Monitoreo y Alertas - Día 4

Change Status

completed · 40 min

Learning Objectives

1. Entender importancia del monitoreo en pipelines
2. Aprender configuración de alertas y notificaciones
3. Comprender métricas clave para trackear
4. Conocer herramientas de observabilidad

Practical exercise to apply the concepts learned.

Ejercicio: Configurar monitoreo completo en un DAG

DAG con monitoreo y alertas:

```python
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.operators.email import EmailOperator
from datetime import datetime, timedelta
import logging

# Configurar logging adicional
logger = logging.getLogger('monitored_dag')

def procesar_datos_con_metricas(**context):
    """Función que trackea métricas detalladas"""
    task_instance = context['task_instance']

    logger.info(f"Iniciando procesamiento - Task: {task_instance.task_id}")

    # Simular procesamiento
    import time
    import random

    # Métrica: tiempo de procesamiento simulado
    processing_time = random.uniform(10, 60)
    time.sleep(processing_time)

    # Métrica: registros procesados
    records_processed = random.randint(1000, 5000)

    # Métrica: tasa de éxito
    success_rate = random.uniform(0.95, 1.0)

    resultado = {
        'registros_procesados': records_processed,
        'tiempo_procesamiento': processing_time,
        'tasa_exito': success_rate,
        'timestamp': datetime.now().isoformat()
    }

    # Guardar métricas en XCom para otras tareas
    task_instance.xcom_push(key='metricas', value=resultado)

    logger.info(f"Procesamiento completado: {resultado}")
    return resultado

def validar_metricas(**context):
    """Validar que las métricas cumplan thresholds"""
    task_instance = context['task_instance']
    metricas = task_instance.xcom_pull(task_ids='procesar_datos', key='metricas')

    if not metricas:
        raise ValueError("No se encontraron métricas")

    # Validar thresholds
    if metricas['tasa_exito'] < 0.9:
        raise ValueError(f"Tasa de éxito baja: {metricas['tasa_exito']:.2%}")

    if metricas['tiempo_procesamiento'] > 300:  # 5 minutos
        logger.warning(f"Tiempo de procesamiento alto: {metricas['tiempo_procesamiento']:.1f}s")

    logger.info("Validación de métricas exitosa")
    return True

def on_failure_alert(context):
    """Callback personalizado para fallos"""
    task_instance = context['task_instance']
```

```
    dag_id = context['dag'].dag_id
    error = str(context.get('exception', 'Unknown error'))

    alert_message = f"""
```

🚨 ALERTA DE FALLO 🚨

DAG: {dag_id} Tarea: {task_instance.task_id} Ejecución: {task_instance.execution_date} Error: {error}

Por favor revisar logs inmediatamente. """.strip()

```
    logger.error(alert_message)

    # Aquí podrías enviar a Slack, PagerDuty, etc.
```

def on_success_summary(context): """Resumen de ejecución exitosa""" task_instance = context['task_instance'] dag_id = context['dag'].dag_id

```
    # Calcular duración total del DAG
    start_date = context['dag_run'].start_date
    end_date = context['dag_run'].end_date

    if start_date and end_date:
        duration = (end_date - start_date).total_seconds()
        logger.info(f"DAG {dag_id} completado en {duration:.1f} segundos")
```

Configurar DAG con monitoreo

dag = DAG('pipeline_monitorado', description='Pipeline ETL con monitoreo completo', schedule_interval='@daily', start_date=datetime(2024, 1, 1), catchup=False, default_args={ 'retries': 2, 'retry_delay': timedelta(minutes=5), 'on_failure_callback': on_failure_alert, 'execution_timeout': timedelta(hours=1) }, # Callbacks del DAG on_success_callback=on_success_summary, # SLA: debe completarse en menos de 2 horas sla_miss_callback=lambda context: logger.warning(f"SLA missed for {context['dag'].dag_id}"))

Tareas del pipeline

procesar = PythonOperator(task_id='procesar_datos', python_callable=procesar_datos_con_metricas, provide_context=True, dag=dag)

validar = PythonOperator(task_id='validar_metricas', python_callable=validar_metricas, provide_context=True, dag=dag)

notificar_exito = EmailOperator(task_id='notificar_exito', to=['*data-team@empresa.com*'], subject='Pipeline ETL Completado - {{ ds }}', html_content=''' <h2>Pipeline ETL Completado Exitosamente</h2> Ejecución: {{ ds }} Duración: {{ dag_run.duration }} Próxima ejecución: {{ next_ds }} ''', dag=dag)

Dependencias

procesar >> validar >> notificar_exito

2. **Configurar alertas adicionales:**

```python
# Tarea adicional para alertas de SLA
def verificar_sla(**context):
    """Verificar cumplimiento de SLA"""
    dag_run = context['dag_run']
    duration = (datetime.now() - dag_run.start_date).total_seconds()

    sla_seconds = 7200  # 2 horas

    if duration > sla_seconds:
        logger.warning(f"SLA violado: {duration:.1f}s > {sla_seconds}s")
        # Enviar alerta crítica

    return duration

verificar_sla_task = PythonOperator(
    task_id='verificar_sla',
    python_callable=verificar_sla,
    provide_context=True,
    dag=dag
)

# Agregar verificación de SLA al final
notificar_exito >> verificar_sla_task
```

Monitorear en producción:

```
# Ver estado del DAG
airflow dags list | grep pipeline_monitorado

# Ver tareas fallidas
airflow tasks failed pipeline_monitorado 2024-01-01

# Ver logs
airflow tasks logs pipeline_monitorado procesar_datos 2024-01-01
```

Verificación: ¿Qué métricas son más importantes para monitorear en un pipeline de datos? ¿Cómo decidirías entre enviar alertas por email vs Slack vs SMS?

Requerimientos:

Apache Airflow con configuración de email
Sistema de logging configurado
Conocimiento de métricas y KPIs