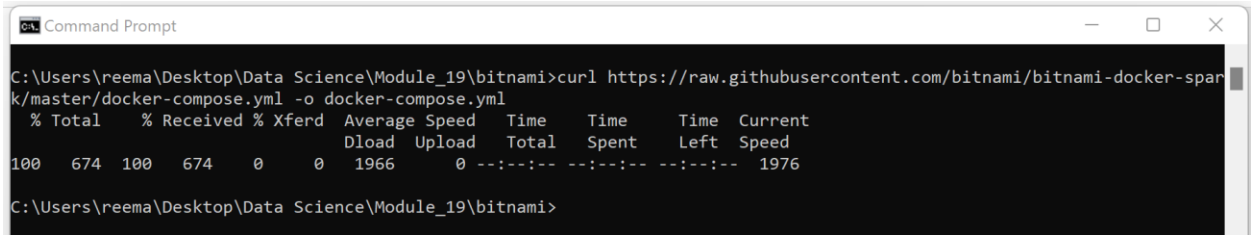
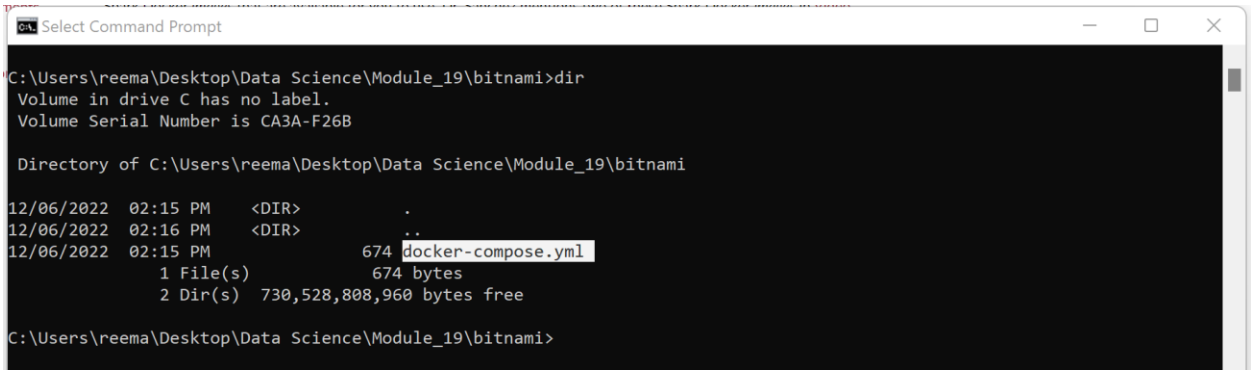
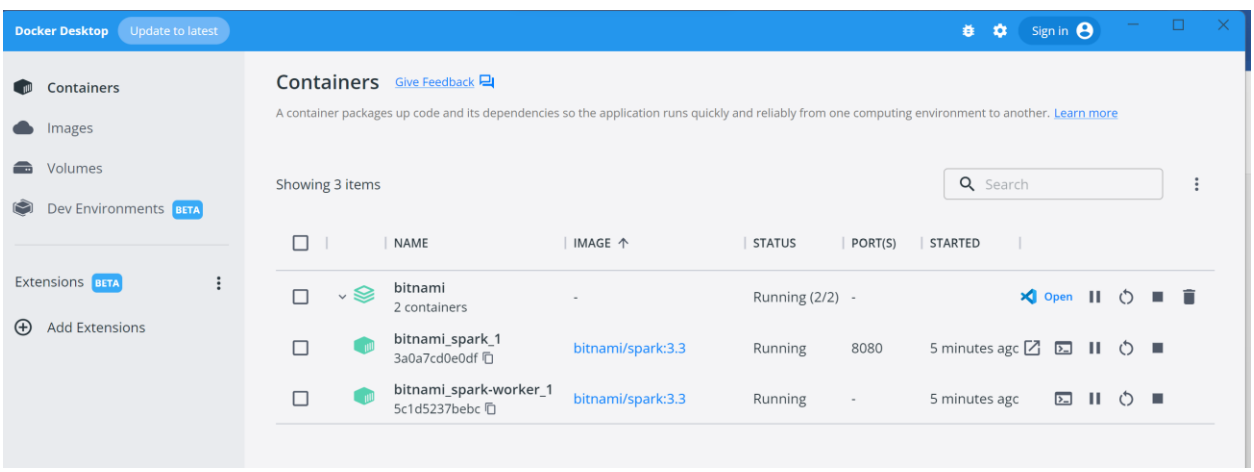
Part 1: PySpark

