

Отчёт по второй лабораторной

Для выполнения данной лабораторной необходимо получить API token Для этого я зарегистрировался на Hugging face

The screenshot shows the Hugging Face user interface. At the top, there's a navigation bar with the Hugging Face logo, a search bar, and links to Models, Datasets, Spaces, Docs, Solutions, and Pricing. Below this is a yellow banner with a message about joining an organization. The main content area is divided into a left sidebar and a right main section. The sidebar contains a user profile card for 'blarney-english' with a 'blarney' badge, and a list of menu items: Profile, Account, Organizations, Billing, Access Tokens (highlighted), and SSH and GPG Keys. The main section is titled 'Access Tokens' and contains a 'User Access Tokens' heading. Below this, there's a paragraph explaining that access tokens authenticate identity to the Hugging Face Hub. A table shows one token with the name 'work' and a 'WRITE' permission. The token value is masked with dots. There are 'Manage' and 'Show' buttons for the token. At the bottom of the main section, there is a 'New token' button.

Для работы в меню Connections создадим подключение

The screenshot shows the Airflow web interface. At the top, there's a navigation bar with the Airflow logo and links to DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. The right side of the bar shows the time '08:13 UTC' and a user profile icon. Below the navigation bar is a 'List Connection' section. It features a search bar, a '+ Actions' button, and a 'Record Count: 1' indicator. A table lists the connections with columns: Conn Id, Conn Type, Description, Host, Port, Is Encrypted, and Is Extra Encrypted. There is one connection listed with the name 'fs_default', type 'docker', host 'docker-proxy', port '2375', and both encryption flags set to 'False'. Each row has a checkbox and icons for edit and delete.

Создадим DAG lab2_dag.py

```

airflow > dags > lab2_dag.py
1  from datetime import datetime
2  from airflow import DAG
3  from docker.types import Mount
4  from airflow.providers.docker.operators.docker import DockerOperator
5  from airflow.sensors.filesystem import FileSensor
6
7  default_args = {
8      'owner': 'vlados',
9      'start_date': datetime(2023, 1, 3),
10     'retries': 1,
11 }
12
13 dag = DAG(
14     'audio_to_text_converter',
15     default_args=default_args,
16     description='DAG for extracting audio, transforming to text, summarizing, and saving as PDF',
17     schedule_interval=None,
18 )
19
20 waiting_file = FileSensor(
21     task_id='waiting_file',
22     poke_interval=10, # Interval to check for new files (in seconds)
23     filepath='/opt/airflow/data', # Target folder to monitor
24     fs_conn_id='fs_default',
25     dag=dag,
26 )
27

