Business Context Aware Data Center Monitoring

Amala Chirayil
amala.chirayil@sjsu.edu
(https://github.com/amalachirayil/CS274_Project)

***Abstract* – In today's fast-paced and technology driven world, Internet commerce has become a de-facto standard. Nowadays there are a lot of options available for a customer over the Internet. In order to be a successful seller online, the provider should provide great customer experience.**

Shopping experience via the Internet requires applications to run with agility and performance. Moreover, the applications should be consistently performing with 100% uptime. Any downtime can cause heavy loss as purchases happen at a time that is convenient to the customer 24 hours 7 days a week.

To ensure that the applications that support business processes run continuously, constant monitoring of the environment in which the application is running needs to be monitored. This monitoring has to be proactive and should be able to predict chances of failures and bottlenecks that may affect the business process, hence predictive monitoring.

Therefore, in this paper, we are going to be examining the importance of predictive analytics on data center monitoring by discussing statistical-based and machine learning models that can be used to learn from system performance metric data. In addition, we are going to look into how we can use Elasticsearch, Beats, and Kibana to collect, analyze, and display data.

Index Terms: Data Center Monitoring, Predictive Analytics, Logistic Regression, Log File, System Performance Metrics

I. INTRODUCTION

The evolution of the Internet has paved the way for numerous businesses of varied types and sizes to be a success in today's fast-paced and technology-driven world. Even though the Internet, originally named Arpanet, started off as a research project funded by the U.S. military [1], it "has expanded beyond the United States to every corner of the globe" [1], giving consumers access to a wide variety of information right at their fingertips.

In addition, the Internet also provided and continues to be a platform for many businesses to conduct their business processes online in order to make profit. This Internet facility is commonly known as ecommerce. According to an article provided on the Salesforce website, ecommerce, also known as electronic commerce, "refers to the process of conducting transactions through the Internet" [2]. In order to utilize this facility, businesses develop ecommerce applications that support their business processes. Automation of these business processes and steps are enabled through multiple technologies. These technologies run on either privately owned data centers or on cloud data centers.

Nowadays, with easy access to the Internet and most businesses selling products and services online, people prefer shopping online rather than take the time to go to a store. For example, Black Friday and Cyber Monday are the two most popular days in the United States where people do an excessive amount of shopping to prepare for the holiday season. In fact, "consumers spent a total of $12.8 billion online in the U.S. during the five-day period from Thanksgiving through Cyber Monday in 2016" [3].

In order to support such high online traffic, businesses must be able to keep their ecommerce application continuously running by proactively monitoring the data center in which their application is deployed in. This is an important and essential task for any business that doesn't want to lose money and instead wants to ensure that the technologies used to build an application are scalable and resilient. In fact, according to a white paper written by Emerson, "the average cost of data center downtime was approximately $5,600 per minute" [4] based on a study conducted in 2011. Network failure, security, power failure, and lack of scalability [5] are the few issues that can arise in the data center and thus cause an application to stop working costing businesses hundreds of thousands of dollars.

The paper is structured as follows. Section II provides background information on the importance of data center monitoring and sets the stage for the rest of the paper. Section III then provides a description of my project and relevant background information associated with it. Section IV dives deeper into the implementation details of the project. Subsequently, Section V provides the results of the implementation. Lastly, Section VI wraps up my project and I also discuss the future work that could be implemented by using my project as its starting foundation.

II. BACKGROUND INFORMATION

A. Setting the Stage: Scenario

Imagine yourself as an entrepreneur and you have an online business where you are selling items to customers. Irrespective of the items that you are selling, you have developed an ecommerce application for your business and this application provides certain facilities to ensure great customer experience. As illustrated in Figure 1, your application provides certain

Figure 1: Your **ecommerce application** provides certain facilities to ensure great customer experience

facilities such as allowing a customer to search for a product, add the product to a shopping cart, make a payment, and finally place an order.

Now, assume that you have installed your ecommerce application on a set of nodes in a large data center. Within these set of nodes, you might also have other applications running and other applications may be coming in ready to execute. In order for your application to run seamlessly, it requires a certain amount of compute and storage. There is a high chance that your application will not require 100% of the compute and storage resources all the time which is why the computers' resources are being shared across many applications. So, over a period of time, you can collect system log information in order to analyze the patterns of compute and storage usage. For example, during the holiday season, your application may get a large number of hits so during this time your application requires a high amount of compute and storage power. During the off-season, your application may get hits, but may not require as much compute and storage power than during the peak season. Then, the question that arises is that, during these hits, how will you be able to ensure that these resources will be available all the time. This is the problem that I am trying to address with my project.

For this purpose, we need to proactively monitor data center resources and not after the fact. For instance, suppose your application is being used by hundreds of customers and in a split-second your application stops working. Now that you've realized your application has stopped working, you go to address the problem, but that is after the fact and in that time, you have already lost customers and money. However, if you had already known that this was going to happen in the near future, you could have proactively performed some maintenance to prevent this issue from occurring.

B. Related Work

My project's foundation was inspired by my internship experience at a large corporation and built after looking into different research papers, journal articles, and products on data center monitoring and log file analysis. AppDynamics is one product that exists in the market today [6]. AppDynamics is a leading Application Performance Management (APM) tool that monitors your application infrastructure and provides code level visibility. As showcased in Figure 2, a software called an Agent is installed on Application Servers and is responsible for collecting system performance metrics. These metrics are then sent to a Controller server where data is processed. An end user, such as a data analyst, can view the data through a web interface to examine what's going on behind the scenes. As a brief summary, this product "learns" application behavior and automatically sets baselines and alerts when the deviation from the baseline is an anomaly [6].

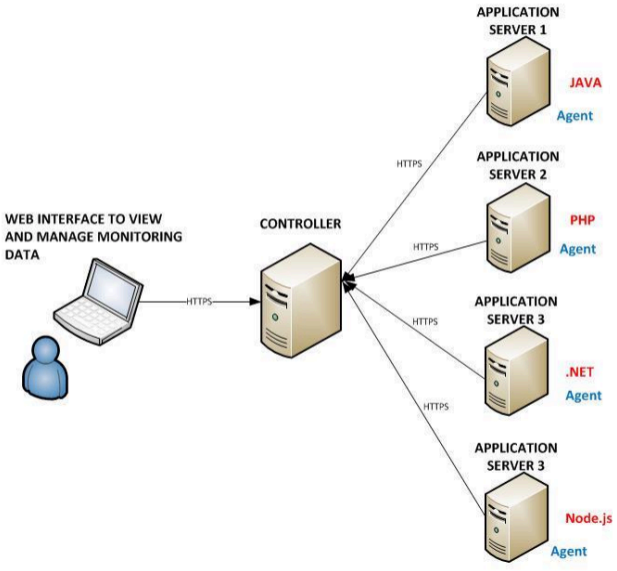


Figure 2: AppDynamics APM tool

III. Project Overview

A. Problem Statement

As mentioned in the previous section, the problem that I am trying to address with my project is business process

