Cloud-Based Linux Server Performance Remote Dashboard Project

Project Objective

The primary objective of this collaboritive (optional) project is to design, implement, and evaluate a performance monitoring solution for a remote, cloud-based Linux infrastructure based on number real-world stresses that affect production servers.

Project Requirements

- **Cloud-Based Deployment:** A VPC and Ubuntu or Debian Linux server vm must be deployed to the Google Cloud Platform.
- **Simulated Load:** The utility stress-ng will be deployed in a custom bash script that will alternately generate high loads of differing time intervals and differing classes (ie cpu, io, filesystem, memory, etc). You may optionally select five (5) classes to stress.
- **Systemd:** Create a systemd service for the stress-ng script that will automatically start as a service whenever server is rebooted.
- **API/Agent:** Create an agent and/or API to allow a remote monitoring solution to collect system statistics (ie memory usage, cpu usage, etc). You may choose five (5) parameters to monitor and collect. The API or Agent should be custom designed and can be developed with Bash, Python, or any other language.
- **Named Pipes:** Utilized named pipes in your solution and identify how it is used in your report.
- **Users and Groups:** Create a unique user and/or group assigned to your API/Agent.
- **Remote Monitoring:** Dashboard GUI must remotely monitor performance metrics of Linux server. Dashboard must be custom designed and can be developed with Bash, Python, or any other language. You may use existing existing libraries such as Django, tKinter or plotly. Dashboard may run from your laptop, personal computer or another GCP virtual machine.
- **Key Performance Metrics:** Monitor essential metrics (ie CPU, memory, disk, network – total of five (5)).
- **Data Visualization:** Dashboard must visually represent performance data using appropriate charts and graphs (real-time **and historical data**).

- **Data Collection Automation:** Data collection from the monitored server(s) must be automated (scripted) by starting app on laptop.
- **Notification System:** based on performance thresholds. Provide user adjustable thresholds to notify user if threshold is surpassed. This can be a message box or color change in the dashboard.
- **Security:** Implement a firewall, iptables, or VPC configuration to limit agent/api access to a single IP address.
- **Failure Handling:** Implement appropriate failure handling mechanisms to monitor operation of agent or api to restart automatically. Use a cron job to check their status every 5 minutes.
- **Version Control:** Use a Git or Github repository to manage version control, development branches and collaboration. All team members will have access to repository and make meaningful updates to the code base reflected by commits.
- **Multiple Servers:** If working as a group, each member must be responsible for a cloud based server deployment. If two students work together, then there shall be two servers and one dashboard. For three students, then have three servers and one dashboard.

Project Deliverables

- **Report:** Submit a modest project report in doc or pdf, that includes:
  - Cover page that includes name of project, github url, date, course section and each member's name
  - Introduction and project objectives (1 paragraph)
  - System architecture and design (2-3 paragraphs)
  - Implementation details (2-3 paragraphs)
  - Experimental results and analysis presented in captioned diagrams, graphs and screenshots from dashboard. Each should be supported with 1 – 2 sentences each.
  - Problems faced (1 -2 paragraphs)
  - What was learned (1 – 2 paragraphs from each team member)
  - Individual contribution table (displays activity type and percentage of effort for each team member)

  - Conclusions and future work (1-2 paragraphs)

- **Code:** Submit well-commented and organized source code for the stress script, agent or api, cron job, and dashboard.
- **Git log:** Git log from git repository that demonstrates multiple commits and activity from **all** the team members.
- **Video operation:** Prepare a video (narration not necessary)
  - **Display the dashboard in action. 1 minute**
  - Manually stop the api or agent to demonstrate automatic restart.
- **Live Test:** Professor will review each student/group to view a live test of your dashboard. Date will be after due date.

Additional Notes:

- Do not use an off-the shelf monitoring system like Nagios, Splunk, or GCP's vm monitoring tools. Develop your own data collection tools.
- You may use libraries or frameworks to assist in developing a remote dashboard.

Reference:

https://wiki.ubuntu.com/Kernel/Reference/stress-ng