# 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



## Activities and Learning

### Task 1: Sistema de Monitoreo (10 minutos)

**¿Por qué monitorear pipelines?**

El monitoreo permite detectar problemas temprano y asegurar rendimiento óptimo.

**Elementos a monitorear:**

**Estado de tareas**: Success, failed, running
**Tiempo de ejecución**: Duración de cada tarea
**Uso de recursos**: CPU, memoria, disco
**Calidad de datos**: Registros procesados, errores
**SLA compliance**: Cumplimiento de acuerdos de servicio

**Configuración básica de monitoreo:**

```
# En dag definition
dag = DAG(
    'monitored_pipeline',
    # ... otros parámetros
    dagrun_timeout=timedelta(hours=2),  # Timeout general
    max_active_runs=1,                  # Una ejecución a la vez
    catchup=False                       # No ejecutar ejecuciones pasadas
)
```

### Task 2: Alertas y Notificaciones (10 minutos)

**Tipos de alertas en Airflow:**

**1. Email alerts:**

```
from airflow.operators.email import EmailOperator

default_args = {
    'email_on_failure': True,
    'email_on_retry': True,
    'email': ['alerts@empresa.com'],
    'retries': 2,
    'retry_delay': timedelta(minutes=5)
}
```

**2. Slack notifications:**

```
from airflow.operators.slack import SlackAPIPostOperator

notificar_slack = SlackAPIPostOperator(
    task_id='alertar_slack',
    channel='#data-alerts',
    text='🚨 Pipeline {{ dag.dag_id }} falló en tarea {{ task.task_id }}',
    slack_conn_id='slack_default',
    dag=dag
)
```

**3. Callbacks personalizados:**

```python
def on_failure_callback(context):
    """Callback ejecutado cuando una tarea falla"""
    task_instance = context['task_instance']
    error_message = str(context.get('exception', 'Unknown error'))

    print(f"Tarea {task_instance.task_id} falló: {error_message}")
    # Aquí puedes enviar alertas personalizadas

tarea = PythonOperator(
    task_id='mi_tarea',
    python_callable=mi_funcion,
    on_failure_callback=on_failure_callback,
    dag=dag
)
```

### Task 3: Métricas y Dashboards (10 minutos)

**Métricas importantes para trackear:**

**Performance metrics:**

```python
def trackear_performance(**context):
    """Trackear métricas de performance"""
    start_time = context['task_instance'].start_date
    end_time = context['task_instance'].end_date

    if start_time and end_time:
        duration = (end_time - start_time).total_seconds()
        print(f"Tarea completada en {duration} segundos")

    return duration
```

**Business metrics:**

Registros procesados por hora
Tasa de éxito de pipelines
Tiempo promedio de respuesta
Costo por ejecución