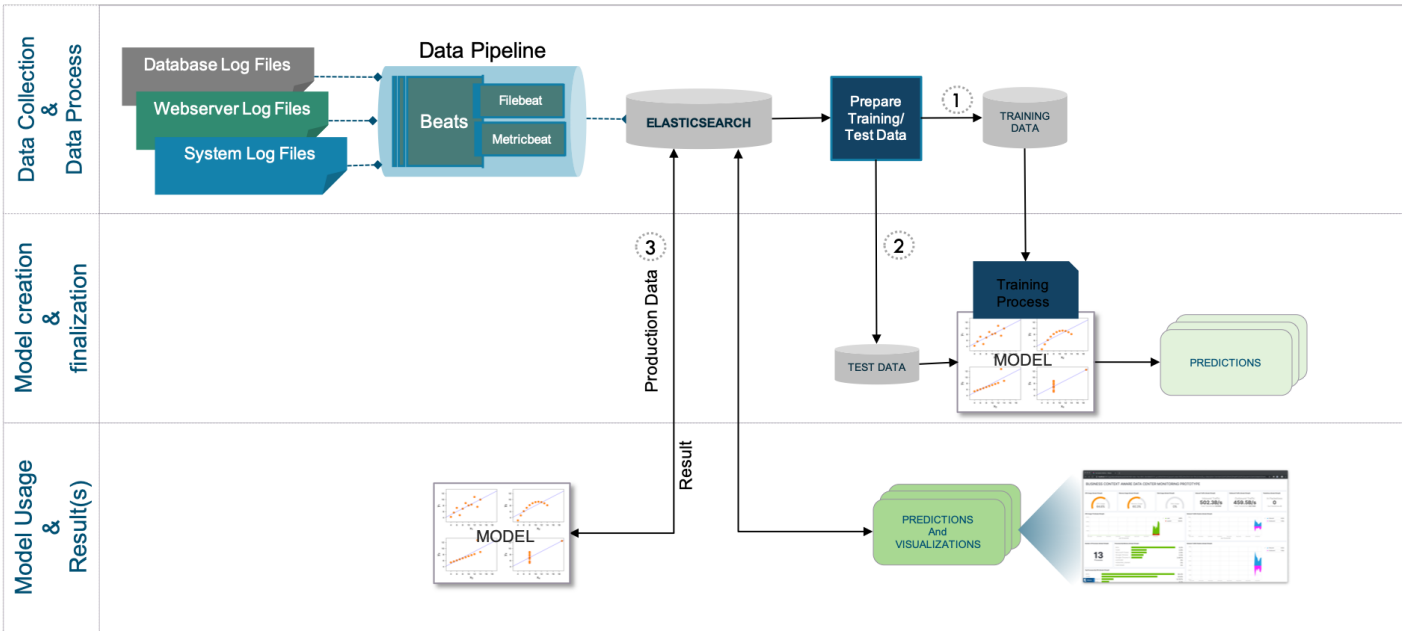


Figure 3: Architecture Diagram of Proposed Solution

interruption caused by inadequate management of compute, storage, and network resources in a data center. This problem affects businesses that are dependent on online applications for conducting their business processes. It's important to address this problem because ensuring scalability and resilience of applications provides great experience for the customer and avoids loss of revenue for the business.

B. Proposed Solution

We will be addressing the problem mentioned before by proactively monitoring data centers to ensure timely alerts/automated actions to scale compute, storage, and network resources in a data center. For this purpose, as depicted in Figure 3, we need to collect and analyze system and database log data so that we can learn from available memory, CPU, disk space, and network traffic in order to predict what kind of situations can arise based on the usage of compute and storage resources so that appropriate maintenance *(e.g. add more virtual machines if it's in a cloud environment)* can be done before the application stops running.

After the system data is collected with the help of two Beats modules (Filebeat and Metricbeat), which are "lightweight data shippers" that are used to capture operational data, this data is stored in Elasticsearch, which is an "open source search and analytics engine for all types of data" [7]. From Elasticsearch, the necessary data is extracted with the help of APIs in order to prepare training and testing datasets for the machine learning models. To find data center resources, such as compute and storage, and their effects on resource contention that may lead to non-performance of the application, a classifier model was deemed appropriate. The two machine learning models that I decided to use was a standard logistic regression model and a multinomial logistic regression model.

Lastly, predictions are captured on a visually appealing dashboard with the help of Kibana so that timely alerts/automated actions can take place.

C. Machine Learning Models

In the classification-based methodology, a dataset, known specifically as a training dataset, is used to form classification models (classifiers). Then, using these classification models, test instances are categorized within these classification models. The key idea in this methodology is that a classifier can be learnt from a given data set and easily distinguish between two or more classes. Standard and multinomial logistic regression models are the two classification and statistical based methods [8,9] that I used for my project.

i. Standard Logistic Regression Model: A standard logistic regression model is a statistical model that is used within the machine learning domain in order to solve binary classification problems. This model predicts the probability of an occurrence utilizing a logit function, which is defined as $f(x) = 1/1+e^{-x}$. Maximum Likelihood Estimation (MLE) is used for estimating the parameters of a model which results in the following logistic regression equation: $y = e^{(b0+b1*x)}/(1 + e^{b0 + b1*x})$, where y is the predicted output, b0 is the bias, b1 is the coefficient for a single input value x. If more than one input value x is being used, logistic regression model will estimate a coefficient for each input.

ii. Multinomial Logistic Regression Model: A multinomial logistic regression model is similar to a standard logistic regression model, but the latter model is utilized in situations where the dependent variable is nominal with more than two levels. It is used to explain the relationship between one or

more independent variables and one multi-class dependent variable.

IV. Implementation

To further understand the full capacity and potential of predictive analytics on data center monitoring, I decided to implement two of the statistical-based methods discussed in the previous section. However, prior to training and testing these statistical-based methods, I had to install and learn how to use Elasticsearch, Beats, and Kibana in order to collect, extract, and display the required system information. Please note a larger image size of each image in this section can be found in the Appendix.

A. *Data Collection: Elasticsearch, Kibana, Beats*

After installing Elasticsearch, Kibana, Filebeat and Metricbeat from the Elastic website, I started the process of collecting my local system performance metric data. The first step was to start Elasticsearch by executing the command **"./bin/elasticsearch**" from the appropriate directory in terminal, as showcased in Figure 4a. Figure 4b represents the output seen after executing this command. In order to ensure that Elasticsearch was running, I navigated to https://localhost:9200 to see if the message in Figure 4c appears.

