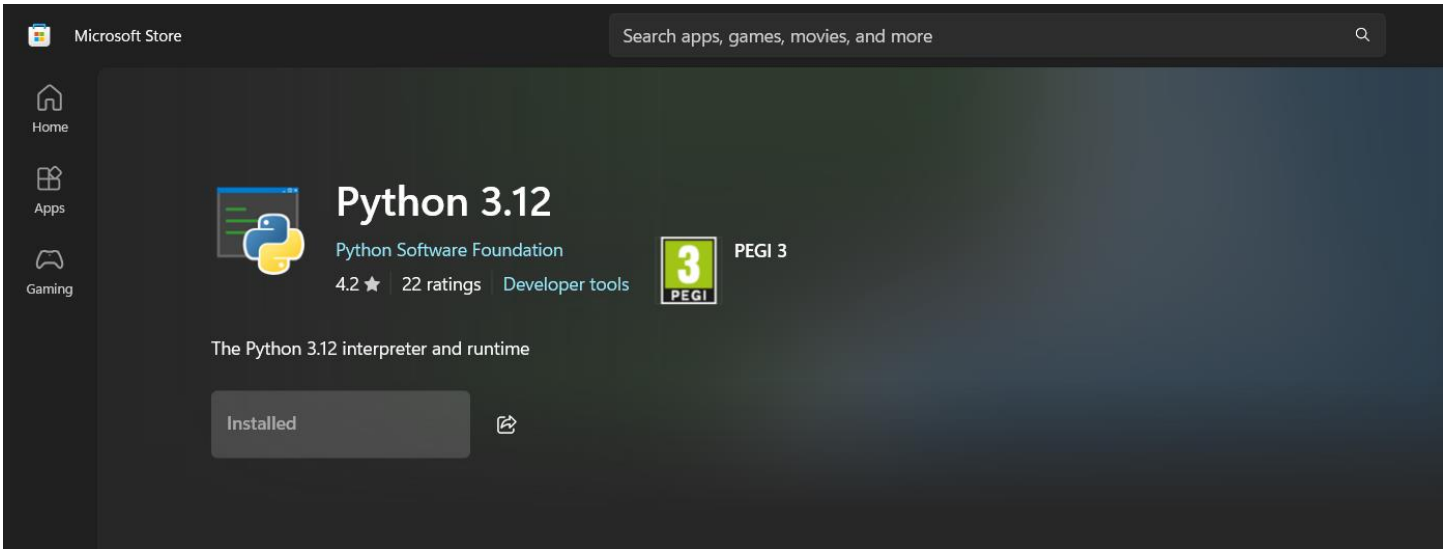
1. Proactive Monitoring: The script ensures continuous monitoring of your application's health, allowing you to detect downtime before it impacts users.
2. Real-Time Alerts: Sending instant email notifications enables quick action, reducing the response time in addressing application issues.
3. Improved Reliability: Automated monitoring helps maintain application uptime, ensuring a more reliable service for users.
4. Cost Efficiency: By identifying and fixing issues early, you can avoid costly downtime and potential revenue loss.
5. Skill Enhancement: Writing and implementing this script improves your skills in web monitoring, error handling, and email automation.
6. Scalable Monitoring: This PoC can be expanded to monitor multiple applications or integrated into a larger system for enterprise-level monitoring

STEP BY STEP OVERVIEW:

STEP 1: INSTALL PYTHON

- Open Microsoft store, search for 'Python 3.x.x' and click on the install.

- This will automatically add ‘python’ to your system’s path environment variable.

