

Урок 7. Построение пайплайнов и визуализация потоков данных в Airflow

1. Ваша задача с использование пандас, записать полученную температуру в таблицу mysql. Таблица должна содержать как минимум текущее время и температуру (т.е. два поля). Таблицу не удаляем, используем append.

```
from sqlalchemy import create_engine
from datetime import datetime
from pandas.io import sql
import requests
```

```
api_key = 'bc6d51747326a116e97fbc66146b6deb'
city = 'Tyumen'
```

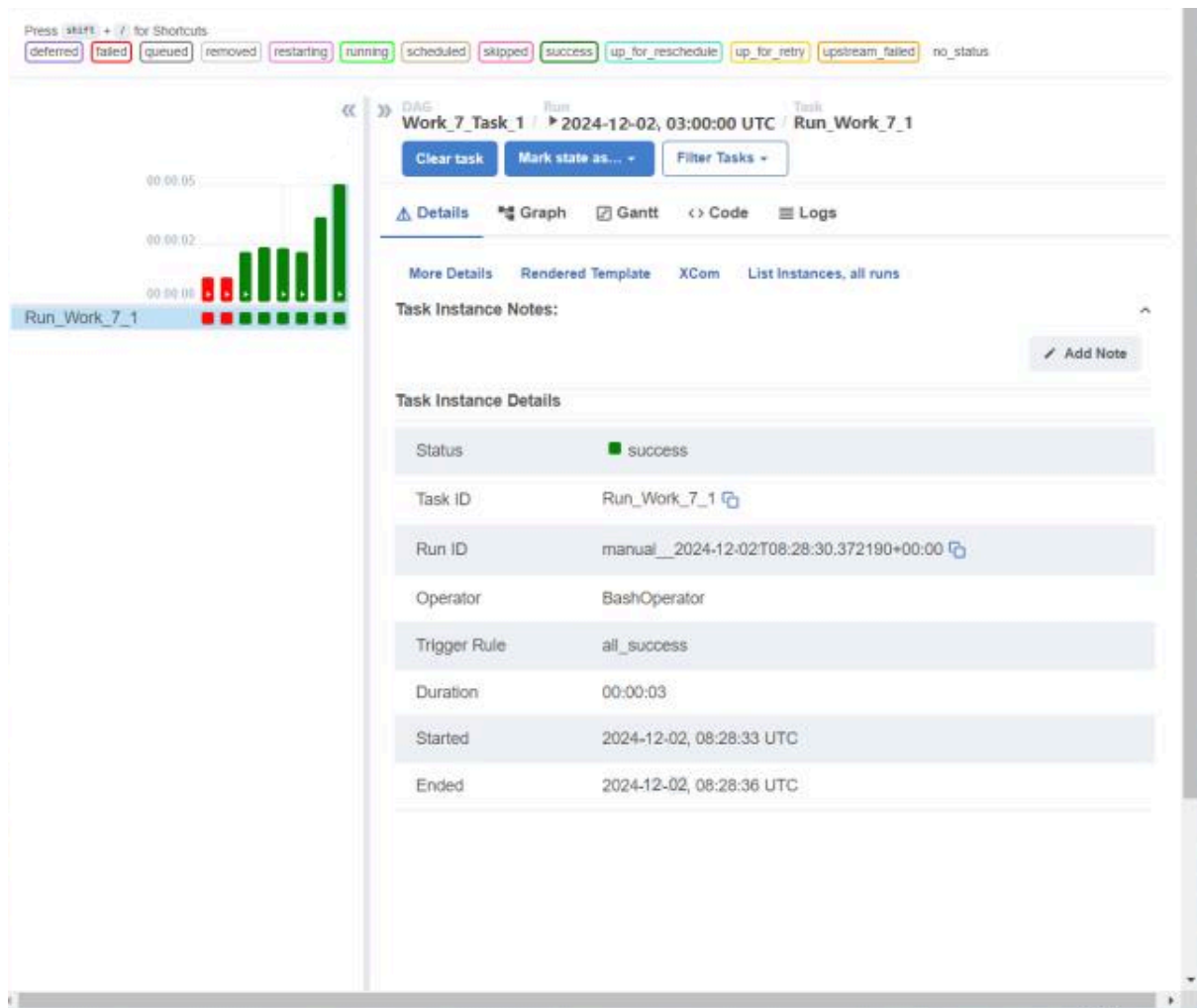
```
def openwear_get_temp(api_key, city):
    url = f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}'
    response = requests.get(url)
    data = response.json()
    temperature = data['main']['temp']
    timestamp = data['dt']
    date_time = datetime.utcfromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S')
    return round(float(temperature) - 273.15, 2), date_time
```

```
def save_weather(api_key, city):
    temperature, date_time = openwear_get_temp(api_key, city)

    con = create_engine("mysql://root:1@localhost:33061/spark")
    sql.execute("""drop table if exists spark.`Temperature_Tyumen`""", con)
    sql.execute("""CREATE TABLE if not exists spark.`Temperature_Tyumen`
        (`date_time` TIMESTAMP NULL DEFAULT NULL, `temperature` FLOAT NULL
        DEFAULT NULL)
        COLLATE='utf8mb4_general_ci' ENGINE=InnoDB""", con)

    with con.connect() as connection:
        ins = f"INSERT INTO spark.`Temperature_Tyumen` (date_time, temperature) VALUES"
        ('{date_time}', {temperature})"
        connection.execute(ins)
```

```
save_weather(api_key, city)
```



