Working environment: VMware workstation 12.1.1

Operating System: Ubuntu 18.04.1 LTS

Python version: Python 2.7

# Airflow

**Introduction**

Use Airflow to author workflows as directed acyclic graphs (DAGs) of tasks. The Airflow scheduler executes your tasks on an array of workers while following the specified dependencies. Rich command lines utilities makes performing complex surgeries on DAGs a snap. The rich user interface makes it easy to visualize pipelines running in production, monitor progress and troubleshoot issues when needed.

**Version**

1.10.0

**Installation**

*# Airflow needs a home, ~/airflow is the default,* *but you can lay foundation somewhere else if you prefer*

*# (optional)*

export AIRFLOW\_HOME=~/airflow

*# install from pypi using pip*

pip install apache-airflow

*# start the web server, default port is 8080*

airflow webserver -p 8080

*# start the scheduler. This can make the server run as deamon.*

airflow scheduler

*# visit localhost:8080 in the browser and enable the example dag in the home page*

*# run task instance*

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

**Tutorial application**

*"""*

*Code that goes along with the Airflow located at:*

*http://airflow.readthedocs.org/en/latest/tutorial.html*

*"""*

**from** **airflow** **import** DAG

**from** **airflow.operators.bash\_operator** **import** BashOperator

**from** **datetime** **import** datetime, timedelta

default\_args = {

'owner': 'airflow',

'depends\_on\_past': **False**,

'start\_date': datetime(2015, 6, 1),

'email': ['airflow@example.com'],

'email\_on\_failure': **False**,

'email\_on\_retry': **False**,

'retries': 1,

'retry\_delay': timedelta(minutes=5),

*# 'queue': 'bash\_queue',*

*# 'pool': 'backfill',*

*# 'priority\_weight': 10,*

*# 'end\_date': datetime(2016, 1, 1),*

}

dag = DAG(

'tutorial', default\_args=default\_args, schedule\_interval=timedelta(1))

*# t1, t2 and t3 are examples of tasks created by instantiating operators*

t1 = BashOperator(

task\_id='print\_date',

bash\_command='date',

dag=dag)

t2 = BashOperator(

task\_id='sleep',

bash\_command='sleep 5',

retries=3,

dag=dag)

templated\_command = """

{*% f*or i in range(5) %}

echo "{{ ds }}"

echo "{{ macros.ds\_add(ds, 7)}}"

echo "{{ params.my\_param }}"

{*% e*ndfor %}

"""

t3 = BashOperator(

task\_id='templated',

bash\_command=templated\_command,

params={'my\_param': 'Parameter I passed in'},

dag=dag)

t2.set\_upstream(t1)

