Epic: Implement a comprehensive Observability solution for GCP platform and workloads

As a cloud engineer, operations, or site reliability engineer, I want to implement observability tools for visibility across our GCP infrastructure to ensure that we can monitor, troubleshoot, and optimize the performance and health of my platform and application workloads.

User Story 1: Log Collection and Analysis

- **As a** Public Cloud User
- **I want to** centrally view and analyze logs from all GCP services (GCE, GKE, Cloud Functions, etc.)
- **So that I can** troubleshoot issues across my cloud infrastructure

  Acceptance Criteria:

  - Logs from all GCP services are collected in a central observability solution.
  - I can search, filter, and analyze logs based on severity, labels, and time ranges.

User Story 2: Infrastructure Metrics Monitoring

- **As a** Site Reliability Engineer (SRE)
- **I want to** monitor key performance metrics of my infrastructure (CPU, memory, network traffic, etc.) using Cloud Monitoring
- **So that I can** ensure the health and performance of GCE, GKE, and other resources are maintained and detect anomalies proactively

  Acceptance Criteria:

  - Metrics from all GCP resources are available and displayed on custom dashboards.
  - Alerts are triggered when critical thresholds are crossed (e.g., CPU > 80%, memory usage > 70%).
  - Alerts are sent to incident management tools like ServiceNow or PagerDuty.

User Story 3: Distributed Tracing for cloud services and applications

- **As a** Developer
- **I want to** implement distributed tracing using Cloud Trace for my service architecture.
- **So that I can** track and identify latency issues across services and optimize response times

  Acceptance Criteria:

  - Traces are automatically collected for requests across multiple services.
  - Latency and performance bottlenecks can be visualized in the Cloud Trace console.
  - I can drill down into specific traces to identify the service or request causing delays.

User Story 4: Real-Time Error Detection

- **As a** DevOps Engineer

- **I want to** automatically detect and aggregate errors from logs using Cloud Error Reporting
- **So that I can** identify recurring issues and address them before they impact user experience and SLOs

  Acceptance Criteria:

  - Cloud Error Reporting aggregates similar errors from application logs.
  - Notifications are sent when a threshold of recurring errors is detected.
  - Errors are classified by service and severity for prioritization.

User Story 5: Uptime Monitoring for APIs and Services

- **As a** Site Reliability Engineer (SRE) or Cloud Engineer
- **I want to** set up uptime checks for critical APIs and services using Cloud Monitoring
- **So that I can** ensure that my services are available to end-users and receive alerts when downtime occurs

  Acceptance Criteria:

  - Uptime checks are configured for all critical services and APIs.
  - Alerts are generated when services become unavailable or exceed latency thresholds.
  - Reports on service uptime and availability are generated and viewable in the Cloud Monitoring dashboard.

User Story 6: Service Level Objective (SLO) Tracking

- **As a** SRE Manager
- **I want to** define and track Service Level Objectives (SLOs) for my applications and services using Cloud Monitoring
- **So that I can** ensure that my services meet performance and availability targets agreed upon with stakeholders

  Acceptance Criteria:

  - SLOs are defined based on key metrics like uptime, response times, and error rates.
  - SLO dashboards are set up, showing real-time and historical performance against targets.
  - Alerts are sent when SLOs are at risk of being breached.

User Story 7: Log Retention and Cost Management

- **As a** Cloud Operations Manager
- **I want to** set retention policies for logs and optimize the cost of storing observability data
- **So that I can** balance between long-term data retention and the cost of storage

  Acceptance Criteria:

    - Retention policies should be set for different types of logs (e.g., error logs are retained for 30 days, debug logs for 7 days).
    - Cost reports for Cloud Logging and Cloud Monitoring are generated.
    - Log storage is optimized by exporting long-term logs to Cloud Storage.

User Story 8: Observability for Service and Resource Limits and Capacity

- **As a** Cloud Operations Engineer
- **I want to** visualize all the limits and available capacity for all my resources in GCP
- **So that I can** be proactive in detecting resource limit breaches before they impact my SLOs

  Acceptance Criteria:

    - All resource and service limits are accessible from single view for all GCP services across different projects.
    - I can get notification of an impending service limit breach.

