Cloud Failure Event Monitoring and Analysis

Sindhu Gummadi
PSID : 2146080

Usha Rani Nalla
PSID: 2146080

Keerthi Pogula
PSID: 2146080

***Abstract*—This project aims to crawl cloud failure events from publicly available status pages like Google, Azure, and record relevant information 1) start and end times of an incident, 2) failure symptoms, 3) root cause, 4) steps taken to repair a failure, and 5) maintenance events. Thereby perform analysis on the collected data and draw meaningful insights.**

I. Introduction

In recent years, cloud computing has become an essential part of the Software/IT industry, providing organizations with flexible and scalable computing resources. However, despite the many benefits of cloud computing, failures and downtime can still occur, which can cause significant disruptions to business operations and lead to customer dissatisfaction. To mitigate the impact of these failures, it is important to have up-to-date information about the status of cloud services, including information about incidents, maintenance events, and other service-related information.

However, despite the many benefits of cloud computing, failures and downtime can still occur, which can cause significant disruptions to business operations. These failures can be caused by various factors such as software bugs, hardware failures, cyber attacks, and human errors. To mitigate the impact of these failures, it is important to have up-to-date information about the status of cloud services, including information about incidents, maintenance events, and other service-related information.

To provide this information, cloud service providers such as Microsoft Azure, and Google Cloud provide public status pages that report the status of their services in real-time. These pages provide detailed information about incidents, including their start and end times, failure symptoms, root causes, and steps taken to repair the failures. Additionally, they provide information about scheduled maintenance events, which can help organizations plan for potential service disruptions.

To make use of this information, it is possible to crawl the public status pages of cloud service providers and collect data on incidents and maintenance events. This data can be stored in a database and analyzed to gain insights into the reliability and availability of cloud services. For example, it is possible to examine the distribution of different types of failures, frequency of a particular occurrence of a failure, peak time of failures, etc. These insights can help organizations identify trends and patterns in service failures, prioritize their response to incidents, and plan for future maintenance events.

In this report, we will explore how to crawl cloud failure events from public available information, including the status pages of cloud providers such as AWS, Azure, and Google Cloud. We will record information such as the start and end times of incidents, failure symptoms, root causes, steps taken to repair failures, and maintenance events. Additionally, we will perform some simple analysis on the collected data to gain insights into the reliability and availability of cloud services.

The report will provide a comprehensive overview of the steps involved in crawling cloud failure events, as well as practical insights into how to analyze this information to gain valuable insights into cloud service reliability and performance. Ultimately, the report aims to help organizations better understand the importance of monitoring cloud service status and to provide practical guidance on how to do so effectively.

II. Importance of Analysing Cloud Failure Information

Monitoring the status of cloud services is critical for organizations that rely on these services to support their business operations. In the event of an incident or failure, having up-to-date information about the status of the service can help organizations respond quickly and effectively, minimizing the impact on business operations and customer satisfaction. Conversely, a lack of information can lead to confusion, delays, and potential customer dissatisfaction.

By fetching information from public cloud status pages, organizations can gain real-time insights into the status of cloud services and any incidents or failures that may be impacting them. This information can help organizations:

1) Respond quickly to incident: : With real-time information about incidents, organizations can respond quickly to minimize the impact on business operations and customer satisfaction. For example, if a cloud service is experiencing downtime, organizations can quickly identify the root cause and take appropriate steps to restore service.

2) Plan for maintenance events: : Scheduled maintenance events can impact business operations, but by knowing in advance when these events are scheduled, organizations can plan accordingly and minimize the impact on their operations.

3) Identify trends and patterns: : By analyzing historical data about cloud failures, organizations can identify trends and patterns in service disruptions. This information can help organizations prioritize their response to incidents and plan for future maintenance events.

4) Enhance customer satisfaction: : By providing timely and accurate information about the status of cloud services, organizations can enhance customer satisfaction and build trust with their customers. Conversely, a lack of information can lead to frustration and potential customer churn.

