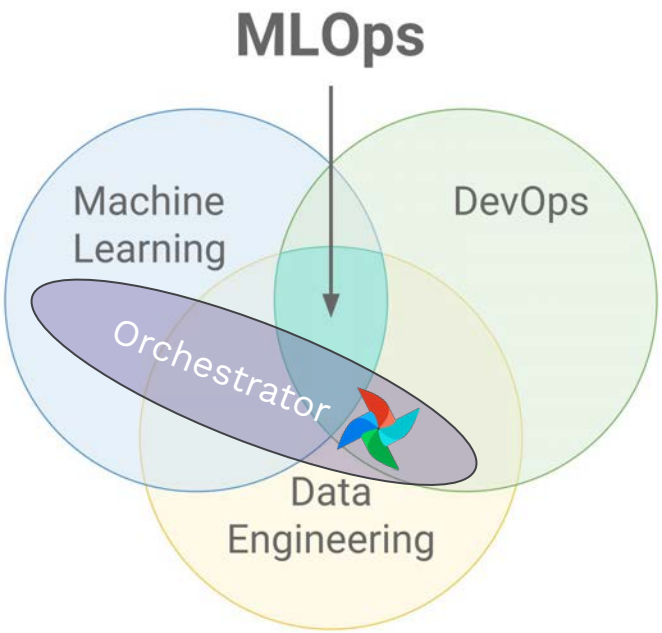
Orchestrating data and ML workflows with

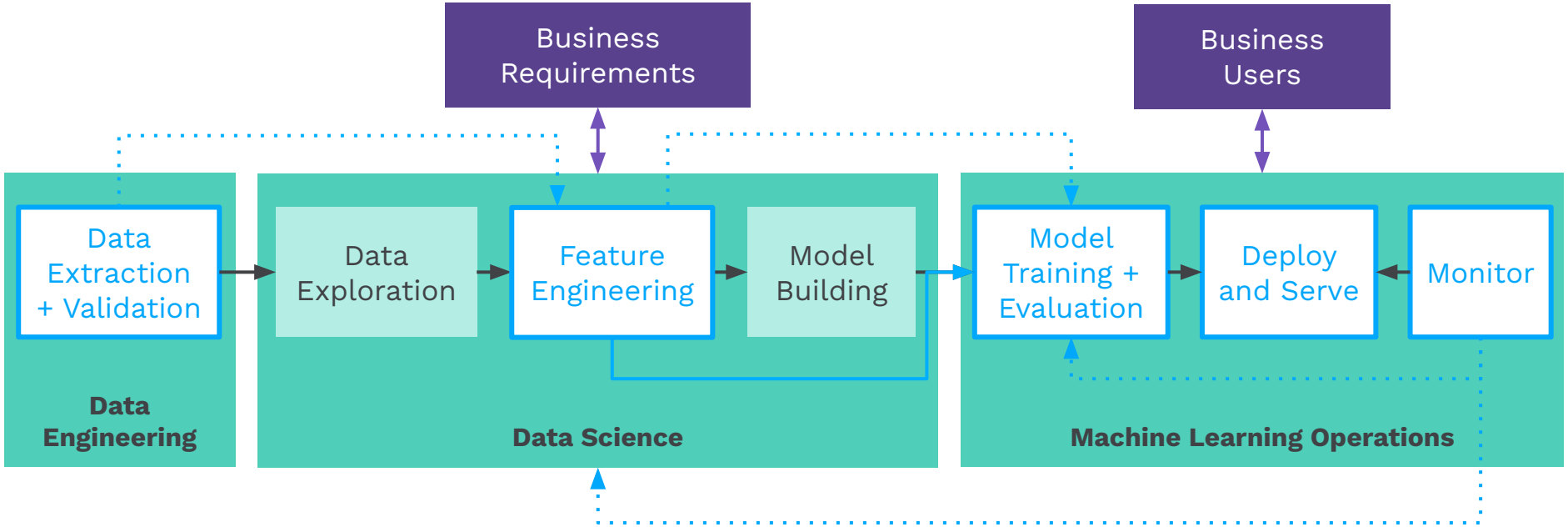
Overview

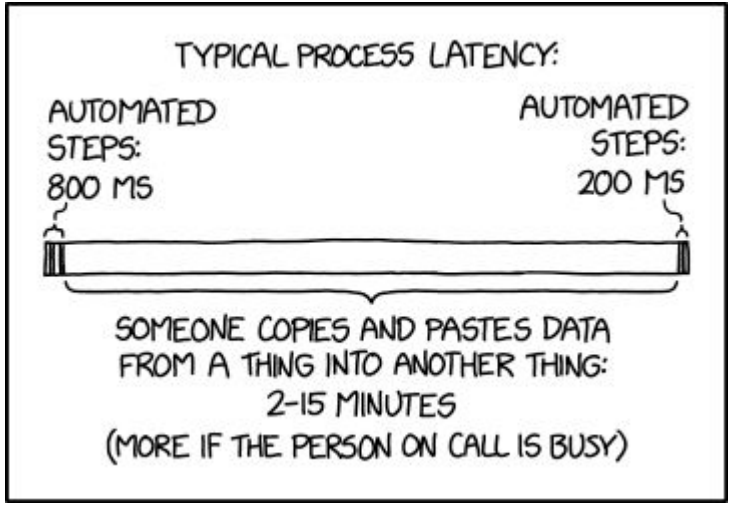
- What is ML orchestration?
- Airflow Crash Course
- The Data
- A data-driven ML pipeline in Airflow
  - Schedule on events using Datasets
  - Asynchronously wait for new data in S3
  - Dynamic tasks = dynamic pipelines
  - Astro SDK to easily interact with object and database storage
  - Custom HuggingFace Operators
  - Slack Alerts
- Short demo
- The Results
- What is next?
- ML & Airflow Resources

ML Orchestration ∈ [ML Ops]



Automatable Components





https://xkcd.com/2565/

Airflow Crash Course

What Is Apache Airflow?

