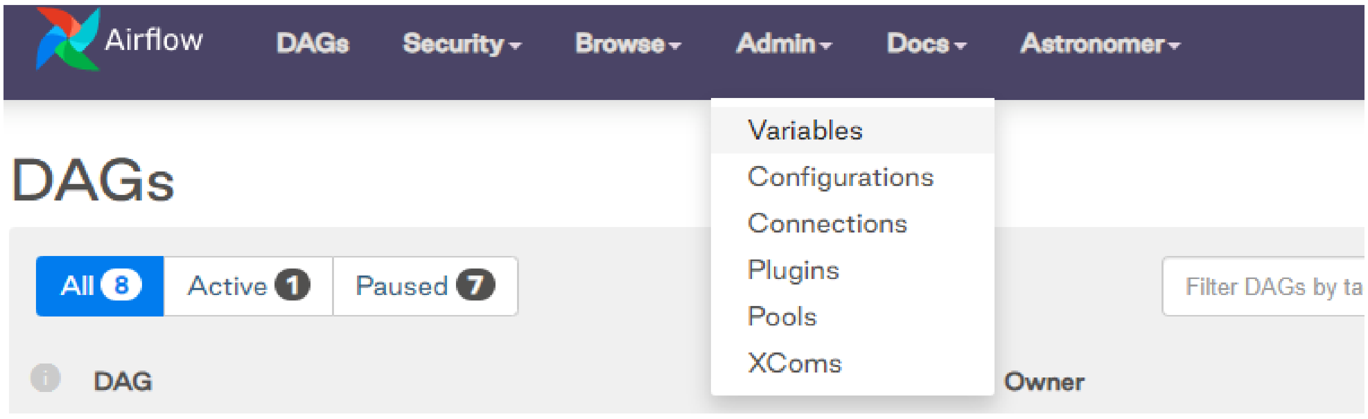
Template Variables

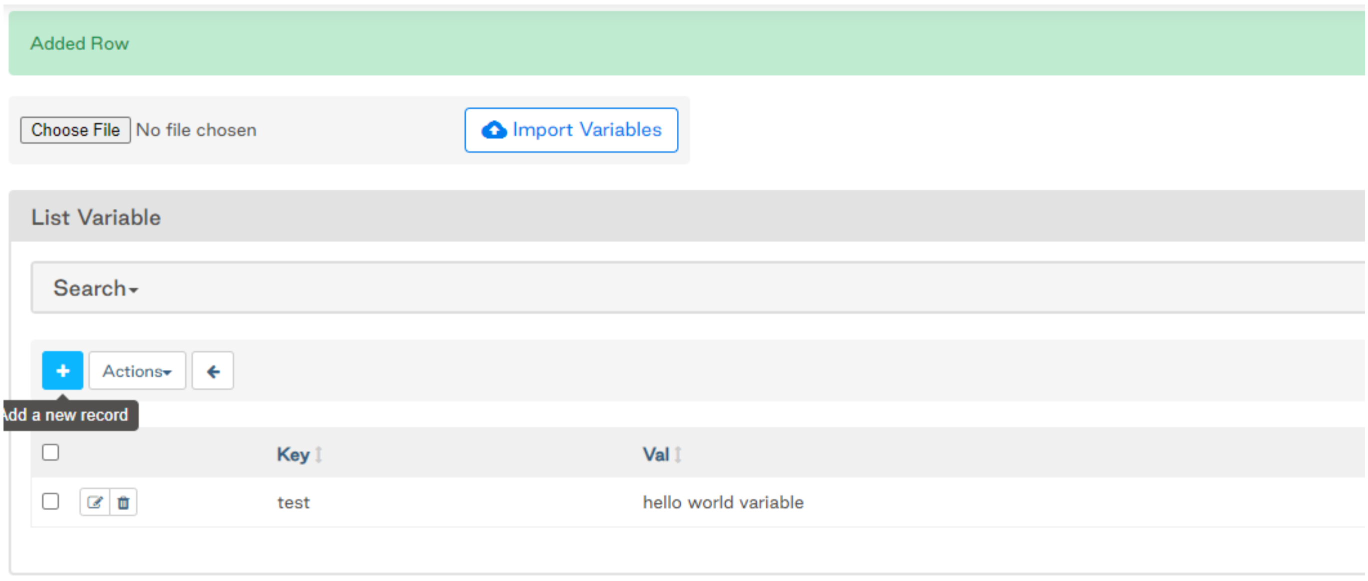
Template variables and macros are the way that we insert dynamic values into your tasks. We mostly use it to pass configuration values as well as the current date boundaries (execution date and next execution date) of your scheduled DAG run.

