

WORKSHOP DAY ONSITE
OCTOBER 4, 2023

CONFERENCE DAY ONSITE
OCTOBER 5, 2023

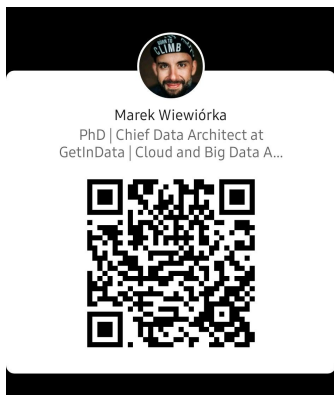
WORKSHOP DAY ONLINE
OCTOBER 6, 2023



**The MLOps platform that makes you
productive, everywhere!**

Marek Wiewiórka
GetInData | Part of Xebia

About me



- Chief Data Architect @**GetInData** | **Part of Xebia**
- Research Assistant at the Warsaw University of Technology
- An Open source contributor to [Snowflake Terraform Provider](#), [SeQuiLa](#) and [Kedro](#) plugins
- Personally a keen long distance runner and gravel bikes enthusiast

GetInData - At a Glance

- Experts in **Big Data, Cloud, Analytics and ML/AI solutions**
- Team of 120+ consultants, **~60% senior level**
- Experience in: **media, e-commerce, retail, fintech, banking, and telco**
- We work with **digital natives where data is core business** (Spotify, Truecaller, Acast, Volt), as well as with traditional enterprises where data is used for improvements
- **A go-to partner** for companies that need tailored and highly scalable data processing and analytics platforms that give competitive advantage and **unlock the full business potential of data.**

SOLUTION AREAS



MLOps & Modern Data Platforms



Data & ML engineering project accelerators

Read: [Kedro plugins](#), [DP Framework](#)



Stream processing & real-time analytics

Technologies



Selected USE CASES

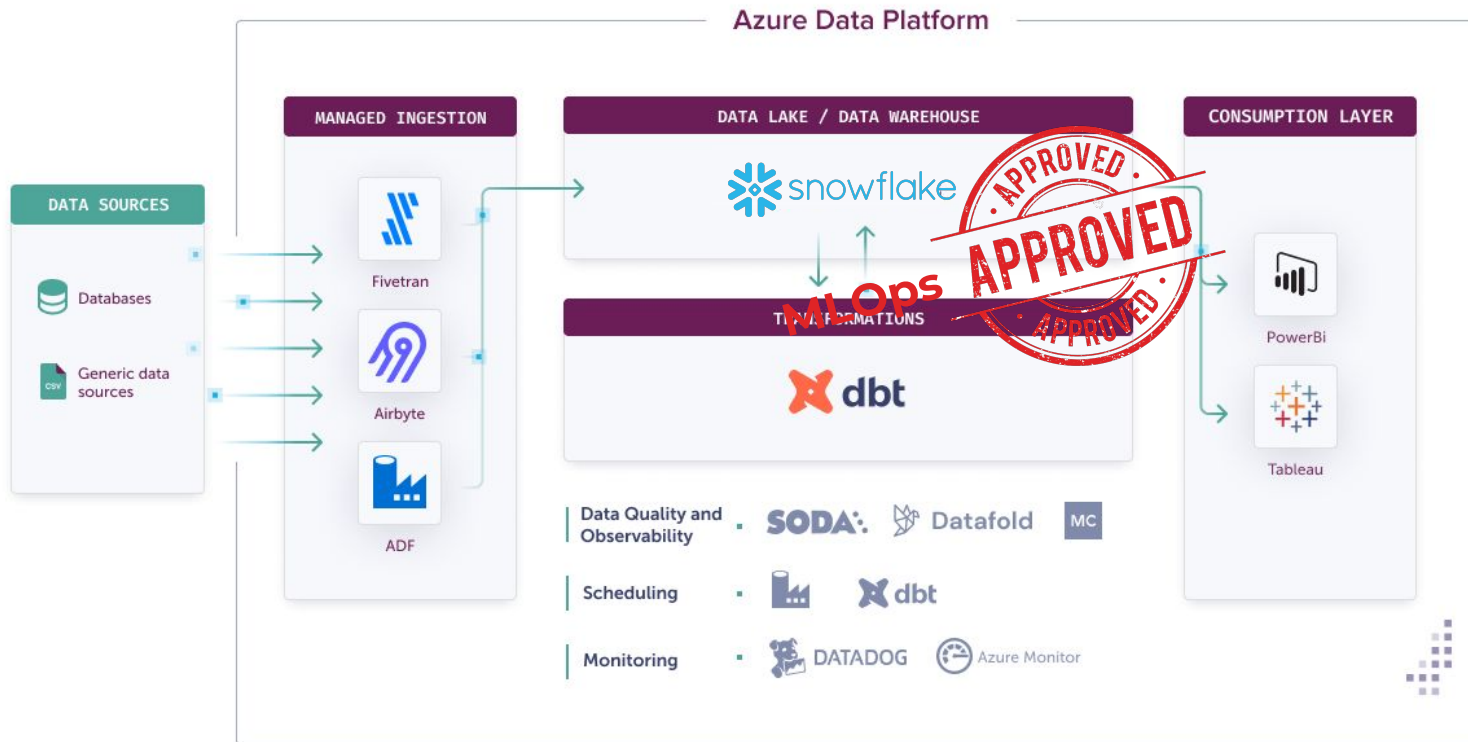
1. Volt.io (Fintech)

