Spellbook

Why should I use Spellbook?

Spellbook provides identity propagation throughout the platform. If you have an Airflow DAG that uses Spark, Spellbook sets up that connection between Airflow and Spark so that all tools know that it's you that is doing the execution. See more:

Upload a new Job to Spellbook

1. You need to have your DAG downloaded to your machine.
    a. Download your DAG from JupyterHub
        i. Right click your .py file
        ii. Click on download from the right click menu
2. Click the create button
3. Name your job
4. Add the DAG id in the job key field
    a. Ensure the DAG id is the same as what is in the DAG file
    b. Ensure the DAG id is not already used by another DAG in Airflow
5. Add in the Classification of the DAG
6. Fill in any other fields you want
7. Add the DAG file you downloaded in step 1
8. Click submit.

Adopting a DAG

If your DAG already exists in Airflow and you'd like to migrate it into Spellbook, do the following:

- Go to Spellbook
- Click the Create button at the top of the screen
- Name your Job
- Start typing your DAG id into the Job key field, your DAG id should show up in the dropdown.

If your DAG id does not show up in the dropdown:

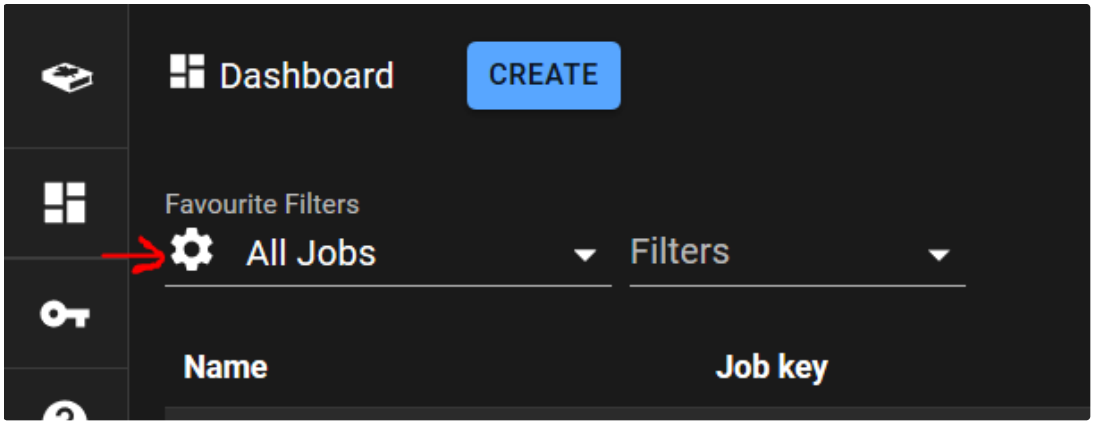
- Ensure you are spelling it correctly
- Ensure it does exist in Airflow
- Check the other Jobs in Spellbook to be sure someone didn't already adopt it
  - If someone did adopt it please contact Hogwarts Support.

- Fill in the rest of the fields as you see fit
  - Note that Classification & type are required.
- Hit submit

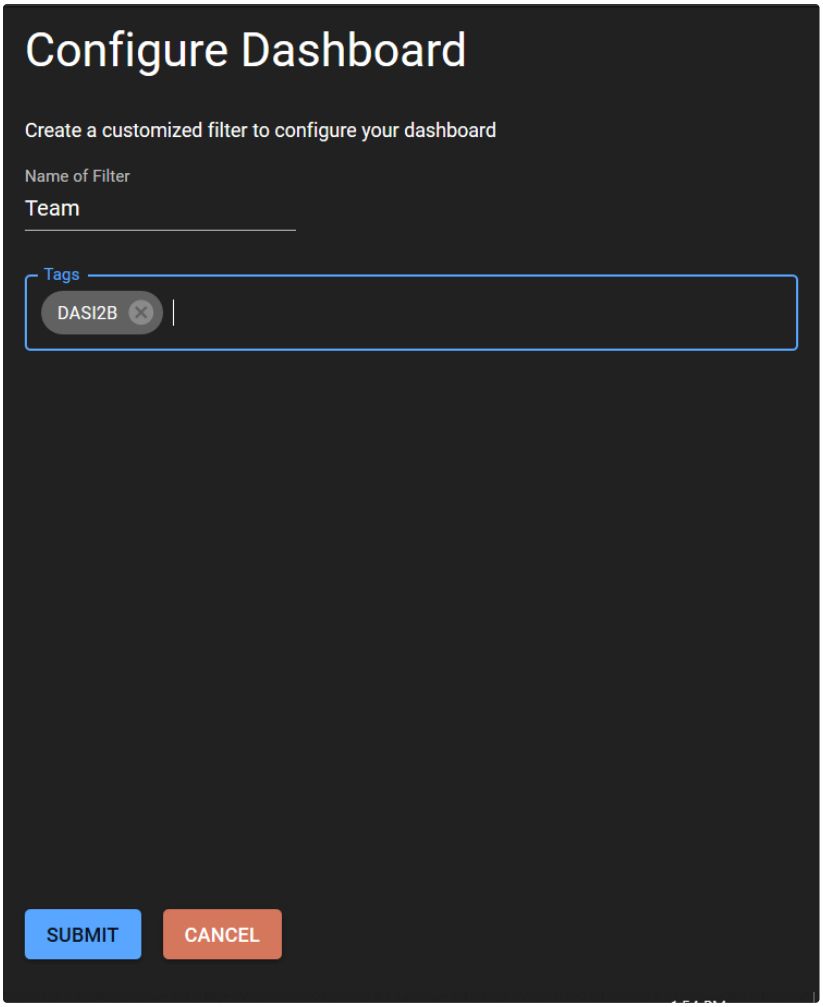
How can I quickly see all the DAGs by my team?