Apache Airflow is an open source tool for **programmatically** authoring, scheduling and monitoring your data pipelines (or any of your Python code).

With over **12M downloads per month**, Airflow is the most popular open source choice for workflow orchestration around the world.

Some of the benefits of using Airflow in production include:

Pipelines as code, written in Python

Infinitely scalable + highly extensible

Large, vibrant OSS community

Airflow UI

Airflow | DAGs | Datasets | Security | Browse | Admin | Docs | 19:58 UTC | AU

DAGs

All 9 | Active 4 | Paused 5 | Filter DAGs by tag | Search DAGs | Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
data_quality_example_dag	airflow	8, 9	1 day, 0:00:00	2022-08-18, 10:46:37	2022-08-27, 19:58:32	1, 1, 1		
dynamic_test	airflow	14, 15	1 day, 0:00:00	2022-08-19, 13:56:04	2022-08-27, 19:58:32	33, 4		
example-dag-complex	airflow	7	1 day, 0:00:00	2022-08-28, 19:53:40	2022-08-28, 19:52:52	22		
load_connections_dag	airflow	1	1 day, 0:00:00	2022-08-17, 19:55:18	2022-08-27, 19:58:04	1		
mapping_xcoms_dag	airflow	2, 11	1 day, 0:00:00	2022-08-27, 19:44:42	2022-08-28, 19:44:42	4, 12		
multiple_parameters_example	airflow	4, 9	1 day, 0:00:00	2022-08-27, 19:44:42	2022-08-28, 19:44:42	2, 15		
snowflake_to_slack_dag	airflow	11, 3	1 day, 0:00:00	2022-08-18, 10:39:30	2022-08-27, 19:58:04	2		
taskflow_example	airflow		None					
zipping_examples	airflow	7, 24	1 day, 0:00:00	2022-08-27, 19:44:42	2022-08-28, 19:44:42	3, 6		

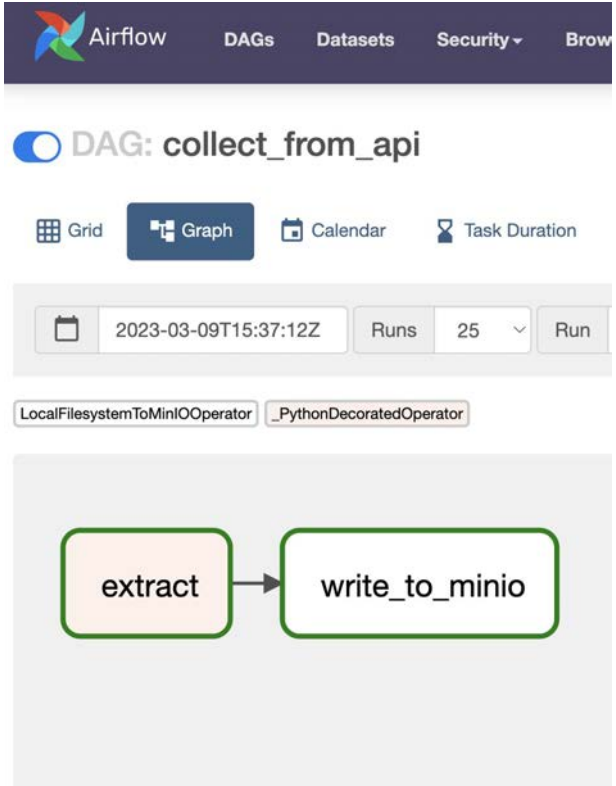
« ‹ 1 › »