In summary, fetching cloud failure information is critical for organizations that rely on cloud services to support their business operations. By monitoring the status of cloud services, organizations can respond quickly to incidents, plan for maintenance events, identify trends and patterns, and enhance customer satisfaction.

III. Types of Cloud Failures

There are several types of cloud failures that can occur in cloud computing environments, including:

1) Service outages: : This occurs when a cloud service is completely unavailable or inaccessible. Service outages can be caused by a variety of factors such as hardware failure, network issues, software bugs, or even natural disasters.

2) Performance degradation: : This occurs when a cloud service experiences reduced performance or slower response times than usual. Performance degradation can be caused by various factors, such as increased traffic, resource constraints, or software bugs.

3) Data loss: : This occurs when data stored in a cloud service is lost or becomes unrecoverable. Data loss can be caused by various factors, such as human error, hardware failure, or software bugs.

4) Security breaches: : This occurs when unauthorized access or malicious activity compromises the security of a cloud service. Security breaches can lead to data theft, data loss, or service disruption.

5) Compliance issues: : This occurs when a cloud service fails to meet regulatory or compliance requirements. Compliance issues can result in fines, legal penalties, or reputational damage.

It is important to note that these types of cloud failures are not mutually exclusive, and multiple failures can occur simultaneously or sequentially. Additionally, the impact of these failures can vary depending on the specific cloud service, the user's location, and other factors. Many of these failures could be reduced by performing an in-depth analysis of the failure information.

IV. Methodology

We used web crawling techniques to gather data from public sources, such as AWS, Azure, and Google Cloud status pages. We focused on collecting data related to failure events, including their start and end times, symptoms of failure, root cause, steps taken to repair, and maintenance events. We collected data for a period of six months, from 2022 to 2023.

V. Data Collection

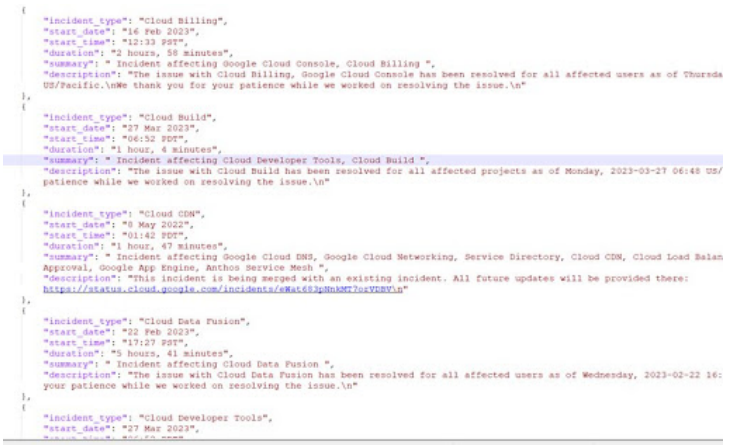
We collected data from the status pages of AWS, Azure, and Google Cloud. We identified all failure events and recorded the following information: start and end times, symptoms of failure, root cause, steps taken to repair, and maintenance events. We stored the data in a cloud database and used Python scripts to parse and analyze the collected data.