2. То что мы делали на четвертом семинаре (Д34) задача с графиком. Нужно с помощью аирфлоу (PythonOperator) сохранить этот график в png/jpeg. Используйте пандас, считайте им таблицу из mysql, постройте график и сохраните его в указанную директорию. На проверку ДЗ высылайте код и скриншоты аирфлоу выполненных задач, логов и сохраненного файла (в pdf).

```
import time, sys, os
from pyspark.sql.session import SparkSession
from pyspark.sql.functions import col, lit
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
from pandas.io import sql
import warnings

warnings.filterwarnings("ignore")
t0=time.time()
con=create_engine("mysql://root:1@localhost:33061/spark")
os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable
spark=SparkSession.builder.appName("AiR Home Work №7").getOrCreate()
```

```

sql.execute("""drop table if exists spark.`AiR_W7T2`""",con)
sql.execute("""CREATE TABLE if not exists spark.`AiR_W7T2` (
    `number` INT(10) NULL DEFAULT NULL,
    `Month` DATE NULL DEFAULT NULL,
    `Payment amount` FLOAT NULL DEFAULT NULL,
    `Payment of the principal debt` FLOAT NULL DEFAULT NULL,
    `Payment of interest` FLOAT NULL DEFAULT NULL,
    `Balance of debt` FLOAT NULL DEFAULT NULL,
    `interest` FLOAT NULL DEFAULT NULL,
    `debt` FLOAT NULL DEFAULT NULL
)
COLLATE='utf8mb4_general_ci'
ENGINE=InnoDB""",con)

```

```

from pyspark.sql.window import Window
from pyspark.sql.functions import sum as sum1
w =
Window.partitionBy(lit(1)).orderBy("number").rowsBetween(Window.unboundedPreceding,
Window.CurrentRow)

```

```

dfG = spark.read.format("com.crealytics.spark.excel")\
    .option("dataAddress", "General!A1:F361")\
    .option("useHeader", "false")\
    .option("treatEmptyValuesAsNulls", "false")\
    .option("inferSchema", "true").option("addColorColumns", "true")\
    .option("usePlainNumberFormat", "true")\
    .option("startColumn", 0)\
    .option("endColumn", 99)\
    .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
    .option("maxRowsInMemory", 20)\
    .option("excerptSize", 10)\
    .option("header", "true")\
    .format("excel")\
    .load("/home/ritorta/HomeWork/W7/Task_2/W7T2.xlsx").limit(1000)\
    .withColumn("interest", sum1(col("Payment of interest")).over(w))\
    .withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

```

```

df120 = spark.read.format("com.crealytics.spark.excel")\
    .option("dataAddress", "120!A1:F135")\
    .option("useHeader", "false")\
    .option("treatEmptyValuesAsNulls", "false")\
    .option("inferSchema", "true").option("addColorColumns", "true")\
    .option("usePlainNumberFormat", "true")\
    .option("startColumn", 0)\
    .option("endColumn", 99)\
    .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
    .option("maxRowsInMemory", 20)\
    .option("excerptSize", 10)\

```

```

.option("header", "true")\
.format("excel")\
.load("/home/ritorta/HomeWork/W7/Task_2/W7T2.xlsx").limit(1000)\
.withColumn("interest", sum1(col("Payment of interest")).over(w))\
.withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