Showing 1-9 of 9 DAGs

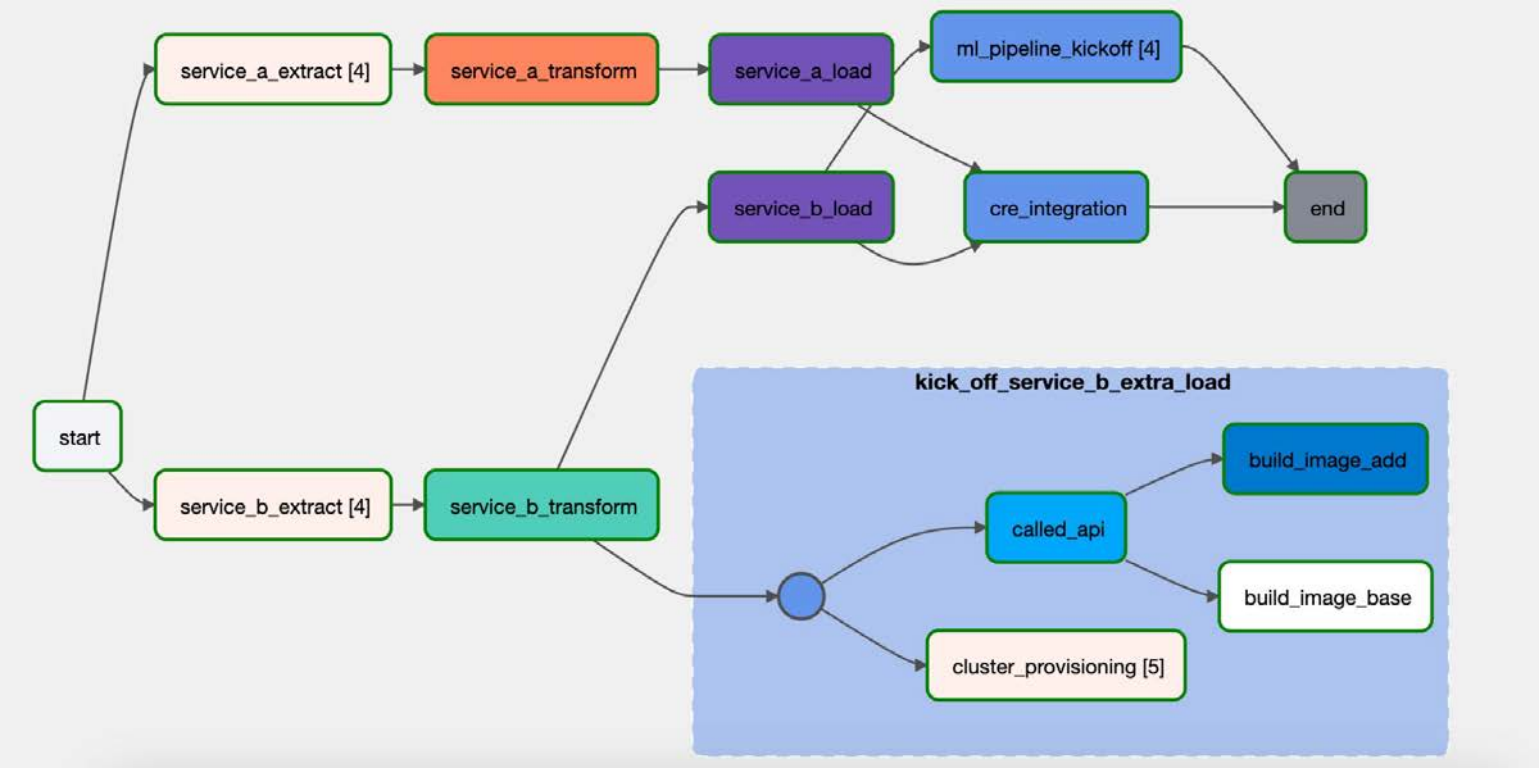
DAGs - tasks - operators

dags > collect_from_api.py > ...

```python
from pendulum import datetime
from airflow.decorators import dag, task
from include.minio import LocalFilesystemToMinIOOperator
import requests
import os


@dag(
    schedule="@daily",
    start_date=datetime(2023, 1, 1),
    catchup=False,
)
def collect_from_api():
    @task
    def extract():
        r = requests.get(os.environ["MY_API"])
        return r.json()

    write_to_minio = LocalFilesystemToMinIOOperator(
        task_id="write_to_minio",
        json_serializeable_information=extract(),
        bucket_name="extract",
        object_name="{{ logical_date }}_api_response.json",
    )


collect_from_api()

```



DAGs complex as you want



Why Airflow?

- Orchestrate data pipelines and ML pipelines in the **same place** with complex dependencies
- **Tool agnostic**, plug and play the newest models
- **Event-driven** scheduling with Datasets and deferrable sensors

- It is all just Python code, run highly customized scripts next to plug and play operators
- Extensible and scalable
- Run tasks in dedicated Kubernetes pods with the @task.kubernetes decorator