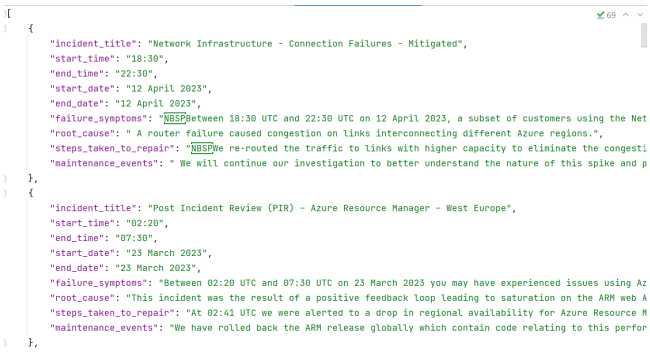
VI. Tools Used

The script uses libraries like BeautifulSoup for parsing the HTML, requests for making HTTP requests, json for storing the data in JSON format, and matplotlib for visualization. Initially we wrote the script for fetching failure events from Google Cloud status page. The same script has been modified to crawl other cloud provider status pages like Azure by updating the URL and the HTML classes used to extract the relevant information. The collected json data from the script is then stored in Google Cloud storage to store data from various service providers and schedule the scripts to run over fixed periods of time.

```
url = "https://status.cloud.google.com/summary"
response = requests.get(url)

html = response.content

soup = BeautifulSoup(html, 'html.parser')
incidents = soup.find_all('psd-product-table')

data = []

# Loop through all the incidents and extract the relevant information
for incident in incidents:
    incident_type = incident.find('span', class_='nAlKgGlv8Vo__product-name').get_text()
    incident_info = incident.find('table', class_='ise88CpWulY__psd-table')
    incident_summary_info = incident_info.find('td', class_='ise88CpWulY__summary')

    if incident_summary_info is not None:
        start_day = incident_info.find('td', class_='ise88CpWulY__date').get_text()
        incident_duration = incident_info.find('span', class_='ise88CpWulY__duration-text').get_text()
        incident_url = 'https://status.cloud.google.com/' + incident_summary_info.find('a').get('href')
        incident_report = requests.get(incident_url).content
        inner_soup = BeautifulSoup(incident_report, 'html.parser')
        main = inner_soup.find('main')
```

Json Structure:





VII. Data preprocessing

Before we can perform any analysis on the data collected from cloud status pages, we need to preprocess it to make it suitable for analysis. This involves cleaning the data and transforming it into a usable format. Some of the preprocessing steps that we followed are:

Data cleaning: This involves removing any irrelevant or incomplete data from the dataset, correcting any errors or inconsistencies, and handling missing values. We had to standardize the data to ensure that it is consistent and comparable across different sources. The data collected from the scraper was in HTML format. We used BeautifulSoup library from python to parse the HTML and fetch the relevant tags. We set up conditions in the script to remove irrelevant data and missing values to ensure data consistency.

Data transformation: This involves transforming the data into a format that can be easily analyzed. The retrieved HTML data had unstructured tags and subtags with lists of strings depicting the required information. We parsed the unstructured data and fetched the relevant tags to get the required information like Start Date, End Date, Type of Failure, Services affected by the failure, Steps taken to resolve the issue, Maintenance events, etc. We also used pattern matching techniques to parse the data and record information in a standardised format. We identified duplicate data and removed them. We identified outliers and removed them to refrain from ineffective analysis. We converted values like time and date to standard formats for easy analysis. Irrelevant and redundant features were removed for an accurate analysis.

Subsequently, we stored the structured data into a series of key-value pairs in a JSON format. We chose this JSON format as it's the most widely used file processing format. Different cloud providers had different information given in the status pages. The files are parsed and processed differently according to the specifications of these cloud providers status pages. We also had to convert categorical data into numerical data so that it can be used in machine learning algorithms.

Feature engineering: This involves creating new features from the existing data that may be useful in the analysis. For example, we created new features to capture the duration of an incident or the number of services affected. We also created new features to separate date, time, timezone of the incidents happened for better analysis.

Data aggregation: This involves aggregating the data over different time periods or grouping it by different categories. For example, we have aggregated the data to compute the number of incidents per month and grouped the data by incident type. This helped us to map the visualizations more effectively. Data aggregation was important to understand and visualize the numbers that would help us estimate the scale of the occurence of these incidents.

VIII. Data storage

We have used Google Cloud for storing the generated JSON data from both Azure and Google cloud platforms. Google Cloud Storage offers several advantages over other cloud storage providers. It can scale to petabytes of data without the need for additional infrastructure. Cloud Storage is designed for high durability and availability, with a 99.999999999 object durability guarantee. GCS offers a pay-as-you-go pricing model, which means you only pay for the storage you use, without any upfront costs. It provides strong encryption for data at rest and in transit, with options for customer-managed encryption keys. Google Cloud Storage integrates seamlessly with other Google Cloud services, such as BigQuery, Cloud Dataflow, and Cloud Pub/Sub. It allows us to store data in multiple regions, providing low-latency access to data from anywhere in the world. Overall, these advantages make Google Cloud Storage a popular choice for businesses looking for a cost-effective, scalable, and reliable storage solution.

