Google Cloud Monitoring

Use Case: Enhancing Observability with Google Cloud Monitoring

GENERAL CHARACTERISTICS

Intent	To enhance observability and incident response using Google Cloud Monitoring.
Scope	Implementation of monitoring and alerting for cloud-native and hybrid environments.
Level	System-level.
Client	Confidential (E-Commerce Platform).
Last Update	03/12/2024
Status	Finalized.
Stage	Deployment and Optimization.

ACTORS

Primary Actor	Site Reliability Engineer (SRE).
Secondary Actors	IT Operations Team, DevOps Engineers, Application Owners.

PREREQUISITES

Static Preconditions	- Google Cloud Project set up with Monitoring API enabled. - Cloud resources and applications identified for observability.
Dynamic Preconditions	- Metrics and logs integrated with Google Cloud Monitoring. - Alerting policies defined based on application and system performance SLAs.
Assumptions	- Client requires proactive monitoring and real-time incident response. - Applications are distributed across multiple regions or hybrid environments.

TRIGGERS

Trigger Event	The client required real-time monitoring and incident alerting to ensure high availability of their e-commerce platform.

EXPECTED OUTCOME

Success Postcondition	- Proactive monitoring detects and resolves issues before they impact users. - Alerting policies notify the team in real-time during incidents.
Failed Postcondition	- Downtime or performance degradation leads to user dissatisfaction and revenue

	loss.

OPERATIONS AND CONCEPTS

Operations	1. Integrated application and infrastructure metrics with Google Cloud Monitoring. 2. Configured dashboards to visualize key performance indicators (KPIs). 3. Set up alerting policies for critical thresholds (e.g., CPU, memory, latency). 4. Enabled uptime checks to monitor application availability across regions. 5. Automated incident notifications using PagerDuty and Slack integrations. 6. Regularly reviewed logs and metrics to identify performance bottlenecks.
Concepts	- Cloud Monitoring: Provides observability into cloud resources and applications. - Alerting Policies: Automatically notify teams when predefined thresholds are breached. - Dashboards: Visualize real-time performance data for better insights.

MAIN SUCCESS SCENARIO

Step 1	Analyzed the client's application architecture and monitoring requirements.
Step 2	Integrated metrics and logs from all cloud resources with Cloud Monitoring.
Step 3	Created dashboards to monitor application health in real-time.
Step 4	Configured alerting policies for critical performance metrics.
Step 5	Set up uptime checks to ensure availability across multiple regions.
Step 6	Enabled automated notifications for incidents via PagerDuty and Slack.
Step 7	Reviewed historical data to identify and resolve performance bottlenecks.