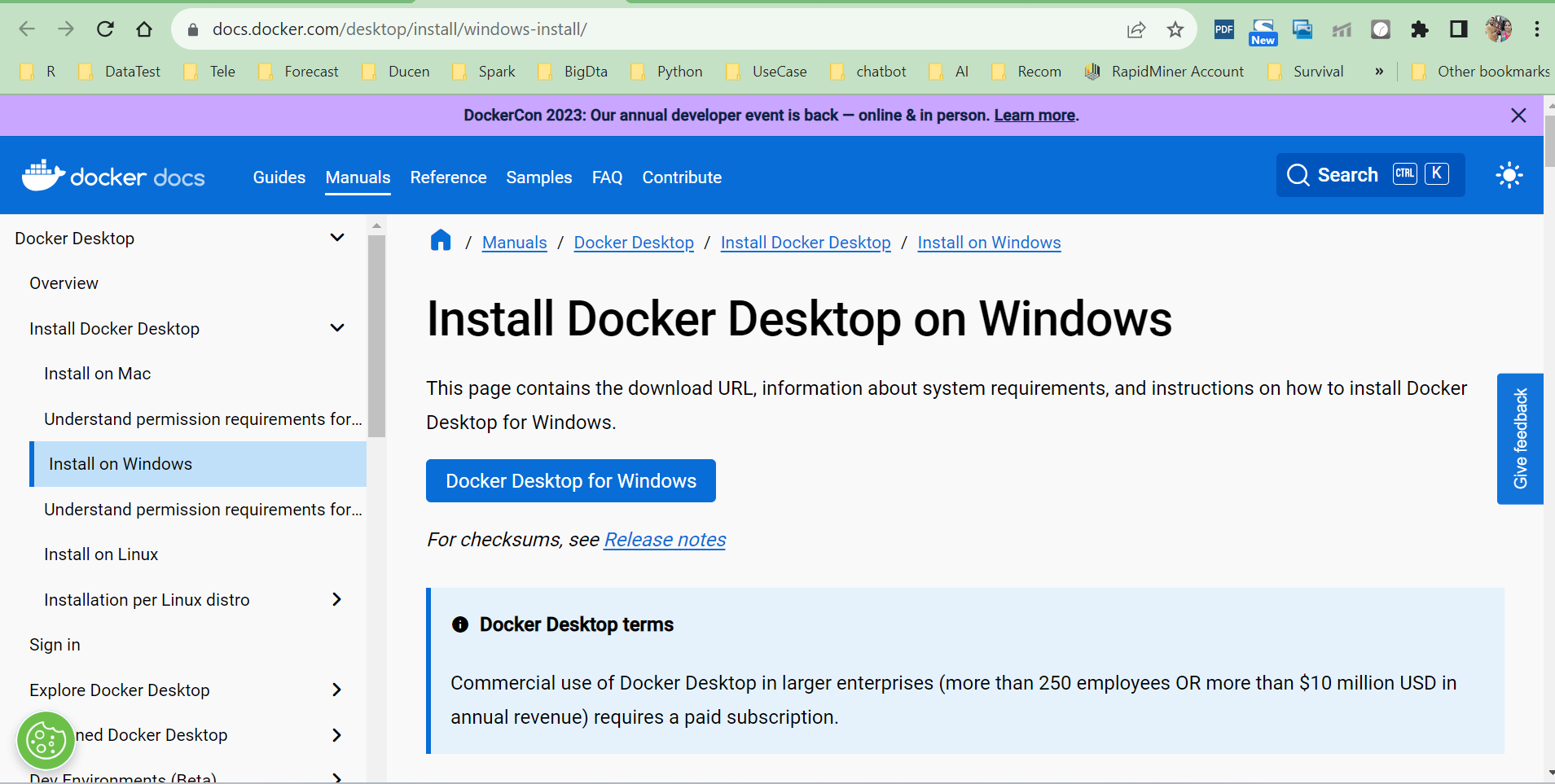
**Airflow installation using Docker:**

1. **Download docker desktop using the following url:**

<https://docs.docker.com/desktop/install/windows-install/>