Setting variables

You can set your own variables to access in your code via the Admin > Variables UI

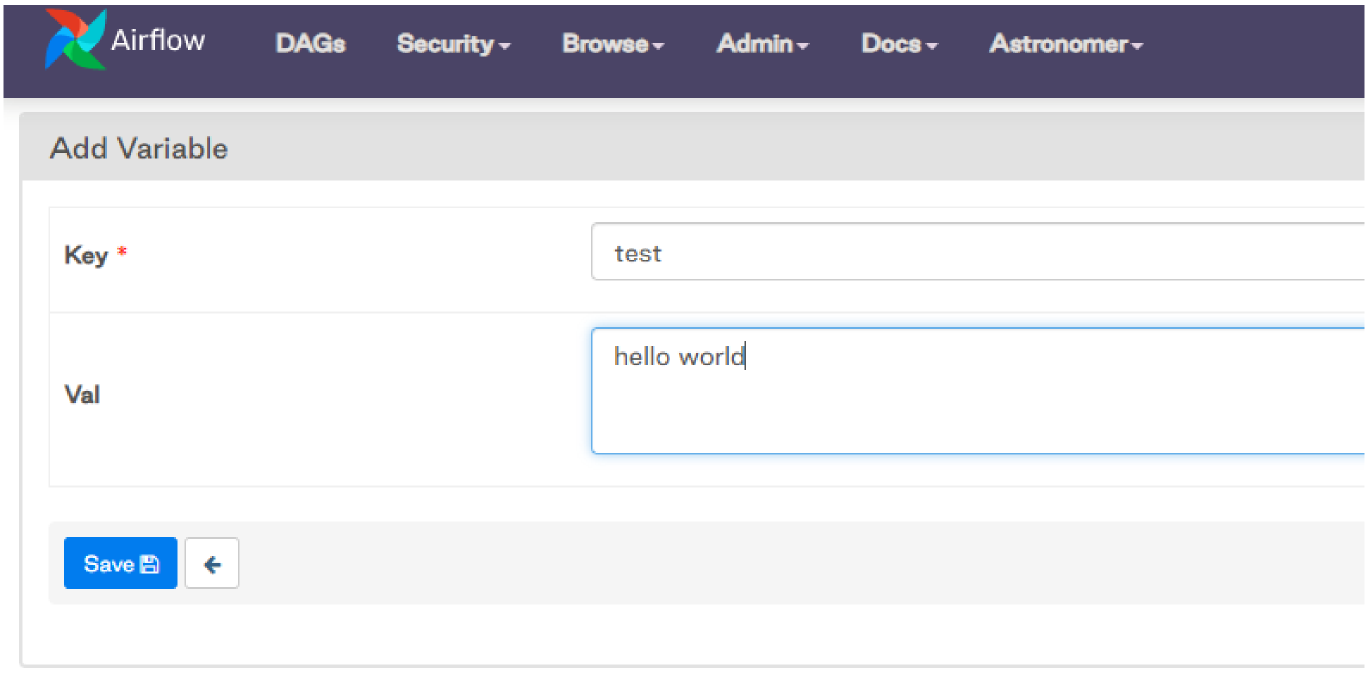


Then create a new variable



Key	Val
test	hello world variable

And specify a key and value



Add Variable

Key * test

Val hello world

Default variables

Aside from setting your own variables, airflow also passes default variables to your DAG, most notable of which is the `execution_date` and some date format shortcuts like `{{ ds }}` which is the execution date in the `YYYY-MM-DD` format.

