Akilandeshwari Srinivasan(451036)

CLCM3102 Lab 8 (Final Lab)

Cloud-Based Notification and Monitoring System Design

Due by – 28th November, 2023

Step 1: System Requirements Gathering

a. Specific Performance Metrics and Events for Monitoring:

1. Response Time: Monitor the application's response time to ensure optimal user experience and identify performance bottlenecks.
2. Server Resource Utilization: Track CPU, memory, and disk usage to proactively manage resource allocation and prevent potential issues.
3. Transaction Success Rates: Monitor successful and failed transaction rates to ensure the reliability of payment processing and order fulfilment.
4. Network Latency: Measure network latency to guarantee smooth communication between different components of the e-commerce system.
5. Error Rates: Track error rates in application components to identify and resolve issues impacting user interactions.
6. Database Performance: Monitor database query times and resource utilization to maintain efficient data retrieval and storage.

b. Critical Events Requiring Immediate Notifications:

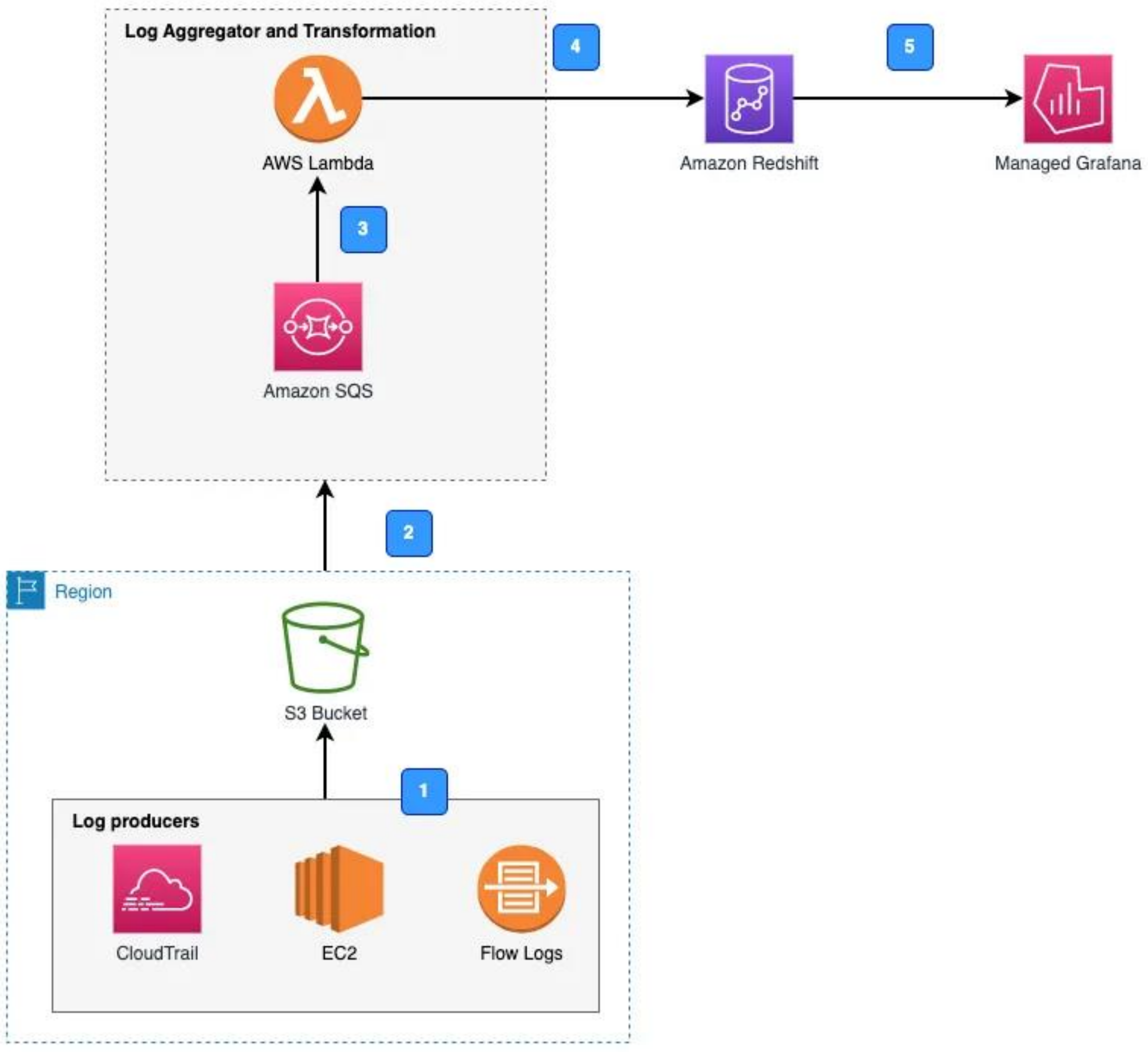
1. System Downtime: Immediate notifications for any unplanned outages or disruptions in the e-commerce application.
2. Security Breaches: Instant alerts for any detected security breaches, unauthorized access attempts, or data breaches.
3. Payment Failures: Immediate notification of transaction failures to minimize revenue loss and provide swift resolution.
4. Server Overload: Alerts for critical resource utilization thresholds, indicating potential server overload and performance degradation.
5. Infrastructure Issues: Immediate notifications for any AWS service disruptions or critical failures in the cloud infrastructure.

