Gestión de Incidentes y Recuperación - Día 3

Change Status

pending · 40 min

Learning Objectives

1. Aprender manejo de incidentes en producción
2. Comprender estrategias de recuperación
3. Conocer importancia de post-mortems
4. Entender preparación para escenarios de disaster

Activities and Learning

Task 1: Tipos de Incidentes Comunes (10 minutos)

Incidentes más frecuentes en pipelines:

1. Fallos de conectividad:

APIs caídas, bases de datos inaccesibles
Timeouts de red, límites de rate alcanzados
Problemas de autenticación/credenciales expiradas

2. Problemas de datos:

Datos corruptos o con formato inválido
Cambios inesperados en esquemas
Volúmenes de datos que exceden capacidad

3. Problemas de recursos:

Memoria insuficiente, CPU al límite
Disco lleno, límites de almacenamiento
Contención de recursos entre pipelines

4. Errores lógicos:

Bugs en código de transformación
Dependencias circulares en DAGs
Configuraciones incorrectas

Task 2: Estrategias de Recuperación (10 minutos)

Runbook de respuesta a incidentes:

Paso 1: Detección automática

```
# Alert triggers
alert_conditions = {
    'pipeline_down': 'up == 0',
    'high_error_rate': 'error_rate > 0.05',
    'sla_violation': 'duration > sla_threshold',
    'data_staleness': 'last_update > 4h'
}
```

Paso 2: Diagnóstico rápido

```
def diagnose_incident(incident_type, symptoms):
    """Diagnóstico automatizado de incidentes"""

    if incident_type == 'pipeline_down':
        return check_infrastructure(symptoms)
    elif incident_type == 'data_quality':
        return check_data_integrity(symptoms)
    elif incident_type == 'performance':
        return check_resource_usage(symptoms)

    return "Unknown incident type"
```

Paso 3: Recuperación automática

```python
def auto_recovery(incident_diagnosis):
    """Intentos de recuperación automática"""

    recoveries = {
        'network_timeout': lambda: retry_with_backoff(),
        'memory_limit': lambda: scale_resources(),
        'data_corruption': lambda: rollback_to_checkpoint(),
        'credential_expired': lambda: refresh_credentials()
    }

    recovery_func = recoveries.get(incident_diagnosis)
    if recovery_func:
        return recovery_func()

    return False  # Requires manual intervention
```

Task 3: Post-Mortems y Mejora Continua (10 minutos)

Estructura de post-mortem efectiva:

1. Facts (Hechos):

¿Qué sucedió exactamente?
¿Cuándo comenzó y terminó?
¿Qué impacto tuvo?

2. Timeline (Cronología):

Momento de detección
Acciones tomadas
Momento de resolución

3. Root Cause (Causa raíz):

¿Por qué sucedió?
¿Fue un error humano, técnico, o proceso?

4. Impact & Resolution (Impacto y resolución):

¿Cuántos usuarios afectados?
¿Cómo se resolvió?
¿Cuánto tiempo tomó?

5. Lessons Learned (Lecciones aprendidas):

¿Qué mejorar?
¿Qué prevenir?
¿Qué automatizar?

Template de post-mortem:

```
# Post-Mortem: Pipeline Down Incident
```