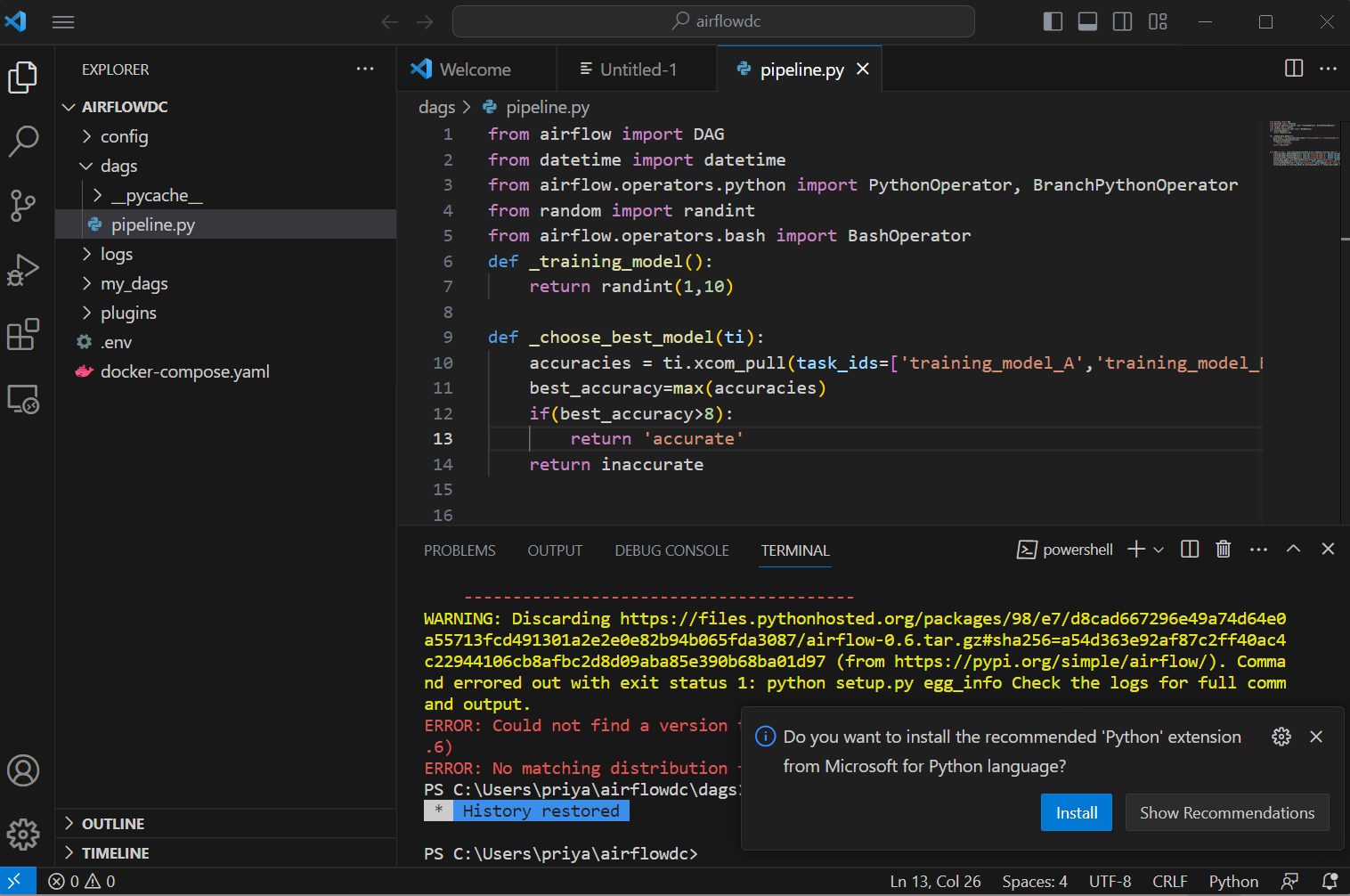
1. **Once installed, check whether the you have linux updated else run the following command in your system to update the linux os:**

“ wsl –update”

