Tech Geek Next »

Top Airflow Interview Questions and Answers (2022)

Airflow Interview Questions and Answers

Q: What is Airflow?
Ans:

Apache Airflow is an open-source workflow management platform. It began in October 2014 at Airbnb as a solution for managing the company's increasingly complex workflows. Airbnb's creation of Airflow enabled them to programmatically author,

(Extract, Transform, Load) workflow orchestration tool.

Q: What issues does Airflow resolve?
Ans:

- Crons are an old technique of task scheduling.
- Scalable
- Cron requires external assistance to log, track, and manage tasks. The Airflow UI is used to track and monitor the workflow's execution.
- Creating and maintaining a relationship between tasks in cron is a challenge, whereas it is as simple as writing Python code in Airflow.
- Cron jobs are not reproducible until they are configured externally. Airflow maintains an audit trail of all tasks completed.

Checkout our related posts :

Jenkins Interview Questions

CI CD DevOps Interview Questions

SonarQube Interview Questions

Openshift Interview Questions

Spring Security Interview Questions

Spring Cloud Interview Questions

oAuth2 Interview Questions

SAML Interview Questions

CloudFormation Interview Questions

Architect Interview Questions

Microservices Interview Questions

Protractor Interview Questions

CloudFormation Interview Questions

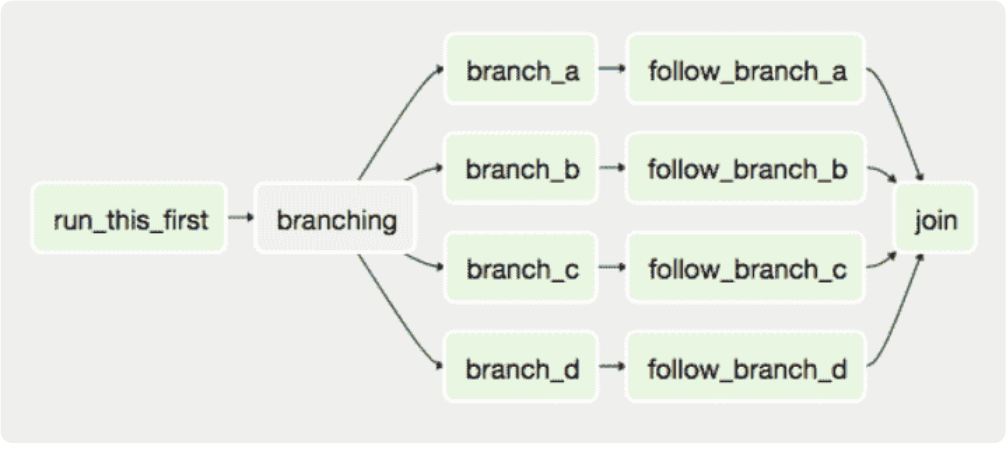
Amazon EC2 Interview Questions

Q: Explain how workflow is designed in Airflow?
Ans:

- A **directed acyclic graph (DAG)** is used to design an Airflow workflow. That is to say, when creating a workflow,

into a graph to form a logical whole.

- The overall logic of your workflow is based on the shape of the graph. An Airflow DAG can have multiple branches, and you can choose which ones to follow and which to skip during workflow execution.



- Airflow could be completely stopped, and able to run workflows would then resume through restarting the last unfinished task.
- It is important to remember that airflow operators can be run more than once when designing airflow operators. Each task should be idempotent, or capable of being performed multiple times without causing unintended consequences.

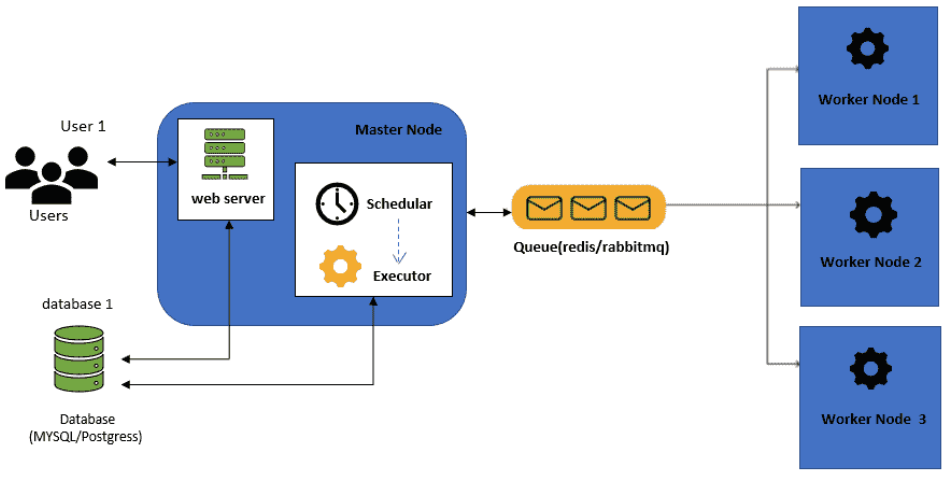
Q: Explain Airflow Architecture and its components?
Ans:

There are four major components to airflow.

