Monitoreo y Observabilidad - Día 2

Change Status

pending · 40 min

Learning Objectives

1. Entender importancia del monitoreo en producción
2. Aprender métricas clave para pipelines
3. Comprender herramientas de observabilidad
4. Conocer estrategias de alerting efectivas

Activities and Learning

Task 1: Métricas Esenciales de Pipeline (10 minutos)

¿Qué monitorear en un pipeline de producción?

Métricas de performance:

Latency: Tiempo desde ingesta hasta resultados
Throughput: Registros procesados por unidad de tiempo
Error rate: Porcentaje de tareas fallidas
Resource usage: CPU, memoria, disco

Métricas de negocio:

Data freshness: Antigüedad máxima de datos
Completeness: Porcentaje de datos completos
Accuracy: Precisión de cálculos y transformaciones
SLA compliance: Cumplimiento de acuerdos de servicio

Ejemplo de métricas personalizadas:

```
from airflow.metrics import Stats

def track_pipeline_metrics(**context):
    """Trackear métricas custom del pipeline"""
    ti = context['task_instance']

    # Métrica: registros procesados
    records = ti.xcom_pull(task_ids='extract', key='records_count') or 0
    Stats.gauge(f'pipeline.{ti.dag_id}.records_processed', records)

    # Métrica: duración de tarea
    duration = (ti.end_date - ti.start_date).total_seconds()
    Stats.timer(f'pipeline.{ti.dag_id}.{ti.task_id}.duration', duration)

    # Métrica: tasa de éxito
    if ti.state == 'success':
        Stats.incr(f'pipeline.{ti.dag_id}.success_count')
    else:
        Stats.incr(f'pipeline.{ti.dag_id}.failure_count')
```

Task 2: Herramientas de Observabilidad (10 minutos)

Stack de monitoreo moderno:

1. Prometheus + Grafana:

```
# prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'airflow'
    static_configs:
      - targets: ['airflow-webserver:8080']
    metrics_path: '/admin/metrics'
```

2. ELK Stack (Elasticsearch, Logstash, Kibana):

Elasticsearch: Almacenamiento y búsqueda de logs

Logstash: Procesamiento de logs
Kibana: Visualización y dashboards

3. Alerting con Alertmanager:

```
# alertmanager.yml
route:
  group_by: ['alertname']
  group_wait: 10s
  group_interval: 10s
  repeat_interval: 1h
  receiver: 'slack-notifications'

receivers:
- name: 'slack-notifications'
  slack_configs:
  - api_url: 'YOUR_SLACK_WEBHOOK'
    channel: '#alerts'
```

Task 3: Estrategias de Alerting (10 minutos)

Tipos de alertas por severidad:

1. Informativas (Info):

Notificaciones de eventos normales
Reportes de finalización exitosa
Métricas de performance

2. Advertencias (Warning):

Degradación de performance
Errores recuperables
SLA cerca de violarse

3. Críticas (Critical):

Fallos en producción
Datos no disponibles
SLA violadas

Configuración de alertas inteligentes:

```
def smart_alert_policy(metric_name, value, threshold):
    """Política de alertas basada en severidad"""

    if metric_name == 'error_rate':
        if value > 0.5:  # 50% error rate
            return 'CRITICAL', 'Error rate extremely high'
        elif value > 0.1:  # 10% error rate
            return 'WARNING', 'Error rate elevated'

    elif metric_name == 'latency':
        if value > 3600:  # 1 hora
            return 'CRITICAL', 'Pipeline severely delayed'
        elif value > 1800:  # 30 minutos
            return 'WARNING', 'Pipeline delayed'

    return 'INFO', 'Metric within normal range'
```