# Installing Apache Airflow

**Using PyPi**:

<https://airflow.apache.org/docs/apache-airflow/stable/installation/installing-from-pypi.html>

<https://airflow.apache.org/docs/apache-airflow/stable/start/local.html>

## [Prerequisites](https://airflow.apache.org/docs/apache-airflow/2.0.1/installation.html#id1)

Airflow is tested with:

* Python: 3.6, 3.7, 3.8
* Databases:
  + PostgreSQL: 9.6, 10, 11, 12, 13
  + MySQL: 5.7, 8
  + SQLite: 3.15.0+
* Kubernetes: 1.16.9, 1.17.5, 1.18.6

**Note:** MySQL 5.x versions are unable to or have limitations with running multiple schedulers – please see: Scheduler. MariaDB is not tested/recommended.

**Note:** SQLite is used in Airflow tests. Do not use it in production. We recommend using the latest stable version of SQLite for local development.

Please note that with respect to Python 3 support, Airflow 2.0.0 has been tested with Python 3.6, 3.7, and 3.8, but does not yet support Python 3.9.

## Installation tools

Only pip installation is currently officially supported.

While there are some successes with using other tools like [poetry](https://python-poetry.org/) or [pip-tools](https://pypi.org/project/pip-tools/), they do not share the same workflow as pip - especially when it comes to constraint vs. requirements management. Installing via Poetry or pip-tools is not currently supported. If you wish to install airflow using those tools you should use the constraints and convert them to appropriate format and workflow that your tool requires.

Typical command to install airflow from PyPI looks like below:

pip install "apache-airflow[celery]==2.3.2" --constraint "https://raw.githubusercontent.com/apache/airflow/constraints-2.3.2/constraints-3.7.txt"

## [Constraints files](https://airflow.apache.org/docs/apache-airflow/2.0.1/installation.html#id7)

Airflow installation might be sometimes tricky because Airflow is a bit of both a library and application. Libraries usually keep their dependencies open and applications usually pin them, but we should do neither and both at the same time. We decided to keep our dependencies as open as possible (in setup.cfg and setup.py) so users can install different version of libraries if needed. This means that from time to time plain pip install apache-airflow will not work or will produce unusable Airflow installation.

In order to have repeatable installation, starting from **Airflow 1.10.10** and updated in **Airflow 1.10.13** we also keep a set of “known-to-be-working” constraint files in the constraints-master, constraints-2-0 and constraints-1-10 orphan branches and then we create tag for each released version e.g. constraints-2.0.1. This way, when we keep a tested and working set of dependencies.

Those “known-to-be-working” constraints are per major/minor python version. You can use them as constraint files when installing Airflow from PyPI. Note that you have to specify correct Airflow version and python versions in the URL.

You can create the URL to the file substituting the variables in the template below.

https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW\_VERSION}/constraints-${PYTHON\_VERSION}.txt

where:

* AIRFLOW\_VERSION - Airflow version (e.g. 2.0.1) or master, 2-0, 1-10 for latest development version
* PYTHON\_VERSION Python version e.g. 3.8, 3.7

## [Installation script](https://airflow.apache.org/docs/apache-airflow/2.0.1/installation.html#id8)

In order to simplify the installation, we have prepared a script that will select [the constraints file](installation:constraints) compatible with your Python version

### **Plain installation**

If you don’t need to install any extra extra, you can use the command set below:

AIRFLOW\_VERSION=2.0.1

PYTHON\_VERSION="$(python --version | cut -d " " -f 2 | cut -d "." -f 1-2)"

# For example: 3.6

CONSTRAINT\_URL="https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW\_VERSION}/constraints-${PYTHON\_VERSION}.txt"

# For example: https://raw.githubusercontent.com/apache/airflow/constraints-2.0.1/constraints-3.6.txt

pip install "apache-airflow==${AIRFLOW\_VERSION}" --constraint "${CONSTRAINT\_URL}"

### Installing with extras

If you need to install [extra dependencies of airflow](https://airflow.apache.org/docs/apache-airflow/2.0.1/installation.html#installation-airflow-extra-dependencies), you can use the script below (the example below installs postgres and google extras.

AIRFLOW\_VERSION=2.0.1

PYTHON\_VERSION="$(python --version | cut -d " " -f 2 | cut -d "." -f 1-2)"

CONSTRAINT\_URL="https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW\_VERSION}/constraints-${PYTHON\_VERSION}.txt"

pip install "apache-airflow[postgres,google]==${AIRFLOW\_VERSION}" --constraint "${CONSTRAINT\_URL}"

## Airflow command is not recognized

If the airflow command is not getting recognized (can happen on Windows when using WSL), then ensure that ~/.local/bin is in your PATH environment variable, and add it in the .bashrc file if necessary:

PATH=$PATH:~/.local/bin

You can also start airflow with python -m airflow

## Initialize Database

$ airflow db init

This should create a folder called airflow in the current folder.

## Create a Directory to Store DAGs

mkdir -p ./airflow/dags/

## Create User

$ airflow users create --role Admin --username admin --email admin --firstname admin --lastname admin --password admin

## Running Airflow

Airflow has an excellent web UI where you can view and monitor your dags. To start the webserver to view the UI simply run the following CLI command. By default, Airflow will use the port 8080 as I am already using that to run something else I am specifying 8081.

$ airflow webserver -p 8081

We also need to start the scheduler in a new terminal.

$ airflow scheduler

Open the browser and navigate to <http://localhost:8081> and login as admin / admin.

The DAGs will be displayed.

# Running Your First DAG Tasks

Here are a few commands that will trigger a few task instances. You should be able to see the status of the jobs change in the example\_bash\_operator DAG as you run the commands below.

*# run your first task instance*

airflow tasks run example\_bash\_operator runme\_0 2015-01-01

*# run a backfill over 2 days*

airflow dags backfill example\_bash\_operator **\**

--start-date 2015-01-01 **\**

--end-date 2015-01-02

Walk through airflow/example\_dags/tutorial.py.

**Path**: /home/asingala/.local/lib/python3.8/site-packages/airflow/example\_dags/