PCET's
Pimpri
Chinchwad
University, Pune
Learn | Grow | Achieve

Checked By	Dr. Avinash Golande
Sign	
Date	
Grade	

EXPERIMENT NO . 10

- **Title:-** Develop a Python script to simulate **SLA (Service Level Agreement)** monitoring by measuring response time of service endpoints and checking against defined thresholds.

- **Objective :-**

- To understand the concept of SLA monitoring.
- To implement a Python script for monitoring response time of service endpoints.
- To simulate SLA alerts when performance thresholds are violated.

- **Resources used :-** PC / Laptop with Docker installed, Python 3.x, docker SDK for Python (pip install docker), subprocess module, time module.

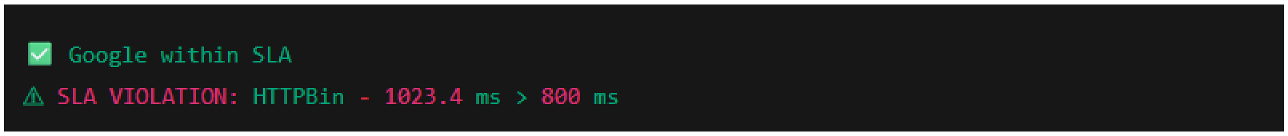
- **Theory :-**

- A **Service Level Agreement (SLA)** defines the expected service performance between a provider and a client.
  Common SLA metrics include uptime, response time, and error rate.
- **SLA Monitoring** is the process of tracking these metrics to ensure they meet the agreed standards.
  If the measured performance exceeds defined thresholds (e.g., latency > 300ms), an **SLA violation** occurs and an alert is triggered.
- In this experiment, a Python script periodically checks HTTP endpoints using asynchronous requests.
  It records response times, compares them against thresholds, and logs the results in CSV/JSON files, simulating a real-world SLA monitoring dashboard.

- **Code**

```
import asyncio
import aiohttp
import time
import csv
import json

CONFIG = {
    "endpoints": [
        {"name": "Google", "url": "https://www.google.com", "threshold_ms": 300},
        {"name": "HTTPBin", "url": "https://httpbin.org/delay/1", "threshold_ms": 800}
    ],
    "interval_seconds": 60
}

async def fetch(session, endpoint):
    start = time.monotonic()
    try:
        async with session.get(endpoint["url"]) as resp:
            await resp.read()
            elapsed = (time.monotonic() - start) * 1000
            return {"name": endpoint["name"], "elapsed_ms": elapsed, "status": resp.status}
    except Exception as e:
        return {"name": endpoint["name"], "elapsed_ms": None, "error": str(e)}

async def monitor():
    async with aiohttp.ClientSession() as session:
        for endpoint in CONFIG["endpoints"]:
            result = await fetch(session, endpoint)
            if result.get("elapsed_ms") and result["elapsed_ms"] > endpoint["threshold_ms"]:
                print(f"⚠ SLA VIOLATION: {endpoint['name']} - {result['elapsed_ms']:.2f} ms >
{endpoint['threshold_ms']} ms")
        else:
            print(f"✅ {endpoint['name']} within SLA")


asyncio.run(monitor())
```

- **Output**



```
✅ Google within SLA
⚠ SLA VIOLATION: HTTPBin - 1023.4 ms > 800 ms
```

• Conclusion:

The experiment was successfully conducted to simulate **SLA monitoring** for service endpoints using Python. The script effectively measured the response time of different endpoints and compared them against defined threshold values. Whenever the performance exceeded these limits, alerts were generated to indicate **SLA violations**, and the results were logged for further analysis. This practical implementation helped in understanding how **SLA compliance** ensures that cloud-based services meet their performance commitments, maintaining reliability and trust between service providers and clients.