**Documentación del Proyecto: Monitoreo de URLs con Python y JMeter**

**1. Descripción del Proyecto**

**Objetivo:** Este proyecto tiene como objetivo monitorear la disponibilidad y el rendimiento de varias URLs a través de pruebas automáticas de carga utilizando **Apache JMeter** y de validación de estado con **Python**.

El proyecto realiza pruebas de rendimiento sobre las URLs utilizando JMeter, analiza los resultados obtenidos y envía alertas por correo electrónico si alguna URL falla o no cumple con los tiempos de respuesta definidos.

**2. Requisitos del Proyecto**

**Requisitos Técnicos**

* **JMeter:** Utilizado para realizar pruebas de carga y obtener métricas de rendimiento de las URLs.
  + **Requisitos:** Java 8 o superior
  + **Instalación:** [Descargar Apache JMeter](https://jmeter.apache.org/download_jmeter.cgi)
* **Python:** Usado para procesar los resultados generados por JMeter y enviar alertas en caso de fallos.
  + **Requisitos:** Python 3.6 o superior
  + **Librerías necesarias:**
    - requests (para monitoreo de URLs)
    - smtplib (para enviar correos electrónicos)
    - xml.etree.ElementTree (para procesar los resultados de JMeter)
  + **Instalación de librerías en Python:**

bash

Copiar

pip install requests

**Herramientas adicionales**

* **Editor de código:** Cualquier editor de texto que soporte Python (ej. Visual Studio Code, PyCharm).
* **Correo SMTP:** Se debe contar con una cuenta de correo para enviar las alertas (por ejemplo, Gmail, Outlook).

**3. Arquitectura del Proyecto**

**Flujo del Proyecto:**

1. **JMeter** realiza pruebas de carga sobre las URLs especificadas en el archivo de prueba .jmx.
2. Los resultados de las pruebas se guardan en un archivo .jtl.
3. **Python** analiza el archivo .jtl generado por JMeter:
   * Verifica si alguna URL ha fallado (código de error no 200).
   * Valida si el tiempo de respuesta de alguna URL excede el umbral definido.
4. Si se detectan problemas, Python envía un correo de alerta a los administradores.

**Estructura de Archivos:**

bash

Copiar

/ProyectoMonitoreo

│

├── /scripts

│ ├── monitoreo\_urls.py # Script principal en Python para monitoreo

│ └── ejecutar\_jmeter.py # Script Python para ejecutar JMeter

│

├── /jmeter

│ ├── test\_plan.jmx # Archivo de prueba de JMeter

│ └── resultados.jtl # Archivo de resultados de JMeter

│

└── README.md # Documentación del proyecto

**4. Instrucciones de Instalación**

**Paso 1: Instalación de JMeter**

1. Descarga **Apache JMeter** desde su página oficial: [Descargar JMeter](https://jmeter.apache.org/download_jmeter.cgi).
2. Descomprime el archivo descargado en una carpeta de tu elección.
3. Verifica que JMeter funcione correctamente ejecutando jmeter desde la línea de comandos en el directorio bin de JMeter:

bash

Copiar

cd /ruta/a/jmeter/bin

./jmeter # En Linux/MacOS

jmeter.bat # En Windows

**Paso 2: Instalación de Python y librerías**

1. Asegúrate de tener **Python 3.6+** instalado en tu sistema. Puedes descargarlo desde [python.org](https://www.python.org/downloads/).
2. Instala las librerías necesarias para el proyecto:

bash

Copiar

pip install requests smtplib

**Paso 3: Configuración del Correo**

Para enviar alertas por correo electrónico, configura un servidor SMTP en el script de Python. Si usas **Gmail**, configura el siguiente bloque en el código de Python:

python

Copiar

smtp\_server = "smtp.gmail.com"

smtp\_port = 587

smtp\_user = "tu\_usuario@gmail.com"

smtp\_password = "tu\_contraseña\_o\_app\_password"

**5. Uso del Proyecto**

**Paso 1: Configuración del Plan de Prueba en JMeter**

1. Abre **JMeter** y crea un nuevo **Test Plan**.
2. Añade un **Thread Group** para definir la cantidad de usuarios simulados y la duración de las pruebas.
3. Añade un **HTTP Request Sampler** para cada URL que quieras monitorear.
4. Configura un **Response Assertion** para verificar que la respuesta de cada URL sea correcta (por ejemplo, código de respuesta 200).
5. Añade un **Listener** (por ejemplo, **View Results Tree**) para almacenar los resultados de las pruebas en un archivo .jtl.

**Paso 2: Ejecutar JMeter desde Python**

1. Abre un archivo Python y utiliza el script ejecutar\_jmeter.py para ejecutar el archivo de prueba .jmx creado en JMeter:

python

Copiar

import subprocess

jmeter\_path = "/ruta/a/jmeter/bin/jmeter.bat" # Ruta del archivo jmeter.bat

test\_plan = "/ruta/a/test\_plan.jmx" # Ruta del archivo JMeter .jmx

result\_file = "/ruta/a/resultados.jtl" # Ruta del archivo donde se guardarán los resultados

command = [jmeter\_path, "-n", "-t", test\_plan, "-l", result\_file]

subprocess.run(command)

**Paso 3: Procesar los Resultados con Python**

Después de ejecutar JMeter, el archivo de resultados .jtl se guardará en la ubicación que hayas especificado. Usa el script monitoreo\_urls.py para analizar este archivo y enviar alertas si alguna URL ha fallado o si el tiempo de respuesta es excesivo:

python

Copiar

import xml.etree.ElementTree as ET

def analyze\_jmeter\_results(result\_file):

tree = ET.parse(result\_file)

root = tree.getroot()

failed\_requests = 0

total\_requests = 0

for child in root.findall('httpSample'):

total\_requests += 1

response\_code = child.get('responseCode')

response\_time = int(child.get('t'))

if response\_code != '200':

failed\_requests += 1

if response\_time > 2000: # Definir el umbral de tiempo de respuesta

print(f"Advertencia: Respuesta lenta en {child.get('URL')} con tiempo de {response\_time} ms")

print(f"Total de solicitudes: {total\_requests}")

print(f"Solicitudes fallidas: {failed\_requests}")

if failed\_requests > 0:

send\_alert\_email(failed\_requests)

def send\_alert\_email(failed\_requests):

subject = f"Alerta de Prueba JMeter: {failed\_requests} solicitudes fallidas"

body = f"Se detectaron {failed\_requests} solicitudes fallidas durante la prueba."

send\_email(subject, body)

analyze\_jmeter\_results("/ruta/a/resultados.jtl")

**6. Consideraciones Finales**

* **Manejo de Errores:** Es importante monitorear el funcionamiento de las URLs en tiempo real. Si alguna URL comienza a fallar, el sistema de alertas enviará correos para que se pueda tomar acción rápidamente.
* **Automatización:** Si deseas ejecutar estas pruebas de forma periódica, puedes configurar el script de Python en un **trabajo programado** (cron en Linux o el Programador de tareas en Windows) para ejecutarse automáticamente.

**7. Conclusión**

Este proyecto te permite monitorear el rendimiento y la disponibilidad de tus URLs utilizando dos potentes herramientas: **JMeter** para las pruebas de carga y **Python** para el análisis de resultados y envío de alertas. De esta manera, puedes obtener una visión integral del estado de tus servicios web y recibir notificaciones oportunas si algo sale mal.