Effective workflow management with Apache Airflow 2.0



**Use-Cases - New features - Best Practices**

In daily business, everything is directly or indirectly associated with digital processes and workflows. This fact makes it even more important to use a central and reliable workflow management platform. As an open-source tool, Apache Airflow appeals to both start-ups and large companies. Thanks to its outstanding scalability, it can be tailored precisely to a wide range of demands. It is easy to learn due to its intuitive handling, but difficult to master because of the extensive configurations and features.

What are the concepts behind Apache Airflow?

How can your organization maximize the benefits of the platform?

#### This whitepaper introduces Apache Airflow 2.0 with focus on the practical application.

Version 01/20



Shaping Business Intelligence

We are driven by the desire to get the best out of data by using artificial intelligence methodologies and data warehousing alike. Telling the story of data in dashboards using the latest business intelligence software is our passion.

Longstanding experience in various industries and a team of highly motivated specialists are the ingredients for our successful products and projects with our renowned customers.

### Enabling Analytics driven Applications!

NextLytics AG [www.nextlytics.com](https://www.nextlytics.com/)



+49 (0)6192 470 71-0



# Challenges of a modern workflow management

The digital century has brought a vast amount of new processes that generate, process and store data on a daily basis. While manual processing is largely eliminated, automation at data level continues to gain importance. At the decision- maker level, for example, recurring analyses and reports are generated automatically. Here, workflow management ensures timely and reliable creation by executing predefined tasks. Workflow management is therefore introduced at the operational level.

workflow example with parallel execution of tasks

process

categorical data

get data

apply model

process

categorical data

process

numerical data

While a workflow focuses on the work to be done, planning, execution and monitoring are the most important aspects of workflow management. Of course, workflow management is not an end-in-itself task. It depends entirely on the quality of the underlying workflows. For this reason, there should be a certain degree of **flexibility within the workflow definition**. Condi- tional execution, the integration of external dependencies and the embedding of task groups are important in this respect.

With progressive digitization and the expansion of the Internet of Things (IoT), the processing of event-based real-time data is an increasing trend. Here, a **dynamic adaptation of the workflow** to the varying data volume should be possible. For Big Data, **scalability and the possibility of containerization** is a further selection criterion for a suitable workflow management platform for workflows with inhomogeneous requirements in terms of hardware and software environment.

#### An overview of different workflow management systems

The number of possible workflow management platforms is large. In addition to several specialized paid tools, there are a whole range of free open source tools that offer an impressive range of functions and have been developed for productive corporate applica- tions. In the following, the widely used tools Luigi, Apache Airflow and Prefect are presented.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Luigi | Apache Airflow | Prefect |
| free of charge |  |  | limited |
| scalability | limited |  |  |
| data pipelines |  | limited |  |
| web interface | limited |  |  |

###### Luigi

The Python module Luigi, originally developed by Spotify, is used to create complex pipelines of batch jobs. The concept is that *tasks* work with the file output *(target*) of their predecessors. A simple visualization is also included in the tool. A major drawback is the high entry hurdle, the difficult maintenance and the limited scalability.

###### Apache Airflow

The free of charge tool Apache Airflow convinces by its comprehensive adaptability. Workflows are defined, planned and monitored. Conceptually, data is stored between the process steps. The realization of data pipelines is not directly possible. Nevertheless, data can be exchanged between tasks via XCOM variables. Planning and triggering is calendar- based using the usual cron notation.

###### Prefect Core

Prefect was developed out of the Airflow community. The focus here is on straightforward data workflows and the exchange of data between tasks using state objects. The defini- tion of the workflows is performed functionally. The user interface is very modern and the Python API is easy to learn. A disadvantage, however, is that the tool is currently still in the development stage and the orchestration of the workflows in the cloud requires a paid upgrade to Prefect Cloud.

#### Workflow management with Apache Airflow

In the section below, the focus is on Apache Airflow as the de facto standard for workflow management. A lot has happened since Airbnb first presented the open source project Airflow in 2015. Meanwhile Airflow is a top level project of the Apache Software Foundation and is being further developed there with great commitment. Not only due to the large community in the background, Airflow enjoys such popularity. The next section provides an overview of the top features.