- **Snowflake-based Modern Data Platform**
- Just **4 months** to build from scratch to insights
- Strong focus on platform security
- The right mix of open-source and cloud-managed technologies

2. Other customer references



From MDP¹ to MDP (MLOps-enabled Data Platform)

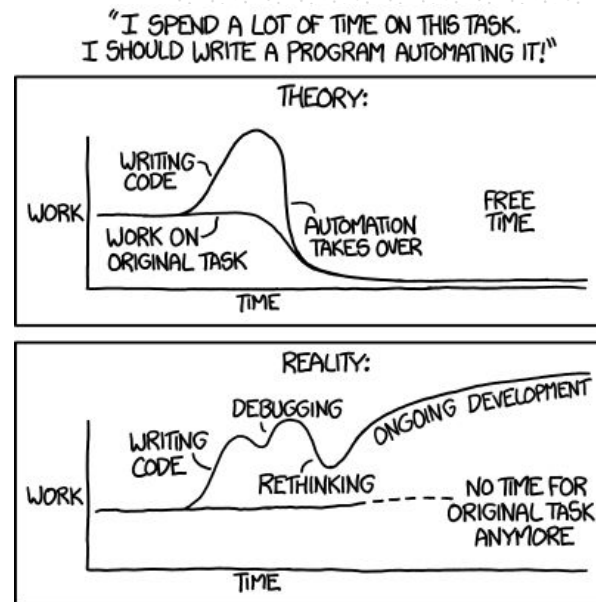


¹MDP - Modern Data Platform

What MLOps is (*not only*) about ?

- Application of the DevOps principles to ML world
- Managing ML model lifecycle
- Tools and platforms
- *Automation* and processes
- Infrastructure as Code

The ultimate goal is: **PRODUCTIVITY**



Source: xkcd by Randall Munroe. Automation takes a life of its own.

GID MLOps “Productivity Manifesto”

- Machine Learning and data science should be **first-class** citizens of Data Platforms
- Lightweight and KISS
- **Open** standards and cloud **agnosticism**
- Short development **feedback loop** (incl. local dev)
- **Fast** new ML projects bootstrapping and **standardization**
- Execution environment **independent** training pipelines
- Easy **collaboration**
- ... MLOps capabilities provisioned **in days not months**

ML projects in layers



**Data
Scientist**

Experimentation + EDA

Machine Learning frameworks

Example technologies:



ML projects in layers



Data
Scientist

Experimentation + EDA

Machine Learning frameworks

?

