SRE Practices: Alerting and Observability Framework

1. Segregation of Alerts Based on Severity

To ensure effective incident management and reduce alert fatigue, alerts will be categorized by severity:

- **Severity 0 (Sev 0): Critical Alerts**
  - Description: Issues requiring immediate resolution to avoid significant business impact.
  - Examples:
    - Breached critical thresholds (e.g., CPU utilization > 90%).
    - Certificate expirations.
    - Service outages or unavailability.
  - Action: Immediate escalation to on-call teams via designated distribution lists.
- **Severity 1 (Sev 1): High-Priority Alerts**
  - Description: High-priority issues needing prompt attention but not immediately business-critical.
  - Examples:
    - Moderate performance degradation (e.g., API latency > 500ms consistently).
    - Resource nearing critical thresholds (e.g., disk space > 80%).
  - Action: Escalation to on-call teams with clear timelines for resolution.
- **Severity 2 (Sev 2): Informational Alerts**
  - Description: Alerts triggered by events within specific time frames or patterns for monitoring purposes.
  - Examples:
    - High volume of requests within a short window.
    - Spikes in non-critical errors (e.g., 404s or 429s).
  - Action: Monitored by teams; no immediate escalation unless thresholds persist.

2. Classification of Alerts: Actionable vs. Non-Actionable

- **Actionable Alerts**
  - Require immediate investigation and corrective actions.
  - Examples:
    - Pods unreachable or unhealthy.
    - Database connection failures.

- **Non-Actionable Alerts**
    - For monitoring or informational purposes only; no action required.
    - Examples:
        - Successful job completions.
        - Scheduled report generation.
    - Action: Limit distribution to essential stakeholders to minimize noise.

3. Standardized Alert Pattern

A consistent naming convention will improve clarity and facilitate alert triaging. The proposed format is:

<Severity>-<Application Name>-<Description>

- Example: Sev0-OrderService-HighCPUUsage

4. Observability Framework with New Relic and Splunk

To achieve end-to-end observability, we will leverage New Relic and Splunk across the three pillars of observability:

- **Metrics**
    - **New Relic**: Monitor application performance metrics, infrastructure health, and key business transactions.
        - Dashboards: CPU usage, memory utilization, request latency.
        - Alerts: Threshold-based alerts for anomalies (e.g., high error rates).
    - **Splunk**: Aggregate and visualize system-level metrics for historical analysis.
- **Logs**
    - **New Relic**: Centralized log management for real-time application logging.
        - Use cases: Debugging and tracing application errors.
    - **Splunk**: Advanced log search and analysis.
        - Use cases: Root cause analysis, compliance, and historical trend identification.
- **Traces**
    - **New Relic**: Distributed tracing for end-to-end visibility of application workflows.
        - Use cases: Identifying bottlenecks and slow API calls.
    - **Splunk**: Augmented with tracing data to correlate logs and metrics for comprehensive insights.

5. Supporting Documentation

- **New Relic**:

    - Guidelines for configuring alerts and dashboards.
    - Best practices for anomaly detection and APM setup.
- **Splunk**:
    - Documentation for building custom dashboards and alert rules.
    - Recommended search queries for efficient log analysis.