##### Web interface Scalability

In addition to the command line interface, the majority of the functions can be used via the web interface. This interface is designed intuitively and has a role-based access control (RBAC). The graphical representation provides an excellent overview even with a large number of workflows.

The components of Apache Airflow are largely interchangeable and therefore also available in scalable configurations. Scaling can be realized with Dask, Kubernetes and a combination of Celery and Redis.

##### Complex workflows Integration to numerous third-party systems



Within the workflows, the order of the tasks can be easily adjusted. Parts of a workflow are reusable and conditional branching is also possible.

Since Airflow is used for orchestration of external systems, a large number of integrations can be found already in the basic installation. Further interfaces are implemented through the community contributions.

##### Simple Python code Customizability through plugins and macros

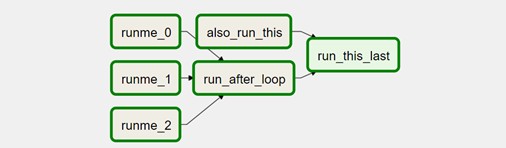


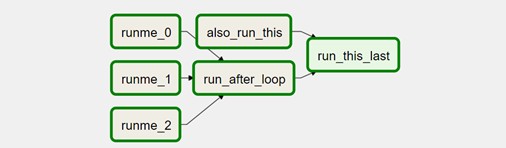
The workflow definitions are implemented using Python code. A basic understanding is sufficient for the creation of a workflow.

For many special use cases there are already user-defined plugins online, which often serve as inspiration for upcoming official extensions of Airflow.

# The basic concept

#### Workflows in Apache Airflow

A workflow is created object-oriented in Apache Airflow using the programming language Python. There the workflow is defined as a **DAG (directed acyclic graph)**.This structure supports the form of workflows, which do not contain any loops, but a fixed order of the individual tasks. For each workflow a start-date and a schedule\_interval are defined, which together define the later schedule of the workflow.



Example representation of a workflow in the graph view in Apache Airflow

While the DAG object is the workflow skeleton, the tasks are defined separately and are assigned to the DAGs accordingly. A distinction is made between operators and sensors. In contrast to the operators, the **sensors** do not execute tasks but check preconditions. These are often used to check for the availability of data and files in close intervals. As soon as the check is positive, the workflow continues.

The actual work is then performed by specialized **operators**. In this case, various third-party systems are easily controlled by their own operators. If connection data is required for this purpose, it is encrypted and stored in the backend of Airflow. Typical purposes of operators are:

* Running a Python script with the PythonOperator
* Transferring data from Spark to a database with the SparkJDBCOperator
* Execute a HTTP request with the HttpOperator

After the tasks have been created, they are put in order. Even workflows branched by a condition are possible with BranchingOperator. The created Python file is then stored in the DAGs folder in the Airflow directory. From here Airflow recognizes the work- flows independently and takes over the further planning, monitoring and execution.

#### Components of Apache Airflow

The execution of the workflows requires the interaction of several components. In its core, Airflow consists of a scheduler, a metadata database and a web server, which offer a special degree of flexibility and scalability due to their configuration options.

A **scheduler**, together with the linked **executor**, ensures the tracking and triggering of the stored workflows. While the scheduler tracks which task can be executed next, the executor takes over the selection of the worker and the following communication. The selection of the executor depends on the desired scaling.