IX. Steps to Store Data in GCS

1) Creation of GCS bucket: : First, we created a GCS bucket where we store our data. We created the bucket using GCS console.

2) Setting up access control: : We can set up access control to our bucket to ensure that only authorized users or applications can access our data. For our project, as the project deals with public information the access permissions are set to global.

3) Uploading data to the bucket: : Once our bucket was created, we uploaded the data to the bucket from the python script directly after the generation of data using blobs. We have achieved this using python libraries such as google.cloud, google.auth.credentials and google.oauth2. Blob is the fundamental unit of storage in GCS, and can be thought of as similar to a file in a file system. The data contains the content being stored, while the metadata provides information such as the object's name, creation time, and access control settings. Blobs can range in size from a few bytes to several terabytes and can be accessed programmatically using GCS APIs.

4) Use GCS for data processing: : Once your data is stored in GCS, we used it performing various visualisation tasks on the data. For this we have used libraries like matplotlib, pandas, spacy.

Overall, GCS provides a reliable, scalable, and secure platform for storing and managing your data, making it a popular choice for businesses and organizations of all sizes.

X. Root Cause Analysis of GC Failures

From the preliminary analysis of the Google Cloud Platform failures, we observed that the highest numbers failures were in the Cloud Networking area. The various components involved in the issue were Internal and External Http Load Balancers, L4 Load Balancer, Packet loss, Latency, Cloud DNS and Security Vulnerabalities. The root cause of the issue was found out to be an issue with the backend service responsible for distributing configuration changes. The service became unresponsive, resulting in the failure of other services that relied on it.

The second major type of failures is on Google Cloud Storage. Typical issues involved with the GCS were Bucket access issues, Slow performance, Billing and quota issues, Storage issues due to events like water intrusion into the data center, failure of multiple redundant coolant systems. When the issue occurred, the resolution steps taken led to customers facing further issues like inaccessibility of regional specific data.

Google compute engine has the next highest number of failures. It has instance start up failures, disk and storage issues, software compatibility issues, and configuration issues. This resulted in an error rate of 20%, requests hanging and latency. Few services like Cloud SQL which rely on Google Compute Engine, Persistent Disk Systems were also affected.

XI. Root Cause Analysis of Azure Failures

Network and Connectivity Issues has the highest number of failures in Azure Failure Events analysis. The incidents involve various technical issues affecting different services. There are incidents relates to physical network disruptions in a single cluster caused by planned maintenance, incidents caused by a software regression that caused intermittent spikes in CPU consumption. Some incidents were triggered by a power event at a regional internet service provider network exchange and a bad configuration push, the decommissioning of a legacy DNS solution that resulted in data streams for the Azure DNS recursive resolver service becoming out of sync with the resolver state.

Azure Resource Manager has the next highest number of failures from the preliminary analysis. The incidents involves multiple issues related to misconfigurations and synchronization problems with backend storage endpoints and networking resources in the ARM services. The misconfiguration in the deployment phase resulted in incorrect endpoint configurations for networking resources, leading to a positive feedback loop that caused saturation on the ARM web API tier due to high-volume, short-held lock contention. Additionally, a synchronization issue occurred between backend components, which impacted the processing of certain ARM requests.