1. Provide a screenshot to show that you correctly pulled the *image* and that the docker-compose.yaml file is present.





2. Provide a screenshot of your Docker Desktop to show that you correctly pulled the *containers.*



3. Provide a screenshot to show that you successfully copied the departuredelays.csv file to the bitnami_spark_1 *container*.

```
Command Prompt

C:\Users\reema\Desktop\Data Science\Module_19\data>docker cp departuredelays.csv bitnami_spark_1:/departuredelays.csv

C:\Users\reema\Desktop\Data Science\Module_19\data>
```

```
Select docker  exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh

$ pwd
/opt/bitnami/spark
$ cd /
$ pwd
/
$ ls -l
total 32684
drwxr-xr-x   1 root root     4096 Dec  4 05:59 bin
drwxr-xr-x   2 root root     4096 Sep  3 12:10 boot
-rwxr-xr-x   1 root root 33396236 Dec  7 05:42 departuredelays.csv
drwxr-xr-x   5 root root      340 Dec  7 05:39 dev
drwxr-xr-x   1 root root     4096 Dec  6 22:19 etc
drwxr-xr-x   2 root root     4096 Sep  3 12:10 home
drwxr-xr-x   1 root root     4096 Aug 23 18:54 lib
drwxr-xr-x   2 root root     4096 Oct 29 12:12 lib64
drwxr-xr-x   2 root root     4096 Oct 29 12:12 media
drwxr-xr-x   2 root root     4096 Oct 29 12:12 mnt
drwxrwxr-x   1 root root     4096 Dec  4 05:58 opt
dr-xr-xr-x 253 root root        0 Dec  7 05:39 proc
drwx------   2 root root     4096 Oct 29 12:12 root
drwxr-xr-x   4 root root     4096 Oct 29 12:12 run
drwxr-xr-x   1 root root     4096 Dec  4 05:59 sbin
drwxr-xr-x   2 root root     4096 Oct 29 12:12 srv
dr-xr-xr-x  11 root root        0 Dec  7 05:39 sys
drwxrwxrwt   1 root root     4096 Dec  6 22:19 tmp
drwxrwxr-x   1 root root     4096 Dec  4 05:58 usr
drwxr-xr-x   1 root root     4096 Oct 29 12:12 var
$
```

4. Provide a screenshot to show that you successfully opened PySpark.

```
docker  exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh

$ pyspark
Python 3.8.15 (default, Nov 28 2022, 06:25:33)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/12/08 06:30:07 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.3.1
      /_/

Using Python version 3.8.15 (default, Nov 28 2022 06:25:33)
Spark context Web UI available at http://3a0a7cd0e0df:4040
Spark context available as 'sc' (master = local[*], app id = local-1670481008529).
SparkSession available as 'spark'.
>>>
```

5. Provide a screenshot to show that you successfully started a PySpark session.

```
Select docker  exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh
$ pyspark
Python 3.8.15 (default, Nov 28 2022, 06:25:33)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/12/08 06:30:07 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.3.1
      /_/

Using Python version 3.8.15 (default, Nov 28 2022 06:25:33)
Spark context Web UI available at http://3a0a7cd0e0df:4040
Spark context available as 'sc' (master = local[*], app id = local-1670481008529).
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
>>>
```

6. Provide a screenshot to show that you successfully defined the spark PySpark session.

```
docker  exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh
>>> spark = (SparkSession
...     .builder
...     .appName("Assignment19.3")
...     .getOrCreate())
22/12/08 06:37:29 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>>
```

7. Provide a screenshot to show that you successfully defined the assignment19_3_data variable.

```
Select docker  exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh
    ^
SyntaxError: invalid character in identifier
>>> spark = (SparkSession
...     .builder
...     .appName("Assignment19.3")
...     .getOrCreate())
22/12/08 06:37:29 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> assignment19_3_data="/departuredelays.csv"
>>>
```

8. Provide a screenshot to show that you successfully defined the df *dataframe* that contains all of the entries in the departuredelays.csv file.

```
Select docker  exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh
>>> spark = (SparkSession
...     .builder
...     .appName("Assignment19.3")
...     .getOrCreate())
22/12/08 06:37:29 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> assignment19_3_data="/departuredelays.csv"
>>> df  = (spark.read.format("csv")
...     .option("interSchema", "true")
...     .option("header", "true")
...     .load(assignment19_3_data))
```

9. Provide a screenshot to show that you successfully created a view of the assignment19_3_table *dataframe*.

```
docker exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh
>>> df.createOrReplaceTempView("assignment19_3_table")
>>>
```