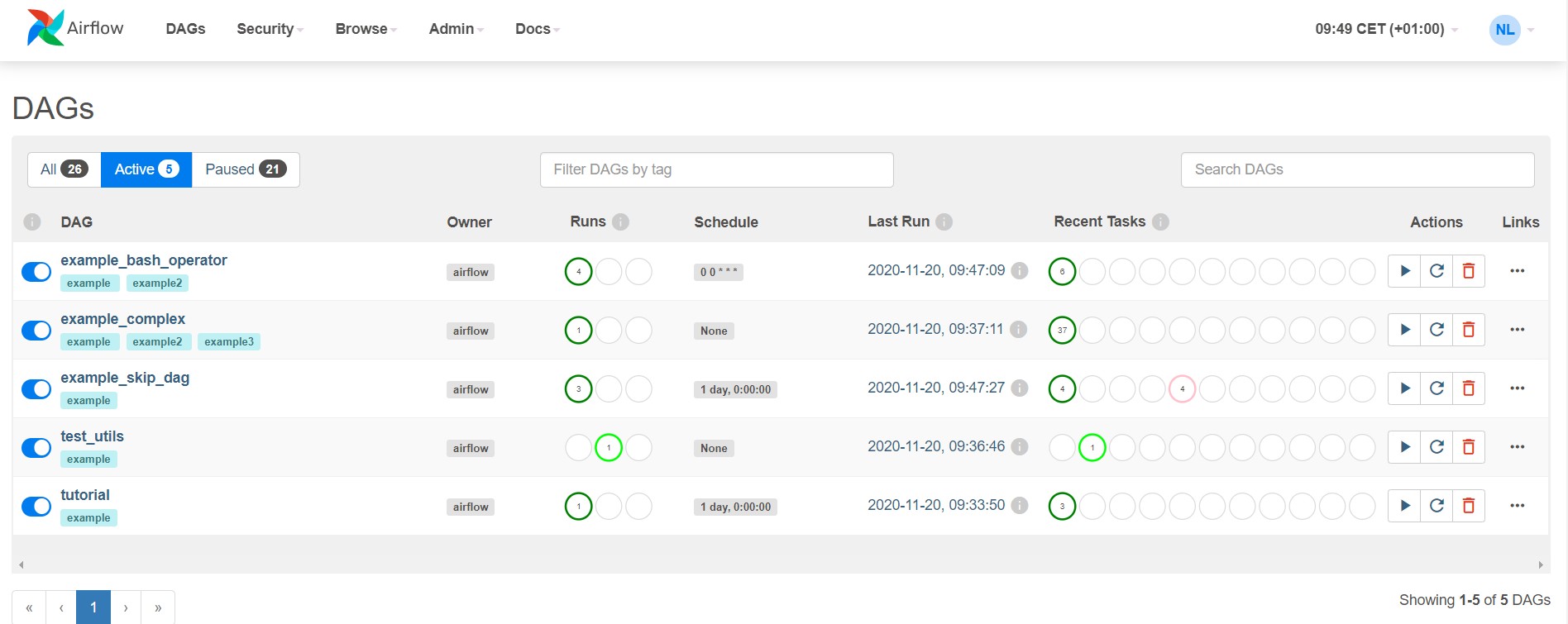
* The *SequentialExecutor* is the default setting and only executes one task at a time. It is suitable for an installation test.
* The *LocalExecutor* has several workers available for the parallelization of tasks. These are located in a single node architecture on the local machine. For few workflows with the same demands, the cost-benefit ratio is optimized here.
* With the *CeleryExecutor* the workers can be distributed on several machines. The increase in availability is essential for a high number of workflows.
* With the *KubernetesExecutor* the worker-pods are created on demand. These can therefore be scaled up and down automatically. Specific requirements regarding the environment or performance can also be realized.

Among other things, the **metadata database** stores infomation of workflow runs and encrypted connection details to external systems. Typically, a postgres database is used for this purpose.

The **web server** enables easy user interaction in a graphical interface. Control functions (e.g. starting and pausing workflows) and a simple and well-structured monitoring are integrated. On the dashboard the status of the DAG runs is displayed.

Web interface Apache Airflow. The status of the workflow runs is visible on the right (DAG Runs).

The status of the tasks of the last workflow is displayed in the middle (Recent Tasks).



#### Use Cases Apache Airflow

The flexibility and scalability of the individual components makes Airflow almost universally applicable. Airflow is just as popular as a workflow management tool for Big Data processes or as a substitute for simple cron jobs. Depending on the application scenario, the architecture can be planned and adapted accordingly. In extreme cases, thousands of workflows per day can be executed in a cluster or just a single workflow that should be managed and monitored via a graphical user interface.

Thus, the desired application purpose is always decisive for the right architecture and configuration.