```

```

audio_extraction = DockerOperator(
    task_id='audio_extraction',
    image='jrottenberg/ffmpeg',
    command='-i /data/video.mp4 -vn -acodec copy /data/audio.aac',
    mounts=[Mount(source='/data', target='/data', type='bind')],
    docker_url="tcp://docker-proxy:2375",
    dag=dag,
)

text_transformation = DockerOperator(
    task_id='text_transformation',
    image='nyurik/alpine-python3-requests',
    command='python /data/text_transformation.py',
    mounts=[Mount(source='/data', target='/data', type='bind')],
    docker_url="tcp://docker-proxy:2375",
    dag=dag,
)

text_summarizing = DockerOperator(
    task_id='text_summarizing',
    image='nyurik/alpine-python3-requests',
    command='python /data/text_summarizing.py',
    mounts=[Mount(source='/data', target='/data', type='bind')],
    docker_url="tcp://docker-proxy:2375",
    dag=dag,
)

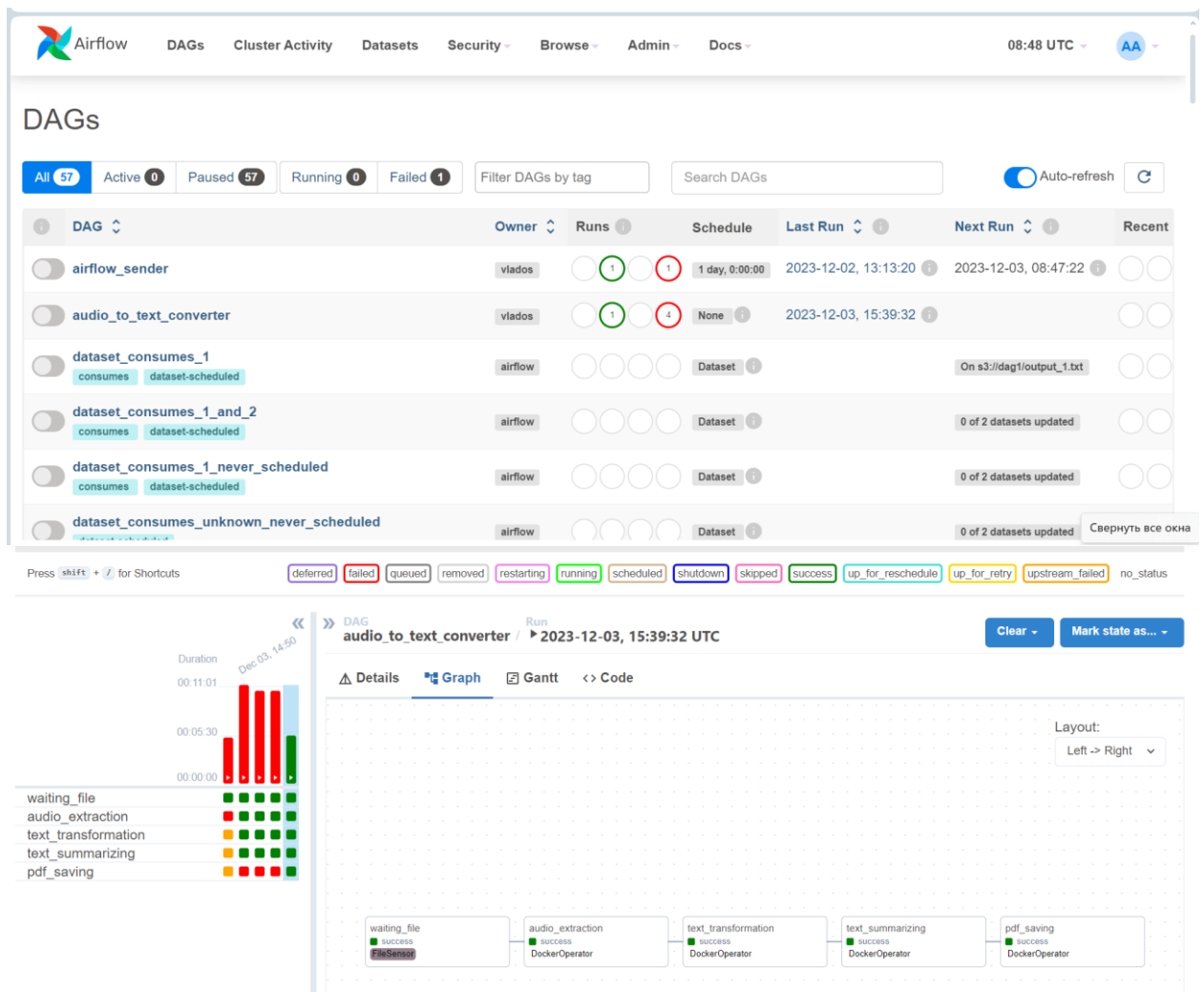
```

```

54
55 pdf_saving = DockerOperator(
56     task_id='pdf_saving',
57     image='blarney/tensorflow_learner:1.0',
58     command='python /data/save_to_pdf.py',
59     mounts=[Mount(source='/data', target='/data', type='bind')],
60     docker_url="tcp://docker-proxy:2375",
61     dag=dag,
62 )
63
64 waiting_file >> audio_extraction >> text_transformation >> text_summarizing >> pdf_saving

```

Теперь заходим в DAG `audio_to_text_converter` (файл `lab2_dag.py` приложен) и запускаем его