Execution environment

Data



MLOps / ML
Engineer

Example technologies:



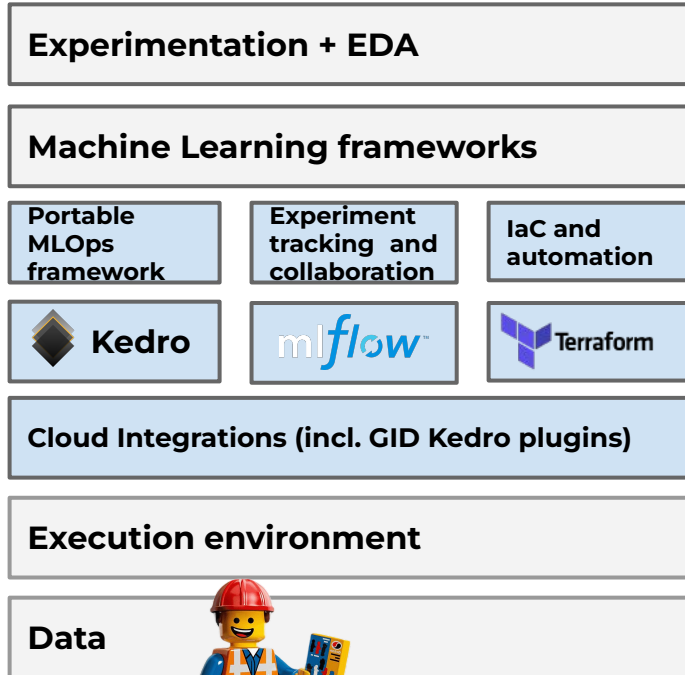
XGBoost



Building blocks of the GID MLOps



Data Scientist



MLOps / ML Engineer

Example technologies:



GID MLOps Platform



Building blocks of the GID MLOps



Data Scientist

Experimentation + EDA

Machine Learning frameworks

Portable MLOps framework

Experiment tracking and collaboration

IaC and automation



Kedro

mlflow



Terraform

Cloud Integrations (incl. GID Kedro plugins)

Execution environment

Data



MLOps / ML Engineer

Example technologies:



XGBoost

GID MLOps Platform



What is Kedro?



Kedro

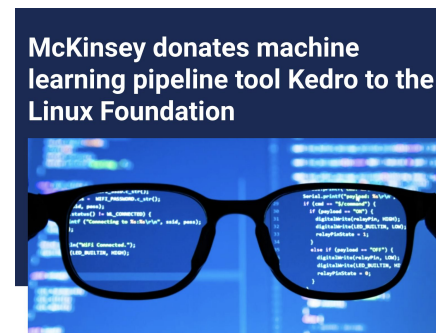
=

Software
Engineering
Principles

+

Data Science

***Kedro** is an open-source Python framework for creating reproducible, maintainable and modular data science code.*



What features does Kedro have?

