**Task 7. Alert project**

**Цель:** Получить базовые навыки работы с Pandas, необходимые для работы с данными и анализа, научиться применять их на практике.

**Задача:** Существует мобильное приложение, которое в режиме реального времени отправляет журналы с логами в формате CSV файлов.

1. Создайте систему, которая сможет обрабатывать и анализировать журналы с логами и отлавливать ошибки.
2. Создайте механизм отправки алертов для обработанных журналов с двумя основными правилами оповещения:

* Оповещение о более чем 10 фатальных ошибках менее чем за одну минуту
* Оповещение о более чем 10 фатальных ошибках менее чем за один час для конкретного bundle\_id

**Обязательные условия:**

* код работает хорошо, без каких-либо ошибок / багов и других сбоев
* хорошо структурированная логика и чистый код (соблюдение стандартов форматирования, организации архитектуры проекта, PEP8 и т.д.)
* решение, должно быть способно анализировать 100 миллионов записей в день
* система должна запускаться одной командой через docker-compose
* правильное оформление git репозитория
* создайте систему таким образом, чтобы в будущем можно было легко добавить новые правила для оповещения

Добавьте файл README, который объясняет архитектуру приложения, как его запустить. Каким образом можно добавить новые правила для оповещения. Необходимо объяснить, почему вы решили реализовать это именно так и почему использовали те или иные технологии.

Загрузите проект в репозиторий GitHub и поделитесь им со **своим тим лидом**, оповестив в Google Chat.

Dataset: [alert\_project\_data](https://drive.google.com/open?id=1FtL2NpnqGJbs35hxfsJTyFjPXTt7U_nH)

Названия колонок датасета: 'error\_code', 'error\_message', 'severity', 'log\_location', 'mode', 'model', 'graphics', 'session\_id', 'sdkv', 'test\_mode', 'flow\_id', 'flow\_type', 'sdk\_date', 'publisher\_id', 'game\_id', 'bundle\_id', 'appv', 'language', 'os', 'adv\_id', 'gdpr', 'ccpa', 'country\_code', 'date'

|  |  |
| --- | --- |
|  | from airflow.sensors.s3\_key\_sensor import S3KeySensor |
|  | from datetime import datetime, timedelta |
|  | from airflow import DAG |
|  | from airflow.operators.dummy\_operator import DummyOperator |
|  | from airflow.utils.dates import days\_ago |
|  |  |
|  | default\_args = { |
|  | 'owner': 'airflow', |
|  | 'start\_date': days\_ago(1), |
|  | 'email\_on\_failure': True, |
|  | 'email\_on\_retry': False, |
|  | 'retries': 2, |
|  | 'retry\_delay': timedelta(minutes=1), |
|  | 'execution\_timeout': timedelta(hours=1), |
|  | 'depends\_on\_past': False |
|  | } |
|  |  |
|  | dag = DAG( |
|  | 's3\_test', |
|  | start\_date=days\_ago(1), |
|  | max\_active\_runs=1, |
|  | schedule\_interval='\* 8 \* \* \*', |
|  | default\_args=default\_args, |
|  | catchup=False |
|  | ) |
|  |  |
|  | start = DummyOperator( |
|  | dag=dag, |
|  | task\_id='start' |
|  | ) |
|  |  |
|  | s3\_key = S3KeySensor( |
|  | task\_id='s3\_key', |
|  | bucket\_key='/data.csv', |
|  | aws\_conn\_id='aws\_test', |
|  | poke\_interval=0, |
|  | timeout=10, |
|  | soft\_fail=True, |
|  | bucket\_name='dp-external-repository', |
|  | verify=False) |
|  |  |
|  |  |
|  | end = DummyOperator( |
|  | dag=dag, |
|  | task\_id='end' |
|  | ) |
|  |  |
|  | start >> s3\_key >> end |

|  |  |
| --- | --- |
|  | from datetime import datetime, timedelta |
|  | from airflow.models import DAG |
|  | from airflow.operators.s3\_key\_sensor import S3KeySensor |
|  | from airflow.operators.python\_operator import PythonOperator |
|  | from airflow.utils.dates import days\_ago |
|  | schedule = timedelta(minutes=5) |
|  | args = { |
|  | 'owner': 'airflow', |
|  | 'start\_date': days\_ago(1), |
|  | 'depends\_on\_past': False, |
|  | } |
|  | dag = DAG( |
|  | dag\_id='s3\_key\_sensor\_demo\_dag', |
|  | schedule\_interval=schedule, |
|  | default\_args=args |
|  | ) |
|  | def new\_file\_detection(\*\*kwargs): |
|  | print("A new file has arrived in s3 bucket") |
|  |  |
|  | file\_sensor = S3KeySensor( |
|  | task\_id='s3\_key\_sensor\_task', |
|  | poke\_interval=60 \* 60, # (seconds); checking file every one an hour |
|  | timeout=60 \* 60 \* 12, # timeout in 12 hours |
|  | bucket\_key='/data.csv', |
|  | bucket\_name=None, |
|  | wildcard\_match=False, |
|  | dag=dag) |
|  | print\_message = PythonOperator(task\_id='print\_message', |
|  | provide\_context=True, |
|  | python\_callable=new\_file\_detection, |
|  | dag=dag) |
|  |  |
|  | file\_sensor >> print\_message |