The full list of default variables can be found here:

https://airflow.apache.org/docs/apache-airflow/stable/macros-ref.html#default-variables

Using template variables in tasks

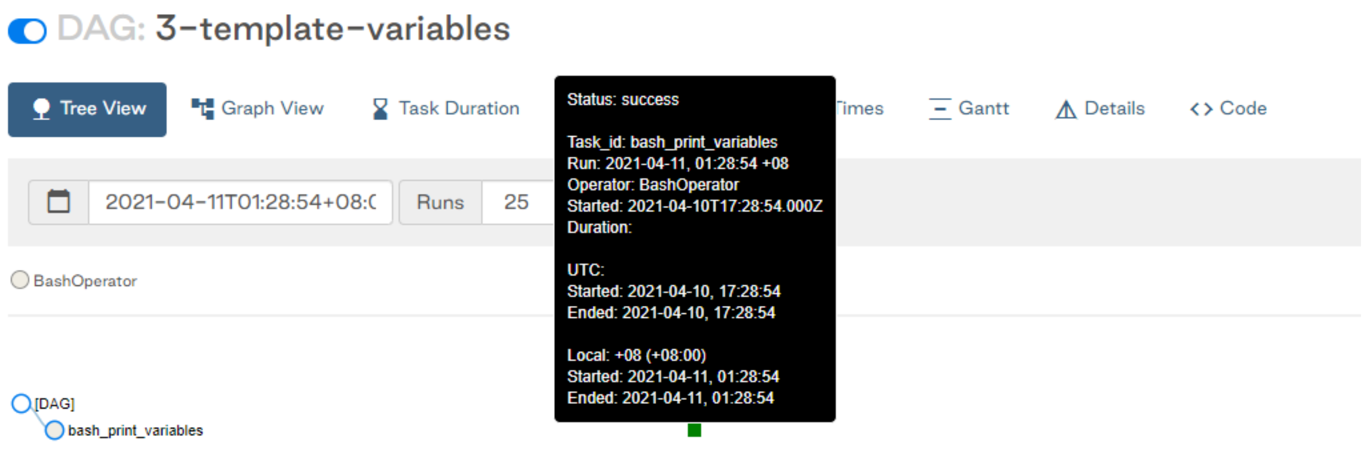
Now in order to use them in our operators, you have to take note that you can only use them in the templated parameters of our operators.

BashOperator

For the `BashOperator` the templated parameters are `bash_command` and `env`. This means we can only use the template variables as parameters like so:

```
bash_task = BashOperator(
    task_id="check_variables"
    bash_command=(
        "echo {{ var.value.test }}" # print custom variable we set in the web ui
        "echo {{ execution_date }};" # print task execution date
        "echo {{ ds }};" # print task execution date in YYYY-MM-DD format
    )
    dag=dag
)
```

After running this, you can see the result by clicking the task in the tree view:



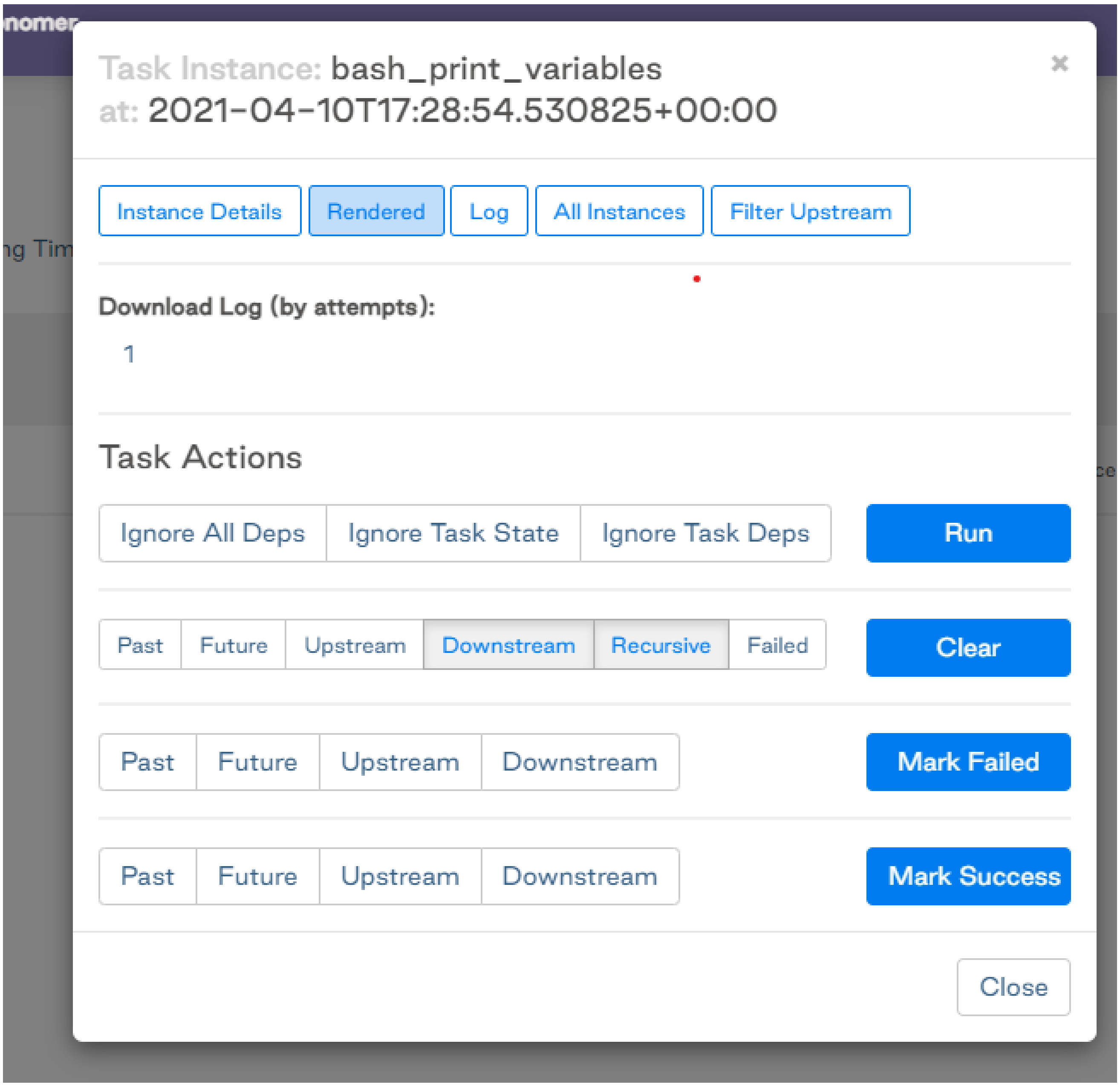
Status: success

Task_id: bash_print_variables
Run: 2021-04-11, 01:28:54 +08
Operator: BashOperator
Started: 2021-04-10T17:28:54.000Z
Duration:

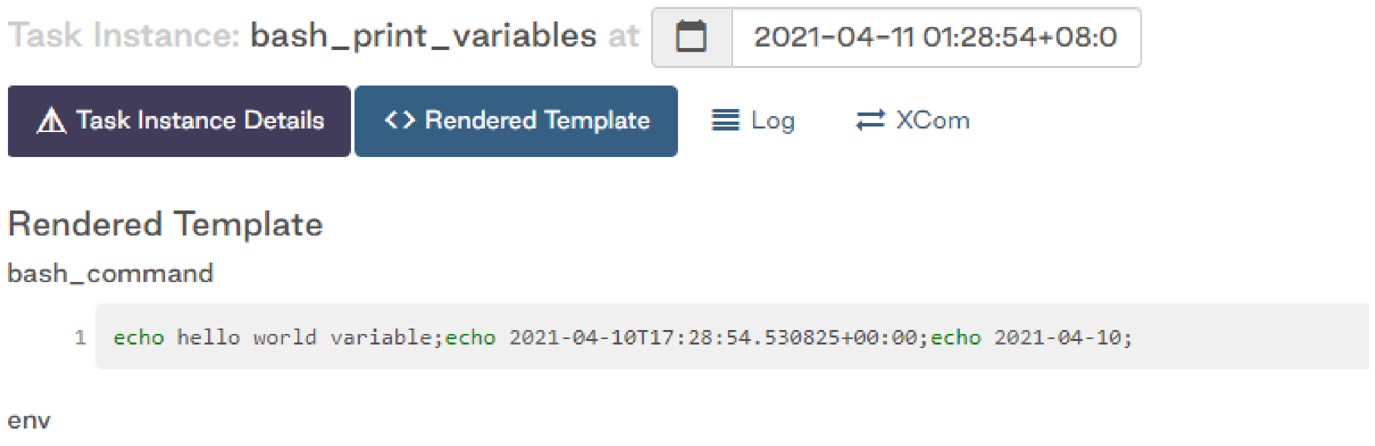
UTC:
Started: 2021-04-10, 17:28:54
Ended: 2021-04-10, 17:28:54