And check the docker version using the following command:

docker --version

1. **Then install visual studio code from the url:** <https://code.visualstudio.com/download>



The folders created in the location will be considered as a docker composer which compose the docker image and runs the script as a dag in Airflow, and create the subfolders : dags, logs, plugins

1. **In order to make the docker compose up, run the following commands in the terminal:**

>>docker ps

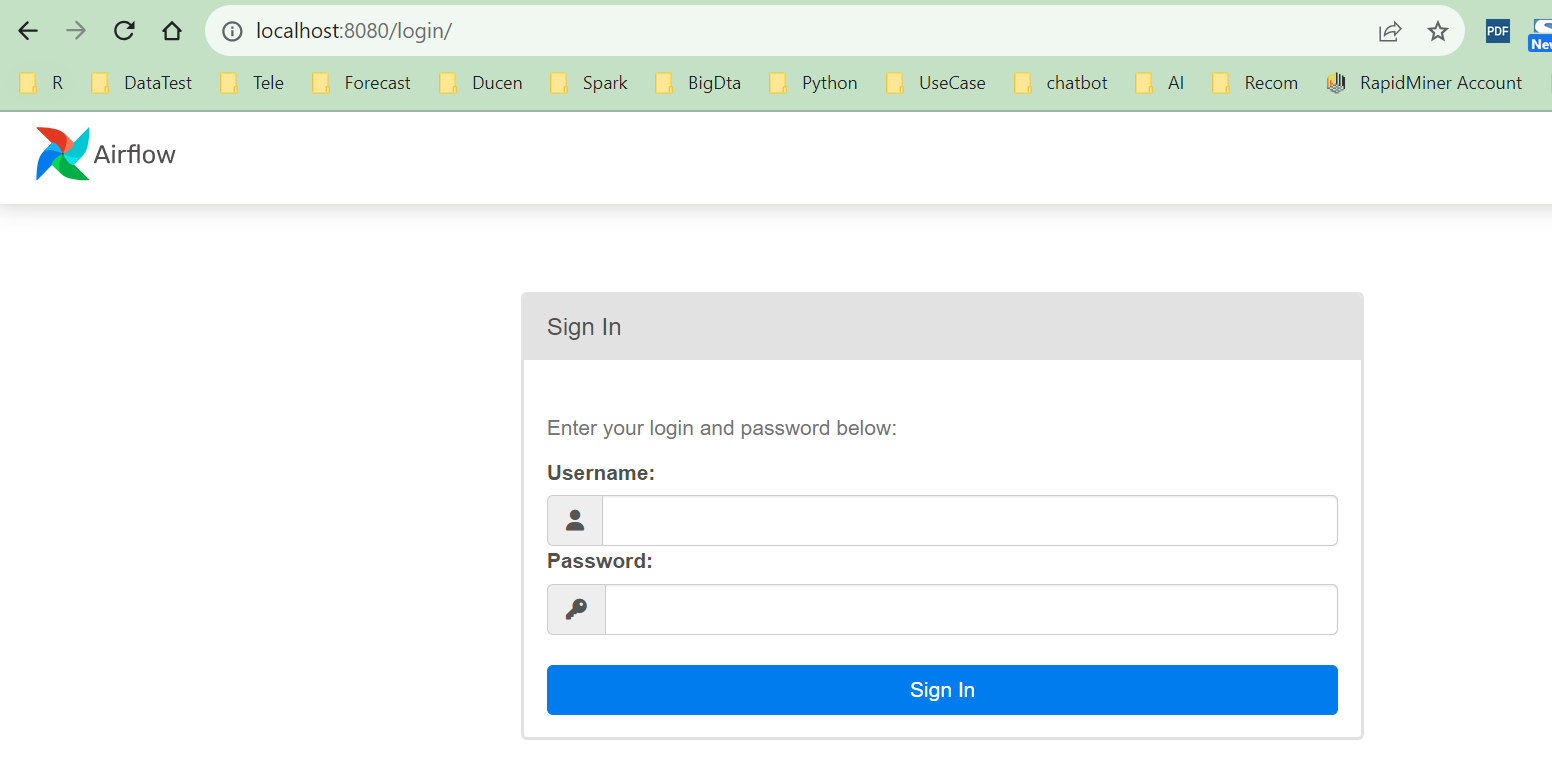
>>curl 'https://airflow.apache.org/docs/apache-airflow/2.7.0/docker-compose.yaml' -o docker- compose.yaml