The Data

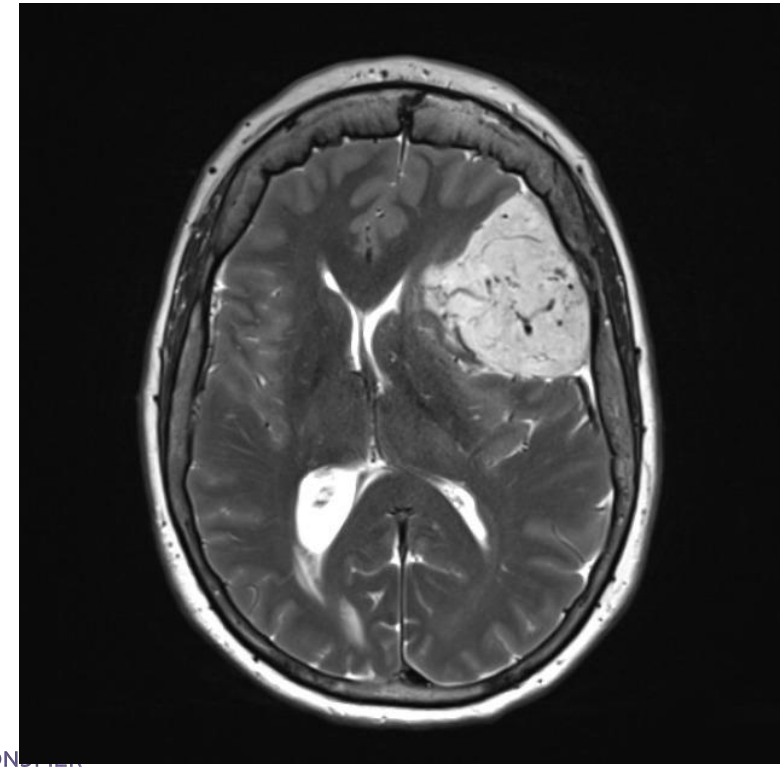
The Data

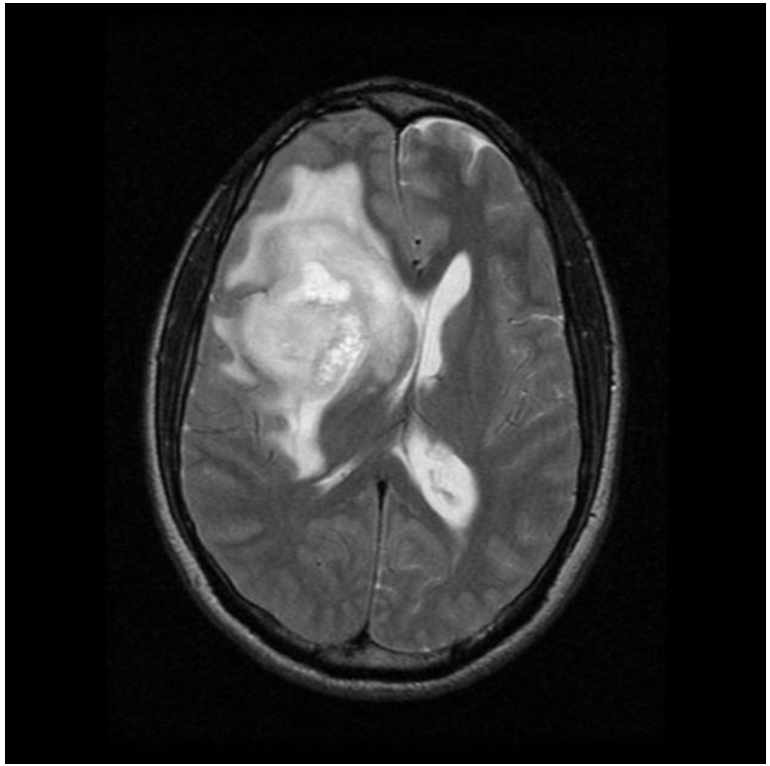
- Glioma: a brain tumor coming from glial cells.
- Meningioma: a brain tumor coming from arachnoidal cap cells.

- 346 T2 weighted images of gliomas
- 329 T2 weighted images of meningiomas
- Train-test-split: 1/4 test
- Test set: 86 gliomas, 82 meningiomas
- Train set: 260 gliomas, 247 meningiomas

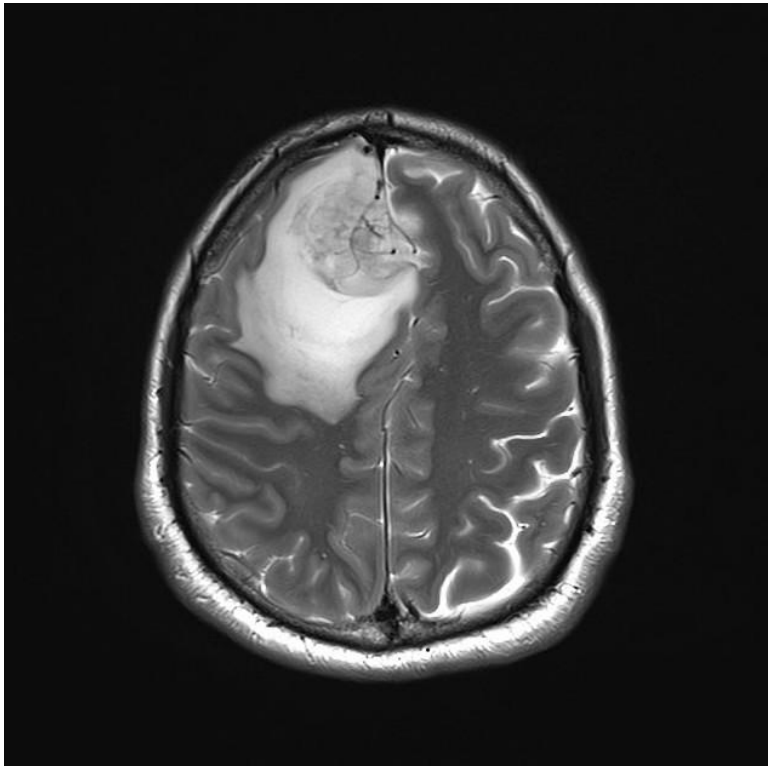
Source: https://www.kaggle.com/datasets/fernando2rad/brain-tumor-mri-images-17-classes

Sometimes it is (relatively) easy





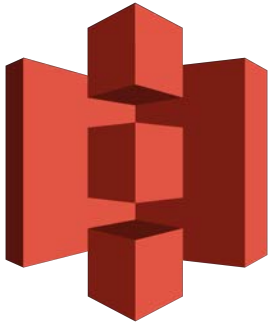
Sometimes it is harder



The pipeline

https://github.com/TJaniF/airflow-ml-pipeline-image-classification

The tools



microsoft/resnet-50

+ Astro Python SDK!

