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: SHALINI D Department: IT

St. JOSEPH'S COLLEGE OF ENGINEERING | St. JOSEPH'S INSTITUTE OF TECHNOLOGY
AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction

Ensuring the availability of your application is critical for maintaining user satisfaction and trust. Monitoring your application proactively can help you detect issues before they impact your users. By creating a Python script, you can automate the process of checking the application's health. The script will periodically send HTTP requests to your application and alert you if it detects any downtime or issues, enabling you to take immediate action.

Objectives

The primary objective of this Python script is to provide an automated and efficient way to monitor your application's availability. The script will:

Send periodic HTTP requests to your application's endpoint.

Check the response status to determine if the application is running correctly.

Send alerts (e.g., email, SMS, or log entry) if the application is down or returning unexpected response.

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\user>python --version
Python 3.12.9
```

Step 3: Install Required Libraries (requests, smtplib)

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

```
pip install requests
```

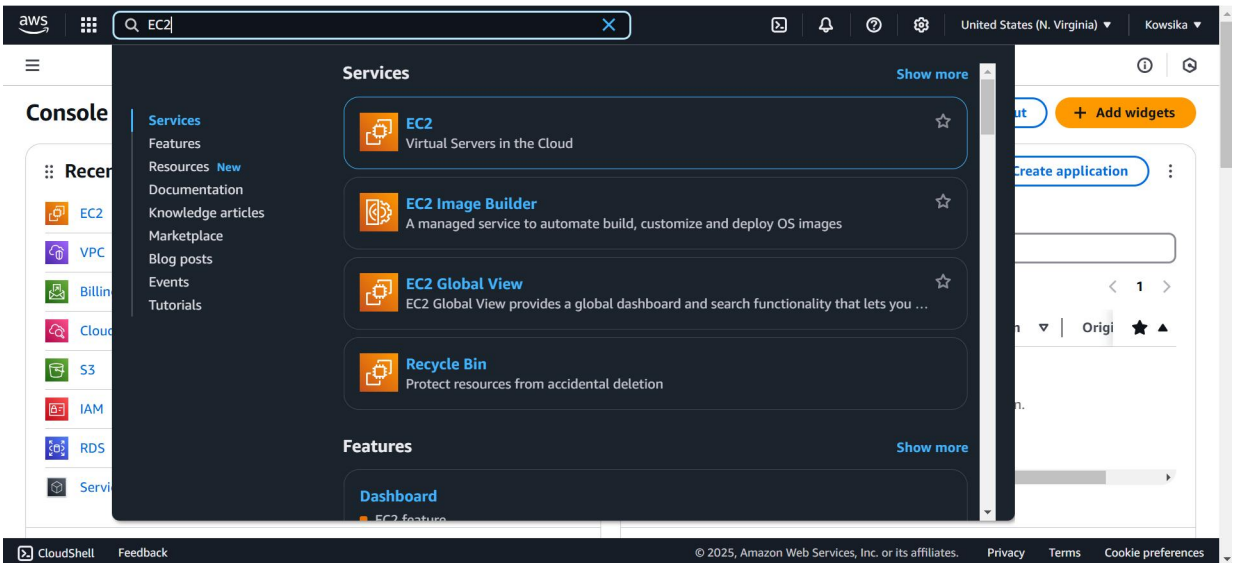
2. The **smtplib** library is included with Python by default, so no installation is needed for it.

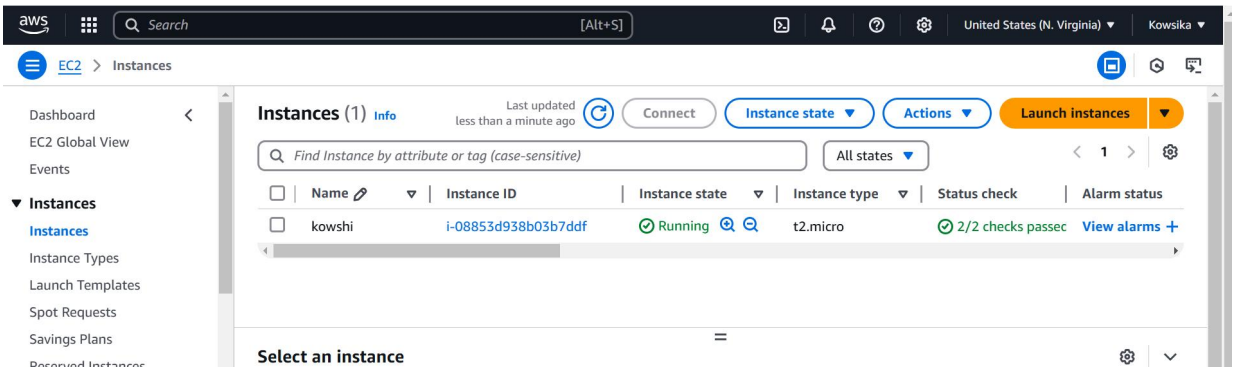
```
C:\Users\user>pip install requests
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: requests in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.12_qbz
5n2kfra8p0\localcache\local-packages\python312\site-packages (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation
.python.3.12_qbz5n2kfra8p0\localcache\local-packages\python312\site-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.python.3.12
_qbz5n2kfra8p0\localcache\local-packages\python312\site-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.pytho
n.3.12_qbz5n2kfra8p0\localcache\local-packages\python312\site-packages (from requests) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\user\appdata\local\packages\pythonsoftwarefoundation.pytho
n.3.12_qbz5n2kfra8p0\localcache\local-packages\python312\site-packages (from requests) (2025.1.31)