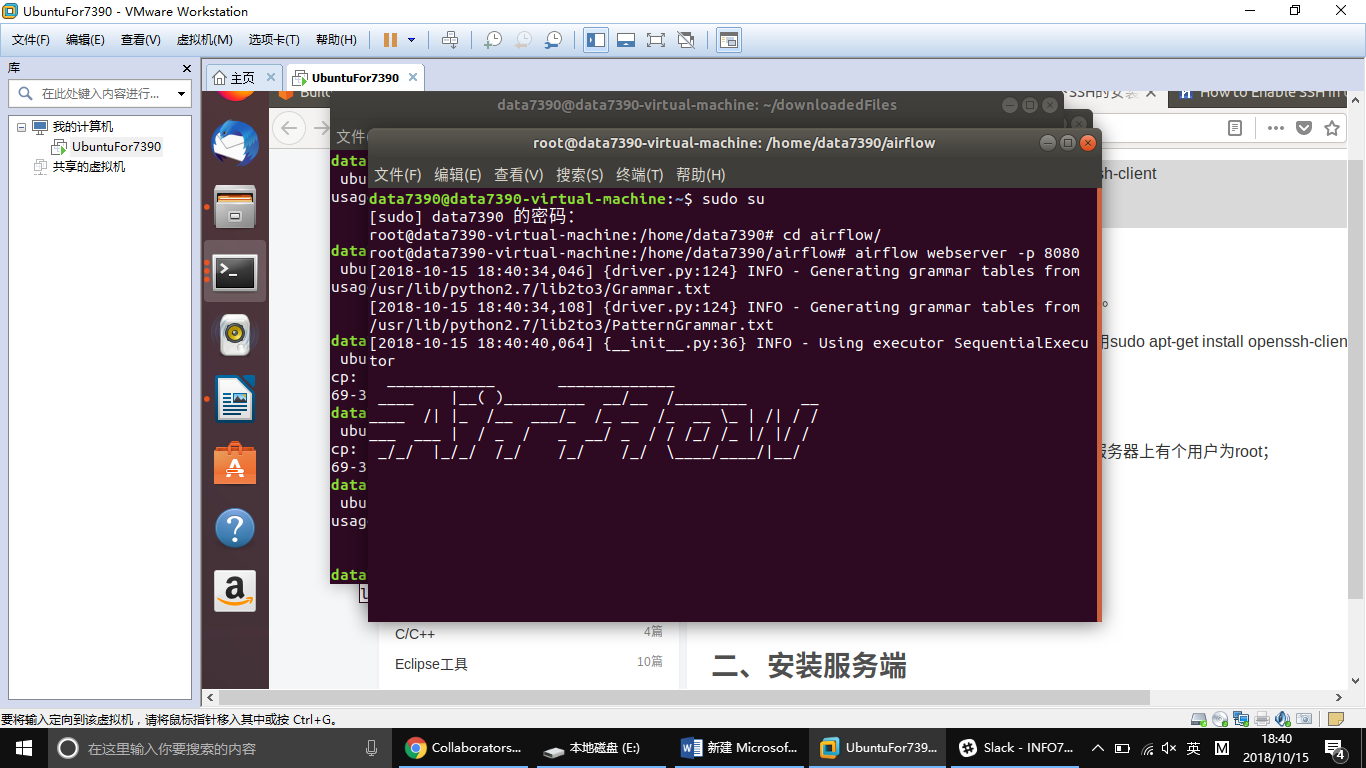
t3.set\_upstream(t1)