Local: +08 (+08:00)
Started: 2021-04-11, 01:28:54
Ended: 2021-04-11, 01:28:54

which will open up the task instance context menu. Then click the "Rendered" menu item.



You can now see the bash commands after you substituted the template variables.



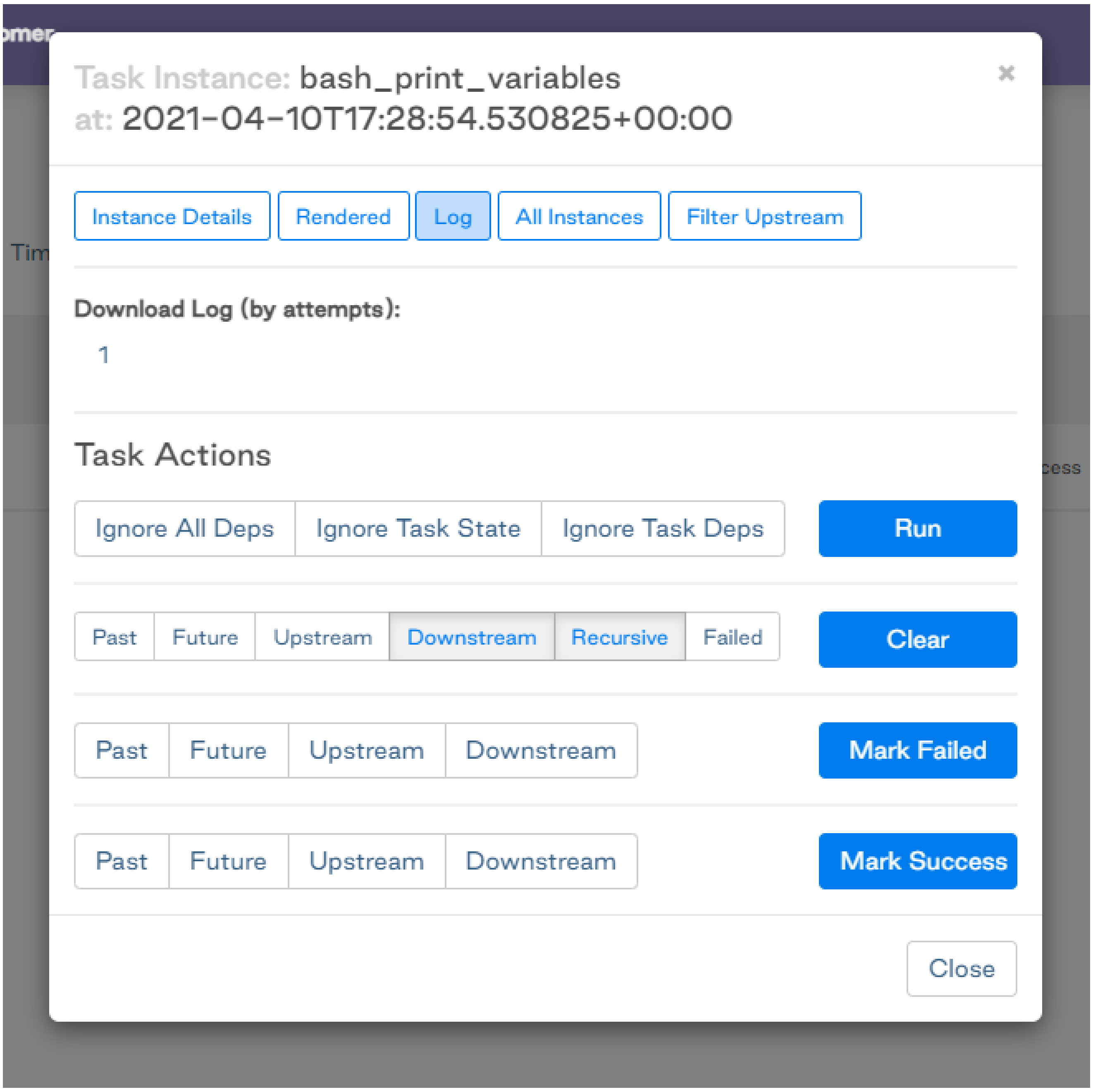
Rendered Template

bash_command

```
echo hello world variable;echo 2021-04-10T17:28:54.530825+00:00;echo 2021-04-10;
```