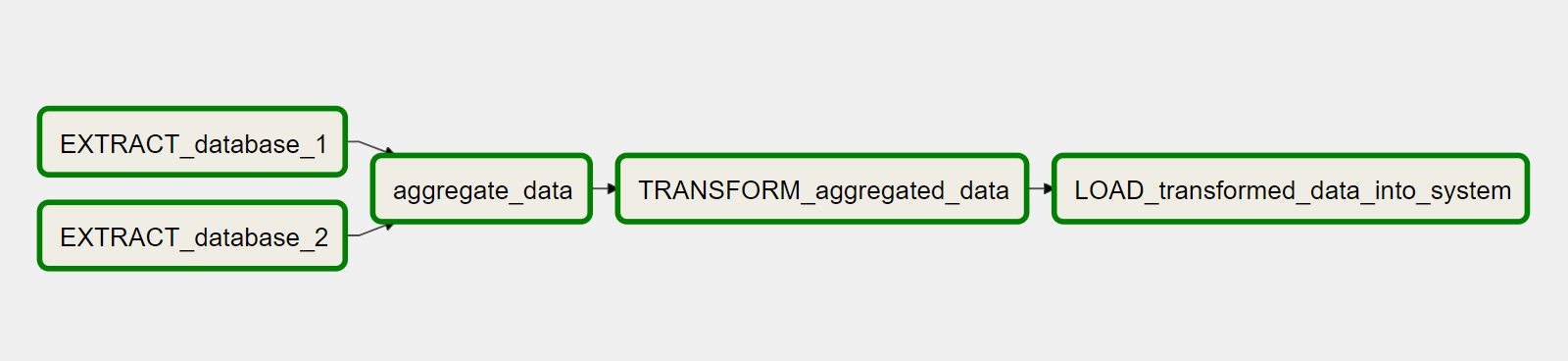
For the entry into Apache Airflow the realization of a practical application example with moderate complexity is recommended. Especially when implementing business-critical workflows, the first step is to gain experience on a test system.

In the following section, some classical application examples in the SAP BW environment are listed and described. These are kept rather abstract and thus can be used in many departments and business areas.

ETL Workflow

In an ETL workflow, the data is first extracted, then transformed and loaded into a database. During this process, data from different unstructured data sources is trans- formed into a target schema. These types of workflows are crucial for technical and business analysis.

Sample ETL workflow with the steps Extract, Aggregate Transform, Load into Airflow



Since the basic idea of Apache Airflow is rooted in ETL best practices, many helpful features are available, such as parallel and prioritized execution of tasks and automatic repetition and notification in case of errors. The logs, metadata and configuration settings are centrally stored and accessible via the web interface.

Ideally, the data is processed incrementally in Airflow. The reduced data volume significantly improves the usability of the results and the performance. Within Airflow, context variables are provided for each run of a workflow to support this approach. The context variables provide the execution time of the current, previous and (planned) future run. The data extraction for processing can thus be dynamically adapted to the execution interval.



Machine Learning Workflow

With the increasing popularity of Machine Learning in companies, the desire for a suitable platform for the central management of Machine Learning workflows is growing. Apache Airflow is the best solution to meet these requirements. The realization of complex dependencies within the workflow and the distributed execution with e.g. Kubernetes are common. In addition, the administration and monitoring of the workflows via the web interface is equally possible for both technical and business users.

In the scope of a Machine Learning application many single steps can be automated. The preparation of training data, model training, model evaluation and even the deployment of the model can be handled with Airflow.



Machine Learning Workflow with integrated steps

The individual steps are either integrated into a single workflow or executed separately in several workflows. Differentiated workflows allow processing to continue with a previous version of the data or model in case of errors.

The process of training data preparation resembles the already presented ETL process. This process can be extended by special feature extraction steps. For the detection of poten- tial error sources, a verification and printout of the data scope and statistical properties of the data is very useful.

Process integration SAP BW

Through the REST API integrated within Airflow, a connection to the SAP Business Ware- house is easily possible. The API is no longer experimental since Airflow 2.0 and has a vast variety of endpoints.

In SAP BW, a REST client can be created for the other side of the integration, which starts selected workflows as required via the Airflow API and sends parameters which are required for execution. In the BW process, the status of the workflow run and the log output of the tasks can be tracked concurrently. For example, a conditional file upload can be started, which only transfers files with modifications into a database.

With the help of the HttpOperator the triggering of BW processes by Airflow is also possible. For this purpose, the development of an own operator for a user-friendly interface is recommended. The logic is then centrally encapsulated and is not distributed over many different workflow definitions.

In addition to the use cases listed here, there are further special application examples in everyday business, which are covered by the custom developments of the Airflow Com- munity. The numerous plug-ins, user-defined operators and macros solve contemporary challenges of workflow management.