8 DAGs

Airflow | DAGs | Datasets | Security | Browse | Admin | Docs | Astronomer | 08:05 UTC | AU

DAGs

All 8 | Active 8 | Paused 0

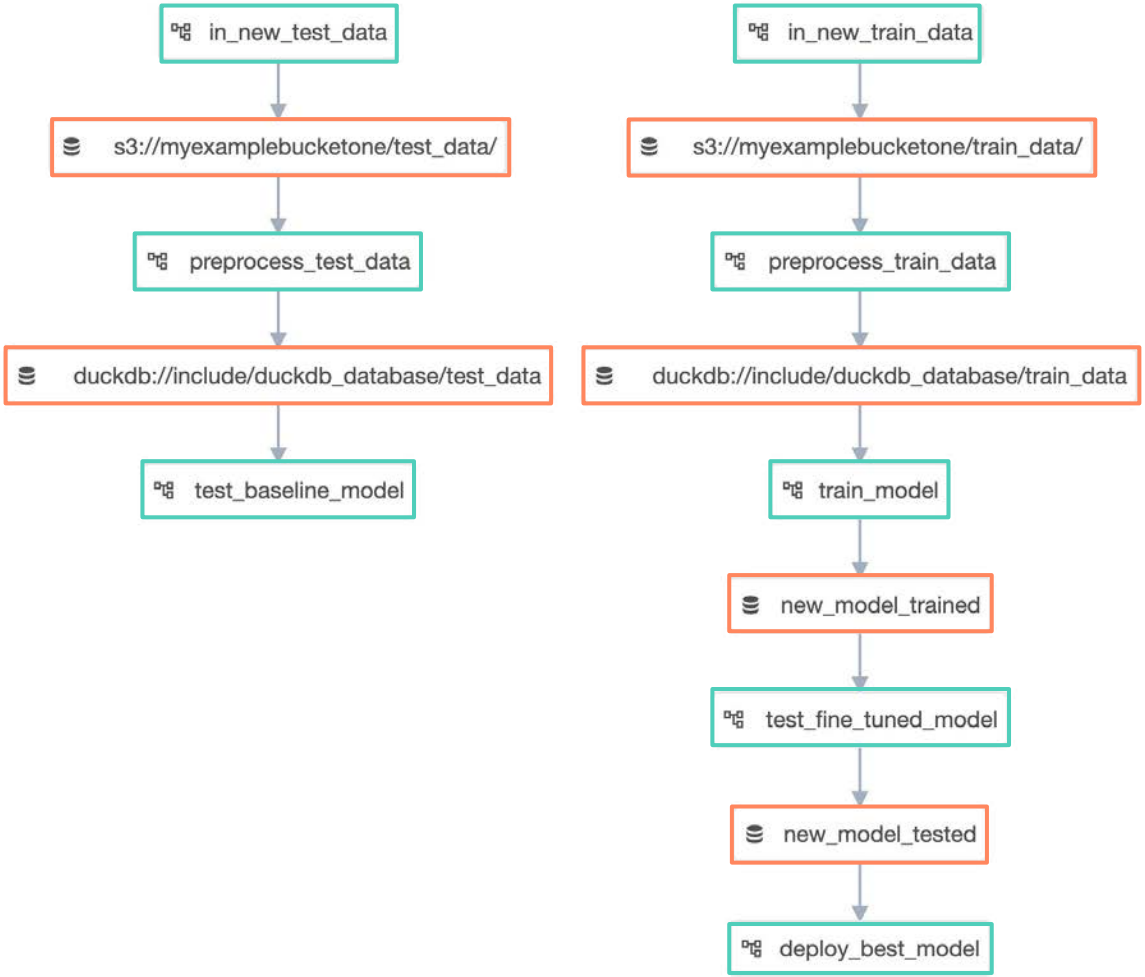
Filter DAGs by tag | Search DAGs | Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
deploy_best_model	airflow	6	Dataset	2023-05-11, 17:06:09	On new_model_tested	5		
in_new_test_data	airflow	1, 1	@continuous	2023-05-11, 15:57:32	2023-05-11, 15:57:32	4, 1, 1		
in_new_train_data	airflow	2, 1	@continuous	2023-05-11, 16:16:36	2023-05-11, 16:16:36	4, 1, 1		
preprocess_test_data	airflow	1	Dataset	2023-05-11, 16:12:52	On s3://myexamplebucketone/test_data_ng/	324		
preprocess_train_data	airflow	2	Dataset	2023-05-11, 18:46:38	On s3://myexamplebucketone/train_data_ng/	961		
test_baseline_model	airflow	1	Dataset	2023-05-11, 16:24:06	On duckdb://include/duckdb_database/test_da...	10		
test_fine_tuned_model	airflow	1	Dataset	2023-05-11, 16:53:11	On new_model_trained	9		
train_model	airflow	2	Dataset	2023-05-11, 19:04:41	On duckdb://include/duckdb_database/train_d...	7		

« ‹ 1 › »