env

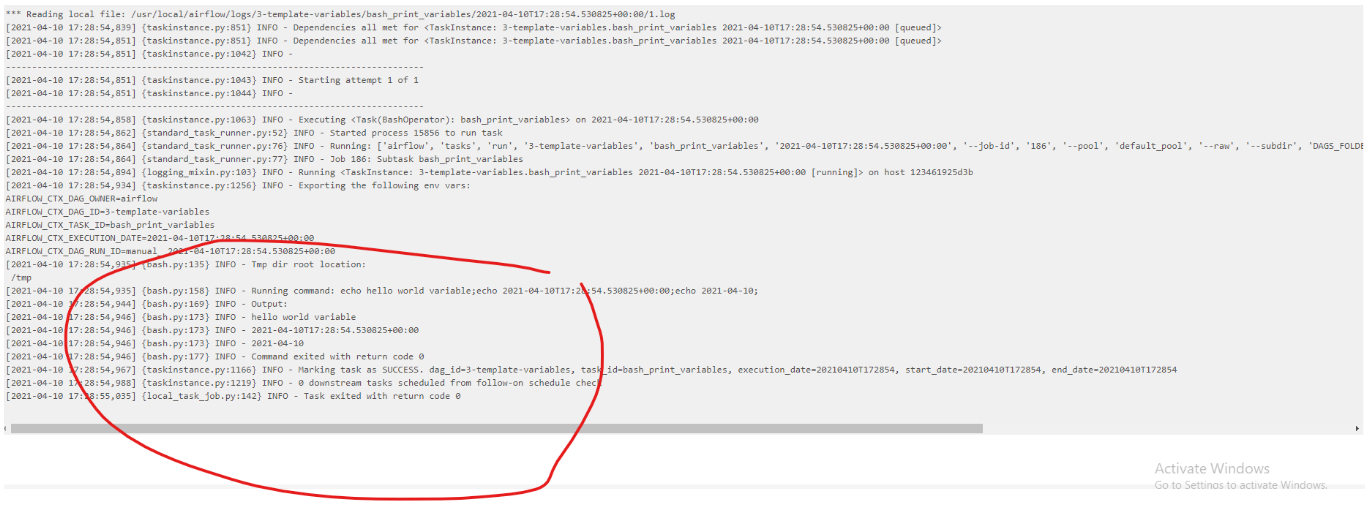
You can also go back to the task logs:


Task Instance: bash_print_variables
at: 2021-04-10T17:28:54.530825+00:00

Download Log (by attempts):
1

Task Actions

To see the variables printed in the logs:


```
*** Reading local file: /usr/local/airflow/logs/3-template-variables/bash_print_variables/2021-04-10T17:28:54.530825+00:00/1.log
[2021-04-10 17:28:54,839] {taskinstance.py:851} INFO - Dependencies all met for <TaskInstance: 3-template-variables.bash_print_variables 2021-04-10T17:28:54.530825+00:00 [queued]>
[2021-04-10 17:28:54,851] {taskinstance.py:851} INFO - Dependencies all met for <TaskInstance: 3-template-variables.bash_print_variables 2021-04-10T17:28:54.530825+00:00 [queued]>
[2021-04-10 17:28:54,851] {taskinstance.py:1042} INFO - 
--------------------------------------------------------------------------------
[2021-04-10 17:28:54,851] {taskinstance.py:1043} INFO - Starting attempt 1 of 1
[2021-04-10 17:28:54,851] {taskinstance.py:1044} INFO - 
--------------------------------------------------------------------------------
[2021-04-10 17:28:54,858] {taskinstance.py:1063} INFO - Executing <Task(BashOperator): bash_print_variables> on 2021-04-10T17:28:54.530825+00:00
[2021-04-10 17:28:54,862] {standard_task_runner.py:52} INFO - Started process 15856 to run task
[2021-04-10 17:28:54,864] {standard_task_runner.py:76} INFO - Running: ['airflow', 'tasks', 'run', '3-template-variables', 'bash_print_variables', '2021-04-10T17:28:54.530825+00:00', '--job-id', '186', '--pool', 'default_pool', '--raw', '--subdir', 'DAGS_FOLDE
[2021-04-10 17:28:54,864] {standard_task_runner.py:77} INFO - Job 186: Subtask bash_print_variables
[2021-04-10 17:28:54,894] {logging_mixin.py:103} INFO - Running <TaskInstance: 3-template-variables.bash_print_variables 2021-04-10T17:28:54.530825+00:00 [running]> on host 123461925d3b
[2021-04-10 17:28:54,934] {taskinstance.py:1256} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_OWNER=airflow
AIRFLOW_CTX_DAG_ID=3-template-variables
AIRFLOW_CTX_TASK_ID=bash_print_variables
AIRFLOW_CTX_EXECUTION_DATE=2021-04-10T17:28:54.530825+00:00
AIRFLOW_CTX_DAG_RUN_ID=manual__2021-04-10T17:28:54.530825+00:00
[2021-04-10 17:28:54,935] {bash.py:135} INFO - Tmp dir root location: 
 /tmp
[2021-04-10 17:28:54,935] {bash.py:158} INFO - Running command: echo hello world variable;echo 2021-04-10T17:28:54.530825+00:00;echo 2021-04-10;
[2021-04-10 17:28:54,944] {bash.py:169} INFO - Output:
[2021-04-10 17:28:54,946] {bash.py:173} INFO - hello world variable
[2021-04-10 17:28:54,946] {bash.py:173} INFO - 2021-04-10T17:28:54.530825+00:00
[2021-04-10 17:28:54,946] {bash.py:173} INFO - 2021-04-10
[2021-04-10 17:28:54,946] {bash.py:177} INFO - Command exited with return code 0
[2021-04-10 17:28:54,967] {taskinstance.py:1166} INFO - Marking task as SUCCESS. dag_id=3-template-variables, task_id=bash_print_variables, execution_date=20210410T172854, start_date=20210410T172854, end_date=20210410T172854
[2021-04-10 17:28:54,988] {taskinstance.py:1219} INFO - 0 downstream tasks scheduled from follow-on schedule check
[2021-04-10 17:28:55,035] {local_task_job.py:142} INFO - Task exited with return code 0
```