```

```

df150 = spark.read.format("com.crealytics.spark.excel")\
.option("dataAddress", "'150!A1:F93")\
.option("useHeader", "false")\
.option("treatEmptyValuesAsNulls", "false")\
.option("inferSchema", "true").option("addColorColumns", "true")\
.option("usePlainNumberFormat", "true")\
.option("startColumn", 0)\
.option("endColumn", 99)\
.option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
.option("maxRowsInMemory", 20)\
.option("excerptSize", 10)\
.option("header", "true")\
.format("excel")\
.load("/home/ritorta/HomeWork/W7/Task_2/W7T2.xlsx").limit(1000)\
.withColumn("interest", sum1(col("Payment of interest")).over(w))\
.withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

```

```

df250 = spark.read.format("com.crealytics.spark.excel")\
.option("dataAddress", "'250!A1:F47")\
.option("useHeader", "false")\
.option("treatEmptyValuesAsNulls", "false")\
.option("inferSchema", "true").option("addColorColumns", "true")\
.option("usePlainNumberFormat", "true")\
.option("startColumn", 0)\
.option("endColumn", 99)\
.option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
.option("maxRowsInMemory", 20)\
.option("excerptSize", 10)\
.option("header", "true")\
.format("excel")\
.load("/home/ritorta/HomeWork/W7/Task_2/W7T2.xlsx").limit(1000)\
.withColumn("interest", sum1(col("Payment of interest")).over(w))\
.withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

```

```

df300 = spark.read.format("com.crealytics.spark.excel")\
.option("dataAddress", "'300!A1:F38")\
.option("useHeader", "false")\
.option("treatEmptyValuesAsNulls", "false")\
.option("inferSchema", "true").option("addColorColumns", "true")\
.option("usePlainNumberFormat", "true")\
.option("startColumn", 0)\
.option("endColumn", 99)\

```

```

.option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
.option("maxRowsInMemory", 20)\
.option("excerptSize", 10)\
.option("header", "true")\
.format("excel")\
.load("/home/ritorta/HomeWork/W7/Task_2/W7T2.xlsx").limit(1000)\
.withColumn("interest", sum1(col("Payment of interest")).over(w))\
.withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

df_combined = dfG.union(df120).union(df150).union(df250).union(df300)

df_combined.write.format("jdbc").option("url","jdbc:mysql://localhost:33061/spark?user=root&
password=1")\
.option("driver", "com.mysql.cj.jdbc.Driver").option("dbtable", "AiR_W7T2")\
.mode("append").save()

"""df_pandas = df_combined.toPandas()"""

df_pandas1 = dfG.toPandas()
df_pandas2 = df120.toPandas()
df_pandas3 = df150.toPandas()
df_pandas4 = df250.toPandas()
df_pandas5 = df300.toPandas()

ax = plt.gca()
ax.ticklabel_format(style='plain')

df_pandas1.plot(kind='line', x='number', y='debt', color='green', ax=ax, label='Debt Genetal')
df_pandas1.plot(kind='line', x='number', y='interest', color='red', ax=ax, label='Interest
General')
df_pandas2.plot(kind='line', x='number', y='debt', color='grey', ax=ax, label='Debt 120')
df_pandas2.plot(kind='line', x='number', y='interest', color='orange', ax=ax, label='Interest
120')
df_pandas3.plot(kind='line', x='number', y='debt', color='purple', ax=ax, label='Debt 150')
df_pandas3.plot(kind='line', x='number', y='interest', color='yellow', ax=ax, label='Interest
150')
df_pandas4.plot(kind='line', x='number', y='debt', color='blue', ax=ax, label='Debt 250')
df_pandas4.plot(kind='line', x='number', y='interest', color='brown', ax=ax, label='Interest
250')
df_pandas5.plot(kind='line', x='number', y='debt', color='black', ax=ax, label='Debt 300')
df_pandas5.plot(kind='line', x='number', y='interest', color='pink', ax=ax, label='Interest 300')

plt.title('Loan Payments Over Time')
plt.grid ( True )
ax.set(xlabel=None)

plt.savefig("/home/ritorta/HomeWork/W7/Task_2/Loan_Payments_Over_Time.jpeg")
plt.show()