Most other incidents occurred in Azure SQL database. In one incident, a specific network traffic pattern combined with a networking stack configuration on the SQL Gateway instances triggered an imbalance on the CPU processing of new connection requests. This caused connectivity issues for some clients attempting to connect to the affected instances, resulting in a degradation of service. The root cause of the issue was identified as the combination of the traffic pattern and the networking stack configuration, which caused an unanticipated bottleneck in the processing of new connection requests. In another incident, a bug caused high resource utilization in the internal cluster service that is responsible for receiving and executing service management operations. This caused a delay in the processing of requests and impacted the overall performance of the affected services. The root cause of the issue was identified as a bug in the internal cluster service, which caused high CPU usage and memory consumption. This, in turn, led to a delay in the processing of requests, resulting in degraded service performance. Also, a manual maintenance operation impacted instances of an internal cluster data service that is responsible for receiving and executing service management operations.

XII. Algorithms

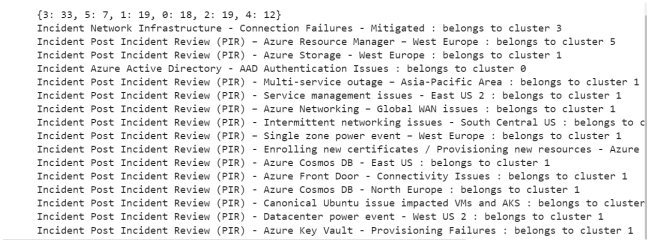
For our data needs, we have decided to use K-means[3] algorithm as we have unstructured data to process.

A. K-means Algorithm

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labeled, outcomes.[5]

The objective of K-means[1] is to group similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number (k) of clusters in a dataset. A cluster refers to a collection of data points aggregated together because of certain similarities. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

We have used K-means on Azure Failure Events data on the Incident Type field. We are using the TfidfVectorizer algorithm[4][6] from the sklearn library. It is used to convert a collection of raw documents to a matrix of TF-IDF features. Since we have textual data and K-means require numerical data, we used this algorithm for data conversion. By using the elbow method, We have decided to use 6 clusters.



Conclusion: From the algorithm's output, we have concluded that the most occurring incidents are in Availability or

Networking. Networking issues cluster is densely packed with different incident types.

B. Entity Detection

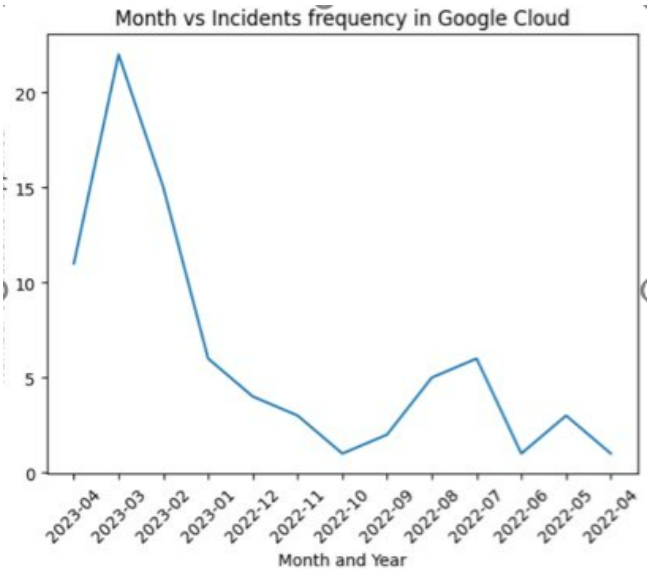
Entity detection, also known as Named Entity Recognition (NER), is a Natural Language Processing (NLP) technique that involves identifying and extracting entities from unstructured text data. An entity refers to a specific object, person, location, organization, or concept mentioned in the text. NER is the process of automatically identifying these entities and classifying them into predefined categories such as person, organization, location, time, etc.[11]
NER can be used in various applications such as text classification[2], information retrieval, question answering, sentiment analysis, and more. For example, in the context of social media monitoring, entity detection can be used to identify the names of companies or products mentioned in tweets, which can be used to track the sentiment of customers towards those brands.

We have cross verified this result from what we extracted from the Cloud status pages and concluded that these two resulted the same.