10. Provide a screenshot to show that you selected the correct entries from your data. Your data should display the first 15 flights from PHL to DFW that had a delay of greater than 150 minutes.

```
docker exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh
>>> spark.sql("""SELECT date,delay, origin, destination
...      FROM assignment19_3_table
...      WHERE delay > 150 AND ORIGIN = 'PHL' AND DESTINATION = 'DFW'
...      ORDER BY CAST(delay AS INT) DESC""").show(15)
+--------+-----+------+-----------+
|    date|delay|origin|destination|
+--------+-----+------+-----------+
|01141620| 1177|   PHL|        DFW|
|02200820|  741|   PHL|        DFW|
|02141850|  316|   PHL|        DFW|
|01022039|  295|   PHL|        DFW|
|03131830|  280|   PHL|        DFW|
|01011425|  279|   PHL|        DFW|
|01110820|  277|   PHL|        DFW|
|03171140|  249|   PHL|        DFW|
|02231425|  242|   PHL|        DFW|
|03281830|  222|   PHL|        DFW|
|02190933|  199|   PHL|        DFW|
|02211425|  198|   PHL|        DFW|
|01081855|  193|   PHL|        DFW|
|02211620|  192|   PHL|        DFW|
|03251425|  182|   PHL|        DFW|
+--------+-----+------+-----------+
only showing top 15 rows

>>>
```

11. Provide a screenshot to show that you selected the correct entries from your data. Your data should display the first 10 flights that have a distance of less than 200 miles and the resulting table should contain all of the columns in the original dataset.

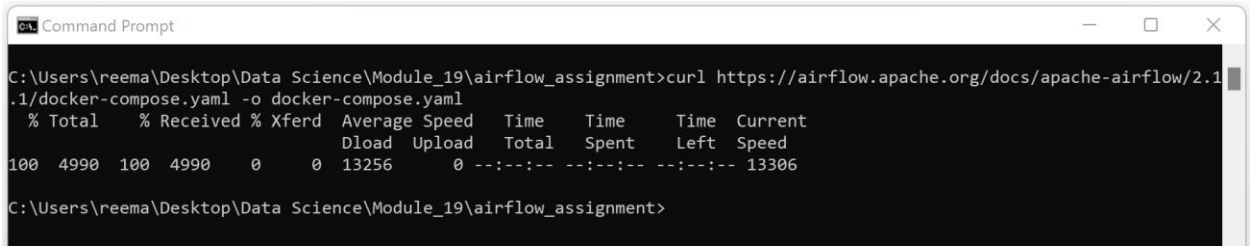
```
docker exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh
>>> spark.sql("""SELECT date,delay, distance, origin, destination
...      FROM assignment19_3_table
...      WHERE distance < 200
...      ORDER BY CAST( distance AS INT) DESC """).show(10)
+--------+-----+--------+------+-----------+
|    date|delay|distance|origin|destination|
+--------+-----+--------+------+-----------+
|03051944|   -4|     199|   CVG|        DTW|
|03041715|   -7|     199|   CVG|        DTW|
|03051715|   -1|     199|   CVG|        DTW|
|03021944|    3|     199|   CVG|        DTW|
|03031944|   -6|     199|   CVG|        DTW|
|03041944|   -4|     199|   CVG|        DTW|
|03051140|   -4|     199|   CVG|        DTW|
|03021715|    0|     199|   CVG|        DTW|
|03061944|   -2|     199|   CVG|        DTW|
|03031715|   -2|     199|   CVG|        DTW|
+--------+-----+--------+------+-----------+
only showing top 10 rows

>>>
```

12. Provide a screenshot to show that you selected the correct entries from your data. Your data should display the first 10 flights that have a distance greater than 600 miles and the resulting table should contain all of the columns in the original dataset.