c. Long-Term Objectives of the Monitoring System:

1. Proactive Issue Identification: Identify and address performance issues before they impact user experience or result in downtime.
2. Scalability: Design the monitoring system to scale with the company's growth, accommodating increased user traffic and data volume.
3. Cost Optimization: Optimize resource allocation based on monitoring data to ensure efficient use of cloud resources and minimize costs.
4. Comprehensive Security Monitoring: Enhance security measures by continuously monitoring for potential vulnerabilities and responding to security incidents promptly.
5. Continuous Improvement: Implement a feedback loop for continuous improvement, using monitoring insights to enhance application architecture and overall system reliability.

By addressing these key requirements, the monitoring and notification system aims to ensure the e-commerce application's reliability, availability, and performance, ultimately contributing to a positive user experience and business success.

Step 2: High-Level Architecture



1. Amazon CloudWatch Agent or AWS CloudTrail collect log lines and store them in Amazon Simple Storage Service (Amazon S3).
2.Amazon S3 sends an object create event to Amazon Simple Queue Service (Amazon SQS).
3.Amazon SQS invokes an AWS Lambda function that transforms the log lines from strings to structured JSON.
4.After the data transformation data are stored in specific format that is suitable for us in the Amazon Redshift data storage.
5.Amazon Redshift data storage is used as input data source for fully Managed Grafana reports.

Step 3: Notification Strategy

a. <u>Explanation of AWS SNS Usage</u>:
AWS Simple Notification Service (SNS) will serve as the notification backbone for critical incidents. In the event of such incidents, AWS SNS will be configured to send alerts to relevant stakeholders and systems. SNS provides a scalable and reliable communication mechanism, supporting various delivery protocols such as email, SMS, and HTTP endpoints. Stakeholders and systems can subscribe to specific SNS topics, ensuring that notifications reach the intended recipients promptly. This setup ensures a streamlined and efficient communication channel during critical events.

b. <u>Types of Events or Thresholds for Notifications</u>:
Notifications will be triggered based on predefined events and thresholds indicative of critical incidents. These may include but are not limited to:

1. System Downtime: Notifications will be triggered when the e-commerce application experiences unplanned outages or downtime.

2. Performance Degradation: Thresholds for response times or error rates exceeding acceptable limits will trigger notifications, indicating potential performance issues.

3. Security Breaches: Alerts will be generated in the case of security incidents, such as unauthorized access attempts or data breaches.

4. Infrastructure Failures: Notifications will be sent for critical issues in the underlying infrastructure, like AWS service disruptions or resource exhaustion.

5. Transaction Failures: Events indicating a significant increase in failed transactions or a decline in successful transactions will prompt notifications.

6. Resource Utilization: Notifications will be triggered if resource utilization surpasses predefined thresholds, helping prevent potential bottlenecks.

Configuring AWS SNS to respond to these events ensures that relevant stakeholders are promptly informed, enabling quick and effective responses to critical incidents in the e-commerce system.

Step 4: Data Collection and Storage
Performance data from the e-commerce application will be collected through instrumentation, utilizing tools like Prometheus or AWS CloudWatch agents. These agents will monitor metrics such as response times and resource utilization. The collected data will be ingested into the system using AWS Simple Queue Service (SQS), acting as a reliable message queue. AWS SQS plays a crucial role in managing incoming data by decoupling the data producers from consumers, ensuring efficient and scalable processing. It queues incoming performance metrics, allowing for asynchronous processing and preventing data loss in case of system spikes or failures.

Step 5: Data Visualization and Reporting:

Grafana is an open-source monitoring and visualization platform, consolidating data from diverse sources. It enables users to create dynamic dashboards with charts and graphs, supporting real-time monitoring and historical analysis. Grafana's alerting system notifies users of potential issues, while its extensibility allows customization with plugins. The platform facilitates collaboration by sharing dashboards and supports user authentication for secure data access. Grafana's versatility extends to reporting, making it a valuable tool for presenting insights to stakeholders. Overall, Grafana is a flexible, user-friendly solution for monitoring, analysis, and reporting in IT and DevOps environments.

Reference:

www.google.com , www.medium.com