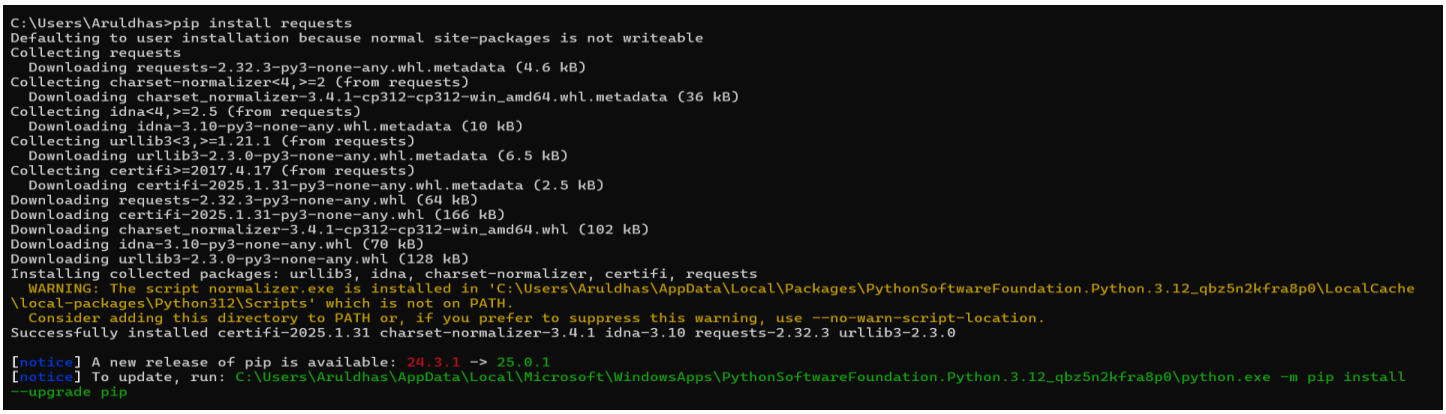


STEP 2: VERIFY THE INSTALLATION

- Open the command prompt and type ‘python –-version’ to verify if python is installed.

STEP 3: INSTALL THE REQUEST LIBRARY

- In the command prompt, type the following command to install the request library.



- The ‘smtplib’ is included in python y default, so no installation is needed for it.

STEP 4: PYTHON SCRIPT

- Firstly, create an EC2 instance.
- Open any text editor, type the following python script & save it as monitor_app.py.
- In the script, add your actual mail-id in the sender’s mail-id and give an app-specific password.
- Also, change the app_URL to your instance URL.
- Save the changes.

monitor_app.py

File Edit View

```
import requests
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import time

# Email configuration
sender_email = "nidhidhas06@gmail.com"
receiver_email = "swathika.gsk@gmail.com"
smtp_server = "smtp.gmail.com"
smtp_port = 587
smtp_user = "nidhidhas06@gmail.com"
smtp_password = "nidhi@2004"  # Your app-specific password

# Application to monitor
app_url = "https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#InstanceDetails:instanceId=i-06fd2a62858a7634b"  # Replace with your app's URL

# Function to send an email alert
def send_alert_email():
    try:
        # Create message
        message = MIMEMultipart()
        message["From"] = sender_email
        message["To"] = receiver_email
        message["Subject"] = "Application Down Alert"

        body = "Your application is down. Please check it immediately!"
        message.attach(MIMEText(body, "plain"))

        # Establishing connection to the email server
        with smtplib.SMTP(smtp_server, smtp_port) as server:
            server.starttls()  # Secure the connection
            server.login(smtp_user, smtp_password)
            server.sendmail(sender_email, receiver_email, message.as_string())

        print("Alert email sent successfully!")

    except Exception as e:
        print(f"Failed to send email: {e}")

# Function to check the application
def check_application():
    try:
        response = requests.get(app_url, timeout=10)  # 10 seconds timeout
        if response.status_code != 200:
            print(f"Warning: Application returned {response.status_code}")
            send_alert_email()
        else:
            print(f"Application is up! Status code: {response.status_code}")

    except requests.exceptions.RequestException as e:
        print(f"Error: {e}")
        send_alert_email()

# Main function to run the script
def monitor_application():
    while True:
        check_application()
        time.sleep(60)  # Check every 60 seconds (1 minute)

if __name__ == "__main__":
    monitor_application()
```

STEP 5: RUN THE PYTHON SCRIPT

- In the command prompt, navigate to the folder where the python script is saved. (cd <folder_path>).

```
C:\Users\Aruldhas>cd "C:\Users\Aruldhas\Desktop\monitor_app"
```

- Also, run the script using 'python monitor_app.py' command.

```
C:\Users\Aruldhas\Desktop\monitor_app>python monitor_app.py
Application is up! Status code: 200
```

STEP 6: TERMINATE THE SCRIPT

- To terminate the scrip, click Ctrl+C in the command prompt window.

```
Traceback (most recent call last):
  File "C:\Users\Aruldhas\Desktop\monitor_app\monitor_app.py", line 62, in <module>
    monitor_application()
  File "C:\Users\Aruldhas\Desktop\monitor_app\monitor_app.py", line 59, in monitor_application
    time.sleep(60)  # Check every 60 seconds (1 minute)
    ^^^^^^^^^^^^^^
KeyboardInterrupt
^C
C:\Users\Aruldhas\Desktop\monitor_app>
```

CONCLUSION:

- Monitor Web Application Health: Periodically send HTTP requests to your application to verify if it is up and running.
- Automated Alerts: Automatically send email alerts whenever the application is down or unreachable, ensuring quick response time.
- Error Handling: Implement error handling to detect and respond to network issues, timeout errors, and non-200 HTTP responses.
- Script Automation: Run the script in an automated manner (every 60 seconds or as configured) to continuously monitor the application's availability.

- Reliability and Maintenance: Improve application reliability by ensuring it’s monitored in real-time and receive alerts on downtime or issues that need attention.
- Email Notification System: Implement an email notification system using SMTP to ensure that administrators or relevant personnel are promptly informed of application downtime.