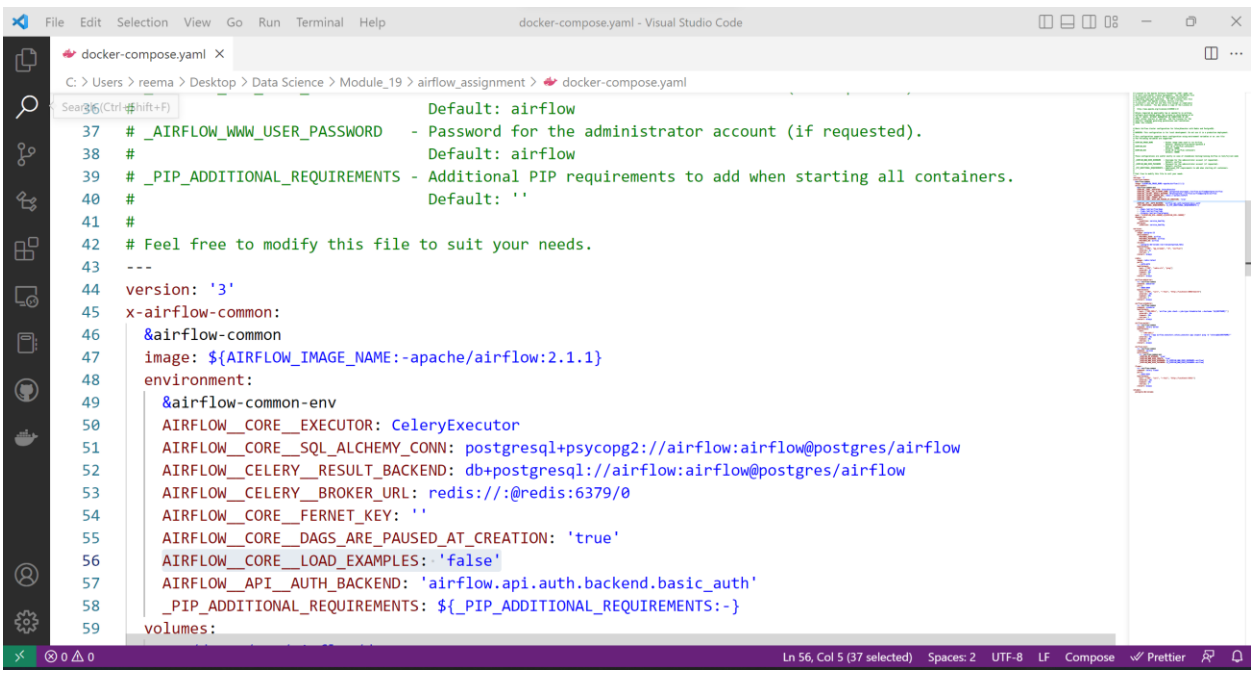
```
docker exec -it 3a0a7cd0e0dfcbac0d306a38a38afee1ba575ed2dc19417054614d7acecf98ee /bin/sh
>>> spark.sql("""SELECT date,delay, distance, origin, destination
...      FROM assignment19_3_table
...      WHERE distance > 600
...      ORDER BY CAST( distance AS INT) DESC """).show(10)
+--------+-----+--------+------+-----------+
|    date|delay|distance|origin|destination|
+--------+-----+--------+------+-----------+
|01090900|   -3|    4330|   JFK|        HNL|
|01050900|   98|    4330|   JFK|        HNL|
|01080900|   14|    4330|   JFK|        HNL|
|01020900|    1|    4330|   JFK|        HNL|
|01040900|  111|    4330|   JFK|        HNL|
|01060900|   -2|    4330|   JFK|        HNL|
|01070900|    3|    4330|   JFK|        HNL|
|01010900|    6|    4330|   JFK|        HNL|
|01110900|   -4|    4330|   JFK|        HNL|
|01030900|  784|    4330|   JFK|        HNL|
+--------+-----+--------+------+-----------+
only showing top 10 rows

>>>
```

Part 2: Airflow

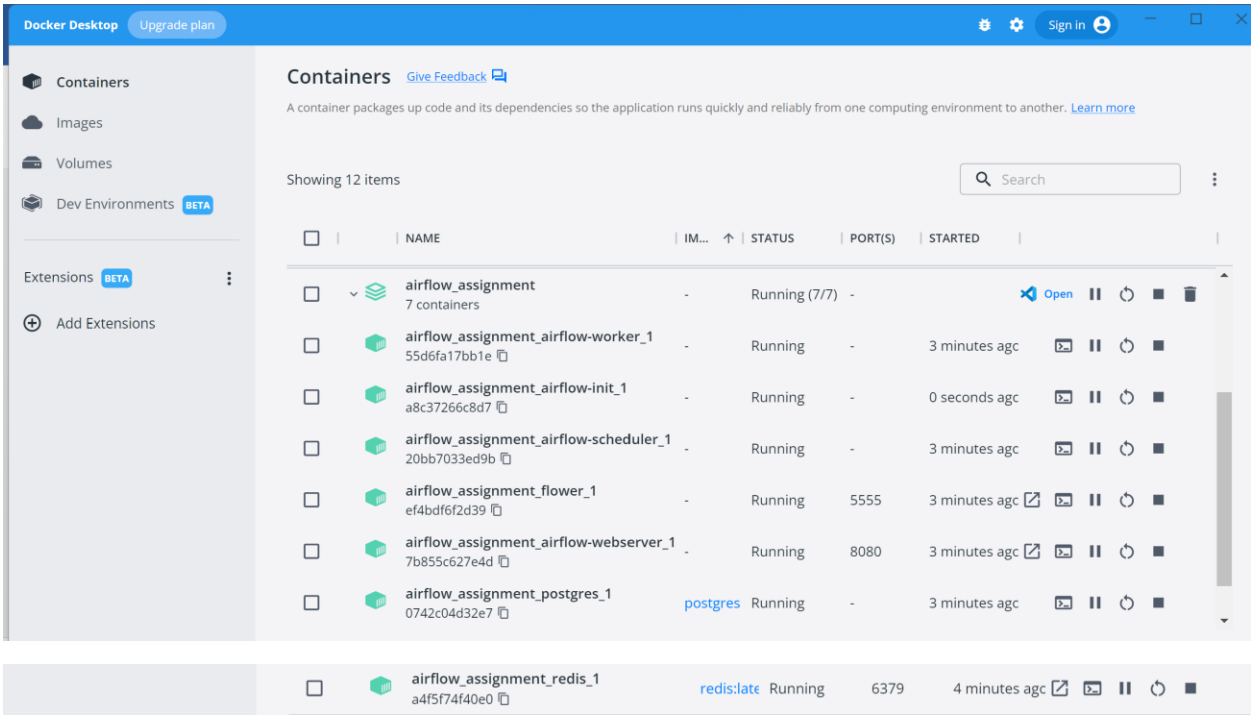
1. Provide a screenshot of your Terminal window response to show that you correctly pulled the Airflow file.