To do this, it requires a sort of team effort. Everyone on your team needs to add an agreed upon tag to their jobs.

1. On the Dashboard screen of Spellbook, click the gear icon underneath 'Favourite Filters'



2. In the drawer on the right that opens, enter a name for your filter.
3. In the tag field, enter the tag your team decided upon, and added to everyone's jobs



4. Submit your new filter!

I have a bash script, how can I automate it?

Click here to expand...

Firstly, your bash script needs to be put into a git repo which sa-spellbook has access to (see adding sa-spellbook to a git repo). The DAG you would want would then look like:

```
from datetime import timedelta
from airflow.models.dag import DAG
from daggers.operators import GitBashOperator

default_args = {
    'owner': '<You!>'
    'start_date': datetime(2021, 5, 10)
}

dag = DAG(
    dag_id='<Your dag id!>',
    default_args=default_args,
    schedule_interval=timedelta(minutes=5)
)

bash_task = GitBashOperator(
    task_id="<descriptive name of what the task does>",
    dag=dag,
```

```
    # The URL would be something like github.com/<user>/<repo>
    # Just ensure that the first part with sa-spellbook and the GIT_ACCESS_TOKEN
    # remains there, that's important.
    repo_url="https://sa-spellbook:${GIT_ACCESS_TOKEN}@<URL>.git",
    # This should be the path from the root of the git repo to the shell file.
    bash_command='path/to/executable.sh'
)

bash_task
```

If your bash script requires parameters:

```
from datetime import timedelta
from airflow.models.dag import DAG
from daggers.operators import GitBashOperator

default_args = {
    'owner': '<You!>'
    'start_date': datetime(2021, 5, 10)
}

dag = DAG(
    dag_id='<Your dag id!>',
    default_args=default_args,
    schedule_interval=timedelta(minutes=5)
)

bash_task = GitBashOperator(
    task_id="<descriptive name of what the task does>",
    dag=dag,
    # The URL would be something like github.com/<user>/<repo>
    # Just ensure that the first part with sa-spellbook and the GIT_ACCESS_TOKEN
    # remains there, that's important.
    repo_url="https://sa-spellbook:${GIT_ACCESS_TOKEN}@<URL>.git",
    # This should be the path from the root of the git repo to the shell file.
    bash_command='path/to/executable.sh {{params.value1}} {{params.value2}}'
    # Of course these values can be named whatever you want
    params={value1: 'hello', value2: 'world'}
)
bash_task
```

Now all that's left is to upload it to spellbook!

I have a Python script, how can I automate it?

Click here to expand...

Firstly, your Python script needs to be put into a git repo which sa-spellbook has access to (see adding sa-spellbook to a git repo). The DAG you would want would then look like:

```
from datetime import timedelta
from airflow.models.dag import DAG
from daggers.operators import GitPythonOperator

default_args = {
    'owner': '<You!>',
    'start_date': datetime(2021, 5, 10)
}
```

```

dag = DAG(
    dag_id='<Your DAG id!>',
    default_args=default_args,
    schedule_interval=timedelta(minutes=5)
)

python_task = GitPythonOperator(
    task_id="<A descriptive name of what the task does>",
    dag=dag,
    # The URL would be something like github.com/<user>/<repo>
    # Just ensure that the first part with sa-spellbook and the GIT_ACCESS_TOKEN
    # remains there, that's important.
    repo_url="https://sa-spellbook:${GIT_ACCESS_TOKEN}@<URL>.git",
    run_module='path.to.module',
    run_method='<Method you want to be run>',
    # If your method requires arguments, pass them in here.
    # If you don't require arguments, no need to have this parameter at all.
    run_kwargs={'arg1': 'hello', 'arg2': 'world'}
)

python_task
```

Now all that's left is to upload it to spellbook!

I have a Notebook, how can I automate it?

Click here to expand...