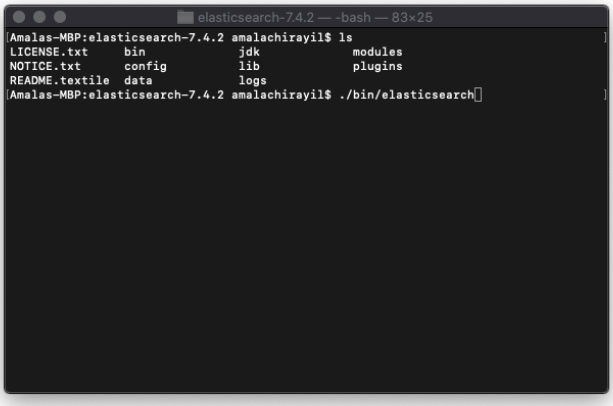


Figure 4a: Starting Elasticsearch

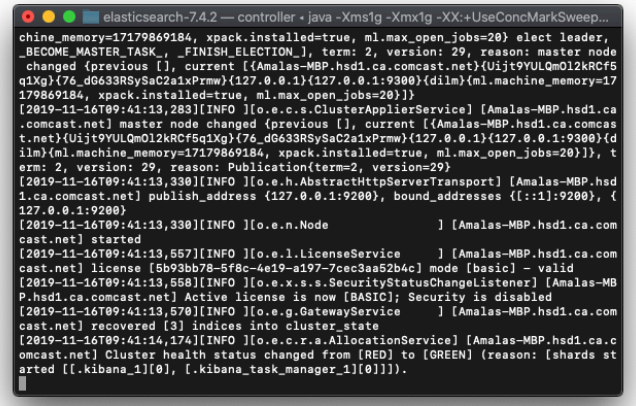


Figure 4b: Output while starting Elasticsearch

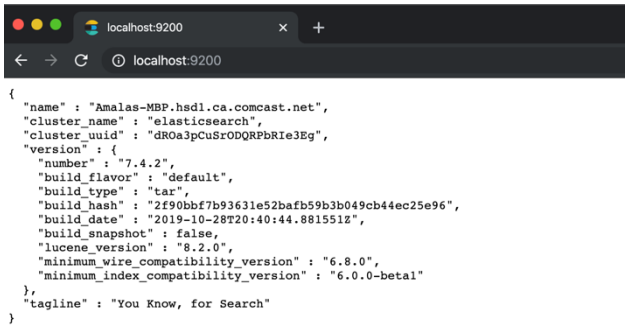


Figure 4c: Verification that Elasticsearch is running

The next step was to start Kibana by executing a similar command to the one used to start Elasticsearch, "**./bin/kibana**" from the appropriate directory in a new terminal, as showcased in Figure 5a. Figure 5b represents the output seen after executing this command. In order to ensure that Kibana was running and a connection was established with Elasticsearch, I navigated to https://localhost:5601 to see if I got a similar output to that of Figure 5c.

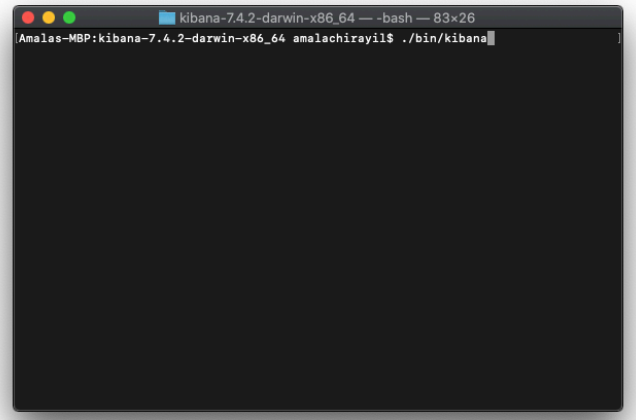


Figure 5a: Starting Kibana

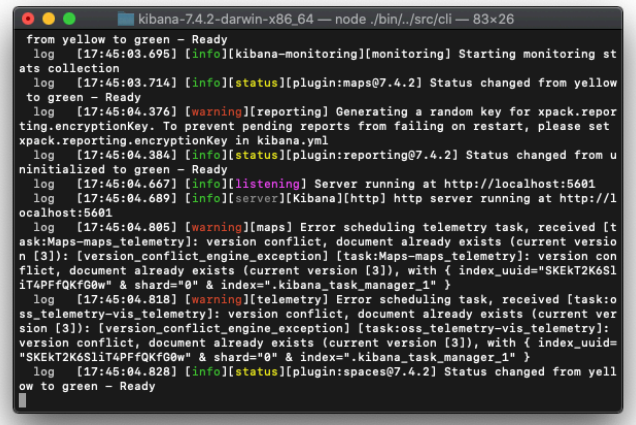


Figure 5b: Output while starting Kibana

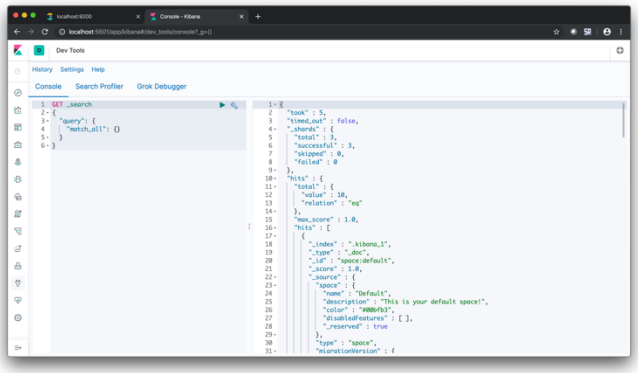


Figure 5c: Verification that Kibana is running

As depicted in Figure 3, the Metricbeat module was used to collect system performance metric data from my local system. Similar to Elasticsearch and Kibana, the command "**./metricbeat -e**" was executed from the appropriate directory in a new terminal, as showcased in Figure 6a. Figure 6b represents the output seen after executing this command. Once a connection to Elasticsearch gets established, an inverted index is created and stores all of the system data based on the fields defined in Metricbeat module [7]. Figure 6c represents all the available and newly created inverted indices that are present. Please note that the inverted index titled "**metricbeat-7.4.2-2019.11.17-000001**" is the index that contains my local system performance metric data.

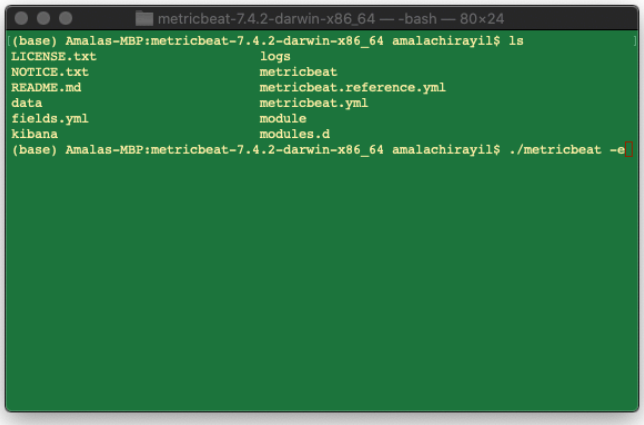


Figure 6a: Starting Metricbeat

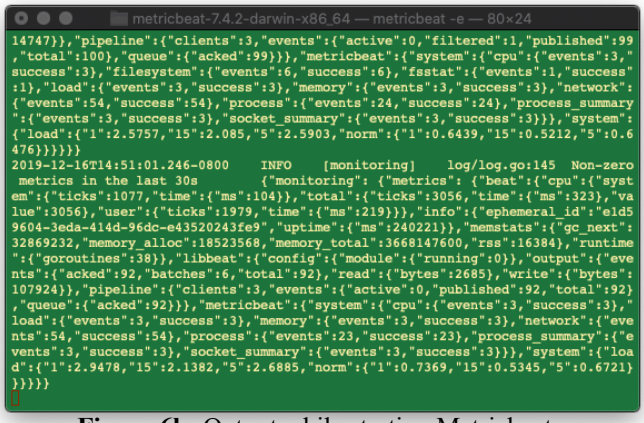


Figure 6b: Output while starting Metricbeat

```
green  open .kibana_task_manager_1            SKEkT2K6SliT4PFfQKfG0w 1 0       2 2  43.4kb  43.4kb
green  open .apm-agent-configuration          rC9LWFWFTZO97lfpM4uC8w 1 0       0 0    283b    283b
yellow open metricbeat-7.4.2-2019.11.17-000001 YSExueeJQw2LCKfKkxwRMA 1 1  395593 0  93.5mb  93.5mb
yellow open filebeat-7.4.2-2019.11.16-000001  4XFkqIuzT9O0Dyne8Dd46g 1 1 49014486 0   6.6gb   6.6gb
green  open .kibana_1                         AuPVvK2kT1CBe0cS0nI9bg 1 0       57 6 320.7kb 320.7kb
```