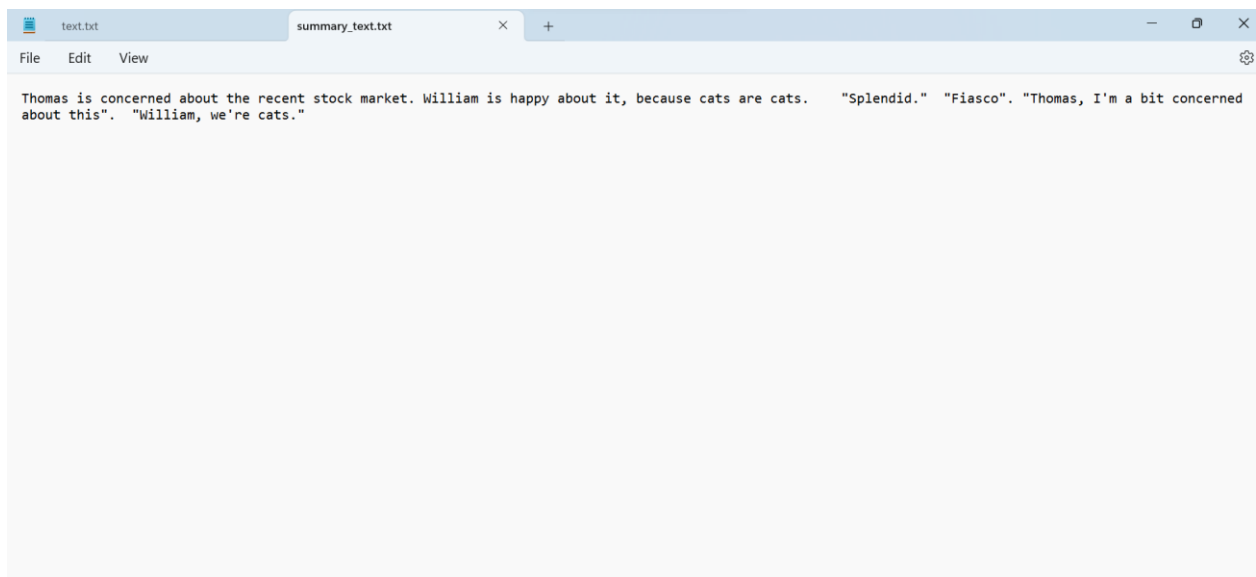
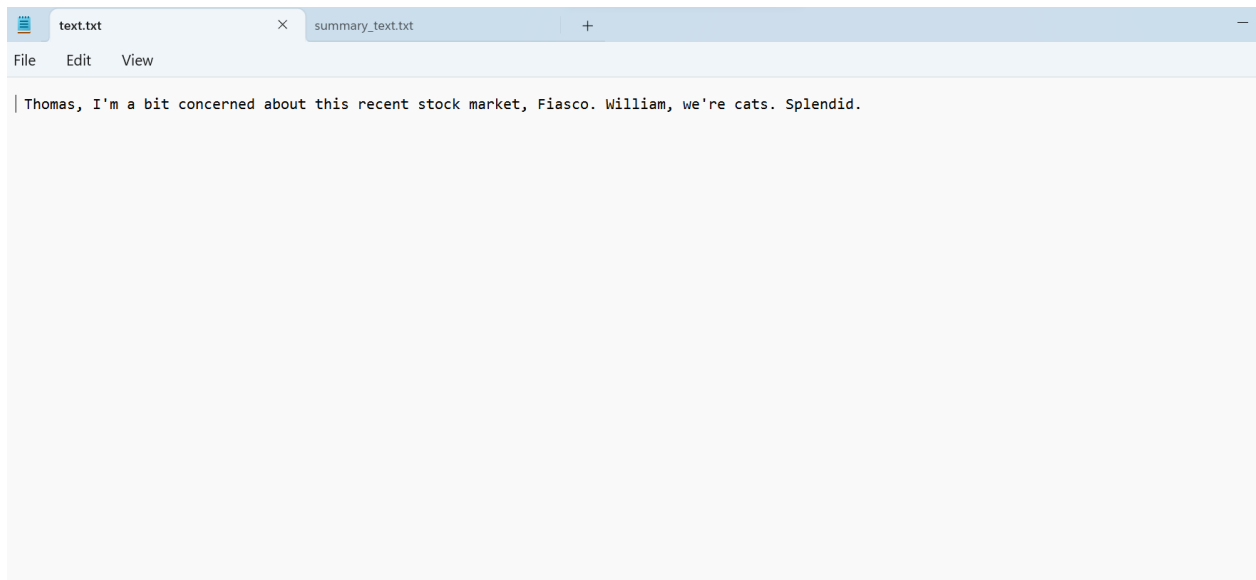


