# 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:YasvanthRajan E Department:AML



# 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**

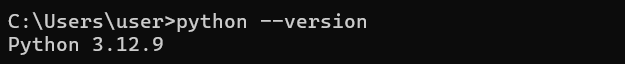
1. Open the **Microsoft Store** on your computer.
2. In the search bar, type **"Python"** and press **Enter**.
3. Find the latest version of Python (e.g., **Python 3.x.x**), and click on it.
4. Click the **Install** button to install Python on your system.
   * This will automatically add Python to your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

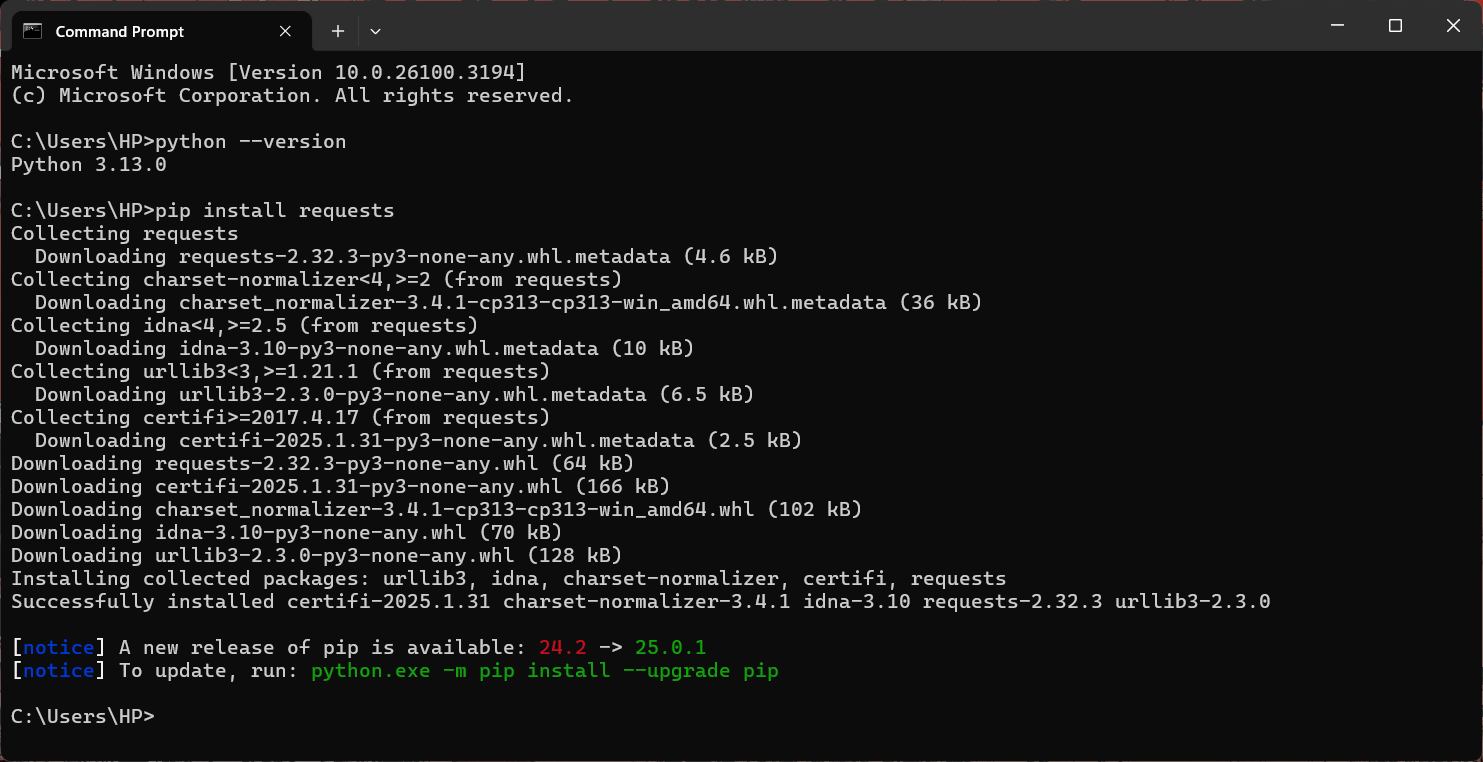
1. This should return the version of Python installed, e.g., Python 3.x.x.
2. If you see the version number, Python is correctly installed.