PythonOperator

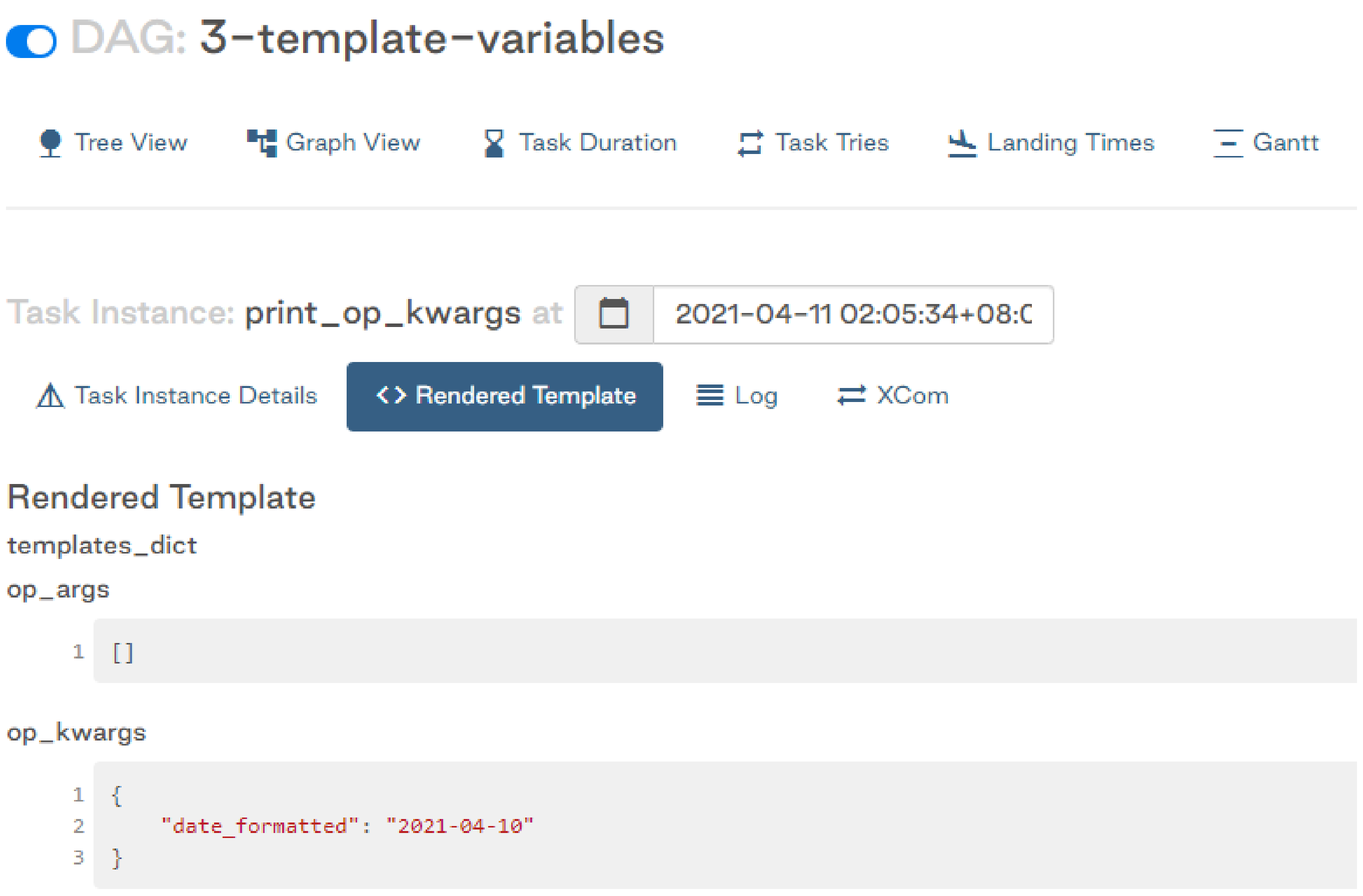
Unlike the `BashOperator` which accepts a string you can template, the `PythonOperator` accepts a python callable object (usually a function). We can check the templated variables here: https://airflow.apache.org/docs/apache-airflow/stable/_api/airflow/operators/python/index.html and we can see that we can use `op_args` or `op_kwargs` to pass templated parameters to the python function we pass to the `PythonOperator`.

```python
def _print_variables(date_formatted):
    print(f"printing variable : {date_formatted}")

...

    print_op_kw_args = PythonOperator(
        task_id="print_op_kwargs",
        python_callable=_print_variables,
        op_kwargs={"date_formatted": "{{ ds }}"}
    )
```

And we can see the rendered argument in our task instance context menu after running:



Rendered Template

templates_dict

op_args

```
[]
```

op_kwargs

```
{
    "date_formatted": "2021-04-10"
}
```

Aside from manually passing variables, Airflow also passes a context keyword arg consisting of most template variables to a python callable if you set `provide_context` parameter to `True`.

```python
def _print_context(**context):
    print(context)

...

    print_context = PythonOperator(
        task_id="print_context",
        python_callable=_print_context,
        provide_context=True
    )
```

Check in the logs that it will print out a dictionary where we have most of our default template variables.

Now to take advantage of this, we can match our parameter name to a key in that context dictionary.

```
def _print_context(**context):
    print(context)

...

    print_context = PythonOperator(
        task_id="print_context",
        python_callable=_print_context,
        provide_context=True
    )
```

Again, you can check in the logs if it was successfully printed out.

Exercise:

Try templating variable on your own. Use both `BashOperator` and `PythonOperator`. Don't look at the sample code example or this PDF to test yourself if you've absorbed this material completely. You can check Airflow docs online if you're having problems. Name your dag