1. **Webserver**

   This is the Airflow UI built on the Flask, which provides an overview of the overall health of various DAGs and helps visualise various components and states of every DAG. For the Airflow setup, the Web Server also allows you to manage users, roles, and different configurations.

2. **Scheduler**



3. **Executor**
    - Executor is another internal component of the scheduler.
    - The executors are the components that actually execute the tasks, while the Scheduler orchestrates them. Airflow has different types of executors, including SequentialExecutor, LocalExecutor, CeleryExecutor and KubernetesExecutor. People generally choose the executor which is best for their use case.
4. **Worker**

    Workers are responsible to run the task that the executor has given them.
5. **Metadata Database**
    - Airflow supports a wide range of metadata storage databases. This database contains information about DAGs, their runs, and other Airflow configurations such as users, roles, and connections.
    - The DAGs' states and runs are shown by the Web Server from the database. This information is also updated in the metadata database by the Scheduler.

Q: What are the types of Executors in Airflow?
Ans:

The executors are the components that actually execute the tasks, while the Scheduler orchestrates them. Airflow has different types of executors, including SequentialExecutor, LocalExecutor, CeleryExecutor and KubernetesExecutor. People generally choose the executor which is best for their use case

Tech Geek

Only one task is executed at a time by SequentialExecutor. The scheduler and the workers both use the same machine.

- **LocalExecutor**

  LocalExecutor is the same as the Sequential Executor, except it can run multiple tasks at a time.

- **CeleryExecutor**
  - Celery is a Python framework for running distributed asynchronous tasks.
  - As a result, CeleryExecutor has long been a part of Airflow, even before Kubernetes.
  - CeleryExecutors has a fixed number of workers on standby to take on tasks when they become available.

- **KubernetesExecutor**

  Each task is run by KubernetesExecutor in its own Kubernetes pod. It, unlike Celery, spins up worker pods on demand, allowing for the most efficient use of resources.

Q: What are the pros and cons of SequentialExecutor?
Ans:

Pros:

1. It's simple and straightforward to set up.
2. It's a good way to test DAGs while they're being developed.

Pros:

1. It isn't scalable.
2. It is not possible to perform many tasks at the same time.

Tech Geek

Q: What are the pros and cons of LocalExecutor?
Ans:

Pros:

1. Able to perform multiple tasks.
2. Can be used to run DAGs during development.

Cons:

1. The product isn't scalable.
2. There is only one point of failure.
3. Unsuitable for use in production.

Q: What are the pros and cons of CeleryExecutor?
Ans:

Pros:

1. It allows for scalability.
2. Celery is responsible for managing the workers. Celery creates a new one in the case of a failure.

Cons:

1. Celery requires RabbitMQ/Redis for task queuing, which is redundant with what Airflow already supports.
2. The setup is also complicated due to the above mentioned dependencies.

Q: What are the pros and cons of KubernetesExecutor?
Ans:

Pros:

1. It combines the benefits of CeleryExecutor and LocalExecutor in terms of scalability and simplicity.
2. Fine-grained control over task-allocation resources. At the task level, the amount of CPU/memory needed can be configured.

complicated.

Q: How to define a workflow in Airflow?
Ans:

Python files are used to define workflows.

DAG (Directed Acyclic Graph)

The DAG Python class in Airflow allows you to generate a **Directed Acyclic Graph**, which is a representation of the workflow.

```
from Airflow.models import DAG
from airflow.utils.dates import days_ago

args = {
    'start_date': days_ago(0),
}

dag = DAG(
    dag_id='bash_operator_example',
    default_args=args,
    schedule_interval='* * * * *',
)
```

You can use the start date to launch a task on a specific date.

The schedule interval specifies how often each workflow is scheduled to run. '* * * * *' indicates that the tasks must run every minute.

Q: How do you make the module available to airflow if you're using Docker Compose?
Ans:

If we are using **Docker Compose**, then we will need to use a custom image with our own additional dependencies in order to make the module available to Airflow. Refer to the following

Tech Geek

Q: How to schedule DAG in Airflow?
Ans:

DAGs could be scheduled by passing a timedelta or a cron expression (or one of the @ presets), which works well enough for DAGs that need to run on a regular basis, but there are many more use cases that are presently difficult to express "natively" in Airflow, or that require some complicated workarounds. You can refer Airflow Improvements Proposals (AIP).