Triggering an Airflow workflow from SAP BW with synchronous process tracking

SAP BW process chain process chain starter

Start DAG in Airflow

Airflow

trigger DAG in Airflow

check status

send parameter

{„system“:“<sytem & client>“}

check status request status dag\_status: running

step 1

step 2

request status

dag\_status: successful

step 3

success

failed

Next step

Next step

#### Apache Airflow 2.0

Since 17.12.2020 there is a new major release of Airflow, which brings new features as well as improved usability, a new design and new integrations with other systems. The next section will list the biggest changes and their meaning in more detail.

Improved scheduler performance and reliability

Guaranteed accessibility by multiple instances of the web server and a scaling at your choice by the selection of the executor has been possible for quite some time within Airflow. The only bottleneck left is the scheduler, which is only available with one instance (in the standard installation!). With a high number of workflows, this results in a long latency time until a workflow starts. A further problem is the possible technical failure of the single scheduler instance, which temporarily prevents the initiation of any workflow.

In the new release this single point of failure is eliminated. Starting with Airflow 2.0, load distribution between several schedulers is possible. This makes the scheduler component highly available and reduces the latency time. The design is active-active, which means that there is no superior scheduler and each scheduler instance executes all functions. Due to the extensive changes in the architecture design, even the performance of a single scheduler benefits. In a [performance test in advance](https://www.youtube.com/watch?v=YLsGVFB8Pws&list=PLGudixcDaxY3RGLSlWoN_cEEXhIT1OPmj&index=10), the scheduler overhead of *a single scheduler instance* could be reduced by 79% in extreme cases (execution of 20,000 dummy tasks).

(Refernce: Keynote: Future of Airflow - [https://www](http://www.youtube.com/watch?v=YLsGVFB8Pws&list=PLGudixcDaxY3RGLSlWoN_cEEXhI-).y[outube.com/watch?v=YLsG](http://www.youtube.com/watch?v=YLsGVFB8Pws&list=PLGudixcDaxY3RGLSlWoN_cEEXhI-)VFB8P[ws&list=PLGudixcDaxY3RGLSlWoN\_cEEXhI](http://www.youtube.com/watch?v=YLsGVFB8Pws&list=PLGudixcDaxY3RGLSlWoN_cEEXhI-)- T1OPmj&index=10)

Sophisticated REST API

While in Airflow 1.x the REST API was still marked as experimental, in the new version it has been greatly expanded and numerous endpoints have been added. The main focus is on replicating all functions of the command-line interface and the web interface. For instance: retrieving the logs for a task or creating and deleting a workflow run. Even connections and variables can be set by API calls.

Functional workflows with the Taskflow API

Before the significance of functional workflows for Airflow becomes clear, the previous way of cross-task communication via XCOM must be understood. Conceptually, no data pipelines are intended within Airflow. XCOM is primarily used to transfer a storage location or a table name via so-called XCOM variables which are explicitly pushed and pulled.

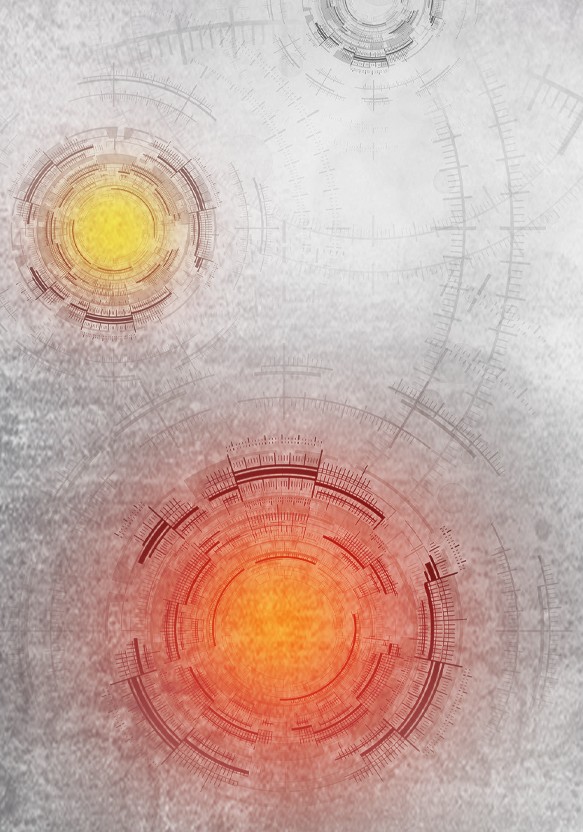
These variables are exchanged via the metadata database and are limited to the size of a BINARY LARGE OBJECT (BLOB). A future support of different XCOM backends like S3 or HDFS is planned to increase the maximum storage size.

