Here's a markdown document on how to create a file watcher in Apache Airflow:

Creating a File Watcher in Apache Airflow

Apache Airflow provides a built-in sensor called `FileSensor` that allows you to monitor the existence of files in a filesystem. This sensor is particularly useful when you need to wait for a file to appear before triggering subsequent tasks in your DAG.

Using FileSensor

The `FileSensor` is part of Airflow's core functionality and can be easily implemented in your DAGs.

Basic Usage

Here's a basic example of how to use the `FileSensor`:

```
from airflow import DAG
from airflow.sensors.filesystem import FileSensor
from datetime import datetime, timedelta

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 1, 1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

dag = DAG('file_watcher_dag', default_args=default_args,
          schedule_interval=timedelta(days=1))

file_sensor_task = FileSensor(
    task_id='watch_for_file',
    filepath='/path/to/file/myfile.txt',
    fs_conn_id='my_filesystem_connection',
    poke_interval=30,
    timeout=600,
```

```
    dag=dag
)
```

In this example: - `filepath`: The path to the file you're watching for. - `fs_conn_id`: The connection ID for the filesystem (set up in Airflow connections). - `poke_interval`: How often (in seconds) the sensor should check for the file. - `timeout`: How long (in seconds) the sensor should wait before timing out.

Advanced Configuration

You can further customize the `FileSensor` with additional parameters:

```
file_sensor_task = FileSensor(
    task_id='watch_for_file',
    filepath='/path/to/file/*.txt',
        # Use wildcards for multiple files
    fs_conn_id='my_filesystem_connection',
    poke_interval=30,
    timeout=600,
    mode='poke',  # or 'reschedule'
    soft_fail=True,
    recursive=True,  # For searching in subdirectories
    dag=dag
)
```

- `mode`: 'poke' (default) keeps the task slot occupied, 'reschedule' frees up the worker slot.
- `soft_fail`: If True, the DAG will continue even if the sensor times out.
- `recursive`: If True, searches for the file in subdirectories as well.

Best Practices

1. **Choose appropriate intervals**: Set the `poke_interval` based on how frequently you expect the file to arrive and how quickly you need to react.

2. **Set reasonable timeouts**: Ensure the `timeout` is long enough to accommodate expected delays but not so long that it holds up your entire pipeline.

3. **Use wildcards wisely**: When watching for multiple files, use wildcards in the `filepath` parameter.

4. **Consider using 'reschedule' mode**: For long-running sensors, 'reschedule' mode can be more efficient as it frees up worker slots.

5. **Implement error handling**: Use `soft_fail=True` if you want the DAG to continue even if the file doesn't appear within the timeout period.

Limitations

- `FileSensor` is designed for local filesystems. For cloud storage or distributed systems, you may need to use specific sensors (e.g., `S3KeySensor` for Amazon S3).
- It only checks for file existence, not content changes or file completeness.

By utilizing the `FileSensor` in Airflow, you can create robust file-watching mechanisms in your data pipelines, ensuring that downstream tasks only run when the required files are available.