Showing 1-8 of 8 DAGs

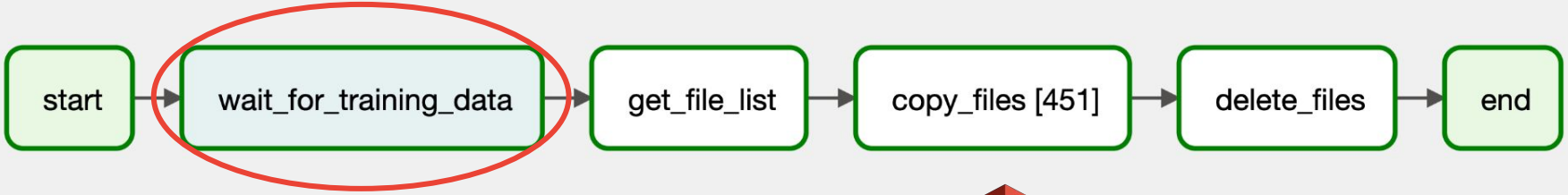
8 DAGs
6 Datasets

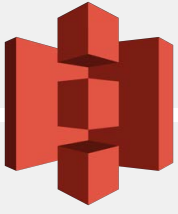


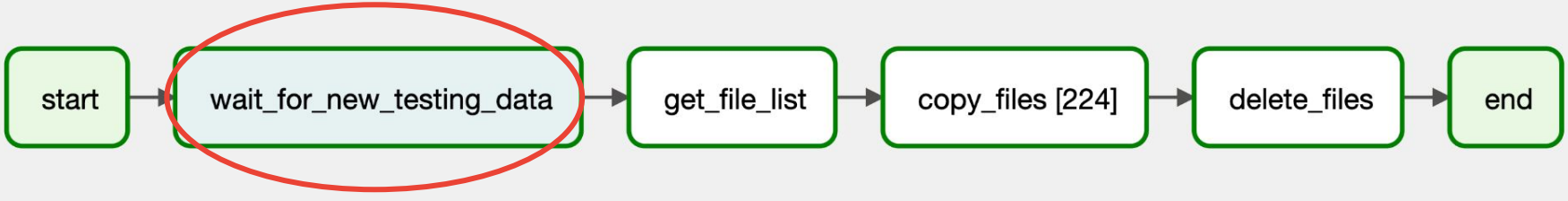
@continuous

```
@dag(
    start_date=datetime(2023, 1, 1),
    schedule="@continuous",
    max_active_runs=1,
    catchup=False,
)
def in_new_train_data():
```

Two DAGs waiting for new train/test data



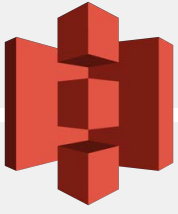




Deferrable operators can save resources!

```
47    # asynchronously wait for a file to drop in S3
48    wait_for_new_training_data = S3KeySensorAsync(
49        task_id="wait_for_training_data",
50        bucket_key=f"{S3_IN_TRAIN_FOLDER_NAME}/*{IMAGE_FORMAT}",
51        bucket_name=S3_BUCKET_NAME,
52        wildcard_match=True,
53        aws_conn_id=AWS_CONN_ID,
54        poke_interval=POKE_INTERVAL,
55    )
56
```

Two DAGs waiting for new train/test data



Dynamic tasks

```
63    # create dynamic arguments from the list of files
64    def map_file_list_to_src_dest_key(astro_file_object):
65        file_path = astro_file_object.path
66        source_key = file_path
67        dest_key = (
68            f"s3://{S3_BUCKET_NAME}/{S3_TRAIN_FOLDER_NAME}/" + file_path.split("/")[-1]
69        )
70        return {"source_bucket_key": source_key, "dest_bucket_key": dest_key}
71
72    # copy files from one location in S3 to another, dynamically mapped for one task per file
73    copy_files = S3CopyObjectOperator.partial(
74        task_id="copy_files",
75        aws_conn_id=AWS_CONN_ID,
76    ).expand_kwargs(in_file_list.map(map_file_list_to_src_dest_key))
77
```

Dynamic tasks

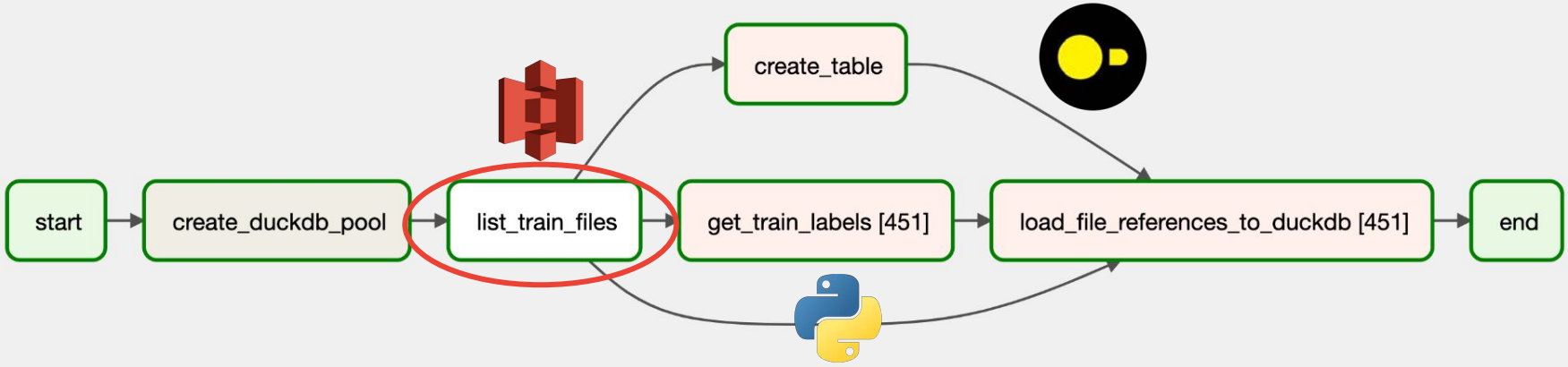
DAG in_new_train_data / Run 2023-05-05, 10:14:40 UTC / Task copy_files []

Clear task | Mark state as... | Filter Tasks

Details | Graph | [] Mapped Tasks

MAP INDEX	STATE	DURATION	START DATE	END DATE
0	success	00:00:02	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
1	success	00:00:01	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
2	success	00:00:02	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
3	success	00:00:01	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
4	success	00:00:01	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
5	success	00:00:01	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
6	success	00:00:02	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
7	success	00:00:02	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
8	success	00:00:01	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
9	success	00:00:01	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
10	success	00:00:01	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC
11	success	00:00:02	2023-05-05, 10:14:51 UTC	2023-05-05, 10:14:53 UTC