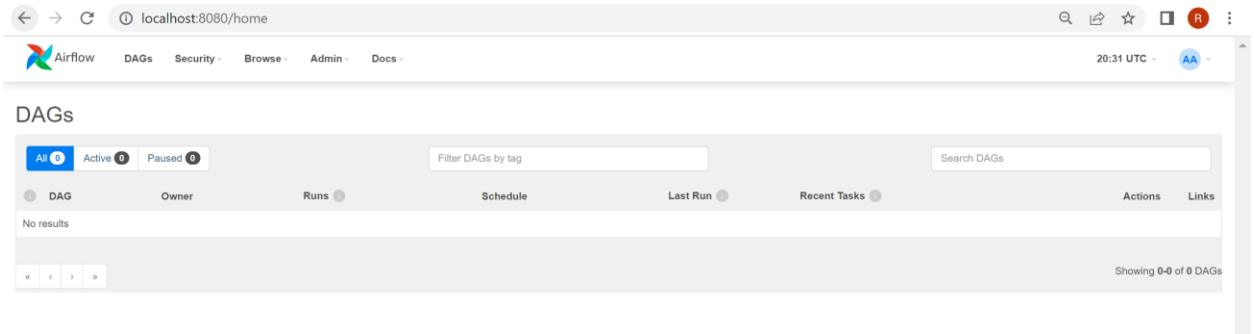
2. Provide a screenshot to show the changed example value (AIRFLOW__CORE__LOAD_EXAMPLES set to false).



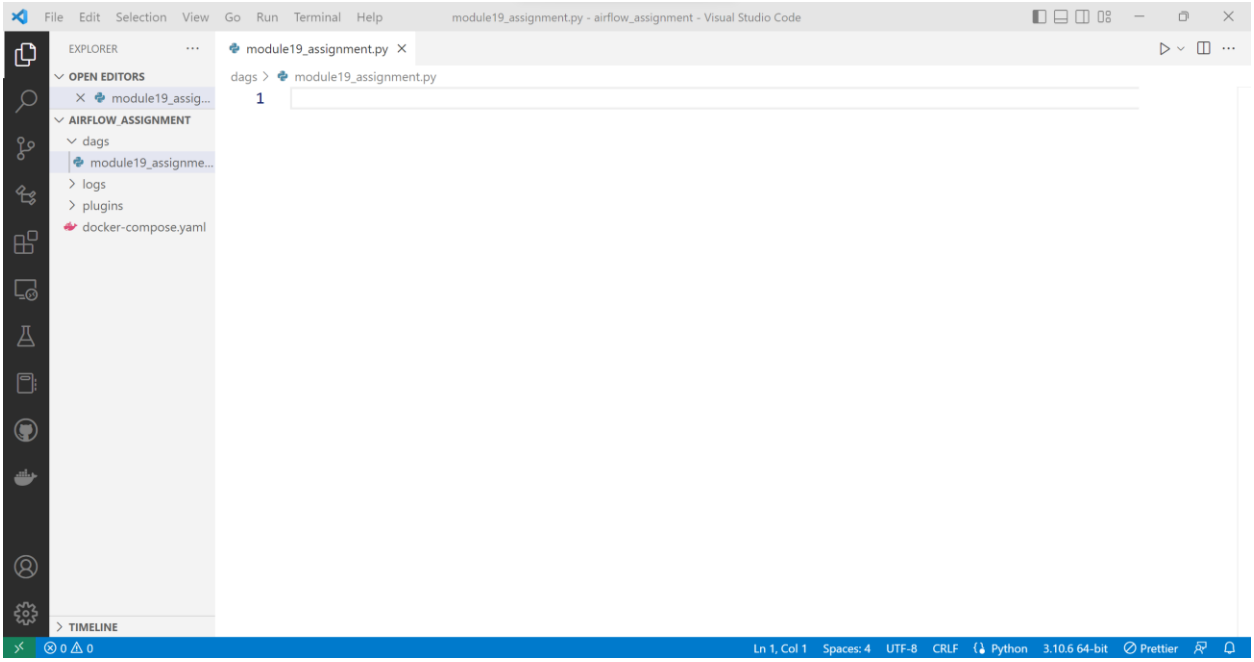
3. Provide a screenshot to show that the Airflow Docker *containers* are running.