Simply use the following command to start a scheduler:

```
airflow scheduler
```

Q: What is XComs In Airflow?
Ans:

XCom (short for cross-communication) are messages that allow data to be sent between tasks. The key, value, timestamp, and task/DAG id are all defined.

Q: What is xcom_pull in XCom Airflow?
Ans:

The xcom push and xcom pull methods on Task Instances are used to explicitly "push" and "pull" XComs to and from their storage. Whereas if do xcom push parameter is set to True (as it is by default), many operators and @task functions will auto-push their results into an XCom key named return value.

If no key is supplied to xcom pull, it will use this key by default, allowing you to write code like this:

```
# Pulls the return_value XCOM from "pushing_task"
value = task_instance.xcom_pull(task_ids='pushing_task')
```

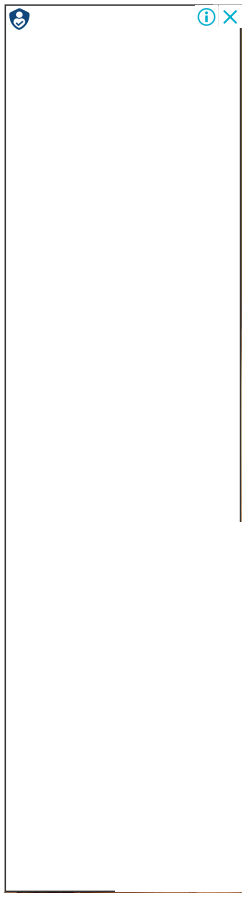
Q: What is Jinja templates?
Ans:

write code that looks like Python syntax. After that, data is passed to the template in order to render the final document.

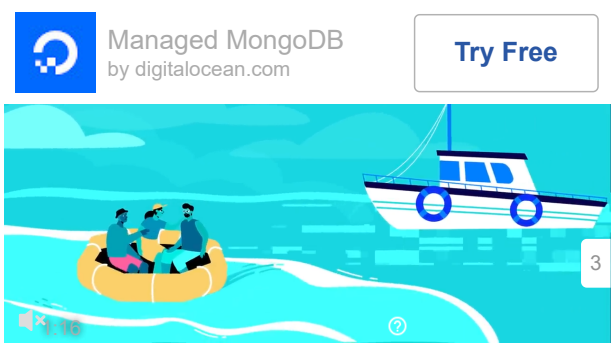
Q: How to use Airflow XComs in Jinja templates?
Ans:

We can use XComs in Jinja templates as given below:

```
SELECT * FROM {{ task_instance.xcom_pull(task_ids='foo', ke
```



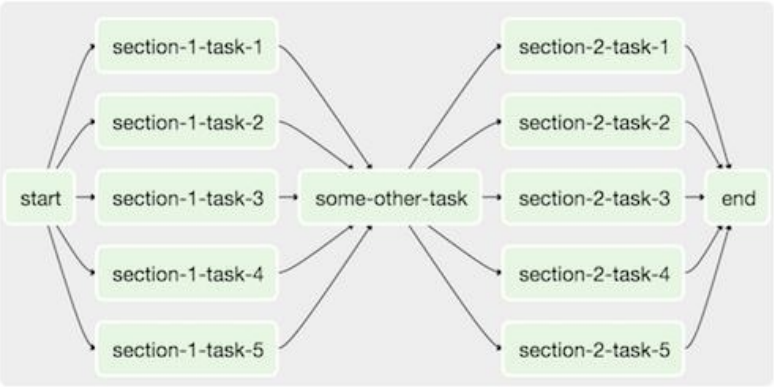
Most frequently Asked Apache Airflow Interview Questions



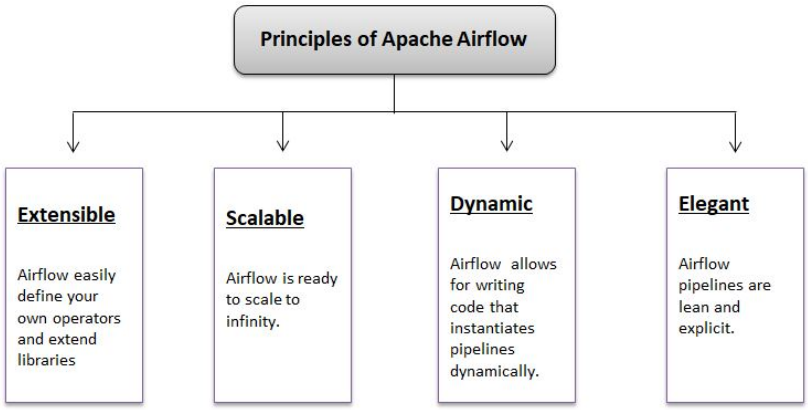
7. How can we use Airflow XComs in Jinja templates?
8. What is DAG in Airflow?
9. What are the alternatives to Airflow?
10. How do we Instantiate a DAG?
11. How do we Importing Modules in Airflow?
12. How can we delete a DAG?