>>docker-compose up -d

1. **Go to the browser and open the following url:**

<http://localhost:8080/home>

default username and password is “Airflow”, “Airflow”



Sometimes it takes 2 minutes to load the site

**6: Create the sample DAG using the following script below:**

from airflow import DAG

from datetime import datetime

from airflow.operators.python import PythonOperator, BranchPythonOperator

from random import randint

from airflow.operators.bash import BashOperator

def \_training\_model():

    return randint(1,10)

def \_choose\_best\_model(ti):

    accuracies = ti.xcom\_pull(task\_ids=['training\_model\_A','training\_model\_B','training\_model\_C'])

    best\_accuracy=max(accuracies)

    if(best\_accuracy>8):

        return 'accurate'

    return inaccurate

with DAG("my\_dag", start\_date=datetime(2023,8,21),schedule\_interval="@daily", catchup=False) as dag:

    training\_model\_A=PythonOperator( task\_id="training\_model\_A", python\_callable=\_training\_model)

    training\_model\_B=PythonOperator( task\_id="training\_model\_B", python\_callable=\_training\_model)

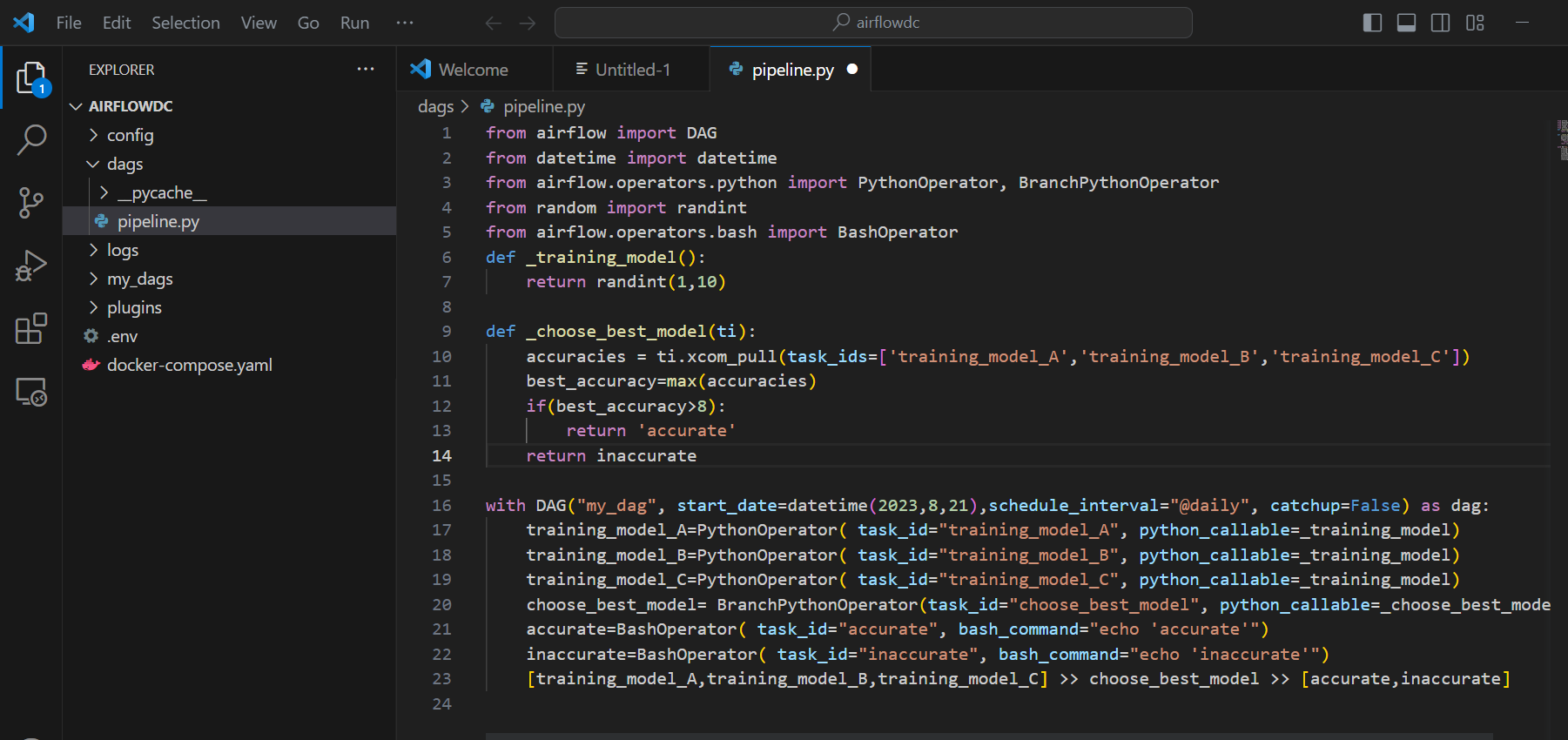
    training\_model\_C=PythonOperator( task\_id="training\_model\_C", python\_callable=\_training\_model)

