Adding Functionalities to Apache Airflow

How to customise your Airflow installation

Plugins: Definition and Utility

- A plugin is a simple way to add functionalities to Apache Airflow without altering its functioning.
- It is built on top of Apache Airflow and not directly merged to the code in order to avoid possible bugs and dependency errors during updates.
- It also can be used as an easy way to write, share and activate new sets of features.
- Different organizations have different stacks and different needs, creating plugins can be a way for companies to customize their Airflow installation in order to reflect their ecosystem.

How Apache Airflow Deals With Plugins

- Airflow offers a generic toolbox for working with plugins. It has a simple plugin manager built-in that can integrate external features to its core by simply dropping files in the `$AIRFLOW_HOME/plugins` folder.
- The python modules in the plugin folder get imported, and hooks, operators, sensors, macros, executors and web views get integrated to Airflow's main collections and become available for use as any Airflow's component.

Extendable Components

- **Operators**: They describe a single task in a workflow.
  - Derived from BaseOperator
- **Sensors**: They are a particular subtype of Operators used to wait for an event to happen.
  - Derived from BaseSensorOperator
- **Hooks**: They are used as interfaces between Apache Airflow and external systems.
  - Derived from BaseHook
- **Executors**: They are used to actually execute the tasks.
  - Derived from BaseExecutor

Extendable Components

- **Admin Views**: Represent base administrative view from Flask-Admin allowing to create web interfaces.
    - Derived from flask_admin.BaseView
- **Blueprints**: Represent a way to organize flask application into smaller and re-usable application. A blueprint defines a collection of views, static assets and templates.
    - Derived from flask.Blueprint
- **Menu Link**: Allow to add custom links to the navigation menu in Apache Airflow.
    - Derived from flask_admin.base.MenuLink

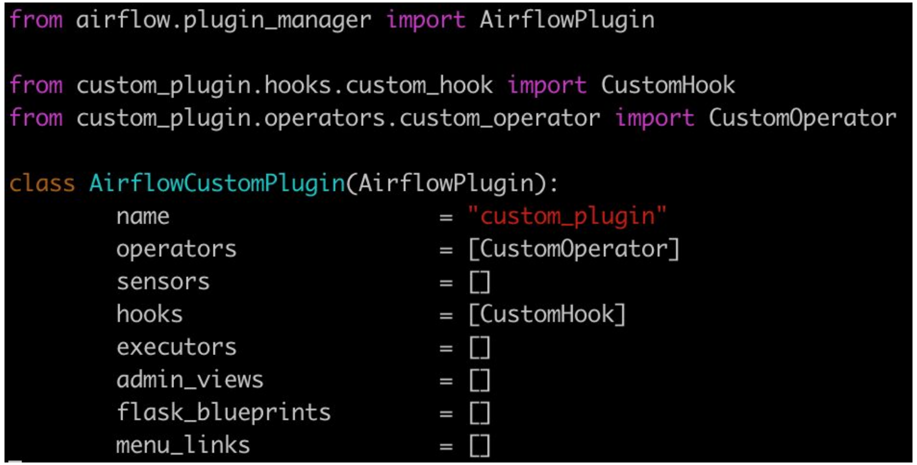
How to Create a Plugin?

To create a plugin you must derive the `airflow.plugins_manager.AirflowPlugin` class and reference the objects you want to plug into Airflow.

Basically you would have a python file (we will use __init__.py) in which a subclass of AirflowPlugin is defined with a name attribute (this name will be used to import your plugin : `from.airflow.operators.{name}`) and the different custom objects (hooks, operators, sensors, …) representing your plugin.

Example

Here is an example of the base file to create your plugin:



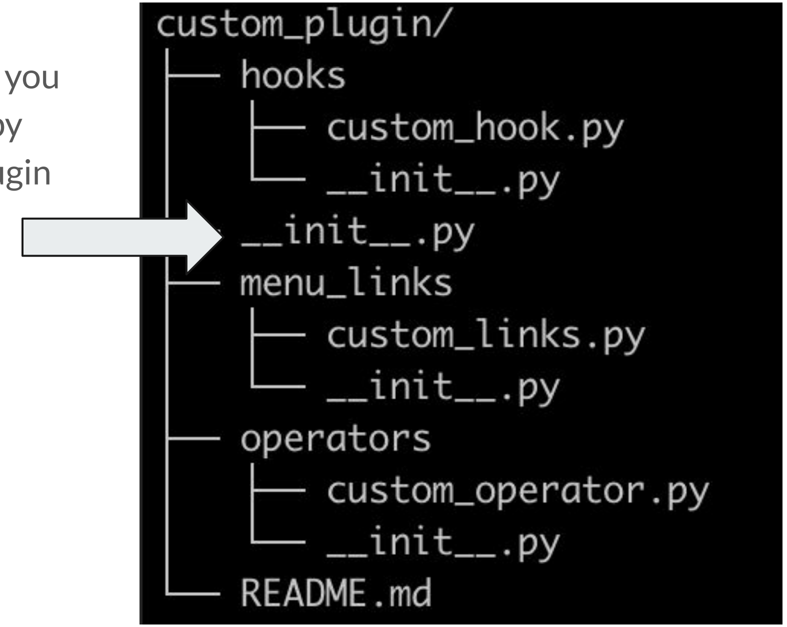
```
from airflow.plugin_manager import AirflowPlugin

from custom_plugin.hooks.custom_hook import CustomHook
from custom_plugin.operators.custom_operator import CustomOperator

class AirflowCustomPlugin(AirflowPlugin):
        name                    = "custom_plugin"
        operators               = [CustomOperator]
        sensors                 = []
        hooks                   = [CustomHook]
        executors               = []
        admin_views             = []
        flask_blueprints        = []
        menu_links              = []
```

Plugin Structure

When you want to create a plugin, I strongly advise you to apply the following structure where the __init__.py next to README.md corresponds to the AirflowPlugin class instantiation:



```
custom_plugin/
├── hooks
│   ├── custom_hook.py
│   └── __init__.py
├── __init__.py
├── menu_links
│   ├── custom_links.py
│   └── __init__.py
├── operators
│   ├── custom_operator.py
│   └── __init__.py
└── README.md
```

Important Notes

- There is actually one more component that you can extend which is the Macros. The macros are a way to pass dynamic information into task instances at runtime. They are tightly coupled with Jinja Template, a python templating language.
- When you create a plugin, be sure to understand exactly what you are doing. Some components require to implement specific functions, for example:
  - An Operator plugin must have an `execute` method.
  - A Sensor plugin must have a `poke` method returning a boolean value.
- When you want to create a new page to Airflow UI, you will need to use an AdminView and a Blueprint together.

Coding Time!

Let's create your first plugin!