"Kurama"

A PaaS to deploy flask application and profiling

A project documentation
submitted in partial fulfillment of the Project requirements for

CLOUD COMPUTING
SS G527
in
Master of Engineering (Software Systems)
By

Shashank S (2019H1120054P) (Group ID : 12)

Ashirwad Pradhan (2019H1120043P) (Group ID : 12)

Suprakash Mukherjee (2019H1120048P) (Group ID : 12)

Pallav Mahamana (2019H1120052P) (Group ID : 12)

Under the supervision of

Dr. K Hari Babu

Assistant Professor
Department of Computer Science and Information Systems

Birla Institute of Technology and Science

Pilani, Pilani Campus, Rajasthan (India)

November, 2019

Terminology:

Kurama : A PaaS to deploy flask application and profiling.

Application : Refers to Application Deployed on the Kurama Service.

1.0 Project Overview

1.1 Problem Domain

A user should be able to deploy flask applications on the *Kurama* service. The user shall be able to generate logs by using a library package provided by the *Kurama* Service. The logs generated will be used as a streaming data. The logs generated in the form of streaming data will be used for profiling services provided by the *Kurama* Platform.

1.2 Anatomy of the Log Data Stream

The data stream of consist of

1. Timestamp of the log that is generated.
2. Log level.
3. Request URL.
4. Response Code.
5. Log Message.
6. Used CPU percentage.
7. Used Disk percentage.
8. Used Swap percentage.
9. Used Virtual Memory percentage.
10. Log UUID.

1.3 Service Provided

1. Send Log data to PostgrseSQL for persistence.

   It is a microservice which reads the streaming log data and stores it in the Kurama database for persistence.

   Input : Log stream terminated by '\n' character.

   Output : Log stream stored in Kurama DB.

2. Alert Service.

   This service provides critical alerts based on the conditions for the resources set by the user. Upon meeting the conditions set by the user an alert email goes to the destination address provided during pipeline configuration.

   Input: Log stream terminated by '\n' character.

   Output: An Alert mail is sent to destination address provided during pipeline configuration.

3. Serializing Service.

   It is a microservice which serializes the streaming log data into JSON file which can be retrieved by the user at a later point in time for further analysis.

   Input: Log stream terminated by '\n' character.

   Output: JSON file.

4. Dashboard Service

   It is a microservice which provides real time analysis and visualization of the streaming log data. This service helps in identifying the performance of the application deployed by the user in terms of the critical system resources used by the application.

   Input : Log stream terminated by '\n' character.

   Output: Graphical dashboard service providing real time visualization of streaming data.

5. Infrastructure.

   All the microservices provided by the Kurama platform will directly run on physical machines which is easy to scale.

   The users application deployed will run on container like environment which isolates each user application from each other.

6. Auto-Scaling.

   Auto-Scaling happens based on the following conditions :

   - When the 95th percentile of the average system load calculated over a period of 5minutes reaches more than 85 percent, then the system will scale up. And if it falls down below 50 percent then the system will scale down.
   - When the 95th percentile of the response time of each request calculated over a period of 10 minutes reaches greater than 200 milliseconds then the service will scale up.

2.0 User Manual

All the configurations of the project is maintained in a file “kuramaconfig.yaml”.

1. Run all the microservices using “python3 <microservice name>.py”
2. The interface is available at port : 5001 and the dashboard is available at port : 5005
3. To simulate the data stream run the file “reqsim.py”