Все нужные файлы создались (во вложении). Генерация произведена на основе этого видео

<https://www.youtube.com/watch?v=vTzeQ2HXVXw>

Имя	Дата изменения	Тип	Размер
 audio.aac	03.12.2023 19:44	AAC Audio File (VLC)	242 КБ
 csv_logger.py	03.12.2023 16:56	Исходный файл Pyth...	0 КБ
 log.csv	04.12.2023 11:44	Файл Microsoft Excel...	15 КБ
 mnist_training.py	04.12.2023 11:43	Исходный файл Pyth...	2 КБ
 model.h5	04.12.2023 11:43	Файл "H5"	17 КБ
 result.pdf	03.12.2023 19:44	Документ Adobe Acr...	2 КБ
 save_to_pdf.py	03.12.2023 19:37	Исходный файл Pyth...	1 КБ
 summary_text.txt	03.12.2023 19:44	Text Document	1 КБ
 text.txt	03.12.2023 19:44	Text Document	1 КБ
 text_summarizing.py	03.12.2023 19:37	Исходный файл Pyth...	1 КБ
 text_transformation.py	03.12.2023 17:20	Исходный файл Pyth...	1 КБ
 video.mp4	03.12.2023 12:36	MP4 Video File (VLC)	420 КБ

Посмотрим, что получилось



Для использования библиотеки `fpdf` (позволяет создать pdf файл), а также библиотек я создал образ Docker и загрузил необходимые библиотеки. Этот же образ используется при обучении нейросети

```
C: > docker > Dockerfile
1  #Deriving the latest base image
2  FROM tensorflow/tensorflow:latest
3
4  RUN pip install scikit-learn numpy pandas fpdf
5
6  #Labels as key value pair
7  LABEL Maintainer="vlados.tensorflow_learner"
8
9
10 # Any working directory can be chosen as per choice like '/' or '/home' etc
11 # i have chosen /usr/app/src
12 WORKDIR /usr/app/src
```