What is Apache Airflow?

Apache Airflow is an open source workflow management that helps us by managing workflow Orchestration with the help of DAGs(Directed Acyclic Graphs).It is written in Python language and the workflow are created through python scripts.Airflow is designed by the principle of Configuration as Code. Apache Airflow began at Airbnb for managing platform and the company's icreasing complex workflows.It is known as data transformation pipeline Extract,Transform,Load(ETL) workflow Orchestration Tool.



What are the Principles of Apache Airflow?



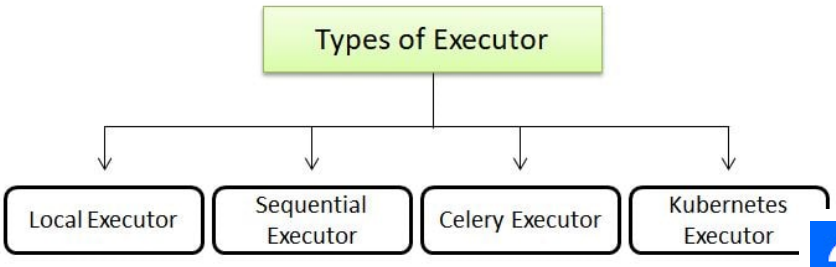
What are the components used by Airflow?

Components used by Airflow are:

- **Web Server** - used for tracking the status of our jobs and in reading logs from a remote File Store.
- **Scheduler** - used for scheduling our jobs and is a multithreaded python process which use DAGb object.
- **Executor** - used for getting the tasks done.
- **Metadata Database** - used for storing the Airflow States.

What are the Types of Executor in Apache Airflow?

There are 4 types of Executor in Apache Airflow:



Local Executor

Helps in running multiple tasks at one time.

Sequential Executor

Helps by running only one task at a time.

Celery Executor

Helps by running distributed asynchronous Python Tasks.

Kubernetes Executor

Helps in running tasks in an individual kubernet pod.

What is XComs?

XComs(Cross Communication) means a mechanism that lets tasks talk with each other, all the default tasks are isolated and can be running on different machines. They can be identified by a Key and also by task_id and dag_id.

What are Jinja Templates?

Jinja Templates helps by providing pipeline authors containing a set of built-in Parameters and Macros. It is normally a template that contains the following:

Variables and Expressions

How can we use Airflow XComs in Jinja templates?

We can use Airflow XComs in Jinja templates by using the following command:

```
SELECT * FROM {{ task_instance.xcom_pull(task_ids='foo', key='Table_Name') }}
```

What is DAG in Airflow?

Directed Acyclic Graph(DAG) helps in maintaining tasks dependencies at a particular time. It is also used for maintaining tasks relations and for ensuring those tasks are executed in an expected Order.

What are the alternatives to Airflow?

Some alternatives to Airflow are as follows:

Luigi

Apache NiFi

Jenkins

AWS Step Functions

Pachyderm

Kubeflow

Argo

Kafka

How do we Instantiate a DAG?

We can Instantiate a DAG by using the following command:

```
dag = DAG(
    'tutorial', default_args=default_args, schedule_interval=timedelta(days=10))
```

How do we Import Modules in Airflow?

Apache Airflow Pipeline is a Python Script which is used to define an Airflow DAG Object.

```
The DAG object; we need this instantiate
from airflow to import DAG
Operators; we need this for operating!
from airflow.operators.bash_operator import BashOperator
```

How can we delete a DAG?

We can Delete a DAG by using the following command:

For CLI:

```
airflow delete_dag my_dag_id
REST API (running webserver locally):
```

For REST API:

```
curl -X "delete"
```

Top Popular Post :

Spring Cloud Interview Questions (/interview/springCloud)

AWS CloudFormation Interview Questions (/interview/CloudFormation)

Spring Batch Interview Questions (/interview/batch)

Apache Camel - File Copy Example (/camel/file-copy)

Angular 8 +PrimeNG Hello World Example - Getting started with PrimeNG (/angular/primeng/prime1)

RxJS Interview Questions (/interview/rxjs)

Cosmos DB Interview Questions (/interview/cosmos)

Apache Camel + Quartz Example (/camel/camel-quartz)

Apache Camel Exception Handling Example (/camel/camel-exception)

Apache Camel Slip Routing EIP Pattern Example (/camel/slip-pattern)