Application Health Analysis

Akhil Reddy Mandadi (akhilreddy.mandadi@sjsu.edu)
Vamsi Mundra (vamsi.mundra@sjsu.edu)
Jagan Gadamsetty (jagan.gadamsetty@sjsu.edu)
Srinivas Rao Samudrala (srinivasrao.samudrala@sjsu.edu)

***Abstract*: All Major Enterprise Applications comprises of different set of inter dependent micro services, and there will be many inter dependent micro services and their API's which will be failing due to many issues. And debugging and troubleshooting the issues will be very difficult during deployment and production phases where it is difficult to track the failures among the comprising API's. Applications Health Analysis will give the detailed report of the API's which are failing frequently and the dependent API's which are getting effected by the failed API's. This lets the organizations monitor their deployments at a Microservice and API level where they get to see the live status of how their applications are performing. This will ease the process of identifying the issues mainly in the production environments where a single failure of an API will cause a feature to break. We can also track the modules of the product which are failing frequently and take necessary actions on it and testing them thoroughly before releasing them.**

Introduction

In the current software development model, any major application is built using many microservices which are dependent on each other for the complete success flow of the application. Many of the micro services may be down due to various reasons and which might affect other micro services and the flow of application. Many developers are working hard to know about which microservices are getting failed, to check that they must go through GB's of log data and check which one is down and what is the reason for the failure. This will lead to a waste of numerous of man hours in identifying the issue in the application going through different logs and it is difficult in viewing the history of the logs if needed to be checked at a later point in time and browsing through the logs is not an easy task as there are different kind of things which will be logged like info an debug logs.

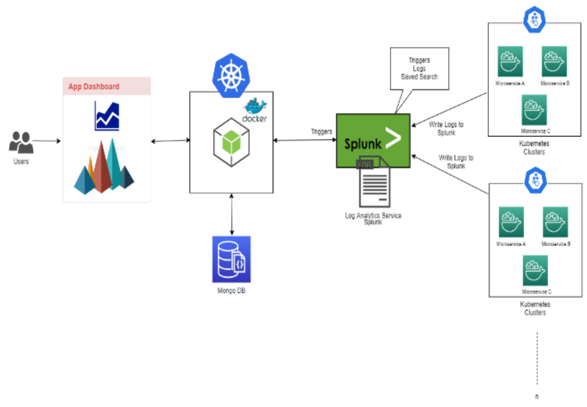
There are different solutions to the approach like using the external logging services like Splunk or Graylog where the developers will log the failures and there are services like Kibana which provide log analytics services but they are not linked to any micro services and with the current architectures where there are hundreds of microservices running for each application and different services writing logs to the same logging service and its difficult to debug the issues and difficult to identify the API's which are failing frequently.

Proposed Work

With the number of applications and cloud deployments growing, comes the problem of log management and a tool to effectively analyze the logs and provide insights to the developers will be of great advantage. Applications Health Analysis solves this problem by providing a detailed report of the Microservices and API's which are failing frequently and the dependent API's which are getting effected by the failed API's. Organizations can monitor the live status of the Microservices and the associated failures. This lets the organizations monitor their deployments at a Microservice and API level where they get to see the live status of how their applications are performing. This will ease the process of identifying the issues mainly in the production environments where a single failure of an API will cause a feature to break. We can also track the modules of the product which are failing frequently and take necessary actions on it and testing them thoroughly before releasing them. This will also provide a prediction of how the API might behave and how many failures might occur at an API and Microservice level in the coming days based on the previous failure history. We leverage the logging capabilities of Splunk to fetch the logs and perform analysis on them.

Application Health Analysis application lets Organizations onboard their Applications into our Application where the Onboarding process involve registering of its microservices, each microservice interdependency with other microservices, the Splunk instance details where the application will write its logs to. Once this process is done, the enterprise can continuously monitor the application's health in the dashboard. Our application includes features like real time microservice health monitoring, microservice health history, API failure history at a microservice level, Gmail notification for API failures, prediction on how many times the microservice and API's are going to fail in next few days. This helps the enterprise understand health of the application well and lets them come up with a solution in advance.

Architecture Design



1. System Architecture

Enterprises are used to write logs to Splunk for status checks in later time, our application uses these logs in Splunk and detects whenever an API fails in a microservice using Splunk SDK and tracks the failure and success of that API by leveraging the alert capabilities of Splunk which enables users to track the health of an application. The frontend is built on React JS, its one-way dataflow makes the application lighter which in turn makes data render smoothly. The server-side computing is built on Nodejs as it offers high performance. Our application's data is stored on MongoDB, which can be easily scaled. Our entire application runs as Docker containers and is deployed on the Cloud. The log analytics service used in our application is Splunk which is used by many multinational enterprises like AT&T, Wells Fargo, Apple, JP Morgan etc. Splunk offer services like Triggers, saved search, Logs and Prediction algorithm for analyzing and monitor the health of a microservice in later time.

Implementation

During the Onboarding process where the enterprise provides the details about the micro services and its dependencies we create an Alert in Splunk for each microservice which continuously monitors the logs of that specific microservice and a webhook is configured which will be hit by Splunk whenever it finds a failure log logged by the application.

Whenever an API of a micro service is failed, Splunk hits the given web hook API based on the alert of the micro service which was created at the time of onboard. The failure alert sends the complete log data which was written by developers to Splunk. Our application parses the complete log data and gets the appropriate information from the logs like, which microservice is down, from how much time the microservice is in the current state and what is the reason for the failure. It sends a notification to the application owner about the micro service which is down and complete log data. Our application then creates an alert to the Splunk which searches for the success of the API which got failed. Our application creates real time saved searches in Splunk which constantly searches for the success of a failed API. When the success log is created for the failed API, the web hook which was given at the time of saved search creation will be called. Our application then receives the complete data of the success API. We parse the complete data and checks which API is successful. We change the current failed state of the API in our data to success and will store the complete failure history. We then remove the saved search created in the Splunk. In this cyclic process we store the complete important log information about the success and failures of the failed API of different microservices in an application which will be useful to the developer to know what are the current microservices which are down and their dependent microservices which are getting effected.

Coming to the User interface walkthrough of our application, the Applications tab takes the user to the applications dashboard where the high-level overview of his on-boarded applications can be viewed. Users can view the current health percentage of all the onboarded applications where-in the health percentage of each application is shown. The bar graph in the applications dashboard shows the count breakup of running and failed microservices for each application on-boarded by the user. The user can view the applications onboarded by him in the applications dashboard and search for a specific application if required and filter applications based on the health percentage using the health range slider. The user can see list of few microservices related to an application on the application card and the user can further know the total microservices present in an application and their individual running status by clicking on the application in the applications dashboard. The user will be able to visualize the total microservices in an application and their current running status. The user will get to know if there are any dependencies for the failed microservices and view the history of any microservice. The live status of the microservice can viewed and the current failures if any and the complete failure history of the all the API of each micro service.

Our application has the capability where we provide the prediction of how the API or an Microservice will perform in the coming days' time where we provide the failure prediction for the next five days. We have leveraged the machine learning capabilities provided by Splunk to predict the failures of the microservices related to an application for the next five days based on its current working status and how it performed in the past by analyzing the logs.

Conclusion

Our Application aims to solve the problem of health monitoring of the application in a consolidated manner from a single dashboard and let the users monitor the health of all the microservices and also look at the API history and the failure logs associated and also provides a prediction service which provides with a prediction of how the Microservice will fare in the coming days by performing the log analysis.