#### **Step 3: Install Required Libraries (requests, smtplib)**

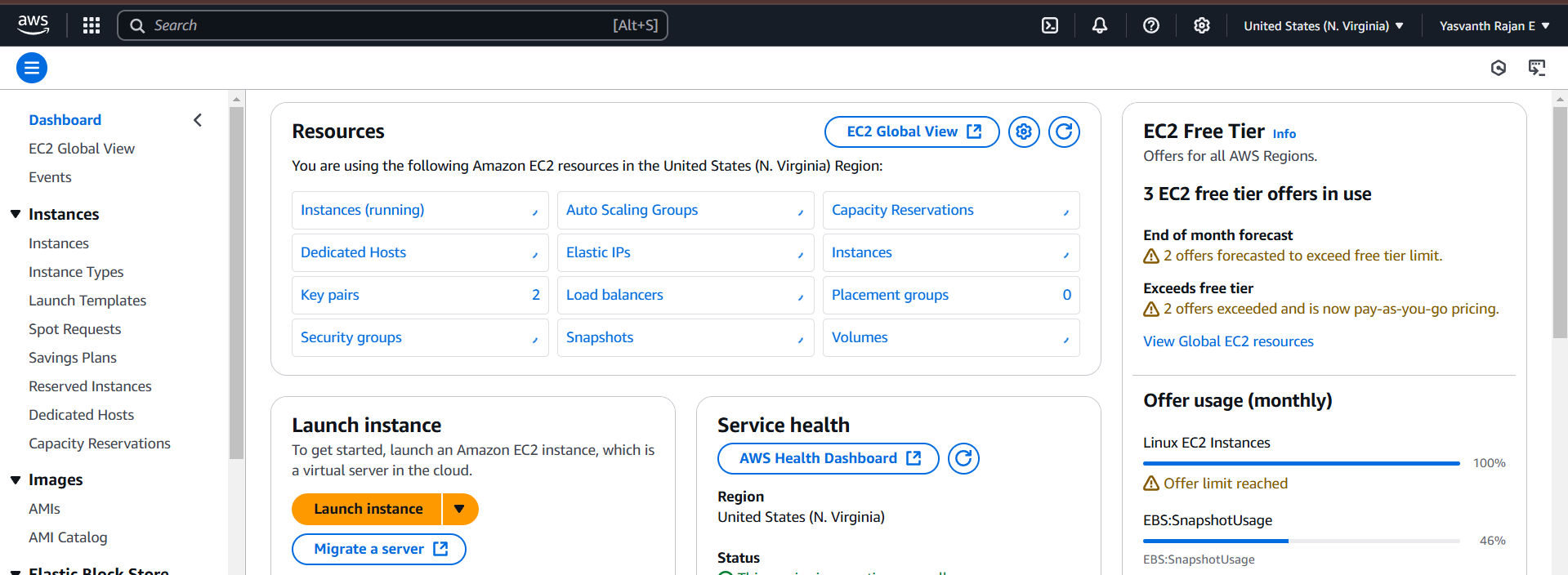
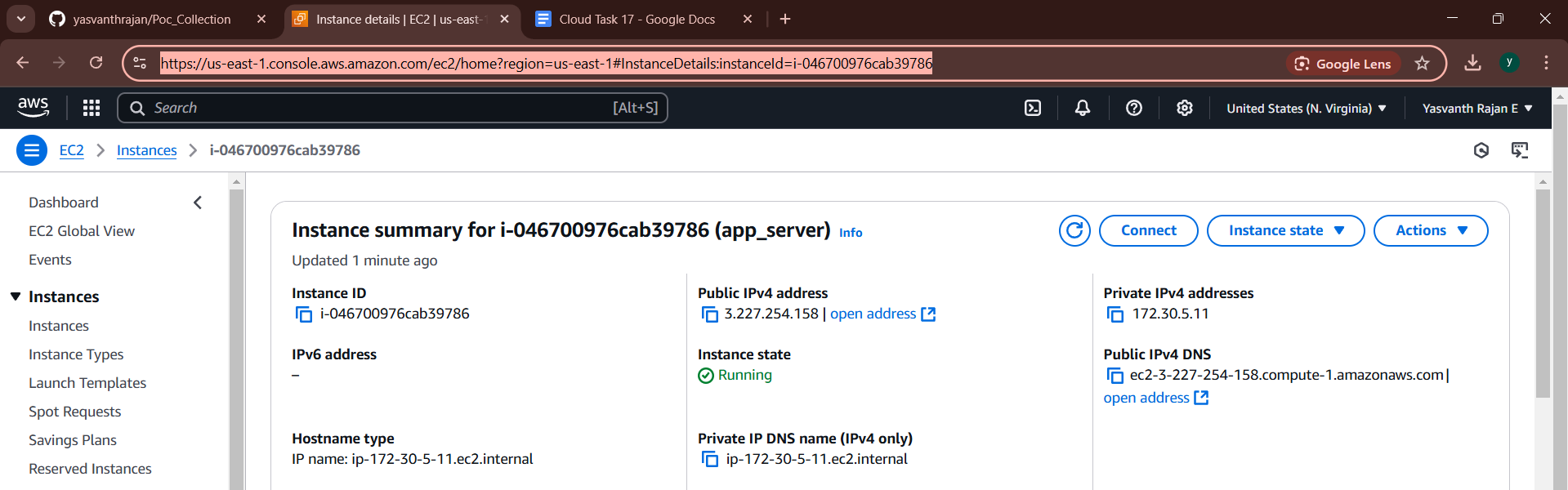
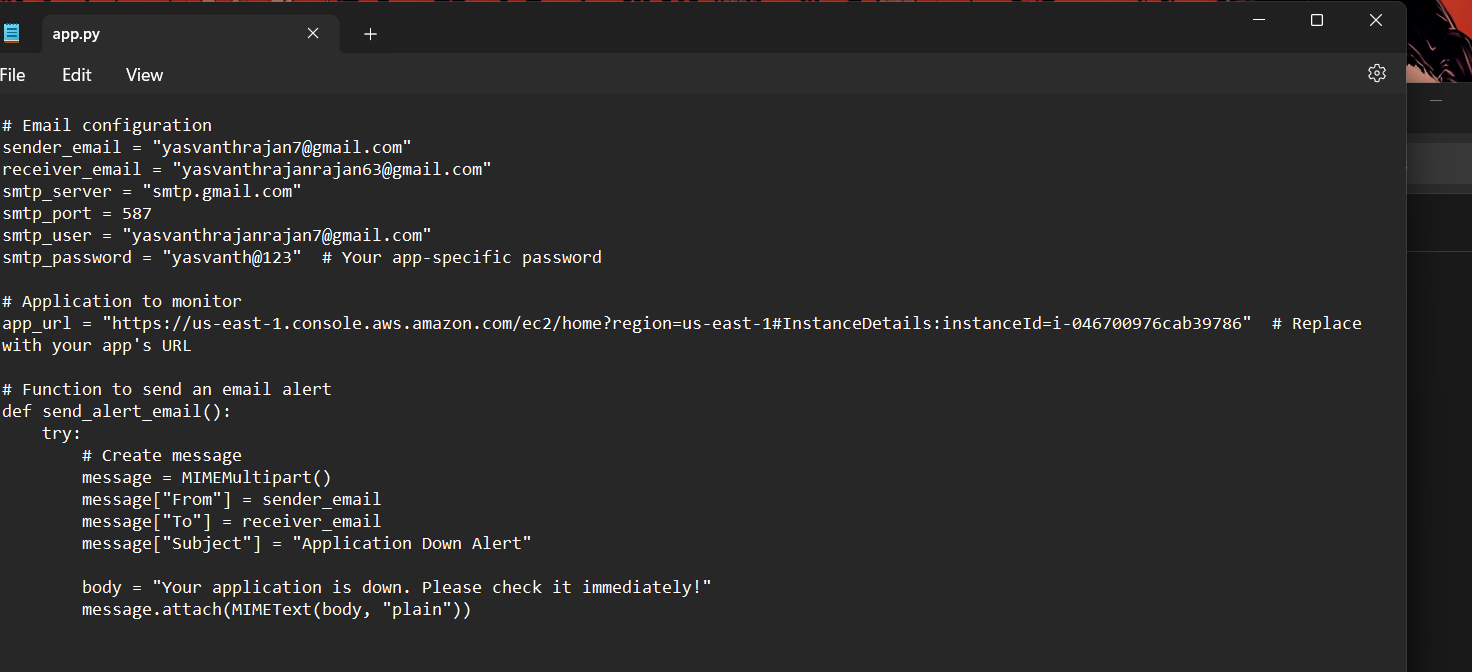
1. In **Command Prompt (CMD)**, type the following command to install the **requests** library:

**pip install requests**

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

#### **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](mailto: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

#### **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

1. Run the script with the following command:

python monitor\_app.py

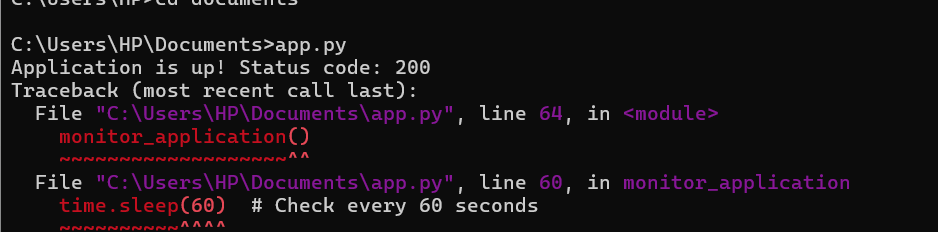
.



#### **Step 7: Stop the Script**

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

.



# 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.