2 DAGs handling preprocessing



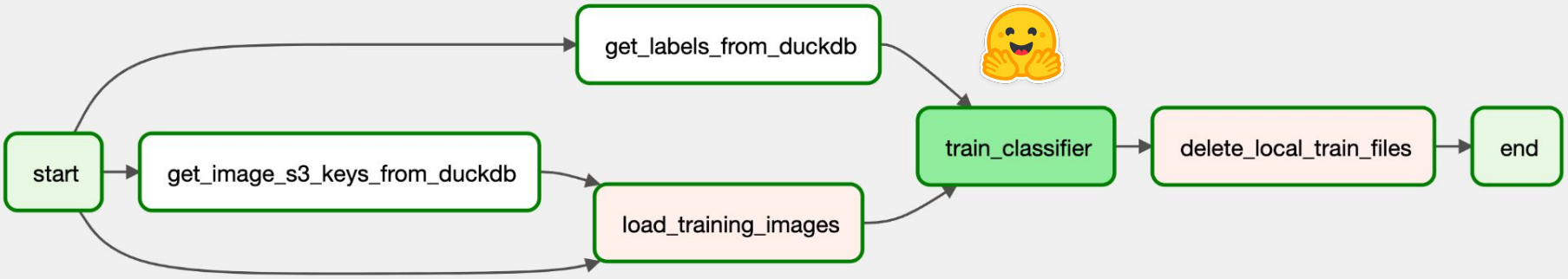
Astro SDK - Part 1

Next generation DAG authoring, if you switch to another blob storage, just change the path and connection ID!

```
53    # list the files in the S3 bucket
54    list_test_files = get_file_list(
55        task_id="list_test_files",
56        path=f"s3://{S3_BUCKET_NAME}/{S3_TEST_FOLDER_NAME}",
57        conn_id=AWS_CONN_ID,
58    )
59
```

https://astro-sdk-python.readthedocs.io

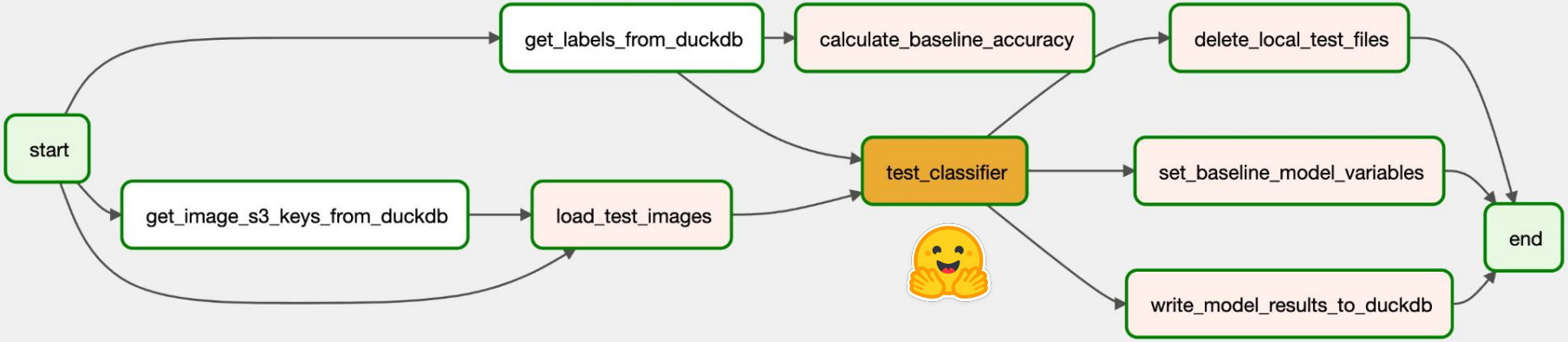
Train the model



Wrapping Model Fine-tuning into a custom operator

```
# fine tune resnet
train_classifier = FineTuneHuggingFaceBinaryImageClassifierOperator(
    task_id="train_classifier",
    model_name=BASE_MODEL_NAME,
    local_images_filepaths=local_images_filepaths,
    labels=get_labels_from_duckdb.map(lambda x: x[0]),
    learning_rate=0.0023,
    model_save_dir=FINE_TUNED_MODEL_PATHS + "/{{ ts }}/",
    train_transform_function=standard_transform_function,
    batch_size=32,
    num_epochs=10,
    shuffle=True,
    outlets=[AirflowDataset("new_model_trained")],
)
```

