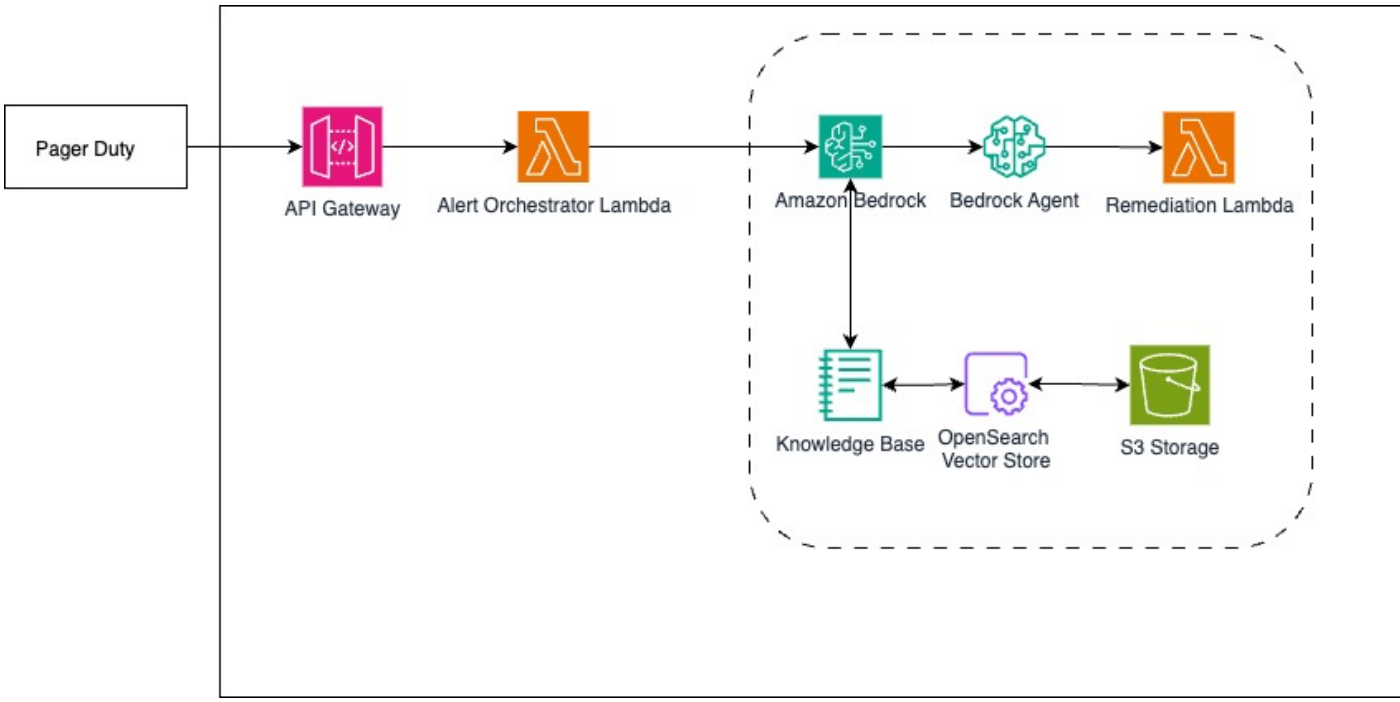
AI-Driven Alert Remediation System - Solution Overview

Executive Summary

The AIDriven Alert Remediation System leverages Amazon Bedrock Agent to automatically a
nalyze alerts from PagerDuty and execute appropriate remediation actions
through AWS Lambda functions. This system reduces mean time to resolution (MTTR) and m
inimizes human intervention for common operational issues.

Architecture Diagram



Key Components

1. **Alert Ingestion**
    - PagerDuty webhook integration
    - API Gateway endpoint for secure webhook reception
    - Initial alert validation and formatting
2. **Bedrock Agent**
    - AI-powered decision making
    - Trained on remediation knowledge base
    - Determines appropriate remediation action
    - Returns structured remediation plans
3. **Remediation Framework**
    - Action Group Lambda for orchestration
    - Specialized remediation Lambdas
    - Comprehensive error handling
    - Audit logging and monitoring

Remediation Capabilities

Alert Type	Remediation Action	Lambda Function
High CPU	Auto-scaling	scale_up
Service Crash	Service Restart	restart_service
Disk Space	Cleanup	cleanup_disk
DB Issues	Connection Reset	reset_database
HA Events	Failover	failover

Implementation Benefits

- **Automated Resolution** : Reduces manual intervention
- **Intelligent Decisions** : AI-powered remediation selection
- **Scalable** : Easy to add new remediation patterns
- **Auditable** : Complete logging of decisions and actions
- **Secure** : IAM-based access control
- **Cost-Effective** : Serverless architecture

Monitoring and Metrics

- CloudWatch metrics for remediation success rates
- Detailed logging of all decisions and actions
- Performance tracking of resolution times
- Agent decision confidence scoring

Security Controls

- IAM role-based access
- API Gateway authentication
- Encrypted communications
- Least privilege principles
- Audit logging

Best Practices

1. **Knowledge Base Management**
    - Regular updates of remediation patterns
    - Documentation of new scenarios
    - Validation of AI decisions
2. **Operational Excellence**
    - Monitoring of remediation success rates
    - Regular testing of remediation actions
    - Continuous improvement of patterns
3. **Security**
    - Regular rotation of credentials
    - Security patches and updates
    - Access review and audit

Future Enhancements

1. Machine learning for pattern recognition
2. Enhanced decision confidence scoring
3. Integration with additional alert sources
4. Advanced remediation workflows
5. Predictive maintenance capabilities

Success Metrics

- Reduction in MTTR
- Decrease in manual interventions
- Improved service availability
- Cost savings in operations
- Increased automation coverage

This solution provides a robust, scalable, and intelligent approach to automated incident rem ediation, leveraging AWS's advanced AI capabilities through
Amazon Bedrock while maintaining security and operational excellence.

Sample Knowledge Base Content

```
# Remediation Patterns and Decision Rules

## High CPU Usage Alerts
When encountering alerts related to high CPU usage:
- If CPU usage > 80% for more than 5 minutes
- If the resource is part of an Auto Scaling group
- Use the 'scale_up' lambda function
- Parameters should include:
  - scalingFactor: 2
  - minInstances: 2

## Service Crash Alerts
For service crash or application failure alerts:
- If health checks are failing
- If the service is unresponsive
- Use the 'restart_service' lambda function
- Parameters should include:
  - forceRestart: true
  - timeout: 300

## Disk Space Alerts
When dealing with disk space issues:
- If disk usage > 85%
- Use the 'cleanup_disk' lambda function
- Parameters should include:
  - retentionDays: 7
  - excludePaths: ["/var/log/archive"]

## Database Connection Issues
For database connection problems:
- If connection timeouts occur
- If connection pool exhaustion is detected
- Use the 'reset_database' lambda function
- Parameters should include:
  - connectionTimeout: 60
  - maxRetries: 3
```

High Availability Failover
For critical service availability issues:
- If primary instance is unhealthy
- If multiple health checks fail
- Use the 'failover' lambda function
- Parameters should include:
  - waitTime: 30
  - validateHealth: true