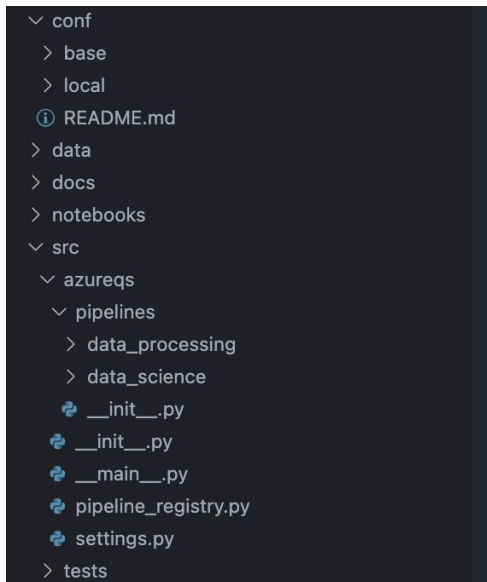
```

  ✓ conf
    > base
    > local
    ⓘ README.md
  > data
  > docs
  > notebooks
  ✓ src
    ✓ azureqs
    ✓ pipelines
      > data_processing
      > data_science
      📄 __init__.py
      📄 __init__.py
      📄 __main__.py
      📄 pipeline_registry.py
      📄 settings.py
    > tests

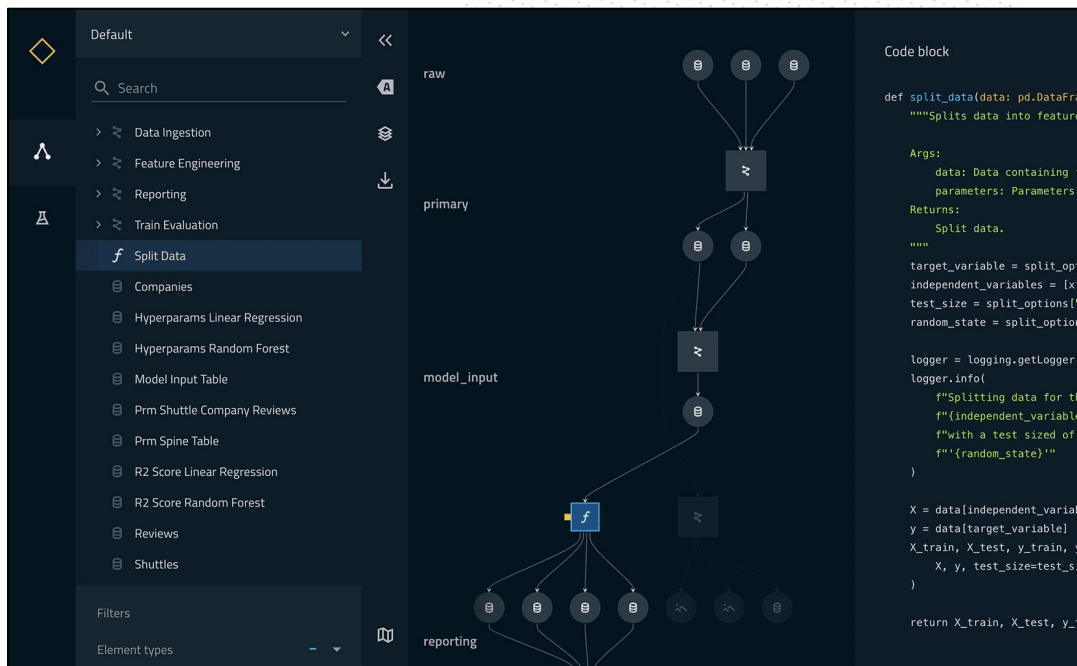
```

**Well defined
project structure
+ project starters**

What features does Kedro have?



Well defined
project structure
+ project starters



Nodes & pipelines
abstractions

Kedro pipeline - data science & engineering

```
6 def create_pipeline(**kwargs) -> Pipeline:
7     return pipeline(
8         [
9             node(
10                func=preprocess_companies,
11                inputs="companies",
12                outputs="preprocessed_companies",
13                name="preprocess_companies_node",
14            ),
15            node(
16                func=preprocess_reviews,
17                inputs="reviews",
18                outputs="preprocessed_reviews",
19                name="preprocess_reviews_node",
20            ),
21            node(
22                func=create_model_input_table,
23                inputs=["preprocessed_reviews", "preprocessed_companies"],
24                outputs="model_input_table",
25                name="create_model_input_table_node",
26            ),
27        ]
28    )
```

```
6 def create_pipeline(**kwargs) -> Pipeline:
7     return pipeline(
8         [
9             node(
10                func=split_data,
11                inputs=["model_input_table", "params:model_options"],
12                outputs=["X_train", "X_test", "y_train", "y_test"],
13                name="split_data_node",
14            ),
15            node(
16                func=train_model,
17                inputs=["X_train", "y_train"],
18                outputs="regressor",
19                name="train_model_node",
20            ),
21            node(
22                func=evaluate_model,
23                inputs=["regressor", "X_test", "y_test"],
24                outputs=None,
25                name="evaluate_model_node",
26            ),
27        ]
28    )
29
```