User Story 9: Observability for Hybrid Cloud Environments

- **As a** Cloud Architect
- **I want to** implement observability across my hybrid cloud environment (GCP and on-premise) using Google Cloud Monitoring and Logging
- **So that I can** have a unified view of performance and health across all workloads

  Acceptance Criteria:

    - On-premises infrastructure is integrated into Google Cloud Monitoring and Logging.
    - A unified dashboard displays metrics, logs, and performance data from both GCP and on-prem environments.
    - Alerts are set for both cloud and on-prem workloads, enabling centralized incident management.

User Story 10: Observability for GCP Network Services

- **As a** Cloud Network Engineer
- **I want to** visualize network traffic and configurations data
- **So that I can** identify network traffic bottlenecks and misconfigurations

  Acceptance Criteria:

    - VPC flow data is integrated into Google Cloud Monitoring and Logging.
    - A unified dashboard displaying network traffic data for GCP VPCs across all projects.

User Story: Proactive notifications

- As a tenant
- I want to be notified proactively for any service disruptions
- So that we could plan our work accordingly
- Acceptance Criteria:
    - Receive advance notification of planned maintenance
    - Get alerts for unexpected outages
    - Have access to a status page showing system health for services being used.

User Story: Unified Observability Dashboard

- As a tenant
- I want a unified observability dashboard
- So that I can get a holistic view of the entire application GCP infrastructure and associated services
- Acceptance Criteria:
    - Integration of metrics, logs and traces in a unified view
    - Ability of custom widgets to accommodate business KPIs along with technical metrics
    - Ability to drill down from high-level overview to detailed component analysis

User Story: Log Analytics and advanced queries

- As a tenant
- I want to be able to use advanced querying capabilities
- So that I can perform complex analysis on our observability data
- Acceptance Criteria:
    - Ability to create custom log-based metrics
    - Ability to implement alerts/policies based on log query results
    - Ability to access visual exploration of data

User Story: Users experience monitoring

- As a tenant
- I want to be able to monitor user experience in real time
- So that we can understand how the backend performance impacts the user experience
- Acceptance Criteria:
    - Ability to correlate the frontend performance data with the backend using traces
    - Dashboard showing end to end user journey performance

User story: Automated Runbooks

- As an SRE engineer
- I want to be able to create automated runbooks.
- So that we can automatically respond to common issues without human intervention
- Acceptance Criteria:

    - Implementation of Function triggered by Monitoring alerts
    - Automated scaling (not Autopilot) of GKE clusters based on load
    - Self-healing capabilities for common application errors

User Story: Unified logging strategy

- As an observability engineer
- I want to implement a unified logging strategy across all our services
- So that we can easily correlate events across systems
- Acceptance Criteria:
    - Centralize all logs
    - Implement structured logging with consistent field (if possible) across services
    - Create log-based metrics for critical events

User Story: Continuous profile of our key services

- As an observability engineer
- I want to implement continuous profiling for our key services
- So that we can optimize resource usage and performance
- Acceptance Criteria:
    - Identify and optimize top 5 resource consuming functions
    - Create dashboard providing resource (such as CPU, Memory) usage pattern over time

User Story: Mapping between relationship of infrastructure components and our services

- As an observability engineer
- I want to implement a topology mapping solution
- So that we can understand the relationship between our services and infrastructure component
- Acceptance criteria:
    - Usage of topology mapping feature such as Cloud Monitoring (GCP)
    - Visualize service dependencies and data flow
    - Setup alerts for change in system topology

User Story: Error reporting Integration

- As an observability engineer
- I want to integrate error reporting with our logging system
- So that we can quickly identify and resolve application errors.
- Acceptance criteria:
    - Automatic error detection from logs
    - Real time alerts for new or frequent errors

    - Dashboard showing error trends and resolutions

User Story: AI driven anomaly Detection [AI/ML use case]

- As a SRE and cloud operations engineer
- I want to be able to implement AI driven anomaly detection
- So that we can proactively identify unusual system behavior
- Acceptance Criteria:
    - Realtime anomaly detection on incoming metrics
    - Ability to get alerted on detected anomalies
    - Ability to train data using historical data

User Story: Predictive Analytics for resource utilization forecasting [AI/ML use case]

- As an SRE and Cloud Operational engineer
- I want to be able to implement predictive analytics
- So that we can forecast resource utilization and potential bottlenecks
- Acceptance Criteria:
    - Ability to train forecasting models using historical Cloud Monitoring data
    - Integration with a scheduler for regular predictive updates
    - Alerting on predictive resource bottlenecks and performance issues.