When I run start serve syntax, I need to get into root model, input

sudo su

Then input

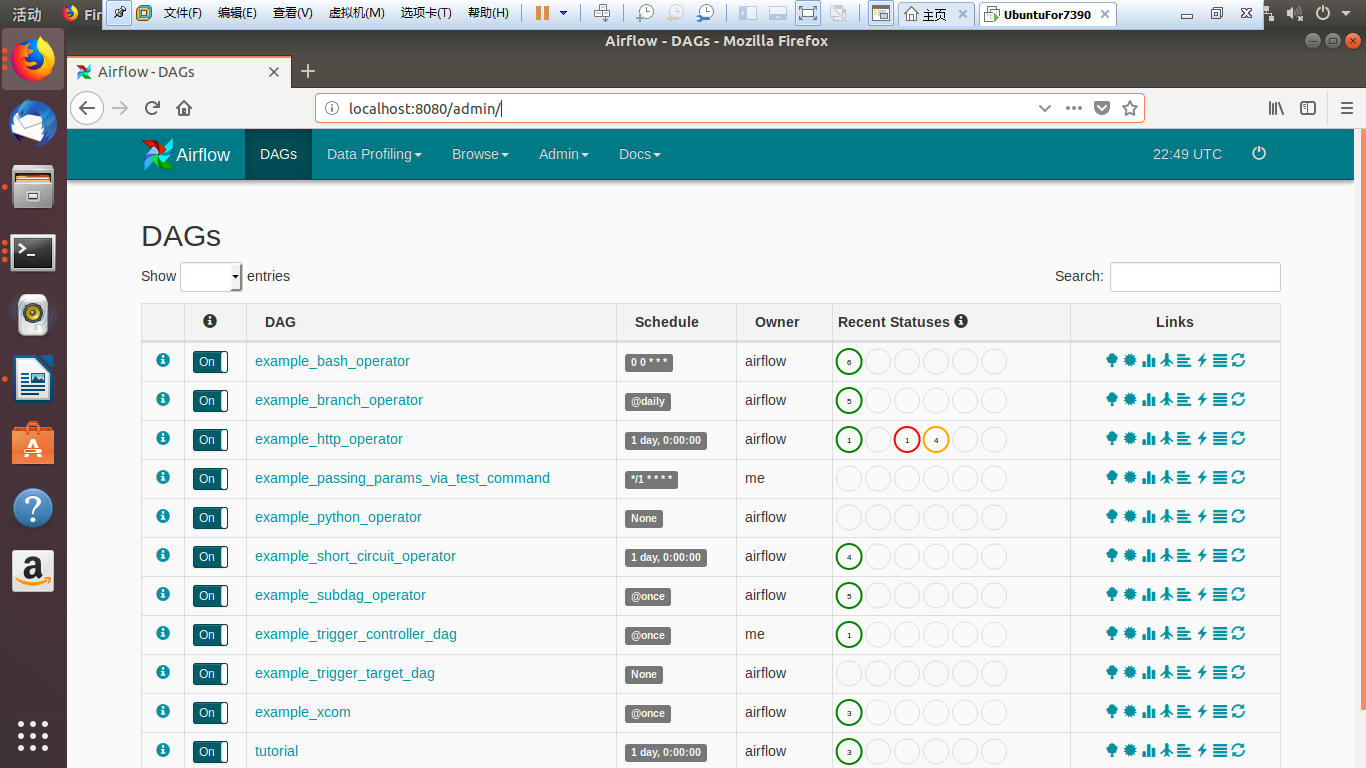
airflow# airflow webserver -p 8080



Then open your browser and input

<http://localhost:8080>

Airflow page will appear. Our tasks will show in different columns.



To run the tutorial script, we need to open a new terminal, find our task’s path. Here I need to start root model. Input