Firstly, your notebook needs to be put into a git repo which sa-spellbook has access to (see adding sa-spellbook to a git repo). The DAG you would want would then look like:

```
from datetime import timedelta
from airflow.models.dag import DAG
from daggers.operators import GitNotebookOperator

default_args = {
    'owner': '<You!>',
    'start_date': datetime(2021, 5, 10)
}

dag = DAG(
    dag_id='<Your DAG id!>',
    default_args=default_args,
    schedule_interval=timedelta(minutes=5)
)

notebook_task = GitNotebookOperator(
    task_id="<A descriptive name of what the task does>",
    dag=dag,
    # The URL would be something like github.com/<user>/<repo>
    # Just ensure that the first part with sa-spellbook and the GIT_ACCESS_TOKEN
    # remains there, that's important.
    repo_url="https://sa-spellbook:${GIT_ACCESS_TOKEN}@<URL>.git",
    # This should be the path from the root of the git repo to the notebook.
    nb_in='notebooks/notebook3.ipynb',
)
```

```

notebook_task
```

If you're using parameters in your notebook:

```
from datetime import timedelta
from airflow.models.dag import DAG
from daggers.operators import GitNotebookOperator

default_args = {
    'owner': '<You!>',
    'start_date': datetime(2021, 5, 10)
}

dag = DAG(
    dag_id='<Your DAG id!>',
    default_args=default_args,
    schedule_interval=timedelta(minutes=5)
)

notebook_task = GitNotebookOperator(
    task_id="<A descriptive name of what the task does>",
    dag=dag,
    # The URL would be something like github.com/<user>/<repo>
    # Just ensure that the first part with sa-spellbook and the GIT_ACCESS_TOKEN
    # remains there, that's important.
    repo_url="https://sa-spellbook:${GIT_ACCESS_TOKEN}@<URL>.git",
    # This should be the path from the root of the git repo to the notebook.
    nb_in='notebooks/notebook3.ipynb',
    # The key of these parameters need to match the key of parameters defined
    # In your notebook
    nb_params={'hellowho': 'the entire planet of Mars'},
)

notebook_task
```

Now all that's left is to upload it to spellbook!

I have a notebook in NBGallery, how do I automate it?

Click here to expand...

Firstly you'll need the notebook's UUID, you can find that with this: NBGallery

```
from datetime import timedelta
from airflow.models.dag import DAG
from daggers.operators.nbgallery import NbgalleryOperator

default_args = {
    'owner': '<You!>',
    'start_date': datetime(2021, 5, 10)
}

dag = DAG(
    dag_id='<Your DAG id!>',
    default_args=default_args,
    schedule_interval=timedelta(minutes=5)
)
```

```
15
16 notebook_task = NbgalleryOperator(
17     task_id="<A descriptive name of what the task does>",
18     dag=dag,
19     nb_id="<your id found through other means",
20     # Can provide params too if you so choose
21     nb_params={'hellowho': 'the entire planet of Mars'}
22 )
23
24 notebook_task
```

This operator is built off the PapermillOperator, so you can provide any of those arguments as well.

I need to read/write from/to a datalake, how do I do it in my DAG?

It is recommended to use the Datalake in a notebook/script, and not directly in a DAG. See Hogwarts Datalakes for more information.

I have a secret I need to add to my DAG, how do I do it?

See Airflow Connections & Variables for how to use secrets

How can I have alerts for when my execution fails?

You can have Airflow email you on a few conditions:

- Your job has failed
- Your job has retried
- Your job has hit an email operator

Email on Failure

To get an email when your job fails, add the following fields to your default_args dictionary:

```
default_args = {
    ...,
    'email': ['<your cyber email>'],
    'email_on_failure': True,
    ...
}
```

Email on Retry

```
default_args = {
    ...,
    'email': ['<your cyber email>'],
    'email_on_retry': True,
}
```

For either of the above configs, you can add those fields to individual operators instead of in the default_args. Default_args gets applied to all operators, however they would get overridden by whats defined in the individual operators.

Email Operator

You can use the email operator to have an email sent to you on whatever condition you want:

```
from airflow.operators.email_operator import EmailOperator

email_task = EmailOperator(
    task_id="<task id explaining what this is for>",
    dag=dag,
    # Or you can specify an email instead of using the default args.
    to=default_args['email'],
    subject="<subject of the email being sent>",
    html_content="<Content of the email, can be written in html>"
)
```

You can put this task at the end of a chain of tasks, and that will email you when that chain terminates successfully. You can also add a `trigger_rule` argument to the operator, which can have any of the following options:

- all_success (default) - trigger when all upstream tasks end successfully
- all_failed - trigger when all upstream tasks end on failure
- all_done - all upstream tasks have completed
- one_success - one upstream task has completed successfully
- one_failed - one upstream task has failed
- none_failed - no upstream task has failed
- none_failed_or_skipped - no upstream task has failed or has been skipped
- none_skipped - no upstream tasks has been skipped

Using a combination of these trigger_rules you can have a variety of email notifications.

How can I add sa-spellbook to my Git repo?

Github

1. Go to your repo
2. Go to settings of that repo
    a. If you don't see a settings tab, you probably are not an admin of that repo.
3. Go to manage access
4. If you see sa-spellbook in the list of collaborators already - Great! You're done.
5. If not, add sa-spellbook as a contributor, and alert the Hogwarts Support chat that you need sa-spellbook added as a contributor.

Gitlab

1. Go to your repo
2. On the left side of the screen, go to members
3. add sa-spellbook as a reporter or higher (anything but guest)