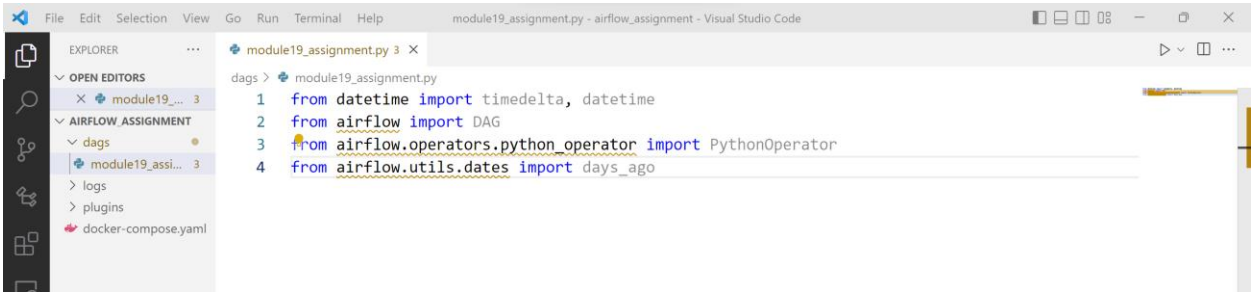
4. Provide a screenshot of your browser window to show that you have successfully logged in to Airflow.



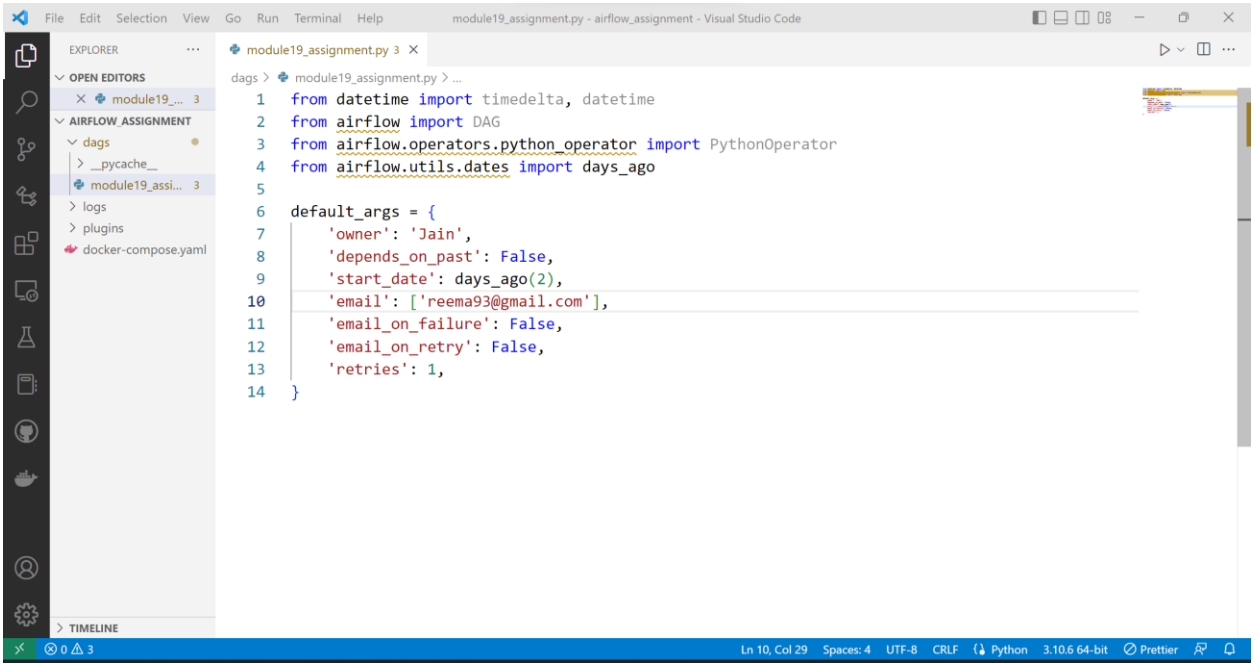
5. Provide a screenshot to show that you created the module19_assignment.py file.