sudo su

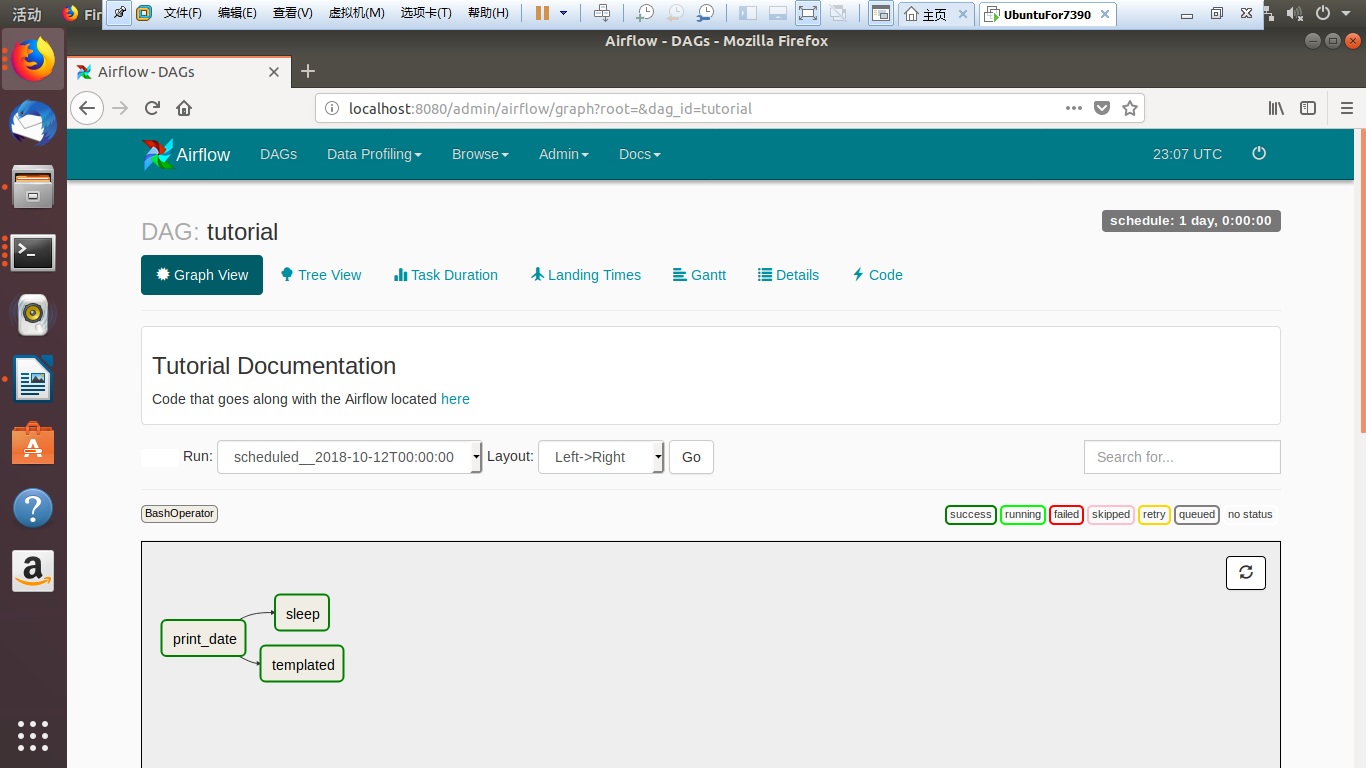
And then input

python tutorial.py

After that, we can test that task. Input

airflow test tutorial print\_date 2018-10-12

Then refresh airflow page. Click tutorial DAG and we can get see tutorial task’s running record..



# Celery

**Introduction**

Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well.

**Version**

Celery 4.2

**Choosing a Broker**

Celery requires a solution to send and receive messages; usually this comes in the form of a separate service called a message broker. In my presentation, I chose RabbitMQ.

Installing RabbitMQ:

sudo apt-get install rabbitmq-server

Installing Celery:

pip install celery

Application:

Create the file tasks.py:

**from** **celery** **import** Celery

app = Celery('tasks', broker='pyamqp://guest@localhost//')