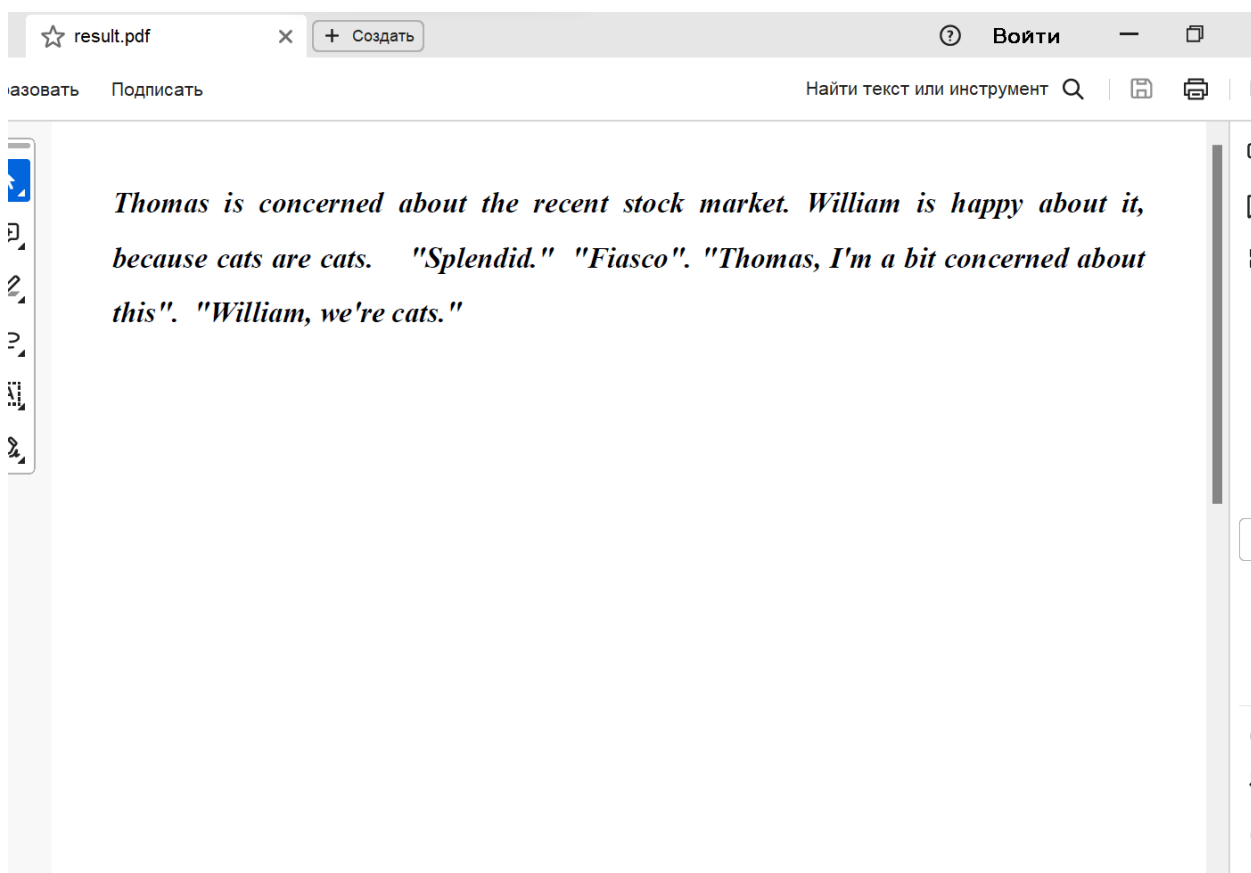
Теперь отправим образ в докерхаб

```
Start a build
PS C:\docker> docker build . -t my_container
[+] Building 299.4s (8/8) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 363B                                0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [internal] load metadata for docker.io/tensorflow/tensorflow:latest 2.7s
=> [auth] tensorflow/tensorflow:pull token for registry-1.docker.io 0.0s
=> [1/3] FROM docker.io/tensorflow/tensorflow:latest@sha256:4689c724a7d65a7d289cc2ae536fa3cd6b636b2df3e23da3a4 193.4s
=> => resolve docker.io/tensorflow/tensorflow:latest@sha256:4689c724a7d65a7d289cc2ae536fa3cd6b636b2df3e23da3a44f 0.0s
=> => sha256:4689c724a7d65a7d289cc2ae536fa3cd6b636b2df3e23da3a44ffdd6ac3af46f 2.83kB / 2.83kB 0.0s
=> => sha256:6a8c4ad355bebd8ef0bea19569618f584ccb4a8adf5bed962e3db9aa4c0010d1 4.54kB / 4.54kB 0.0s
=> => sha256:345716e6b3b788c2ad1d84667e57458de4bfe33d247000c7b55cbb90c1a971d 861B / 861B 0.9s
=> => sha256:5cc77b1d99a1e092e17bf48f5ead9028d5ac88e76729a10f02f74cb72a82d870 769B / 769B 1.3s
=> => sha256:aec849d3972efa43b64ee3c3ba659c0f787f8f59c82fb3e48c87cbb22a12e 29.54MB / 29.54MB 54.4s
=> => sha256:5cd1e9a0284bf5a49731ecf8d7436d199fef8adc18745d487843caa92d93e9b1 163B / 163B 1.5s
=> => sha256:f190ee1fa1d7baf03b64c1a00e4417777c2ba26c3478a1fdf46049b8415f919a 37.63MB / 37.63MB 54.0s
=> => sha256:909fb360ac91957ba7f421486cec96a01a9d732795d9c754f9d8cd8bc4e792370 133.67MB / 133.67MB 117.0s
=> => sha256:bc9c616aad4fac3c21d24d39c503ae697731ba183e0bef0ac576ff529fc1baec 970B / 970B 54.6s
=> => sha256:77cb63a520e5d19dd7e7e1a3b29b35aca75db372592dd4b14d144ed9e6ff5273 128B / 128B 54.8s