XIII. Visualization

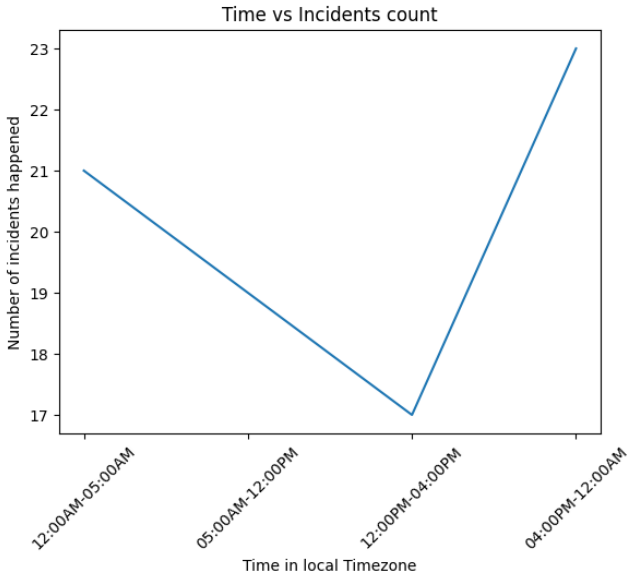
Below is the list of plots we have drawn from the data analysis. We primarily used matplotlib[10] library for the plots.

1) Google Cloud Storage Failures Analysis: Month vs Incidents frequency



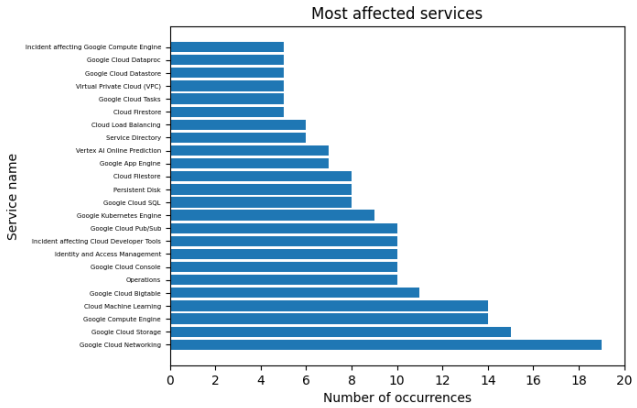
Above figure shows the incidents count from the year 2022 to 2023 in Google Cloud. We can conclude that the incidents count in Google Cloud has increased from the year 2022 to 2023.

2) Google Cloud Storage Failures Analysis: Time of the day vs Incident count



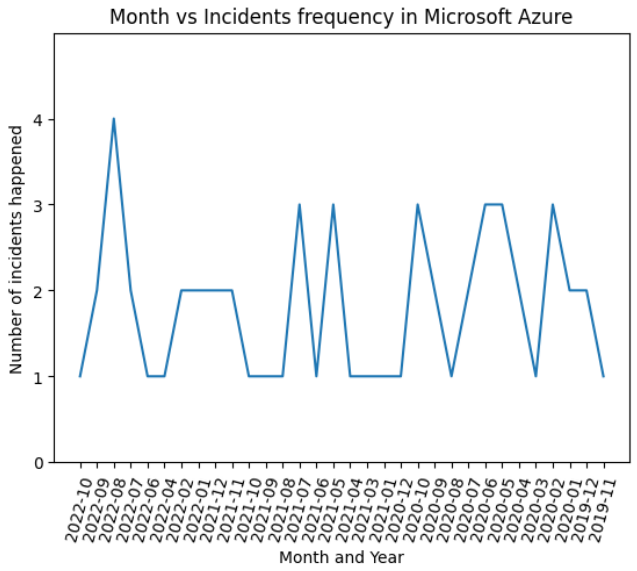
The plot above shows the incident count at different times of the day in Google Cloud for a particular timezone. We can conclude from it that most of the incidents happened at early in the morning and less incidents happened later in the day.