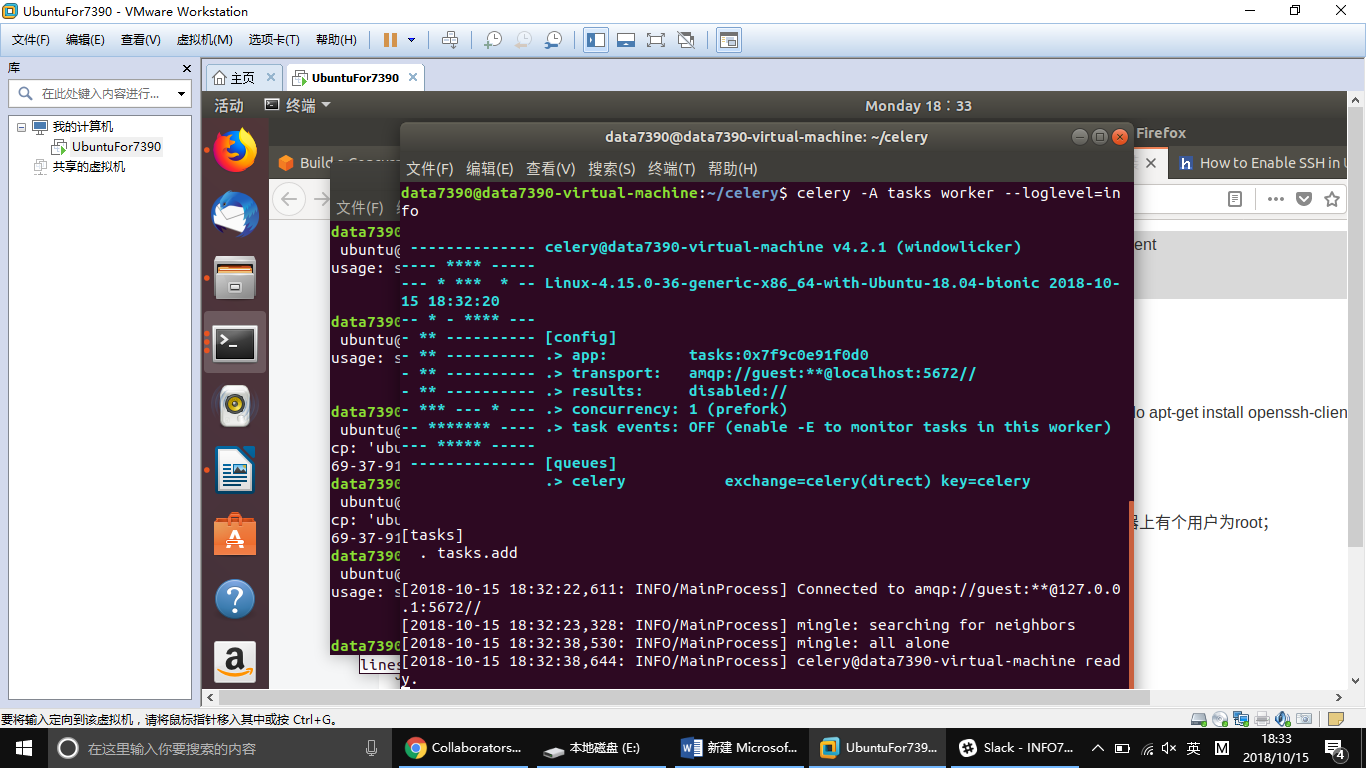
**@app.task**

**def** **add**(x, y):

**return** x + y

Running the Celery worker server

celery -A tasks worker --loglevel=info



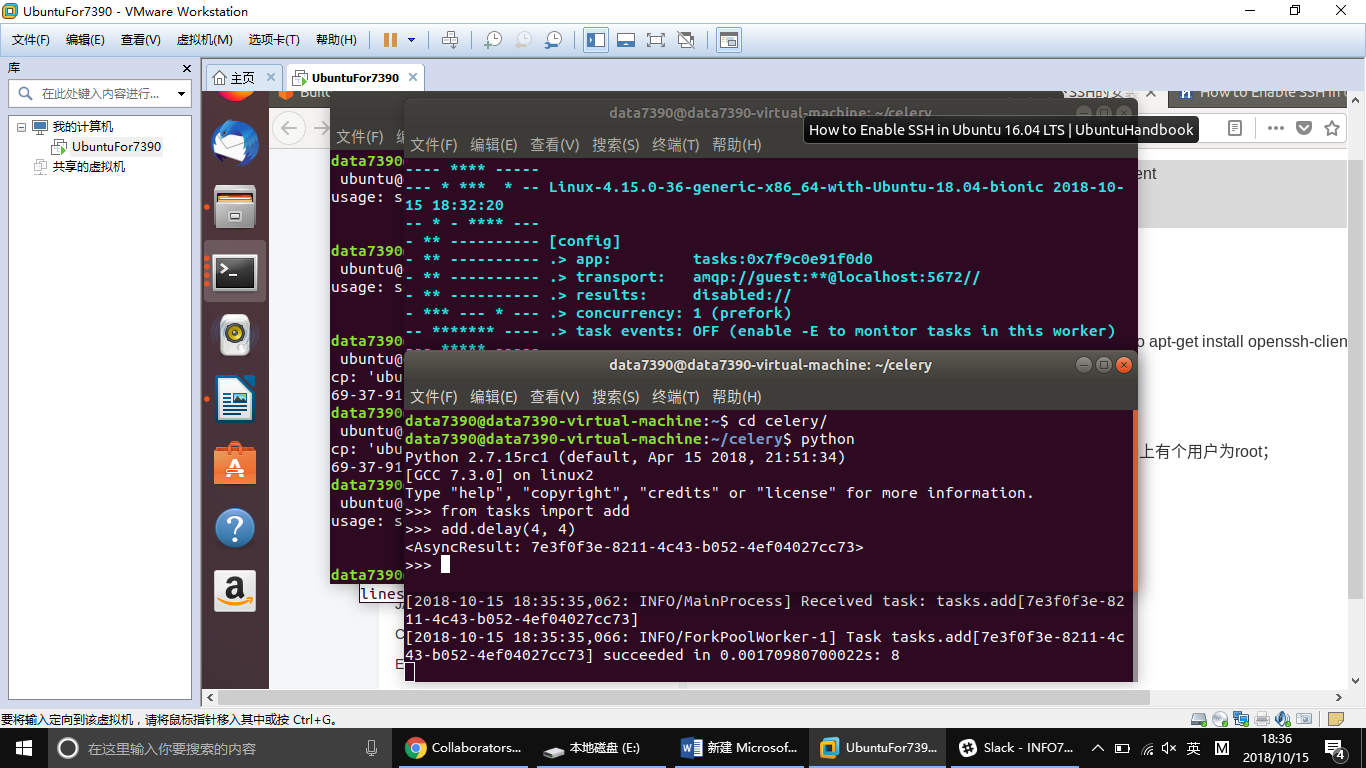
Calling the task:

After running celery server, you can open a new terminal and call that task:

**>>> from** **tasks** **import** add

**>>>** add.delay(**4**, **4**)

Then, the result will be printed at the previous terminal.



**Reference**

<https://airflow.apache.org/start.html>

<http://docs.celeryproject.org/en/latest/getting-started/first-steps-with-celery.html>