Figure 6c: Inverted indices

In addition to the Metricbeat module, the Filebeat module was used to collect log data from MongoDB which was utilized in an ecommerce application. Since the focus of this project was not to create an ecommerce application, I used an application that was developed using the MERN stack by Rizwan Khan (https://github/com/Rizwan17/mystore-front-end and https://github.com/Rizwan17/mystore-back-end) to collect MongoDB log data. In order to be able to collect database log data, the MongoDB module specified within Filebeat had to be enabled as showcased in Figure 7a. The command "**./filebeat -c filebeat.yaml -e**" was executed from the appropriate directory in a new terminal. Figure 7b represents the output seen after executing this command. Once a connection to Elasticsearch gets established, an inverted index is created and stores all of MongoDB log data based on the fields defined in the Filebeat module [7]. The title of the inverted index that stores the MongoDB log data is "**filebeat-7.4.2-2019.11.16-000001**" as can be seen in Figure 6c.

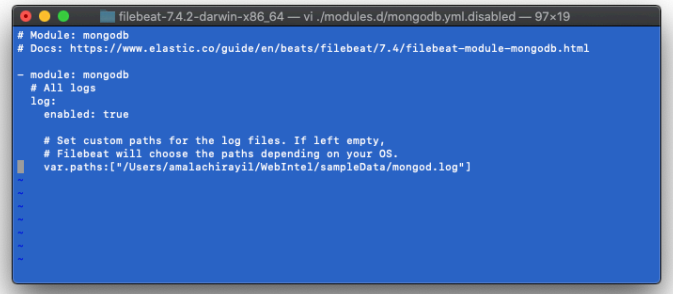


Figure 7a: Configuring MongoDB module within Filebeat

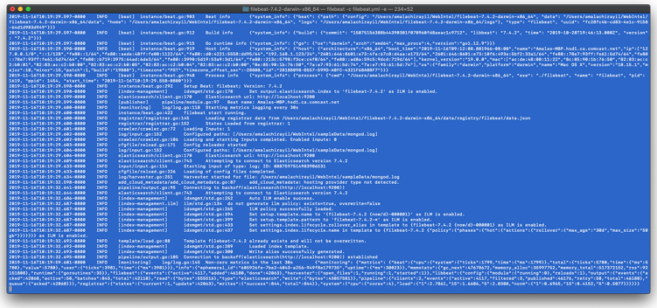


Figure 7b: Output while starting filebeat

B. Implementation of Machine Learning Models

i. Standard Logistic Regression Model: The standard logistic regression model was trained on a dataset that consisted of performance metrics of 1,750 virtual machines from GWA-T-12 Bitbrains distributed data center [10]. Please refer to Table 1 to see the format of this data.

Using Python, pandas, numpy, seaborn, and functions from the sklearn library, I trained two

standard logistic regression models, one for CPU usage and another one for Memory usage. After training these two models, I tested them using performance metrics from my local system to exhibit and simulate how predictive analytics would work in real-time. Table 2 displays the format of the data captured by MetricBeat of my local system. The code can be found on my GitHub page and is titled as "CPU Logistic Regression Model" and "Memory Logistic Regression Model" for the CPU model and memory model, respectively.

Name	Description
Timestamp	number of milliseconds since 1970-01-01
CPU cores	Number of virtual CPU cores provisioned
CPU capacity	The capacity of the CPUs in terms of MHz
CPU usage	In terms of MHz
CPU usage	In terms of percentage
Memory provisioned	Capacity of the memory in the VM in terms of KB
Memory usage	Memory that is actively used in terms of KB
Disk read throughput	In terms of KB/s
Disk write throughput	In terms of KB/S
Network received throughput	In terms of KB/s
Network transmitted throughput	In terms of KB/s

Table 1: GWA-T-12 Bitbrains Dataset Format

Name	Description
Timestamp	YYYY-MM-DD
system.cpu.cores	Number of CPU cores provisioned
system.cpu.user.pct	In terms of percentage
system.memory.total	In terms of bytes
system.memory.actual.free	In terms of bytes
system.diskio.read.count	Total number of reads completed successfully
system.diskio.write.count	Total number of writes completed successfully

Table 2: Local System Metric Dataset Format

ii. Multinomial Logistic Regression Model: The multinomial logistic regression model was trained on a dataset that consisted of the performance metrics of a MongoDB database from a staging environment and data collected from an ecommerce application that I used.

Using Python, pandas, numpy, seaborn, and functions from the sklearn library, I trained a single multinomial logistic regression model. This model was chosen because the target variable can be one of 3 classes (tech stack is stressed, tech stack is about to be stressed, tech stack not stressed) and severity level of the mongo log has 4 labels. I split this dataset into 75% training dataset and 25% testing dataset. Table 3 showcases the format for this data. The code can be found on my GitHub page and is titled as "Multinomial Logistic Regression Model to Determine Stress Level of Tech Stack."

Name	Description
Timestamp	YYYY-MM-DD:HH:mm:ss
Severity	4 types (F-Fatal, E-Error, W-Warning, I-Informational)
Component	E.g. NETWORK, ACCESS
Context	E.g. initandlisten
Message	E.g. waiting for connections on port 2701.

Table 3: MongoDB Log Dataset Format

V. Results

A. *Memory Model*

The memory model was trained on 500 data records from GWA-T-12 Bitbrains and tested on 100 data records from my local system log. As showcased in the confusion matrix below, in Figure 8, the model was only able to achieve 28% accuracy.

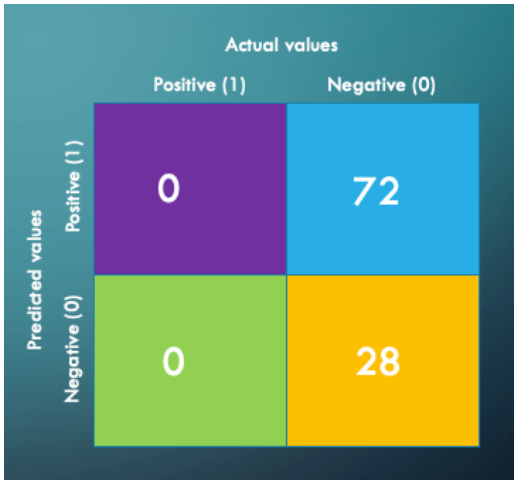


Figure 8: Memory Model Confusion Matrix

B. *CPU Model*

The CPU model was trained on 500 data records from GWA-T-12 Bitbrains and tested on 100 data records from my local system log. As showcased in the confusion matrix below, in Figure 9, the model was able to achieve 98% accuracy.

