

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: Santhoshini T

Department: CSE

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.

Overview

Python Script to Monitor an Application

1. **Setting Up the Monitoring Environment:** Install Python and required libraries (requests, smtplib), and configure an email account for alerts.
2. **Making Periodic HTTP Requests:** Use requests to periodically send HTTP requests to the application URL.
3. **Defining Success and Failure Conditions:** Check HTTP response status codes (200 for success, non-200 for failure) to detect application status.
4. **Sending Email Alerts:** If the application is down, use smtplib to send an email alert to a specified recipient.
5. **Automating Periodic Checks:** Use an infinite loop (while True) to repeatedly check the application's status at regular intervals (e.g., every 60 seconds).
6. **Logging and Handling Errors:** Log any errors or failures and handle exceptions to ensure script reliability.

Objectives

Python Script to Monitor an Application

1. **Understanding Web Monitoring Fundamentals:** Learn how to periodically check the availability of web applications using HTTP requests.
2. **Practical Scripting Skills:** Gain hands-on experience in writing Python scripts that interact with web services and handle errors.
3. **Automated Alerting:** Develop a script that automatically detects application downtime and sends email notifications.
4. **Handling HTTP Status Codes:** Learn how to interpret HTTP status codes (e.g., 200, 404, 500) to assess the health of an application.

5. **Email Automation:** Gain experience in automating email alerts via SMTP to notify administrators when issues are detected.
6. **Improving Reliability:** Explore how to build a simple and reliable monitoring system that runs continuously to ensure your application is always available.

Importance

Python Script to Monitor an Application

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 from Microsoft Store

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



Step 2: Verify Python Installation

1. Open the **Command Prompt (CMD)**:
2. Type the following command to verify that Python is installed:

python --version

3. This should return the version of Python installed, e.g., Python 3.x.x.
4. If you see the version number, Python is correctly installed.

```
C:\Users\hi>python --version
Python 3.11.9
```

Step 3: Install Required Libraries (requests, smtplib)

1. In **Command Prompt (CMD)**, type the following command to install the **requests** library:

```
pip install requests
```

2. The **smtplib** library is included with Python by default, so no installation is needed for it.

```
C:\Users\hi>python --version
Python 3.11.9

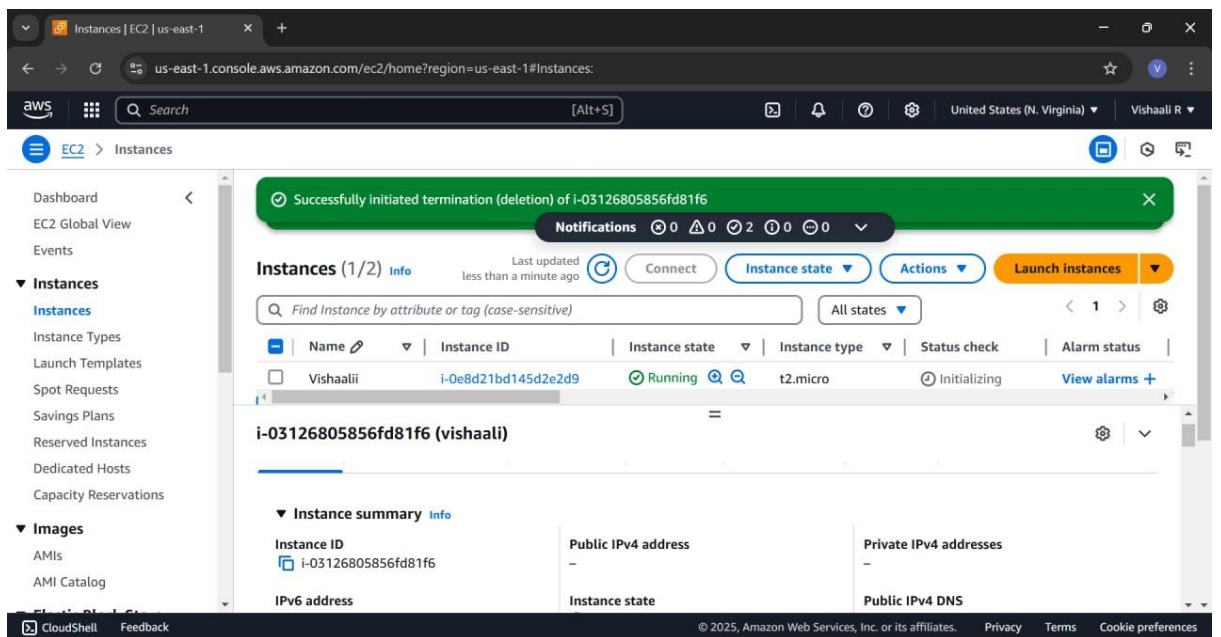
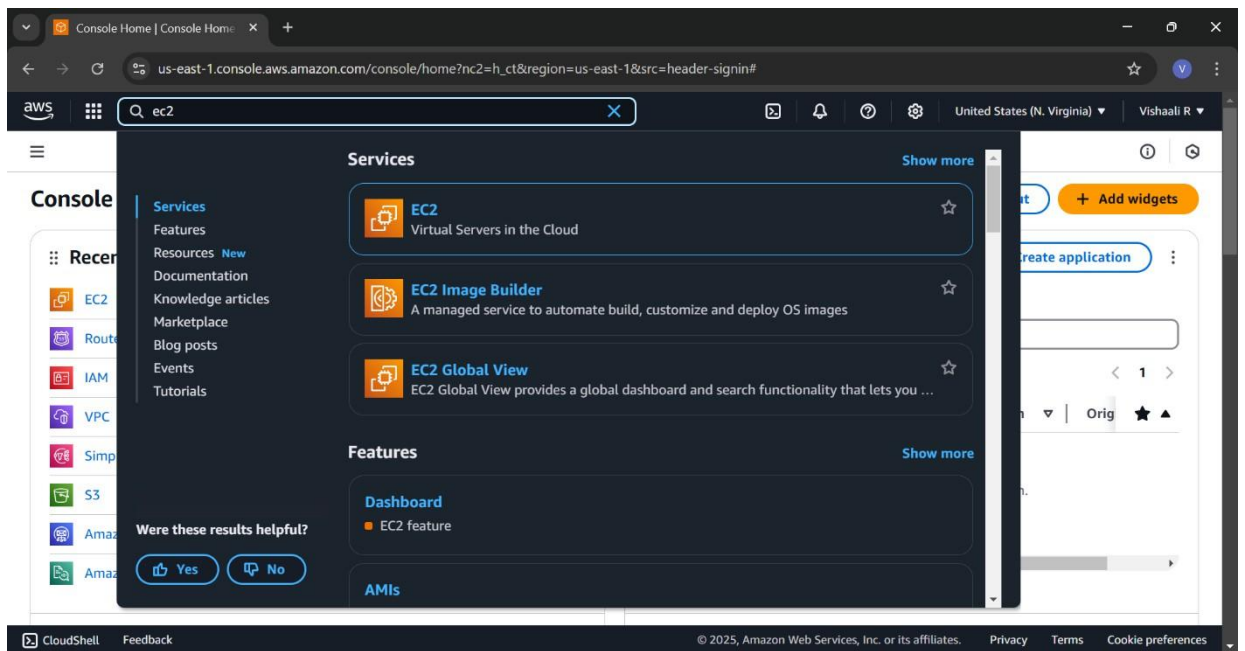
C:\Users\hi>pip install requests
Requirement already satisfied: requests in c:\users\hi\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfradp0\localcache\local-packages\python311\site-packages (2.32.3)
Requirement already satisfied: charset-normalizer<4,=>2 in c:\users\hi\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfradp0\localcache\local-packages\python311\site-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,=>2.5 in c:\users\hi\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfradp0\localcache\local-packages\python311\site-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,=>1.21.1 in c:\users\hi\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfradp0\localcache\local-packages\python311\site-packages (from requests) (2.3.0)
Requirement already satisfied: certifi<2017.4.17 in c:\users\hi\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfradp0\localcache\local-packages\python311\site-packages (from requests) (2025.1.31)

[notice] A new release of pip is available: 26.0 -> 25.0.1
[notice] To update, run: c:\users\hi\appdata\local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11.qbz5n2kfradp0\python.exe -m pip install --upgrade pip
```