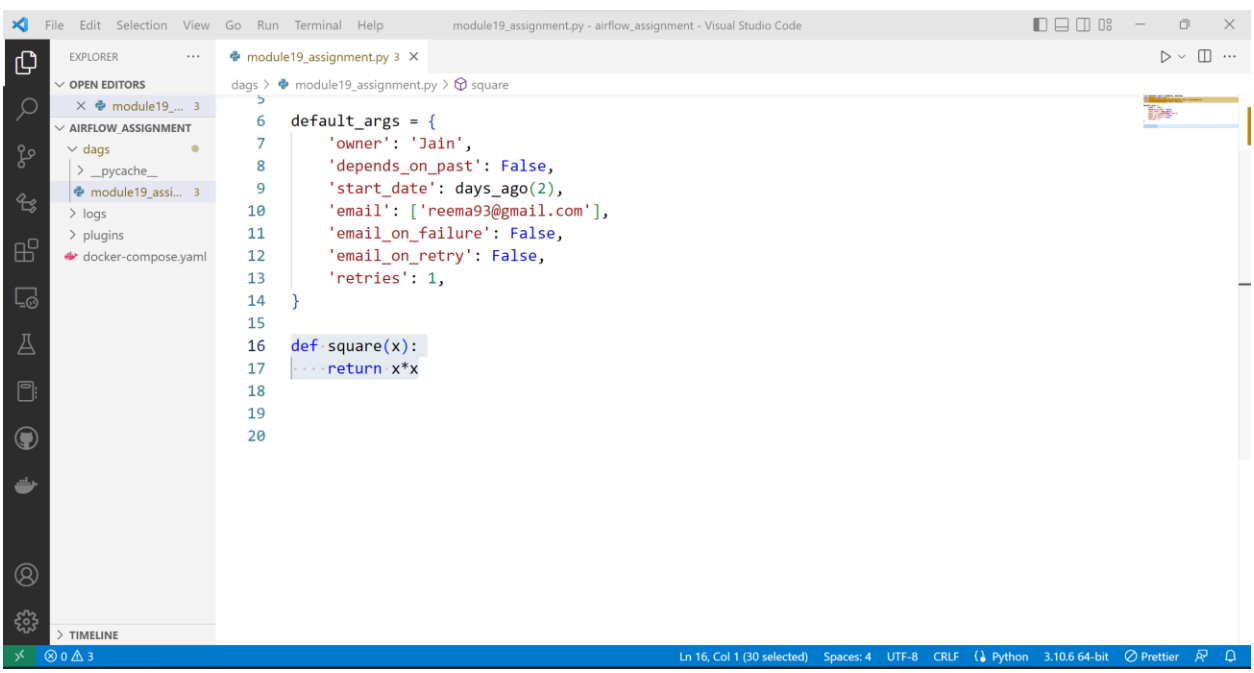
6. Provide a screenshot to show that you correctly imported the required *libraries*.



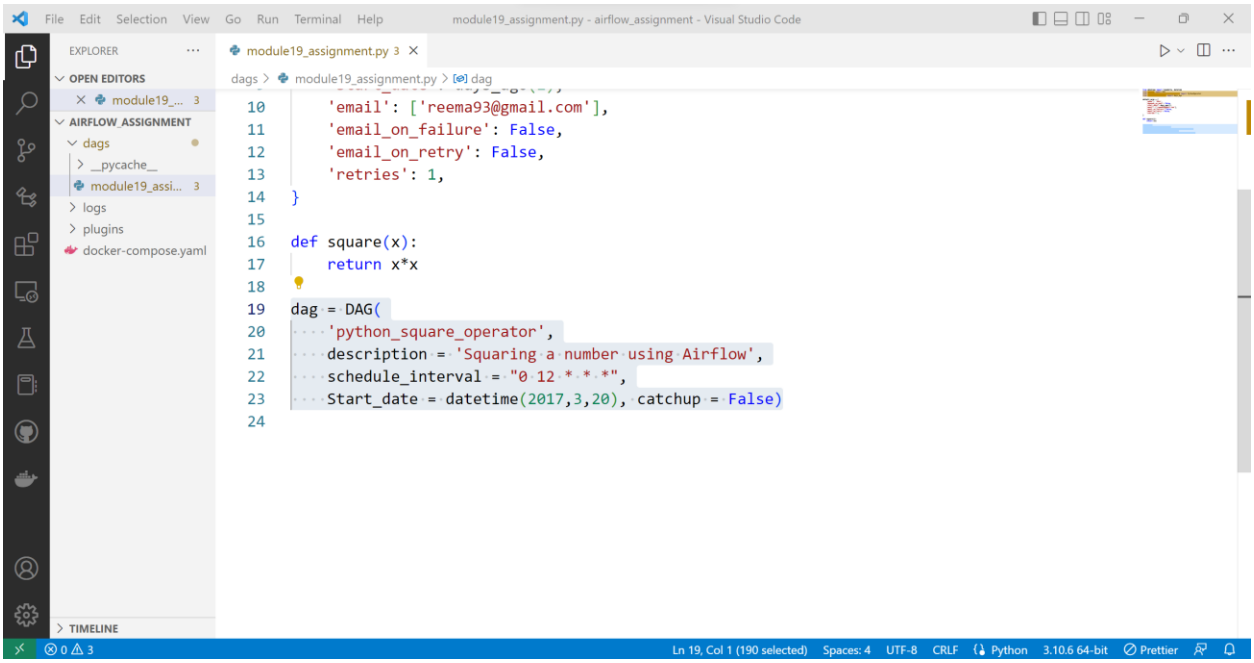
7. Provide a screenshot to show that you set up your DAG correctly, including your last name and email address.