[notice] A new release of pip is available: 24.3.1 -> 25.0.1
[notice] To update, run: C:\Users\user\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.12_qbz5n2kf
ra8p0\python.exe -m pip install --upgrade pip
```

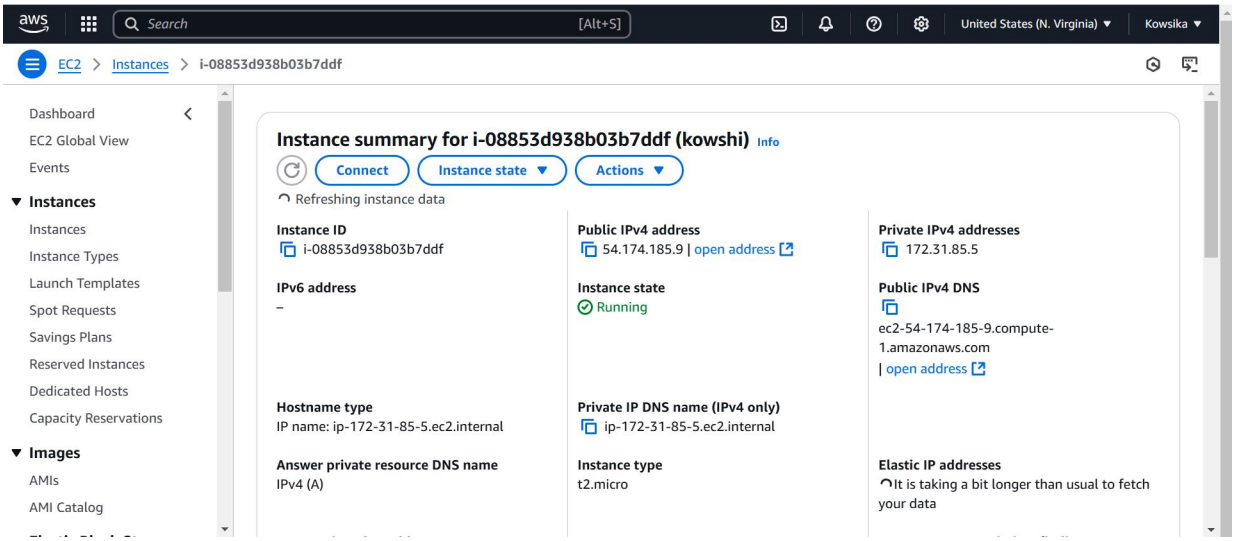
3.

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


EC2
United States (N. Virginia)
Kowsika
Services
Show more
EC2
Virtual Servers in the Cloud
EC2 Image Builder
A managed service to automate build, customize and deploy OS images
EC2 Global View
EC2 Global View provides a global dashboard and search functionality that lets you ...
Recycle Bin
Protect resources from accidental deletion
Features
Show more
Dashboard
Services
Features
Resources New
Documentation
Knowledge articles
Marketplace
Blog posts
Events
Tutorials
Console
Add widgets
Create application
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search
[Alt+S]
United States (N. Virginia)
Kowsika
EC2 > Instances
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Instances (1) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
kowshi
i-08853d938b03b7ddf
Running
t2.micro
2/2 checks passec
View alarms
Select an instance


Search
[Alt+S]
United States (N. Virginia)
Kowsika
EC2 > Instances > i-08853d938b03b7ddf
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Instance summary for i-08853d938b03b7ddf (kowshi) Info
Connect
Instance state
Actions
Refreshing instance data
Instance ID
i-08853d938b03b7ddf
IPv6 address
–
Hostname type
IP name: ip-172-31-85-5.ec2.internal
Answer private resource DNS name
IPv4 (A)
Public IPv4 address
54.174.185.9 | open address
Instance state
Running
Private IP DNS name (IPv4 only)
ip-172-31-85-5.ec2.internal
Instance type
t2.micro
Private IPv4 addresses
172.31.85.5
Public IPv4 DNS
ec2-54-174-185-9.compute-1.amazonaws.com | open address
Elastic IP addresses
It is taking a bit longer than usual to fetch your data

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 = "mjnevesh06@gmail.com"
receiver_email = "kowshika3134@gmail.com"
smtp_server = "mjnevesh06@gmail.com"  # Use your email provider's SMTP server
smtp_port = 587  # Standard port for TLS
smtp_user = "mjnevesh06@gmail.com"  # Your email (used for login)
smtp_password = "nev79@06"  # Your email password or 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-0446b40c51eec972a"  # 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}")
```

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

Outcome
By creating and running a Python script to monitor your application, you will achieve the following outcomes:

Proactive Monitoring: Your application will be monitored continuously, allowing you to detect issues before they affect users. This proactive approach ensures minimal downtime and maintains user satisfaction.

Automated Alerts: The script will automatically send alerts if the application is down or not responding as expected. These alerts can be configured to notify you via email, SMS, or any other preferred method, enabling you to take immediate action.

Improved Reliability: Regular monitoring and timely alerts contribute to the overall reliability of your application. You'll be able to address potential problems quickly, reducing the risk of extended downtime.

Enhanced User Experience: By ensuring your application is always available, you provide a consistent and reliable experience for your users, which can lead to increased trust and loyalty.

Data Insights: Monitoring data can help you identify patterns and potential areas for improvement in your application. This information can be valuable for optimizing performance and planning future updates.