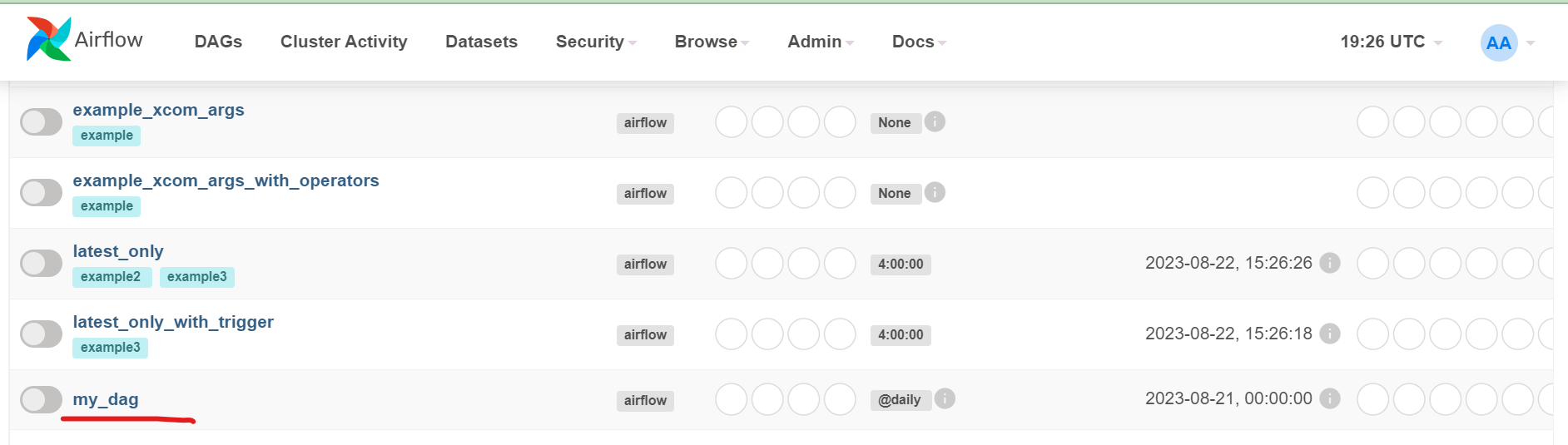
    choose\_best\_model= BranchPythonOperator(task\_id="choose\_best\_model", python\_callable=\_choose\_best\_model)

    accurate=BashOperator( task\_id="accurate", bash\_command="echo 'accurate'")

    inaccurate=BashOperator( task\_id="inaccurate", bash\_command="echo 'inaccurate'")

    [training\_model\_A,training\_model\_B,training\_model\_C] >> choose\_best\_model >> [accurate,inaccurate]





The dependencies on the tasks or the operators can be seen under the graph section as below:

