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: CHANDRU S

Department: INFORMATION TECHNOLOGY

Introduction

Ensuring the high availability and reliability of an application is essential for delivering a seamless user experience. This Proof of Concept (PoC) aims to develop a lightweight Python script to monitor an application's health by sending periodic HTTP requests. The script will detect downtime and promptly notify administrators via email alerts.

Overview

This PoC focuses on creating a Python-based monitoring solution with the following key components:

1. **Setting Up the Monitoring Environment** – Install Python and necessary libraries (requests, smtplib) and configure an email account for alerts.
2. **Making Periodic HTTP Requests** – Use the requests library to send recurring HTTP requests to the application's URL.
3. **Defining Success and Failure Conditions** – Assess HTTP response status codes (200 for success, non-200 for failure) to determine application status.
4. **Sending Email Alerts** – Utilize smtplib to send email notifications when downtime is detected.
5. **Automating Periodic Checks** – Implement a loop to conduct regular application status checks (e.g., every 60 seconds).
6. **Logging and Handling Errors** – Log failures and handle exceptions to enhance script reliability.

Objectives

This PoC aims to achieve the following objectives:

1. **Understanding Web Monitoring Fundamentals** – Learn how to periodically check web application availability using HTTP requests.
2. **Developing Practical Scripting Skills** – Gain hands-on experience in writing Python scripts that interact with web services and manage errors.
3. **Implementing Automated Alerting** – Create a system that detects downtime and sends email notifications automatically.
4. **Handling HTTP Status Codes** – Interpret status codes (e.g., 200, 404, 500) to assess application health.
5. **Automating Email Notifications** – Use SMTP to send alerts to administrators when issues arise.
6. **Enhancing Reliability** – Build a simple yet effective monitoring system that continuously ensures application availability.

Step-by-Step Overview

Step 1:

- Open the **Microsoft Store** on your computer.
- In the search bar, type "**Python**" and press **Enter**.
- Find the latest version of Python (e.g., **Python 3.x.x**) and click on it.
- Click the **Install** button to install Python on your system.
    - This will automatically add Python to your system's PATH environment variable.

Step 2:

- Open the Command Prompt (CMD).
- Type the following command to verify that Python is installed:
  python --version
- This should return the installed Python version, e.g., **Python 3.x.x**.
- If you see the version number, Python is correctly installed.

```
C:\Users\chandru>python --version
Python 3.12.0
```

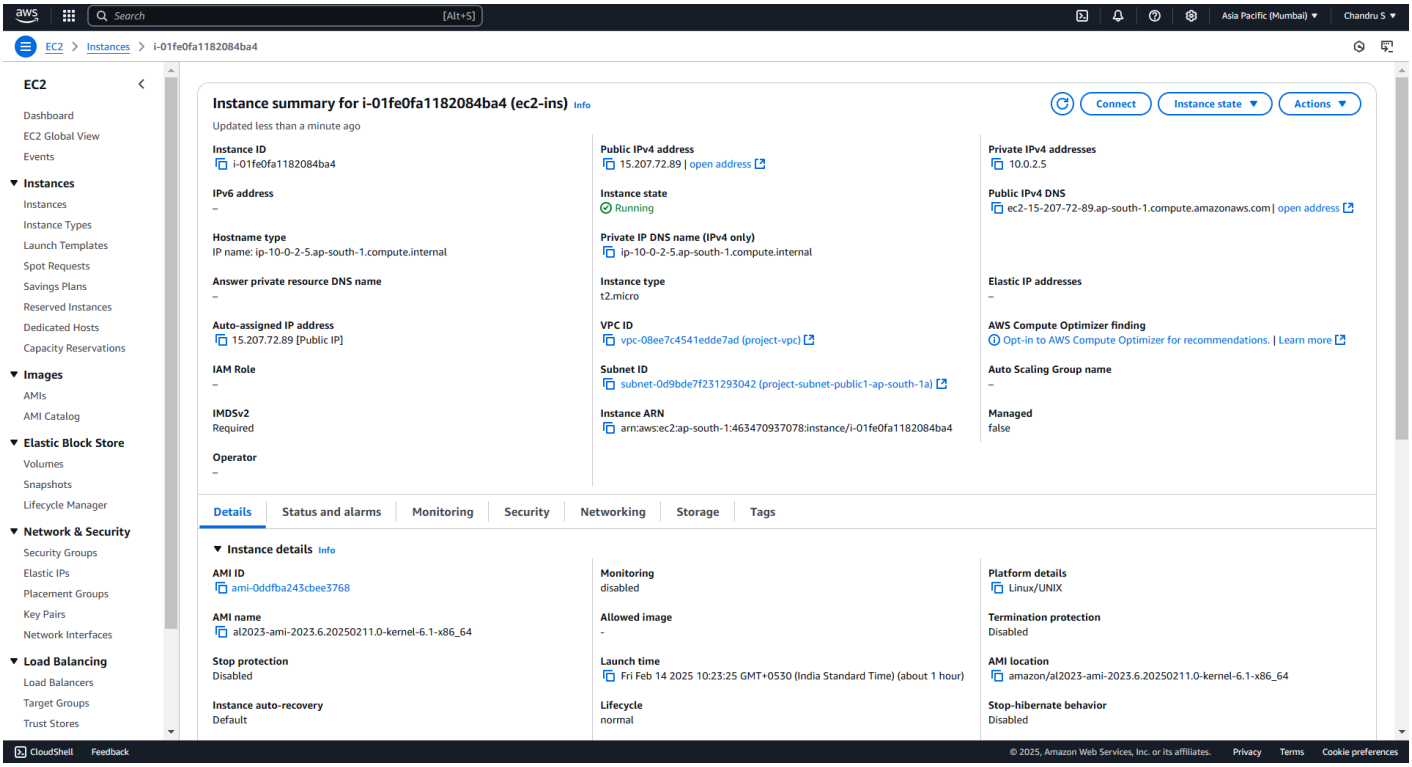
Step 3:

- In Command Prompt (CMD), type the following command to install the requests library:
  pip install requests
- The **smtplib** library is included with Python by default, so no installation is needed.

```
C:\Users\chandru>pip install requests
Requirement already satisfied: requests in c:\users\chandru\appdata\local\programs\python\python312\lib\site-packages (2
.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\chandru\appdata\local\programs\python\python312\lib\
site-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\chandru\appdata\local\programs\python\python312\lib\site-package
s (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\chandru\appdata\local\programs\python\python312\lib\site-p
ackages (from requests) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\chandru\appdata\local\programs\python\python312\lib\site-p
ackages (from requests) (2025.1.31)
```

Step 4:

- Create an **EC2 Instance**.
- Open any text editor (e.g., Notepad, VS Code).
- Copy and paste the Python script to monitor your EC2 instance.
- Replace your_email@example.com with your actual Gmail address (e.g., your_email@gmail.com).
- Set smtp_user to your **Gmail address** as well.
- Enter your **app-specific password** (not your Gmail password) for the smtp_password field. If you don't have an app-specific password, create one in your Google Account settings (Security section under App passwords).
- Change app_url to your **Instance URL**.
- Save the file with a **.py** extension.



task17.py - C:\Users\chandru\AppData\Local\Programs\Python\Python312\task17.py (3.12.0)

File Edit Format Run Options Window Help

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

# Email configuration
sender_email = "chandrusaravanan13@gmail.com"
receiver_email = "cyberphoenix023@gmail.com"
smtp_server = "chandrusaravanan13@gmail.com"
smtp_port = 587
smtp_user = "chandrusaravanan13@gmail.com"
smtp_password = "chan1001#"  # 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-01fe0fa1182084ba4"  # 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}")
```

Step 5:

In Command Prompt (CMD), run the python script with the following command:
python path\to\your\script\directory\file_name.py

```
C:\Users\chandru>python C:\Users\chandru\AppData\Local\Programs\Python\Python312\task17.py
Application is up! Status code: 200
```

Step 6:

To stop the script at any time, press **Ctrl + C** in the Command Prompt window.

```
C:\Users\chandru>python C:\Users\chandru\AppData\Local\Programs\Python\Python312\task17.py
Application is up! Status code: 200
Traceback (most recent call last):
  File "C:\Users\chandru\AppData\Local\Programs\Python\Python312\task17.py", line 62, in <module>
    monitor_application()
  File "C:\Users\chandru\AppData\Local\Programs\Python\Python312\task17.py", line 59, in monitor_application
    time.sleep(60)  # Check every 60 seconds (1 minute)
    ^^^^^^^^^^^^^^
KeyboardInterrupt
^C
```

Outcome

- **Monitor Web Application Health:** Periodically send HTTP requests to verify if the application is up and running.
- **Automated Alerts:** Automatically send email alerts whenever the application is down or unreachable, ensuring a quick response.

- **Error Handling:** Implement error handling to detect and respond to network issues, timeout errors, and non-200 HTTP responses.
- **Script Automation:** Run the script at regular intervals (e.g., every 60 seconds) to continuously monitor application availability.
- **Reliability and Maintenance:** Improve application reliability by ensuring real-time monitoring and alerts for any issues.
- **Email Notification System:** Use SMTP to notify administrators or relevant personnel of application downtime promptly.