```

```

spark.stop()
t1=time.time()
print('finished',time.strftime('%H:%M:%S',time.gmtime(round(t1-t0))))

```

Unnamed-1\spark\AIR_W7T2 - HeidiSQL 12.6.0.6765

Файл Редактировать Поиск Запрос Инструменты Переход Помощь

Фильтр баз данных Фильтр таблиц

Unamed-1

- Airflow
- information_schema
- mysql
- performance_schema
- spark
 - AIR_W7T2 192,0 KiB
 - Temperature_Tyumen 64,0 KiB
 - ws1_w3t5v2 16,0 KiB
 - ws1_w3t5v2a 16,0 KiB
 - ws1_w3t5v2b 16,0 KiB
 - WSL_W6T1 64,0 KiB
- sys

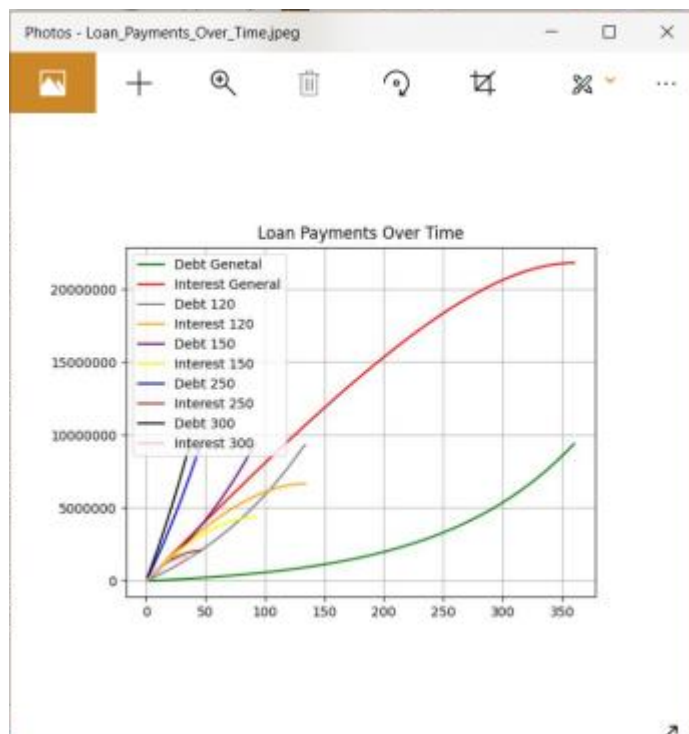
Хост: 127.0.0.1 База данных: spark Таблица: AIR_W7T2 Данные

spark.AIR_W7T2: 669 >> Далее Показать все Сортировка Столбцы (8/8) Фильтр

#	number	Month	Payment amount	Payment of the principal debt	Payment
1	1	2025-01-13	120 000	38 327,9	
2	2	2025-02-13	120 000	35 949,6	
3	3	2025-03-13	120 000	38 973,2	
4	4	2025-04-13	120 000	36 622,2	
5	5	2025-05-13	120 000	36 951	
6	6	2025-06-13	120 000	39 951,1	
7	7	2025-07-13	120 000	37 641,5	
8	8	2025-08-13	120 000	40 625,3	
9	9	2025-09-13	120 000	38 250,4	
10	10	2025-10-13	120 000	38 464,8	
11	11	2025-11-13	120 000	46 668,1	
12	12	2025-12-13	120 000	39 231,2	
13	13	2026-01-13	120 000	42 178,5	
14	14	2026-02-13	120 000	39 964,2	
15	15	2026-03-13	120 000	42 894,1	
16	16	2026-04-13	120 000	40 710,1	

116 SELECT * FROM `spark`.`AIR_W7T2` LIMIT 1000;
 117 SHOW CREATE TABLE `spark`.`AIR_W7T2`;
 118 SELECT * FROM `spark`.`AIR_W7T2` LIMIT 1000;

r1 : c2 Подключено: MariaDB or MySQL 8.0.3 Время работы: 00:14 h Серверное вр Ожидание.