With the introduction of a functional workflow definition (TaskFlow API) in the new major release Airflow 2.0 the way information is exchanged between tasks does not change - XCOM is still used - but workflows are created much easier using Python functions. The output of the previous function can be passed as a variable to the following function.

The passing of messages is thus explicitly specified and thus the task order is determined implicitly and does not need to be specified separately.

Abolishing the Flask-Admin UI

The new release focuses on security! While two user interfaces were available in Airflow 1.10, only the Role-Based-Access-Control (RBAC) interface will be available in the new version. The role-based access reduces administrative work and allows password protected access to the web interface.



Smart Sensors

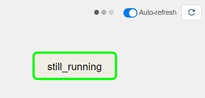
Previously, it was possible to reach the maximum number of parallel running tasks but while the machines were almost idle. This case occurs when many sensors are used in the workflows. These have a poke\_intervall to check a certain criterion. Until the criterion is reached or the allowed time window is exceeded, the sensor task remains active and thus occupies one of the available workers. Thereby the checking of the criterion is connected with very low workload.

Airflow 2.0 implements the idea of bundling such sensor tasks centrally and executing them in batches. This dramatically reduces the amount of processes. The feature is marked as early access and may therefore be subject to further changes in the future.

Reusable parts through TaskGroups

The reuse of code parts in the workflow definition saves time. While SubDAGs were the method of choice in the previous release, they are now replaced by the TaskGroups. In contrast to the SubDAGs, these support parallel execution even in the grouped area. In the web interface, these grouped areas are also highlighted separately.

In addition to the highlights presented here, one point should not remain unmentioned, which will delight the long-time Airflow user: The status of a workflow run can now be tracked even more effortlessly. In the graph view, an **auto-refresh** has been provided at the request of the community.The constant updating of the web view is thus a thing of the past in Airflow 2.0.



In order to stay up to date, the [roadmap of Apache Airflow](https://cwiki.apache.org/confluence/display/AIRFLOW/Airflow%2B2.0%2B-%2BPlanning) can be followed. Some announced features like the DAG versioning are already in work, but were postponed to a later release due to time constraints.

# 7 recommendations for the optimal usage of Apache Airflow

Last but not least, here are some practical tips on how to make your workflow management with Apache Airflow as optimal and robust against errors as possible.

###### Idempotent tasks

Errors within the exectution of workflows are inevitable. These are caught by the workflow management and the corresponding task packages are repeated. To prevent the repeti- tions from causing problems, the tasks should be designed idempotent. This means that repeated execution should always lead to the same result. This ensures that there are no duplicate entries in the target table. For this purpose, old, erroneous entries should first be deleted before new results are written to the database.

###### Be aware of the execution times

In the ETL context, the execution\_time is not the same as the runtimes. While the execution\_time is the start time stamp of a planning period, the workflow run always starts at the end of this period. If a workflow should run every Tuesday at 08:00 AM, the corresponding workflow will not start until Tuesday at 07:59 AM one week later.

###### Consideration of maintenance

Regular maintenance is a component of every IT system. In a workflow, this leads to data sources and third-party systems being temporarily unavailable. In Airflow itself, the workflows can be paused for such a case. To ensure that your system is not subsequently brought to its knees under the load of the caught up runs, you should use the Latest- OnlyOperator. Here, only the most current run is executed and the runs in the main- tenance period are not rescheduled.

###### Clear workflow responsibility

Within a workflow there should be a distinct purpose. The complexity should not be increased unnecessarily. If parts of the workflow are not necessarily connected, these parts should be moved to another workflow. This way, an error at the beginning of the work- flow does not lead to a complete failure. Especially in the Machine Learning area, data preparation for the training, the training of the model and the prediction can be separated from each other. If the training of the model fails, the forecast can still be performed with a predecessor model.

