Incident Response Runbook - HX Infrastructure Phase 3.4

Overview

This runbook provides comprehensive procedures for incident detection, classification, remediation, and escalation within the HX Infrastructure Phase 3.4 Production Operations framework.

Incident Classification

Severity Levels

Critical (P1)

- **Definition**: Complete service outage or data loss
- **Response Time**: Immediate (< 5 minutes)
- **Examples**:
- All application endpoints down
- Database unavailable
- Security breach detected
- Data corruption

High (P2)

- **Definition**: Significant service degradation
- **Response Time**: < 15 minutes
- **Examples**:
- Single service failure
- Performance degradation > 50%
- Partial functionality loss
- High error rates (> 10%)

Medium (P3)

- **Definition**: Minor service impact
- **Response Time**: < 1 hour
- **Examples**:
- Non-critical service issues
- Performance degradation < 50%
- Monitoring alerts
- Resource warnings

Low (P4)

- **Definition**: Minimal or no service impact
- **Response Time**: < 4 hours
- **Examples**:
- Informational alerts
- Maintenance notifications

- Documentation updates

Automated Incident Detection

Detection Methods

The system automatically detects incidents through:

1. **System Health Monitoring**
   - CPU usage > 80%
   - Memory usage > 85%
   - Disk usage > 90%
   - System load > 5.0

2. **Service Monitoring**
   - Service failures (systemd status)
   - Application endpoint failures
   - Database connectivity issues

3. **Performance Monitoring**
   - Response time > 2000ms
   - Error rate > 5%
   - Throughput degradation

Manual Incident Detection

```
# Run comprehensive health check
./scripts/automation/monitoring/health_check.sh --type all --format summary

# Check specific service
systemctl status nginx postgresql docker

# Monitor real-time metrics
watch -n 5 './scripts/automation/monitoring/health_check.sh --type system --format
summary'
```

Incident Response Procedures

Immediate Response (First 5 Minutes)

1. **Acknowledge the Incident**
   ```bash
   # Check incident status
   cat /var/log/hx-infrastructure/incidents/detected-*.json | tail -1 | jq '.'

# View active incidents
cat /var/www/html/incident_dashboard.json | jq '.active_incidents'
```

1. **Assess Impact and Severity**
   `bash`
   `# Check affected services`
   `ansible-playbook -i inventory/production \`

```
    playbooks/production/operations/health_monitoring.yml \
    --tags "quick_assessment"
```

2. **Initiate Auto-Remediation**
   `bash`
   ```
   # Trigger automated remediation
   ansible-playbook -i inventory/production \
     roles/incident_response/tasks/auto_remediation.yml
   ```

Auto-Remediation Actions

Service Failures

```
# Restart failed critical services
systemctl restart nginx postgresql docker

# Verify service recovery
systemctl status nginx postgresql docker
```

Resource Issues

```
# High CPU - Kill resource-intensive processes
ps aux | sort -nrk 3,3 | head -5
kill -TERM <high-cpu-pid>

# High Memory - Clear system caches
sync && echo 3 > /proc/sys/vm/drop_caches

# High Disk - Clean temporary files
find /tmp -type f -atime +1 -delete
find /var/tmp -type f -atime +1 -delete
```

Application Issues

```
# Restart application services
docker-compose restart
systemctl restart hx-infrastructure-app

# Clear application caches
redis-cli FLUSHALL
```

Manual Remediation

Database Issues

```
# Check database connectivity
psql -h localhost -U postgres -d hx_infrastructure -c "SELECT 1;"

# Check database performance
psql -h localhost -U postgres -d hx_infrastructure -c "
  SELECT count(*) as active_connections,
         avg(extract(epoch from (now() - query_start))) as avg_query_time
  FROM pg_stat_activity
  WHERE state = 'active';"

# Restart database if needed
systemctl restart postgresql
```

Network Issues

```
# Check network connectivity
ping -c 3 8.8.8.8
curl -I http://google.com

# Restart network services
systemctl restart networking
systemctl restart nginx
```

Load Balancer Issues

```
# Check nginx configuration
nginx -t

# Reload nginx configuration
systemctl reload nginx

# Check upstream servers
curl -H "Host: internal" http://localhost/health
```

Escalation Procedures

Automatic Escalation Triggers

- Critical incidents (P1)
- Auto-remediation failures
- Multiple service failures
- Security incidents

Escalation Steps

Level 1 - Team Lead

```
# Send escalation notification
ansible-playbook -i inventory/production \
  roles/incident_response/tasks/escalate_incidents.yml \
  -e "escalation_level=1"
```

Level 2 - Engineering Manager

- Escalate if no response in 15 minutes
- Multiple critical incidents
- Customer impact confirmed

Level 3 - Executive Team

- Major outage (> 1 hour)
- Data breach or security incident
- Regulatory compliance issues

Notification Channels

Slack Integration

```
# Send Slack notification
curl -X POST -H 'Content-type: application/json' \
  --data '{"text":"Critical incident on production: Service outage detected"}' \
  $SLACK_WEBHOOK_URL
```

Email Alerts

```
# Send email alert
echo "Critical incident detected on $(hostname)" | \
  mail -s "CRITICAL: HX Infrastructure Alert" ops-team@company.com
```