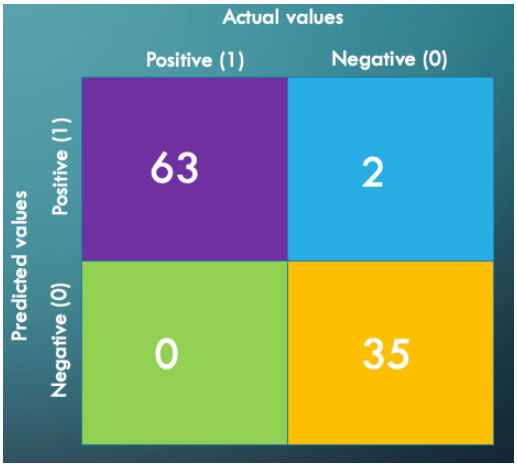


Figure 9: CPU Model Confusion Matrix

C. *MongoDB Model*

The MongoDB model was trained on 75% of 500 records and tested on the remaining 25%. As showcased in the confusion matrix in Figure 10, it was only able to achieve 69% accuracy.

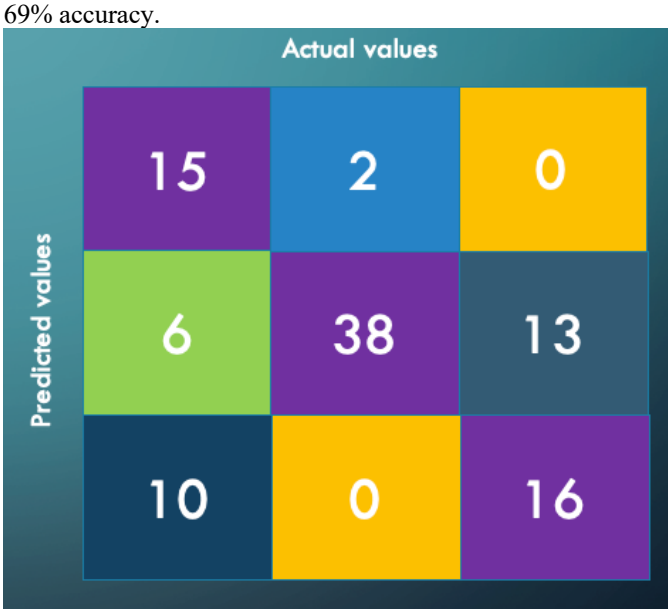


Figure 10: MongoDB Model Confusion Matrix

D. *Real-time Data Visualization*

Please refer to Figure 11 in the Appendix to see a snapshot of the real-time data visualization of my local system.

VI. Conclusion and Future Work

Through this project, I wanted to address the issue of business process interruption that are caused by inadequate management of compute, storage, and network resources in a data center. I explained the importance of the topic and provided a detailed description of how I implemented my solution.

It is known that in the real-world, a system that is 100% perfect and complete does not exist. There is also something more to discover and work on. Similarly, with my project I have listed out a couple of points that I would consider as a future enhancement and they are as follows:

- Explore other machine learning classification models such as Support Vector Machine and Random Forest and see how they perform
- Enhance data visualization by inserting business process visuals
- Create a mechanism to send alert via email or text message
- Research data preprocessing methods to preprocess data prior to using the data as a training and testing dataset

References

[1] Carolyn Duffy Marshall, "*The Evolution of the Internet*", Network World, 9, February 2009

[2] Salesforce, *"Overview: What is ecommerce?"*, https:/salesforce.com/products/commerce-cloud/resources/what-is-ecommerce/

[3] Mark Gaydos, "*Is Your Data Center Ready for Black Friday and Cyber Monday Onslaughts*", Industry Perspectives, 17, November 2017

[4] A White Paper from the Experts in Business-Crtical Continuity (Emerson), "*Understanding the Cost of Data Center Downtime: An Analysis of the Financial Impact on Infrastructure Vulnerability*", White Paper

[5] Phillipa Gill, Navendu Jain, Nachiappan Nagappan, "*Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications*", Dept. of Computer Science at University of Toronto, Microsoft

[6] AppDynamics Business White Paper, "*A Modern Approach to Monitoring Performance in Production*", 2014

[7] Elasticsearch, https://elastic.co

[8] Jeffrey R. Wilson, Kent A. Lorenz "*Standard Binary Logistic Regression Model*", Part of the ICSA Book Series in Statistics book series (ICSABSS, volume 9)

[9] Dr. Jon Starkweather, Dr. Amanda Kay Moske, "*Multinomial Logistic Regression*," UNT

[10] https://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains

:

APPENDIX

Figure 4a

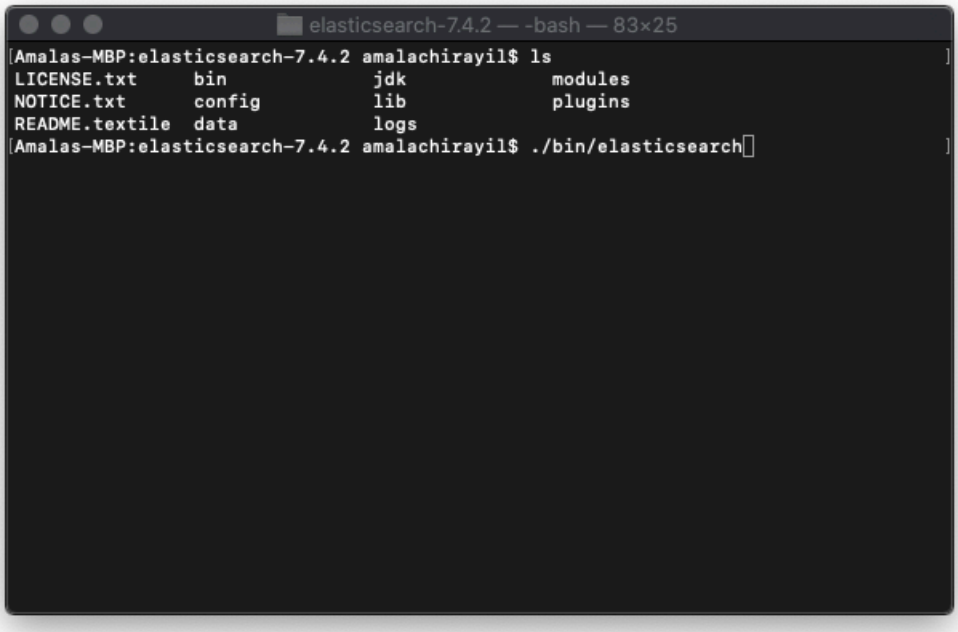


Figure 4b

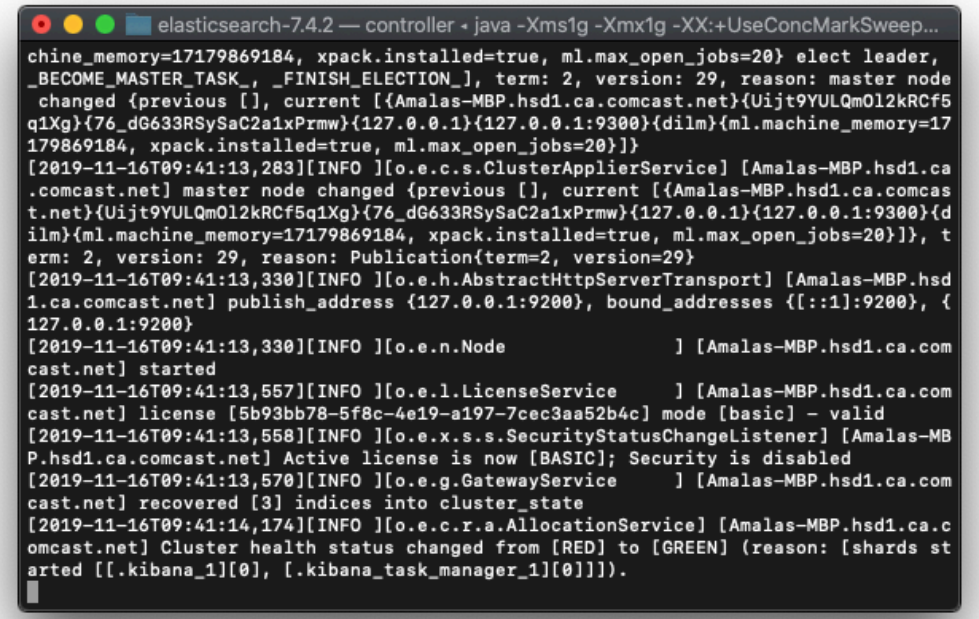
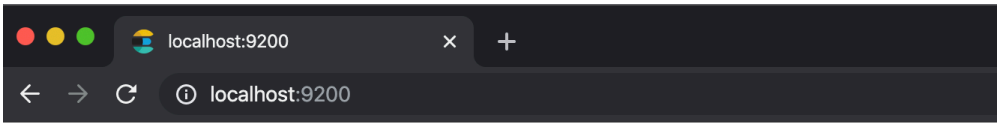