###### Encapsulating the logic

For a desired behaviour of an operator, it can be functionally adapted and extended by a user defined operator class. The passed parameters can also be replaced or standardized. For example, the BashOperator can be adapted to use a predefined Anaconda envi- ronment for the execution of a Python script.

###### Setting priorities

If the number of workflows increases very quickly, temporary bottlenecks can occur due to overlapping execution times. The priority\_weight parameter can be used to control which workflow is preferred and to what extent. This is set per workflow or per individual task.

###### Using variables for more flexibility

In Airflow there are many possibilities to keep the workflows flexible. With encrypted variables in the backend and the context variables at runtime there are two options to choose from. For example, the file path for a FileSensor can be created as a variable in the backend and can be altered afterwards via the user interface.

# Conclusion

As an open source platform with regular functional enhancements, Apache Airflow meets the modern requirements of workflow management. In particular, the expansion with regard to Big Data and the extensive API are among the advantages compared to alternative solutions. With the introduced functional workflows, the design of data pipelines is perspectively simplified and thus a weakness of Airflow is addressed. Apache Airflow‘s status as the de facto workflow management platform thus remains justified.

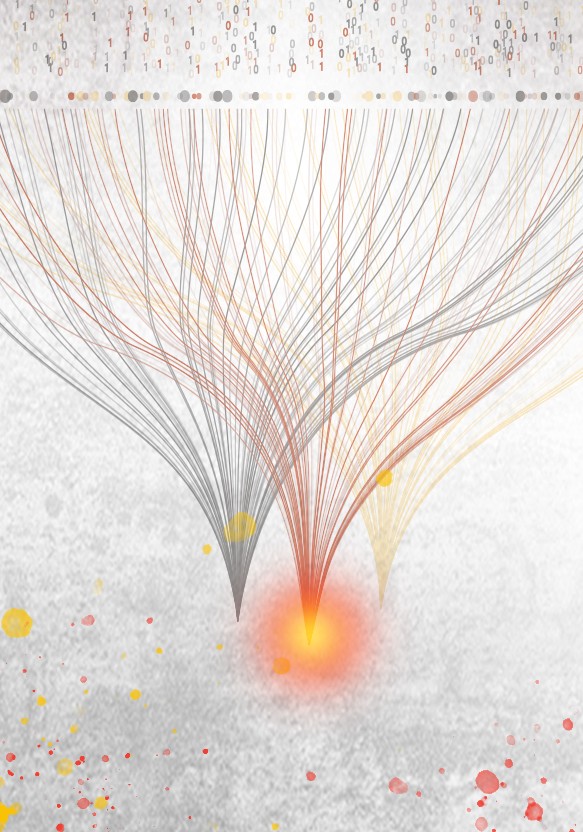
Do you have any questions or need support in switching to Apache Airflow?

We will be happy to assist you with the conceptual design and implementation of your workflow management with Airflow and demonstrate how to get the optimal benefit/effort ratio for your needs.



[Talk to us!](https://bit.ly/3mrJ2t8)

Autorin: Co-Autor:

Luise Wiesalla

Workingstudent Machine Learning & Data Science

#### Lars Schymonski

Consultant

Machine Learning & Data Science

**Ask your question without obligation and we are happy to help! Your contact in sales:**

**Your contact for technical questions:**

**Your contact in sales:**

Further information about the topic

[Machine Learning](https://bit.ly/3mqWBbD) [Apache Airflow](https://bit.ly/3nJ6XFz)

Alexander Rabe

Consultant Machine Learning & Data Science

[Alexander.Rabe@NextLytics.com](mailto:Alexander.Rabe@NextLytics.com)

#### Lars Schymonski

Consultant Machine Learning & Data Science

[Lars.Schymonski@NextLytics.com](mailto:Lars.Schymonski@NextLytics.com)

#### Sebastian Uhlig

Senior Consultant SAP BW & BI [sebastian.uhlig@NextLytics.com](mailto:sebastian.uhlig@NextLytics.com)

## Always stay up to date...



**... with our newsletter**

#### Don‘t miss any news about SAP BI, Dashboarding, SAP Analytics Cloud, Lumira Designer or Machine Learning and sign up for our newsletter now.

[Sign up for our newsletter](https://bit.ly/3modSCk)

**Unlocking AI for BI!**

NextLytics AG [www.nextlytics.com](https://www.nextlytics.com/)



+49 (0)6192 470 71-0