=> => extracting sha256:345716e6b3b788c2ad1d84667e57458de4bfe33d247000c7b55cbb90c1a971d 0.0s
=> => extracting sha256:5cc77b1d99a1e092e17bf48f5ead9028d5ac88e76729a10f02f74cb72a82d870 0.0s
=> => extracting sha256:5cd1e9a0284bf5a49731ecf8d7436d199fef8adc18745d487843caa92d93e9b1 0.0s
=> => extracting sha256:f190ee1fa1d7baf03b64c1a00e4417777c2ba26c3478a1fdf46049b8415f919a 0.6s
=> => sha256:f2e46a0a82df72a17e81b6c585aae77a18510b5e96186888c75b54ef277bf82 1.13kB / 1.13kB 87.3s
=> => sha256:831122755b02c47a8a94d5a9ad0da7717c9cbfe7f8a5230d24f764f7d35e9925 1.13kB / 1.13kB 88.0s
=> => extracting sha256:909fb360ac91957ba7f421486cec96a01a9d732795d9c754f9d8cd8bc4e792370 6.5s
=> => extracting sha256:bc9c616aad4fac3c21d24d39c503ae697731ba183e0bef0ac576ff529fc1baec 0.0s
=> => extracting sha256:77cb63a520e5d19dd7e7e1a3b29b35aca75db372592dd4b14d144ed9e6ff5273 0.0s
=> => extracting sha256:3268aa633e510fe5595e2faade355f5ca532d32987ed8087a6e1ffcb92b98f9b 2.3s
=> => extracting sha256:c6cbc212b10ddd3e38925320760e6bbe7fb3047ddae8ce92fb893dde185e95e1 16.6s
=> => extracting sha256:f2e46a0a82df72a17e81b6c585aae77a18510b5e96186888c75b54ef277bf82 0.0s
=> => extracting sha256:831122755b02c47a8a94d5a9ad0da7717c9cbfe7f8a5230d24f764f7d35e9925 0.0s
=> [2/3] RUN pip install scikit-learn numpy pandas fpdf          101.1s
=> [3/3] WORKDIR /usr/app/src                                     0.0s
=> exporting to image                                             2.0s
=> => exporting layers                                             2.0s
=> => writing image sha256:e678fd59c718222517c8b8a4e037708fd21952497aade61a53ec9cc9c6d53b70 0.0s
=> => naming to docker.io/library/my_container                     0.0s
```