Figure 4c




```
{
  "name" : "Amalas-MBP.hsd1.ca.comcast.net",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "dROa3pCuSrODQRPbRIe3Eg",
  "version" : {
    "number" : "7.4.2",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "2f90bbf7b93631e52bafb59b3b049cb44ec25e96",
    "build_date" : "2019-10-28T20:40:44.881551Z",
    "build_snapshot" : false,
    "lucene_version" : "8.2.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```


Figure 5a

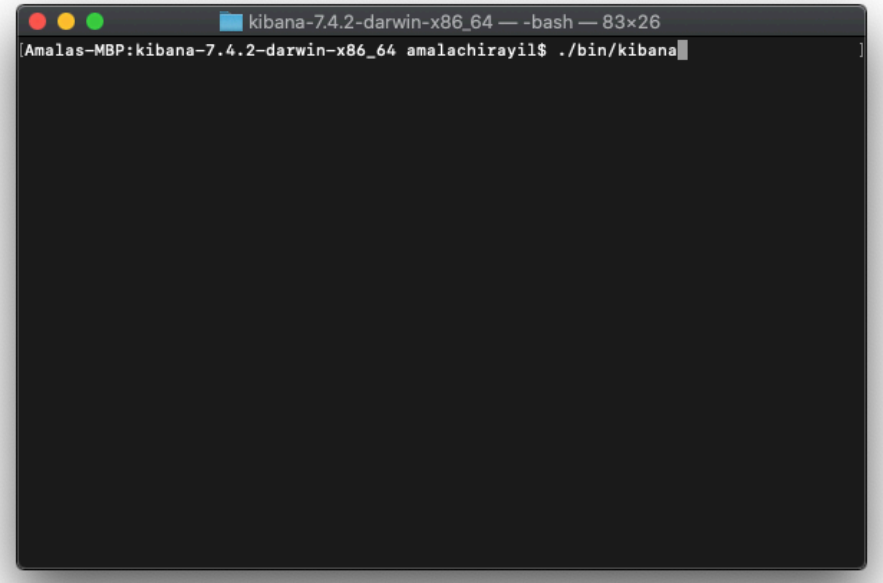


Figure 5b

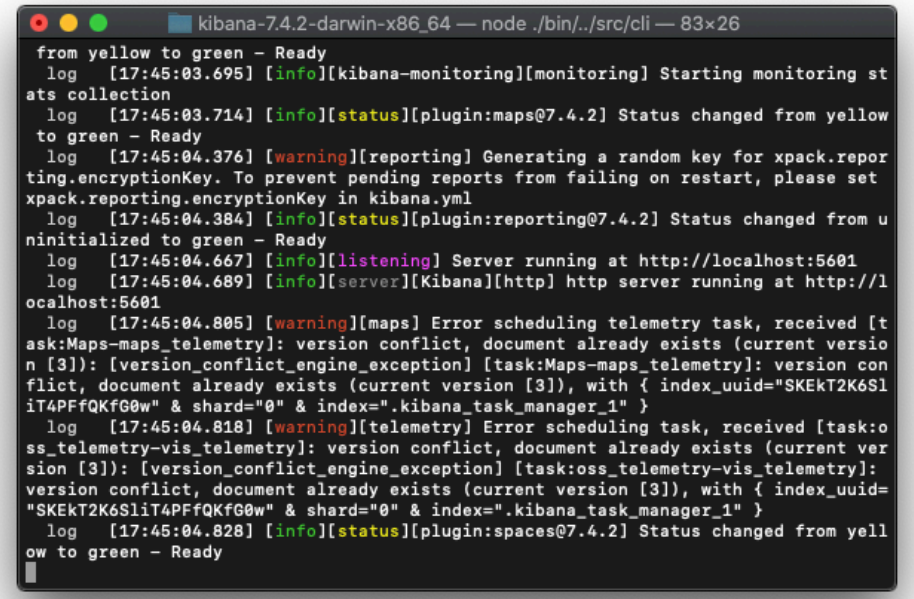


Figure 5c

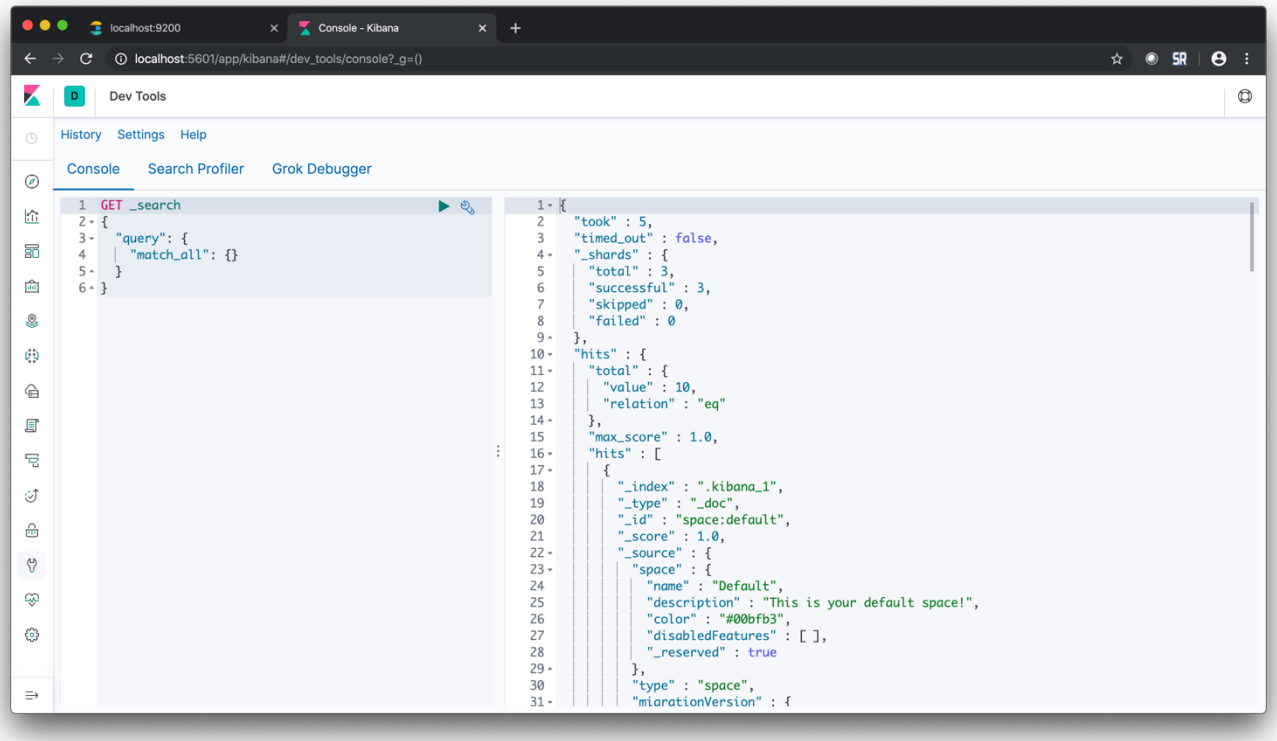


Figure 6a

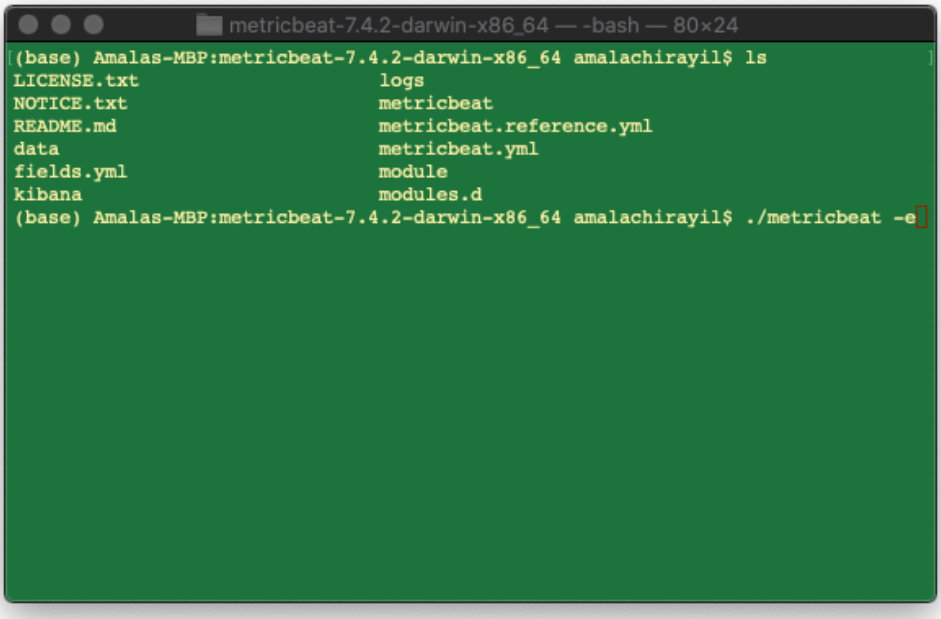


Figure 6b

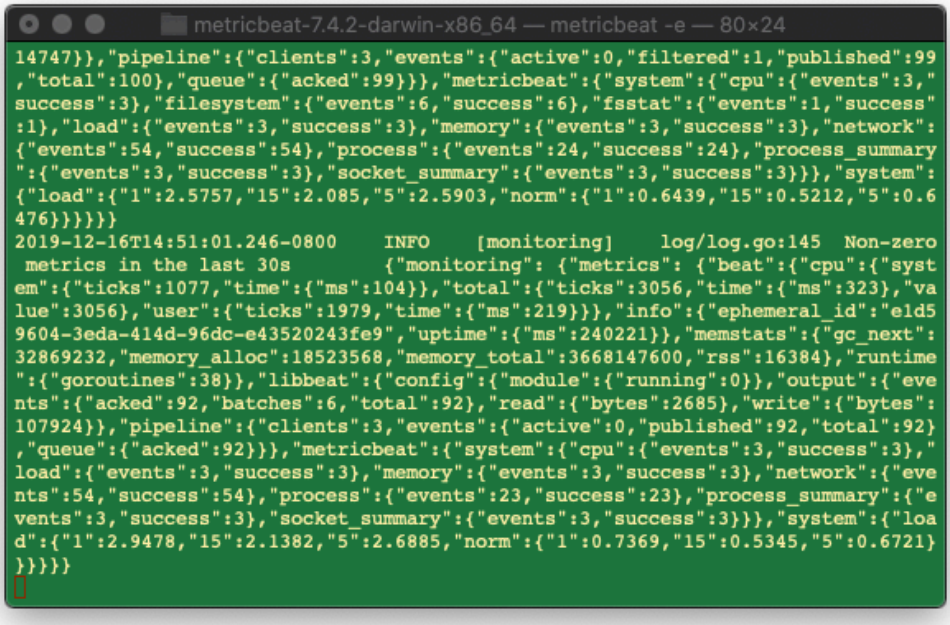


Figure 6c