3. Зарегистрируйтесь в OpenWeatherApi (<https://openweathermap.org/api>) 3.1 Создайте ETL, который получает температуру в заданной вами локации, и дальше делает ветвление: - В случае, если температура больше 15 градусов цельсия — идёт на ветку, в которой есть оператор, выводящий на экран «тепло»; - В случае, если температура ниже 15 градусов, идёт на ветку с оператором, который выводит в консоль «холодно». - Оператор ветвления должен выводить в консоль полученную от API температуру. - Приложите скриншот графа и логов работы оператора ветвления.

```
from datetime import datetime
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator, BranchPythonOperator
from datetime import datetime, timedelta
from airflow.operators.python import PythonOperator
from datetime import datetime
import requests
import os
from dotenv import load_dotenv

def openwear_get_temp(**kwargs):

    #API
    dotenv_path = '/home/ritorta/HomeWork/API_KEY.env' # Проверить путь к API
    load_dotenv(dotenv_path)
    openweather_api = os.getenv('OPENWEATHER_API')

    ti = kwargs['ti']
    city = "Tyumen"
    api_key = openweather_api
    url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}"
    payload = {}
    headers = {}
    response = requests.request("GET", url, headers=headers, data=payload)
    return round(float(response.json()['main']['temp'])-273.15, 2)

def openwear_check_temp(ti):
    temp = int(ti.xcom_pull(task_ids='Tyumen_get_temperature'))
    print(f'Temperature now is {temp}')
    if temp >= 15:
        return 'Tyumen_Temp_warm'
    else:
        return 'Tyumen_Temp_cold'

with DAG(
    'Tyumen_check_temperature_warm_or_cold',
    start_date=datetime(2024, 4, 25),
```

```

        catchup=False,
        tags=['W7T3'],
    ) as dag:
        Tyumen_get_temperature = PythonOperator(
            task_id='Tyumen_get_temperature',
            python_callable=openwear_get_temp,
        )

        Tyumen_check_temperature = BranchPythonOperator(
            task_id='Tyumen_check_temperature',
            python_callable=openwear_check_temp,
        )

        Tyumen_Temp_warm = BashOperator(
            task_id='Tyumen_Temp_warm',
            bash_command='echo "It is warm"',
        )

        Tyumen_Temp_cold = BashOperator(
            task_id='Tyumen_Temp_cold',
            bash_command='echo "It is cold"',
        )

    Tyumen_get_temperature >> Tyumen_check_temperature >> [Tyumen_Temp_warm,
    Tyumen_Temp_cold]

```

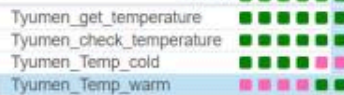

Next Run: 2024-12-02, 00:00:00

≡ Ganitt

 Audit Log

Auto-refresh ☐

deferred failed queued removed restarting running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status



Filter Tasks =

[Details](#) [Graph](#) [Gantt](#) [Code](#) [Logs](#)

[More Details](#)
[Rendered Template](#)
[XCom](#)
[List Instances, all runs](#)

Task Instance Notes:

 Add Note

Task Instance Details

Status ■ success

Task ID Tyumen_Temp_warm 

Run ID manual_2024-12-02T08:28:06.798638+00:00 [🔗](#)