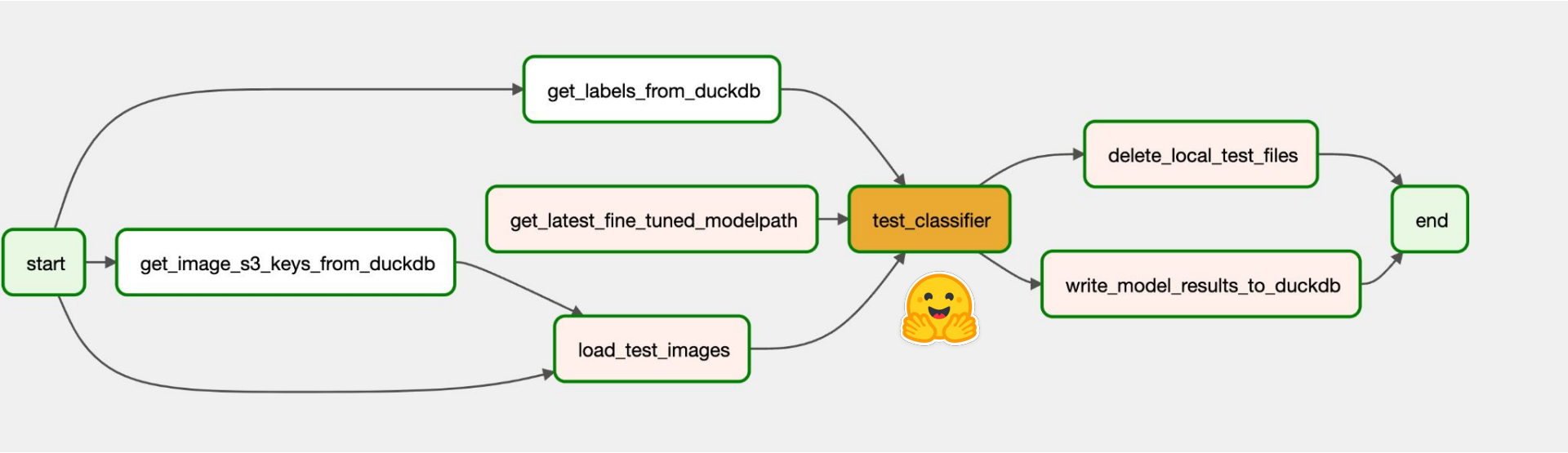
Get a baseline



Wrapping model testing into a custom operator

```
test_classifier = TestHuggingFaceBinaryImageClassifierOperator(
    task_id="test_classifier",
    model_name=BASE_MODEL_NAME,
    local_images_filepaths=local_images_filepaths,
    labels=get_labels_from_duckdb.map(lambda x: x[0]),
    test_transform_function=standard_transform_function,
    batch_size=500,
    shuffle=False,
)
```

Test fine-tuned model



Airflow Notifiers

```
108     # test the fine-tuned model
109     test_classifier = TestHuggingFaceBinaryImageClassifierOperator(
110         task_id="test_classifier",
111         model_name=get_latest_fine_tuned_model(
112             fine_tuned_models_folder=FINE_TUNED_MODEL_PATHS
113         ),
114         local_images_filepaths=local_images_filepaths,
115         labels=get_labels_from_duckdb.map(lambda x: x[0]),
116         test_transform_function=standard_transform_function,
117         batch_size=500,
118         shuffle=False,
119         # if this task is successful send a slack notification
120         on_success_callback=SlackNotifier(
121             slack_conn_id=SLACK_CONNECTION_ID,
122             text=SLACK_MESSAGE,
123             channel=SLACK_CHANNEL,
124         ),
125         outlets=[AirflowDataset("new_model_tested")],
126     )
```

Customized Slack alerts

Pipeline Alerts APP 7:06 PM

🎉 Model Test Successful 🎉

The test_classifier task finished testing the model: include/fine_tuned_models/2023-05-11T16:34:19.374235+00:00/!

Fine-tuned model results:
Average test loss: 0.0
Accuracy: 0.37735849056603776
Precision: 0.19047619047619047
Recall: 0.046511627906976744
F1-Score: 0.07476635514018691
AUC: 0.4068174577891048

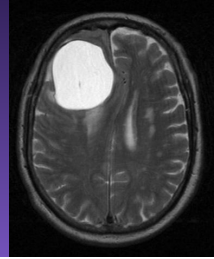
Comparison:
Base Rate Accuracy of the test set: 0.5408805031446541
Pre-fine-tuning average test loss: 0.0
Pre-fine-tuning test accuracy: 0.4528301886792453

Deploy the best model - Astro SDK Part 2

```python
# pick the best model from the duckdb records for the latest test set
@aql.transform(pool=DUCKDB_POOL_NAME)
def pick_best_model(in_table):
    return """SELECT model_name
            FROM {{ in_table }}
            WHERE test_set_num = (SELECT MAX(test_set_num) FROM {{ in_table }})
            ORDER BY auc DESC
            LIMIT 1;"""

@aql.dataframe
def deploy_model(df: pd.DataFrame):
    print(df["model_name"])

model_deploy = deploy_model(
    df=pick_best_model(
        in_table=Table(conn_id=DB_CONN_ID, name=RESULTS_TABLE_NAME),
    )
)
```

Short demo

The Results



microsoft/resnet-50 **before** fine-tuning:

- Accuracy: 0.482
- F1 Score: 0.360
- AUC: 0.472

microsoft/resnet-50 **after** fine-tuning:

- Accuracy: 0.845
- F1 Score: 0.845
- AUC: 0.844

And that was only 30 epochs...

What is next?

- Dynamic task mapping over custom ML operators for hyper parameter tuning
- In production: The Kubernetes Executor and KubernetesPodOperator can run heavy tasks in dedicated K8s pods with different resource requirements.
- At scale: consider using a relational database with the option of parallel writing

- You'd like to see more HuggingFaceOperators? Create your own and share! https://apache-airflow-slack.herokuapp.com/

Airflow ❤️ ML - Resources

Airflow Quickstart

- MLFlow Airflow provider (GH repository)
- Airflow and Weights and Biases Demo (GH repository)
- How to Orchestrate Machine Learning Workflows with Airflow (webinar)
- 8 Things I Wish I Knew About Airflow Before I Started Orchestrating Machine Learning Workflows (blog post)
- Airflow and Sagemaker (tutorial)

Tell us what you integrations and content you want to see!

Take Home Messages

- Was this clinically useful? No.
- Can the pipeline be easily adjusted for other use cases? Yes!

Airflow is a central place that can orchestrate both data and ML pipelines, fully tool-agnostic!

Thank you.