```
green  open .kibana_task_manager_1          SKEkT2K6SliT4PFfQKfG0w 1 0        2 2  43.4kb  43.4kb
green  open .apm-agent-configuration        rC9LWFWFTZO97lfpM4uC8w 1 0        0 0    283b    283b
yellow open metricbeat-7.4.2-2019.11.17-000001 YSExueeJQw2LCKfKkxwRMA 1 1   395593 0  93.5mb  93.5mb
yellow open filebeat-7.4.2-2019.11.16-000001   4XFkqIuzT9O0Dyne8Dd46g 1 1 49014486 0   6.6gb   6.6gb
green  open .kibana_1                       AuPVvK2kT1CBe0cS0nI9bg 1 0       57 6 320.7kb 320.7kb

```

Figure 7a

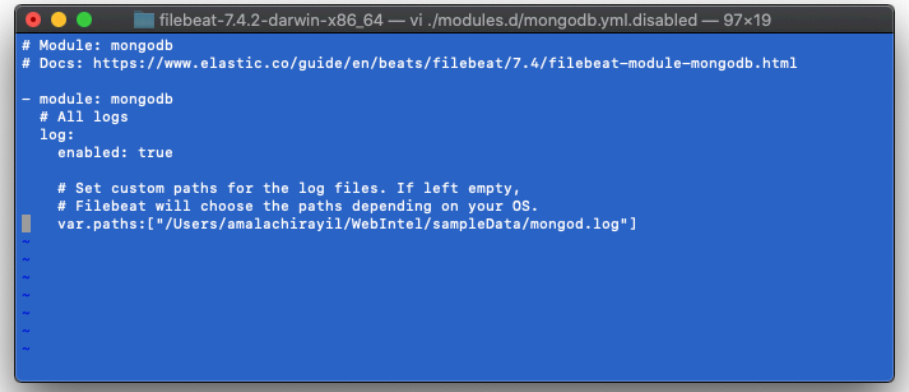


Figure 7b

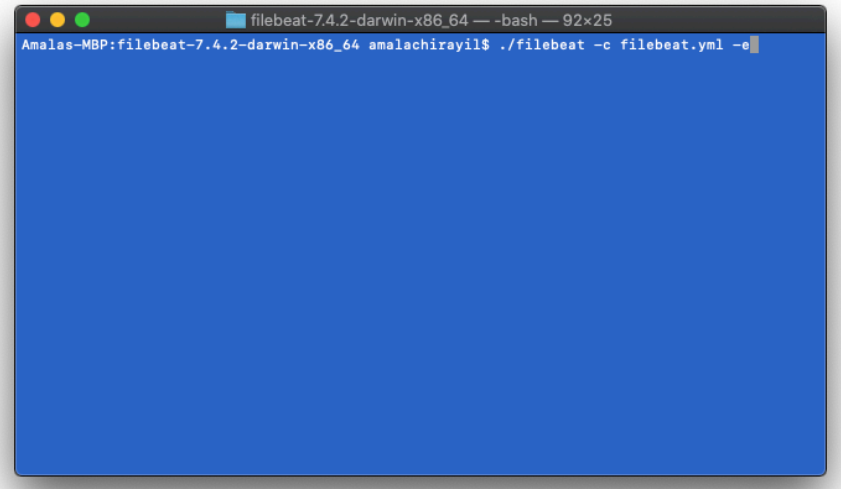
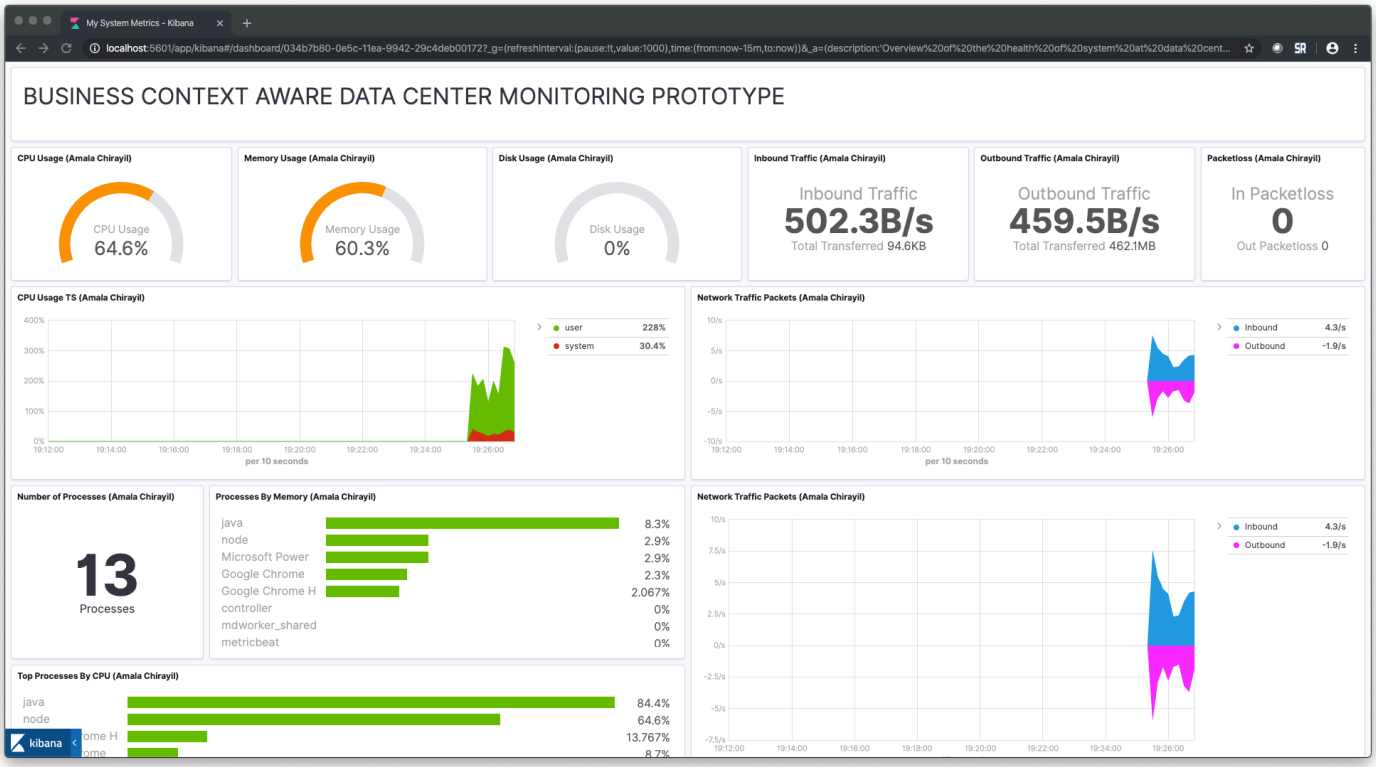


Figure 11


My System Metrics - Kibana
BUSINESS CONTEXT AWARE DATA CENTER MONITORING PROTOTYPE
CPU Usage (Amala Chirayil)
CPU Usage
64.6%
Memory Usage (Amala Chirayil)
Memory Usage
60.3%
Disk Usage (Amala Chirayil)
Disk Usage
0%
Inbound Traffic (Amala Chirayil)
Inbound Traffic
502.3B/s
Total Transferred 94.6KB
Outbound Traffic (Amala Chirayil)
Outbound Traffic
459.5B/s
Total Transferred 462.1MB
Packetloss (Amala Chirayil)
In Packetloss
0
Out Packetloss 0
CPU Usage TS (Amala Chirayil)
user 228%
system 30.4%
per 10 seconds
Network Traffic Packets (Amala Chirayil)
Inbound 4.3/s
Outbound -1.9/s
per 10 seconds
Number of Processes (Amala Chirayil)
13
Processes
Processes By Memory (Amala Chirayil)
java 8.3%
node 2.9%
Microsoft Power 2.9%
Google Chrome 2.3%
Google Chrome H 2.067%
controller 0%
mdworker_shared 0%
metricbeat 0%
Network Traffic Packets (Amala Chirayil)
Inbound 4.3/s
Outbound -1.9/s
Top Processes By CPU (Amala Chirayil)
java 84.4%
node 64.6%
13.767%
kibana