PagerDuty Integration

```
# Trigger PagerDuty alert
curl -X POST https://events.pagerduty.com/v2/enqueue \
  -H 'Content-Type: application/json' \
  -d '{
    "routing_key": "'$PAGERDUTY_ROUTING_KEY'",
    "event_action": "trigger",
    "payload": {
      "summary": "Critical incident on production",
      "source": "'$(hostname)'",
      "severity": "critical"
    }
  }'
```

Incident Documentation

During Incident

1. **Create Incident Record**
   `bash`
   ```
     # Incident documentation is automated
     # Check incident ID
     ls /var/log/hx-infrastructure/incidents/$(date +%Y-%m-%d)/
   ```

2. **Update Incident Status**
   `bash`

```
# View incident timeline
cat /var/log/hx-infrastructure/incidents/$(date +%Y-%m-%d)/INC-*_timeline.txt
```

Post-Incident

Immediate Actions (Within 1 Hour)

- [ ] Verify service restoration
- [ ] Update incident status
- [ ] Notify stakeholders
- [ ] Document root cause (preliminary)

Follow-up Actions (Within 24 Hours)

- [ ] Complete post-incident analysis
- [ ] Update monitoring/alerting
- [ ] Implement preventive measures
- [ ] Update documentation

Post-Incident Analysis Template

```
# Post-Incident Analysis: INC-{INCIDENT_ID}

## Incident Summary
- **Date/Time**:
- **Duration**:
- **Severity**:
- **Services Affected**:

## Timeline
- **Detection**:
- **Response**:
- **Resolution**:

## Root Cause
- **Primary Cause**:
- **Contributing Factors**:

## Impact Assessment
- **Users Affected**:
- **Revenue Impact**:
- **SLA Breach**:

## Response Evaluation
- **What Went Well**:
- **What Could Be Improved**:

## Action Items
- [ ] Immediate fixes
- [ ] Long-term improvements
- [ ] Process updates
- [ ] Training needs
```

Monitoring and Alerting

Key Dashboards

- **Grafana Main**: http://monitoring:3000/d/hx-infrastructure
- **Incident Dashboard**: http://production/incident_dashboard.json

- **System Health**: http://production/health

Alert Thresholds

```
# Current alert thresholds
cpu_usage: 80%
memory_usage: 85%
disk_usage: 90%
response_time: 2000ms
error_rate: 5%
```

Monitoring Commands

```
# Real-time system monitoring
watch -n 5 'free -h && df -h && uptime && systemctl status nginx postgresql docker'

# Application monitoring
curl -s http://localhost/health | jq '.'
curl -s http://localhost/metrics | grep -E "(response_time|error_rate)"

# Log monitoring
tail -f /var/log/hx-infrastructure/application.log | grep -E "(ERROR|CRITICAL|FATAL)"
```

Recovery Procedures

Service Recovery

```
# Full service recovery playbook
ansible-playbook -i inventory/production \
  playbooks/production/operations/service_management.yml \
  -e "operation=full_recovery"
```

Database Recovery

```
# Database recovery procedures
ansible-playbook -i inventory/production \
  playbooks/backup/restore_database.yml \
  -e "restore_point=latest"
```

Application Recovery

```
# Application stack recovery
docker-compose down
docker-compose up -d
./scripts/automation/monitoring/health_check.sh --type endpoints
```

Communication Templates

Initial Incident Notification

```
Subject: [INCIDENT] Production Issue Detected - {SEVERITY}

We are currently investigating a {SEVERITY} incident affecting {SERVICES}.

Incident ID: {INCIDENT_ID}
Detected: {TIMESTAMP}
Impact: {DESCRIPTION}
Status: Investigating

Updates will be provided every 15 minutes.
```

Resolution Notification

```
Subject: [RESOLVED] Production Incident - {INCIDENT_ID}

The production incident has been resolved.

Incident ID: {INCIDENT_ID}
Duration: {DURATION}
Root Cause: {CAUSE}
Resolution: {RESOLUTION}

Post-incident analysis will be completed within 24 hours.
```

Contact Information

Emergency Contacts

- **Primary On-Call**: +1-555-0101
- **Secondary On-Call**: +1-555-0102
- **Engineering Manager**: +1-555-0103
- **Security Team**: +1-555-0104

Communication Channels

- **Slack**: #hx-infrastructure-incidents
- **Email**: incidents@hx-infrastructure.local
- **War Room**: https://meet.company.com/incident-response

Tools and Resources

Incident Management Tools

- **Incident Dashboard**: http://production/incident_dashboard.json
- **Log Aggregation**: /var/log/hx-infrastructure/
- **Monitoring**: http://monitoring:3000
- **Metrics**: http://monitoring:9090

Useful Commands

```bash
# Quick incident overview
./scripts/automation/monitoring/health_check.sh --type all

# Service status check
systemctl status nginx postgresql docker consul

# Resource usage
htop
iotop
nethogs

# Network diagnostics
ss -tulpn
netstat -i
```

Training and Drills

Monthly Incident Response Drills

- Simulate various incident types
- Practice escalation procedures
- Test communication channels
- Review and update procedures

Training Resources

- Incident Response Training (../training/incident_response.md)
- System Architecture (../../ARCHITECTURE.md)
- Monitoring Setup (../dashboards/monitoring_setup.md)