3) Google Cloud Storage Failures Analysis: Most affected Services vs Incident count



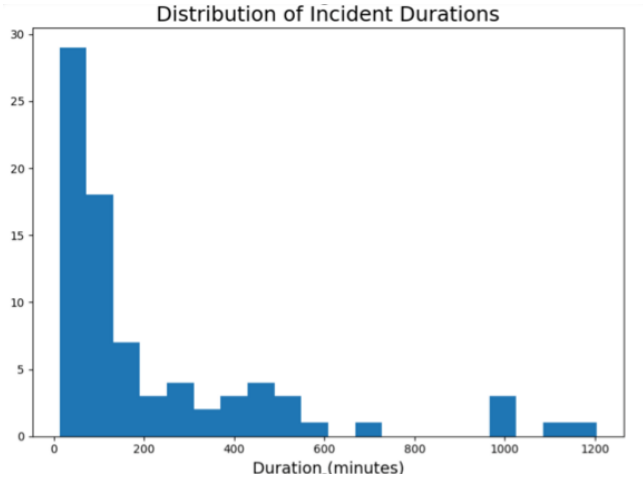
The plot above shows the most affected services vs incident count on Google Cloud. We can say that most incidents occurred in Google Cloud Networking service. The next service with the highest number of incidents is Google Cloud Storage.

4) Microsoft Azure Cloud Failures Analysis: Month vs Incidents frequency



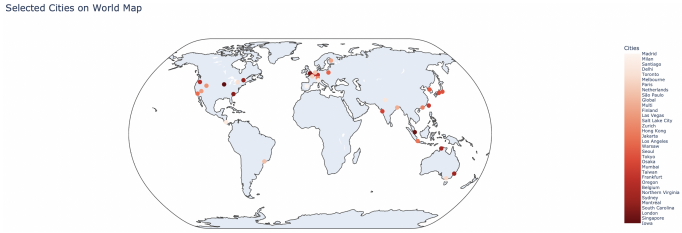
Above figure shows the incidents count from the year 2019 to 2022 in Google Cloud. There's no particular pattern in the plot. Therefore, we can say that incidents in Google Cloud happened more or less at the same frequency in all the years that we observed.

5) Microsoft Azure Cloud Failures Analysis: Incident Count vs Duration of resolution



The plot above shows the amount of time it took in minutes to resolve the issues against the number of incidents that got resolved in that duration. We can conclude from the graph that most of the resolved were resolved in less than 200 minutes.

A. *Cities having the most Cloud Failures*



The plot above shows the world map of different cities having the most number of cloud failure events. From it, we can see that Iowa has the most number of cloud failures and Madrid has the least.

XIV. Future Development

Our analysis can be extended to other cloud providers. Also, it can be scheduled to run at specific times and can be integrated with other projects. A web page or an application can be developed on this concept to have continuous access to various cloud failure events visualization.

XV. Conclusion

From different cloud failure incidents frequency count, we understood that GCS performed better than other service providers when it came to resolving the issues. In GCS Services, most cloud failure incidents occurred in Google Cloud Networking service. The next service with the highest number of incidents is Google Cloud Storage. From K-means, we understood that Availability and Networking issues are the most occurring failure events. Most of the incidents happened early in the morning and fewer incidents happened later in the day. Iowa has the most number of cloud failures and Madrid has the least.

REFERENCES

[1] K-means clustering in Python: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[2] Text classification in Python: https://towardsdatascience.com/text-classification-in-python-dd95d264c802

[3] K-means clustering in Python: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[4] TfIDVectorizer : https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[5] Kmeans: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336

[6] TfIdVectorizer : https://www.sciencedirect.com/science/article/abs/pii/S0167404820303576

[7] Google Cloud Status https://status.cloud.google.com/summary

[8] Empirical Analysis Cloud Incident Data: https://ieeexplore.ieee.org/document/6529290

[9] Azure Cloud Status: https://azure.status.microsoft/en-us/statushistoryapi/?serviceSlug=all&regionSlug=all&startDate=all&shdrefreshflag=true

[10] Visualisations https://matplotlib.org/

[11] Entity Detection: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8629225