SRE Incident Management Best Practices for MetLife Insurance

Overview

MetLife Insurance is modernizing its digital infrastructure to improve reliability and customer satisfaction. However, the current incident management process is manual and fragmented, leading to delays in detection, diagnosis, and resolution. This document outlines best practices for Site Reliability Engineering (SRE) to streamline incident response and improve service uptime across key APIs.

Core Services Overview

1. policy-api

Purpose:
Handles all operations related to insurance policy lifecycle management, including creation, updates, renewals, cancellations, and retrieval of policy details.

Key Functions:

- Policy creation and underwriting
- Policyholder data updates
- Coverage modifications
- Policy status queries

Dependencies:

- Customer data (via customer-api)
- Payment verification (via payment-api)

Risks if Down:

- Inability to issue or modify policies
- Customer onboarding delays
- Compliance and regulatory risks

---

2. claims-api

Purpose:
Manages the end-to-end claims process, from submission to adjudication and payout.

Key Functions:

- Claim intake and validation
- Status tracking
- Integration with fraud detection systems
- Communication with policy-api for coverage verification

Dependencies:

- Policy verification (via policy-api)
- Customer identity (via customer-api)
- Payment processing (via payment-api)

Risks if Down:

- Delayed claim processing
- Increased customer dissatisfaction
- Regulatory non-compliance

---

3. payment-api

Purpose:
Facilitates all financial transactions including premium payments, refunds, and claim disbursements.

Key Functions:

- Payment authorization and capture
- Refund processing
- Integration with third-party payment gateways
- Transaction logging and reconciliation

Dependencies:

- Policy and claim data
- Customer account details

Risks if Down:

- Failed or delayed payments
- Financial reporting issues

- Trust erosion due to payment failures

---

4. customer-api

Purpose:
Centralizes customer data management, including personal information, preferences, and authentication.

Key Functions:

- Customer profile creation and updates
- Authentication and authorization
- Data sharing with other APIs
- Consent and privacy management

Dependencies:

- Identity verification systems
- External CRM platforms

Risks if Down:

- Login failures
- Inability to access or update customer data
- Broken user journeys across services

Incident Management Best Practices

1. Proactive Monitoring

- Implement unified observability tools (e.g., Grafana, Prometheus, Datadog) across all APIs.
- Set up service-level indicators (SLIs) and objectives (SLOs) for latency, error rate, and availability.
- Use synthetic monitoring to simulate user interactions.

2. Intelligent Alerting

- Configure alerts based on thresholds and anomaly detection.
- Use context-rich alerts with metadata (affected service, severity, recent changes).
- Avoid alert fatigue by tuning sensitivity and grouping related alerts.

3. Automated Triage and Diagnosis

- Integrate AIOps platforms to correlate logs, metrics, and traces.
- Use runbooks and playbooks for common incident types.
- Employ automated root cause analysis (RCA) tools to reduce manual effort.

4. Incident Response Workflow

- Adopt a standardized incident lifecycle: **Detection → Triage → Mitigation → Resolution → Postmortem**.
- Use incident management platforms (e.g., PagerDuty, Opsgenie) for coordination.
- Assign clear roles: Incident Commander, Communications Lead, Technical Lead.

5. Post-Incident Review

- Conduct blameless postmortems within 48 hours of resolution.
- Document RCA, impact, timeline, and remediation steps.
- Track incident trends to identify systemic issues.

6. Continuous Improvement

- Regularly update runbooks and playbooks.
- Invest in chaos engineering to test resilience.
- Automate remediation for recurring issues (e.g., restart services, scale resources).