Operator	BashOperator
----------	--------------

Trigger Rule	all_success
--------------	-------------

Duration 00:00:00

Started 2024-12-02, 08:28:12 UTC

Ended 2024-12-02, 08:28:12 UTC

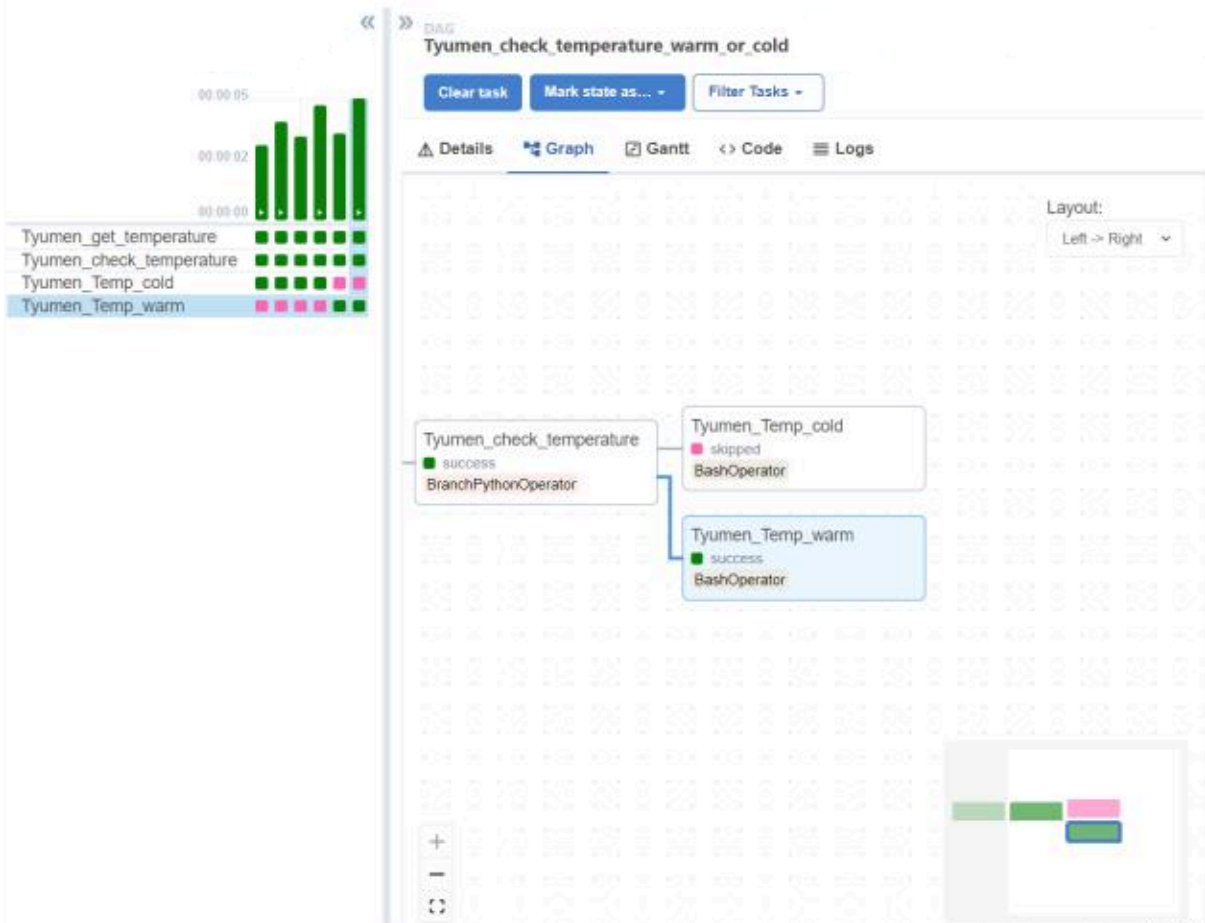
DAG: Tyumen_check_temperature_warm_or_cold Schedule: 1 day, 0:00:00 Next Run: 2024-12-02, 00:00:00

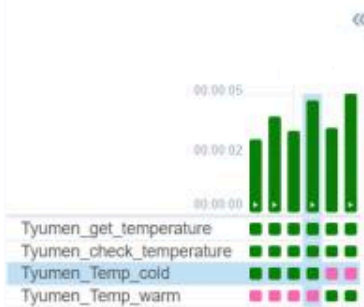
Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

02.12.2024 08:41:20 25 All Run Types All Run States Clear Filters Auto-refresh

Press **SHIFT** + **F** for Shortcuts

deferred failed queued removed restarting running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status





DAG
Tyumen_check_temperature_warm_or_cold

Clear task Mark state as... Filter Tasks

[Details](#) [Graph](#) [Gantt](#) [Code](#) [Logs](#)

[More Details](#)
[Rendered Template](#)
[XCom](#)
[List instances, all runs](#)

Task Instance Notes:

 Add Note

Task Instance Details

Status ■ success

Task ID Tyumen_Temp_cold

Run ID: manual__2024-T2-02T08:26:25.959084+00:00

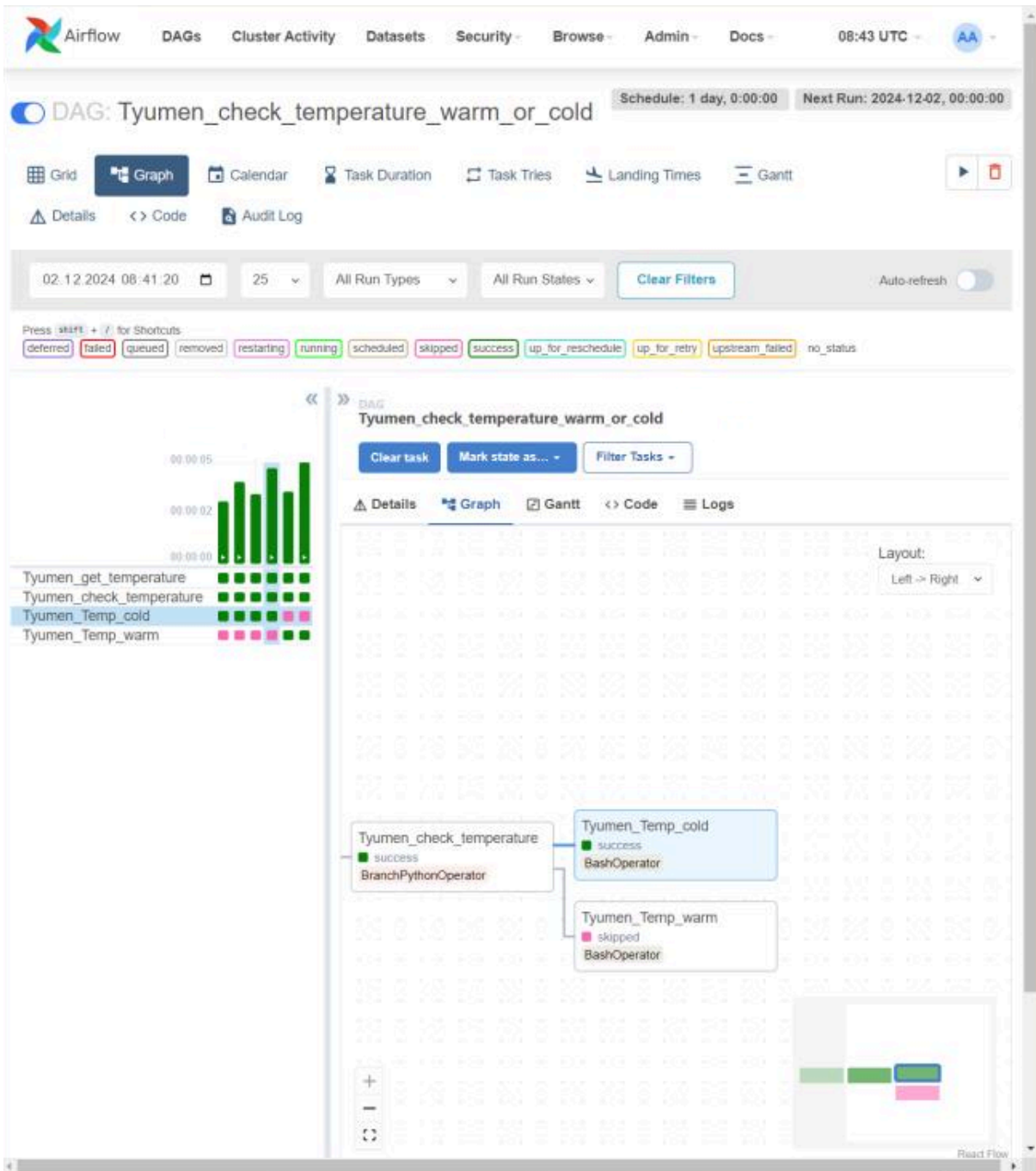
Operator	BashOperator
----------	--------------

Trigger Rule	all_success
--------------	-------------

Duration 00:00:00

Started 2024-12-02, 08:26:30 UTC

Ended 2024-12-26, 08:26:31 UTC



Triggered Work_7_Task_2, it should start any moment now.

Triggered Work_7_Task_1, it should start any moment now.

DAGs

All 4 Active 3 Paused 1 Running 1 Failed 0 Filter DAGs by tag Search DAGs

☐ Auto-refresh

DAG	Owner	Runs	Schedule	Last Run
<input checked="" type="checkbox"/> Tyumen_check_temperature_warm_or_cold WTT3	airflow	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	1 day, 0:00:00	2024-12-02, 08:28:06
<input type="checkbox"/> Work_6_Task_1	Ritorta	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	0 6 ***	2024-11-24, 08:26:19
<input checked="" type="checkbox"/> Work_7_Task_1	Ritorta	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	0 6 ***	2024-12-02, 08:28:30
<input checked="" type="checkbox"/> Work_7_Task_2	Ritorta	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	0 6 ***	2024-12-02, 08:28:30