Пушим в докерхаб

```
PS C:\docker> docker push blarney/tensorflow_learner:1.0
The push refers to repository [docker.io/blarney/tensorflow_learner]
ac04e3abc1f7: Pushed
b4e26ff48ccf: Pushed
75acb1242fe3: Mounted from tensorflow/tensorflow
1d7a2a211a6b: Mounted from tensorflow/tensorflow
2db699de670e: Mounted from tensorflow/tensorflow
0cf31f98a4b6: Mounted from tensorflow/tensorflow
f663f4c9c5b6: Mounted from tensorflow/tensorflow
104e4c35057a: Mounted from tensorflow/tensorflow
eb864c00a034: Mounted from tensorflow/tensorflow
94235a128255: Mounted from tensorflow/tensorflow
0ac81db158f3: Mounted from tensorflow/tensorflow
8e8c3d39273b: Mounted from tensorflow/tensorflow
f99aba8580cb: Mounted from tensorflow/tensorflow
256d88da4185: Mounted from tensorflow/tensorflow
1.0: digest: sha256:8bf49649451c2fb64f85ac69022dc743911268463f851c92c4cc64056ce32111 size: 3250
```

Итоговый результат работы DAG представлен ниже



Как видно, DAG отработал **превосходно**

Теперь перейдём к созданию DAG для обучения нейросети lab2_dag2.py

```
airflow > dags > lab2_dag2.py
1  from datetime import datetime
2  from airflow import DAG
3  from airflow.providers.docker.operators.docker import DockerOperator
4  from airflow.sensors.filesystem import FileSensor
5  from docker.types import Mount
6
7  default_args = {
8      'owner': 'vlados',
9      'start_date': datetime(2023, 12, 3),
10 }
11
12 dag = DAG(
13     'mnist_learning',
14     default_args=default_args,
15     schedule_interval=None,
16 )
17
18 load_data_train_model = DockerOperator(
19     task_id='load_data_train_model',
20     image='blarney/tensorflow_learner:1.0',
21     command='python /data/mnist_training.py',
22     mounts=[Mount(source='/data', target='/data', type='bind')],
23     docker_url="tcp://docker-proxy:2375",
24     dag=dag,
25 )
26
27 load_data_train_model
```

Модель mnist_training.py

```
airflow > data > mnist_training.py
1  from sklearn.datasets import load_wine
2  from sklearn.model_selection import train_test_split
3  import tensorflow
4  from keras import models
5  from keras import layers
6  from tensorflow.keras.utils import to_categorical
7
8  ⚡
9  wine = load_wine(as_frame=True)
10 wine.frame.head()
11 X = wine.data
12 y = wine.target
13 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4,random_state=123)
14
15 train_labels = to_categorical(y_train)
16 test_labels = to_categorical(y_test)
17 def declare_model_for_learning():
18     model = models.Sequential()
19     model.add(layers.Dense(12, activation='relu', input_shape=(13,)))
20     model.add(layers.Dense(12, activation='relu'))
21     model.add(layers.Dense(3, activation='softmax'))
22     return model
23
24 my_model = declare_model_for_learning()
25 my_model.compile(optimizer=tensorflow.keras.optimizers.SGD(learning_rate=0.05), loss='categorical_crossentropy')
26 my_model.save_weights('/data/model.h5')
27
28 my_model.load_weights('/data/model.h5')
29 csv_logger = [tensorflow.keras.callbacks.CSVLogger('/data/log.csv', append=True, separator=';')]
30
```