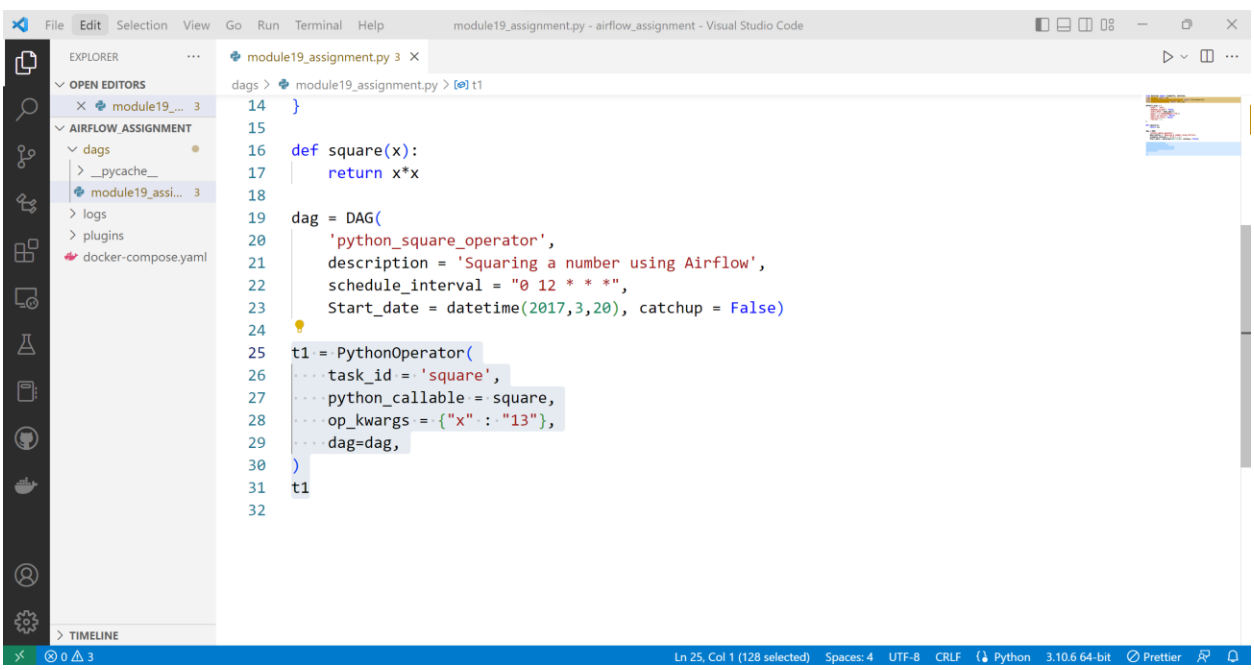
8. Provide a screenshot to show that you defined the square() *function* correctly.



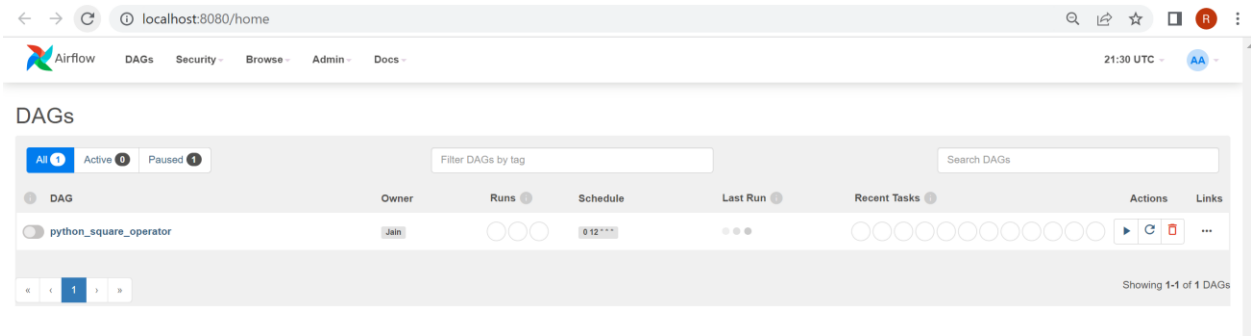
9. Provide a screenshot to show that you correctly defined the DAG object.



10. Provide a screenshot to show that you defined the DAG *Task* correctly.



11. Provide a screenshot of the Airflow UI to show that your DAG is configured correctly.



12. Provide a screenshot of the log to show that the DAG ran successfully.