3.

Step 4: Write the Python Script

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



Instance details | EC2 | us-east-1

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstanceDetails:instanceId=i-0e8d21bd145d2e2d9

Search [Alt+S]

United States (N. Virginia) Vishali R

EC2 > Instances > i-0e8d21bd145d2e2d9

Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

Instance summary for i-0e8d21bd145d2e2d9 (Vishaalii) Info

Updated less than a minute ago

Instance ID

i-0e8d21bd145d2e2d9

IPv6 address

-

Hostname type

IP name: ip-172-31-31-32.ec2.internal

Answer private resource DNS name

IPv4 (A)

Auto-assigned IP address

18.212.252.59 [Public IP]

Public IPv4 address

18.212.252.59 | open address

Instance state

Running

Private IP DNS name (IPv4 only)

ip-172-31-31-32.ec2.internal

Instance type

t2.micro

VPC ID

vpc-003aed95b395244f9

Private IPv4 addresses

172.31.31.32

Public IPv4 DNS

ec2-18-212-252-59.compute-1.amazonaws.com | open address

Elastic IP addresses

-

AWS Compute Optimizer finding

Opt-in to AWS Compute Optimizer for reco

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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 = "vishaaliraja3@gmail.com"
receiver_email = "jeffrinajames@gmail.com"
smtp_server = "vishaaliraja3gmail.com"
smtp_port = 587
smtp_user = "vishaaliraja3@gmail.com"
smtp_password = "Vishaali666" # Your app-specific password

# Application to monitor
app_url = "https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstanceDetails:instanceId=i-0e8d21bd145d2e2d9" # 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 6: Run the Python Script

1. In **Command Prompt (CMD)**, navigate to the folder where the Python script is saved using the `cd` command:

```
cd path\to\your\script\directory
```

2. Run the script with the following command:

```
python monitor_app.py
```

.

```
C:\Users\hi>python C:\Users\hi\OneDrive\Desktop\vish.py  
Application is up! Status code: 200
```


Step 7: Stop the Script

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

```
C:\Users\hi>python C:\Users\hi\OneDrive\Desktop\vish.py
Application is up! Status code: 200
Application is up! Status code: 200
Application is up! Status code: 200
Application is up! Status code: 200
Application is up! Status code: 200
Application is up! Status code: 200
Application is up! Status code: 200
Application is up! Status code: 200
|
```

Outcome

- **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.