Заходим в Dag mnist_learning

DAGs

All57

Active0

Paused57

Running0

Failed1

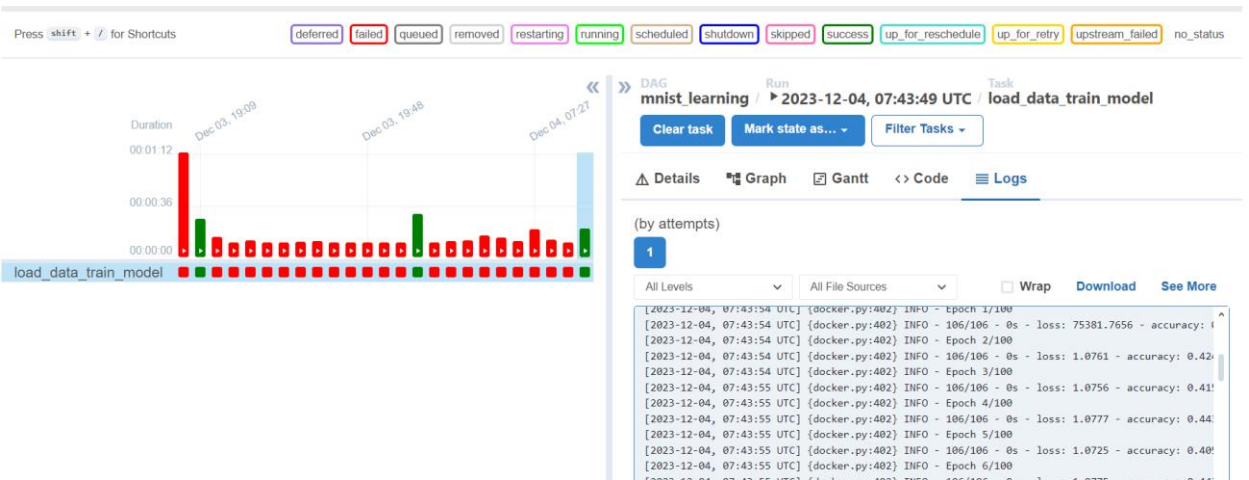
Filter DAGs by tag

Search DAGs

Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent
<div><div></div>airflow_sender</div>	vlados	<div><div>1</div><div>1</div></div> <div>1 day, 0:00:00</div>	2023-12-02, 13:13:20	2023-12-03, 08:48:55		
<div><div></div>mnist_learning</div>	vlados	<div><div>3</div><div>30</div></div> <div>None</div>	2023-12-04, 07:43:49			
<div><div></div>audio_to_text_converter</div>	vlados	<div><div>1</div><div>4</div></div> <div>None</div>	2023-12-03, 15:39:32			
<div><div><div>dataset_consumes_1_and_2</div><div>consumesdataset-scheduled</div></div></div>	airflow	<div><div></div><div></div><div></div></div> <div>Dataset</div>		0 of 2 datasets updated		
<div><div><div>dataset_consumes_1_never_scheduled</div><div>consumesdataset-scheduled</div></div></div>	airflow	<div><div></div><div></div><div></div></div> <div>Dataset</div>		0 of 2 datasets updated		
<div><div><div>dataset_consumes_unknown_never_scheduled</div><div>dataset-scheduled</div></div></div>	airflow	<div><div></div><div></div><div></div></div> <div>Dataset</div>		0 of 2 datasets updated		

Как видно, работает



Логи также сохранились

PREREQUISITES	airflow > data > log.csv
__pycache__	
airflow_dag.py	
lab2_dag.py	
lab2_dag2.py	
data	
audio.aac	
csv_logger.py	
log.csv	
mnist_training.py	
model.h5	
result.pdf	
save_to_pdf.py	
summary_text.txt	
text_summarizing.py	
text_transformation.py	
text.txt	
video.mp4	
logs	
plugins	
.gitkeep	
docker-compose.yaml	
Dockerfile	
images	
mlflow	
OUTLINE	
TIMELINE	

```
1 epoch;accuracy;loss;val_accuracy;val_loss
2 0;0.40566039085388184;1641.1126708984375;NA;NA
3 1;0.4245283007621765;1.074064016342163;NA;NA
4 2;0.4150943458080292;1.0743898153305054;NA;NA
5 3;0.43396225571632385;1.074536919593811;NA;NA
6 4;0.4433962404727936;1.0756022930145264;NA;NA
7 5;0.4245283007621765;1.0749510526657104;NA;NA
8 6;0.4433962404727936;1.0749015808105469;NA;NA
9 7;0.4433962404727936;1.0774645805358887;NA;NA
10 8;0.4433962404727936;1.0767619609832764;NA;NA
11 9;0.4433962404727936;1.0778062343597412;NA;NA
12 10;0.4433962404727936;1.0764371156692505;NA;NA
13 11;0.43396225571632385;1.0745817422866821;NA;NA
14 12;0.4433962404727936;1.078050971031189;NA;NA
15 13;0.4433962404727936;1.0777451992034912;NA;NA
16 14;0.40566039085388184;1.0726640224456787;NA;NA
17 15;0.4433962404727936;1.0771993398666382;NA;NA
18 16;0.4433962404727936;1.0744603872299194;NA;NA
19 17;0.4433962404727936;1.0776898860931396;NA;NA
20 18;0.4433962404727936;1.0779529809951782;NA;NA
21 19;0.4245283007621765;1.075922507476807;NA;NA
22 20;0.4433962404727936;1.0758016109466553;NA;NA
23 21;0.4433962404727936;1.078128695487976;NA;NA
24 22;0.43396225571632385;1.075610637664795;NA;NA
25 23;0.4433962404727936;1.0748225450515747;NA;NA
26 24;0.40566039085388184;1.0705697536468506;NA;NA
27 25;0.4433962404727936;1.0755130052566528;NA;NA
28 26;0.4433962404727936;1.0765016078948975;NA;NA
29 27;0.4433962404727936;1.0753997564315796;NA;NA
```

Отлично! Всё работает!