

Monitoring, Logging, and Performance Optimization

by Fariz Wakan

May 15th, 2024



Logging & Monitoring in Airflow

The screenshot shows a web browser window displaying the Apache Airflow documentation page for Logging & Monitoring. The browser's address bar shows the URL: `airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment/logging-monitoring/index.html`. The page features the Apache Airflow logo and a navigation menu with links to Community, Meetups, Documentation, Use-cases, Announcements, Blog, and Ecosystem. On the left side, there is a sidebar with a version dropdown set to 2.9.0, a search bar labeled 'Search docs', and a 'CONTENT' section listing various topics like Overview, Quick Start, Installation of Airflow™, Security, Tutorials, How-to Guides, UI / Screenshots, Core Concepts, and Authoring and Scheduling. The main content area has a breadcrumb trail: Home / Administration and Deployment / Logging & Monitoring. The title 'Logging & Monitoring' is prominently displayed. Below the title, the text states: 'Since data pipelines are generally run without any manual supervision, observability is critical.' It then explains that Airflow has support for multiple logging mechanisms and a built-in mechanism to emit metrics for gathering, processing, and visualization in other downstream systems. It also mentions that Airflow supports the ability to detect errors in the operation of Airflow itself, using an Airflow health check, and that it supports real-time error notification via integration with Sentry. At the bottom right, there is a button with a GitHub icon and the text 'Suggest a change on this page'.

Logging & Monitoring — Airflow

airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment/logging-monitoring/index.html

New Chrome available

Apache Airflow

Community Meetups Documentation Use-cases Announcements Blog Ecosystem

Version: 2.9.0

Search docs

CONTENT

- Overview
- Quick Start
- Installation of Airflow™
- Security
- Tutorials
- How-to Guides
- UI / Screenshots
- Core Concepts
- Authoring and Scheduling
- Administration and Deployment
- Production Deployment

Home / Administration and Deployment / Logging & Monitoring

Logging & Monitoring

Since data pipelines are generally run without any manual supervision, observability is critical.

Airflow has support for multiple logging mechanisms, as well as a built-in mechanism to emit metrics for gathering, processing, and visualization in other downstream systems. The logging capabilities are critical for diagnosis of problems which may occur in the process of running data pipelines.

In addition to the standard logging and metrics capabilities, Airflow supports the ability to detect errors in the operation of Airflow itself, using an Airflow health check. Since Airflow is generally used for running data pipelines in production, it also supports real-time error notification via integration with Sentry.

- Logging and Monitoring architecture
- Logging for Tasks
- Advanced logging configuration

Suggest a change on this page

Best Practices in Airflow

The screenshot shows a web browser window displaying the Apache Airflow documentation page for Best Practices. The browser's address bar shows the URL `airflow.apache.org/docs/apache-airflow/stable/best-practices.html#`. The page features the Apache Airflow logo and a navigation menu with links to Community, Meetups, Documentation, Use-cases, Announcements, Blog, and Ecosystem. On the left, a sidebar lists various topics, with 'Best Practices' and 'Writing a DAG' highlighted. The main content area is titled 'Home / Best Practices' and 'Best Practices'. It explains that creating a new DAG is a three-step process: writing Python code, testing the code, and configuring environment dependencies. A list of these steps is provided. Below this, it states that the tutorial will introduce best practices for these steps. The section 'Writing a DAG' begins with the text: 'Creating a new DAG in Airflow is quite simple. However, there are many things that you need to take care of to ensure the DAG runs or failure does not produce unexpected results.' On the right, another sidebar lists sub-topics under 'Best Practices', including 'Writing a DAG', 'Creating a Custom Operator', 'Creating a task', 'Deleting a task', 'Communication', 'Top level Python Code', and 'How to check if my code is "top-level" code'. A 'Suggest a change on this page' button is visible in the bottom right corner.

Best Practices — Airflow Doc

airflow.apache.org/docs/apache-airflow/stable/best-practices.html#

New Chrome available

Apache Airflow

Community Meetups Documentation Use-cases Announcements Blog Ecosystem

Home / Best Practices

Best Practices

Creating a new DAG is a three-step process:

- writing Python code to create a DAG object,
- testing if the code meets your expectations,
- configuring environment dependencies to run your DAG

This tutorial will introduce you to the best practices for these three steps.

Writing a DAG

Creating a new DAG in Airflow is quite simple. However, there are many things that you need to take care of to ensure the DAG runs or failure does not produce unexpected results.

Integration

Public Interface of Airflow

▼ Best Practices

▼ Writing a DAG

Creating a Custom Operator/Hook

Creating a task

Deleting a task

Communication

Top level Python Code

How to check if my code is "top-level" code

Dynamic DAG Generation

Airflow Variables

Timetables

Triggering DAGs after changes

Example of watcher

Best Practices

Writing a DAG

Creating a Custom Operator

Creating a task

Deleting a task

Communication

Top level Python Code

How to check if my code is "top-level" code

Suggest a change on this page



Thank you!

See you in the next session 💪