Kedro node

```
6 def create_pipeline(**kwargs) -> Pipeline:
7     return pipeline(
8         [
9             node(
10                func=preprocess_companies,
11                inputs="companies",
12                outputs="preprocessed_companies",
13                name="preprocess_companies_node",
14            ),
15            node(
16                func=preprocess_reviews,
17                inputs="reviews",
18                outputs="preprocessed_reviews",
19                name="preprocess_reviews_node",
20            ),
21            node(
22                func=create_model_input_table,
23                inputs=["preprocessed_reviews", "
24                outputs="model_input_table",
25                name="create_model_input_table_node",
26            ),
27        ]
28    )
```

```
49 def create_model_input_table(
50     reviews: pd.DataFrame, companies: pd.DataFrame, ratings: pd.DataFrame
51 ) -> pd.DataFrame:
52     """Combines all data to create a model input table.
53
54     Args:
55         reviews: Preprocessed data for reviews.
56         companies: Preprocessed data for companies.
57         ratings: Raw data for ratings.
58
59     Returns:
60         Model input table.
61
62     """
63     reviews_with_ratings = reviews.merge(ratings, left_on="id", right_on="rating_id")
64     model_input_table = reviews_with_ratings.merge(
65         companies, left_on="company_id", right_on="id"
66     )
67     model_input_table = model_input_table.dropna()
68     return model_input_table
```

What about parameters?

```
6 def create_pipeline(**kwargs) -> Pipeline:
7     return pipeline(
8         [
9             node(
10                func=split_data,
11                inputs=["model_input_table", "params:model_options"],
12                outputs=["X_train", "X_test", "y_train", "y_test"],
13                name="split_data_node",
14            ),
15            node(
16                func=train_model,
17                inputs=["X_train", "y_train"],
18                outputs="regressor",
19                name="train_model_node",
20            ),
21            node(
22                func=evaluate_model,
23                inputs=["regressor", "X_test", "y_test"],
24                outputs=None,
25                name="evaluate_model_node",
26            ),
27        ]
28    )
29
```

What about parameters?

```
6 def create_pipeline(**kwargs) -> Pipeline:
```

```
7     return pipeline(  
8         [  
9             node(  
10                func=split_data,  
11                inputs=["model_input_table", "  
12                outputs=["X_train", "X_test", "  
13                name="split_data_node",  
14            ),  
15            node(  
16                func=train_model,  
17                inputs=["X_train", "y_train"],  
18                outputs="regressor",  
19                name="train_model_node",  
20            ),  
21            node(  
22                func=evaluate_model,  
23                inputs=["regressor", "X_test", "  
24                outputs=None,  
25                name="evaluate_model_node",  
26            ),  
27        ]  
28    )  
29
```

```
  ✓ conf
```

```
  ✓ base
```

```
  ✓ parameters
```

```
    ! data_processing.yml
```

```
    ! data_science.yml
```

```
    ! azureml.yml
```

```
    ! catalog.yml
```

```
    ! logging.yml
```

```
    ! parameters.yml
```

```
  > local
```

```
  > data
```

```
  > docs
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

```
10
```

```
11
```

```
12
```

```
13
```

```
14
```

```
15
```

```
model_options:
```

```
  test_size: 0.2
```

```
  random_state: 3
```

```
features:
```

```
  - engines
```

```
  - passenger_capacity
```

```
  - crew
```

```
  - d_check_complete
```

```
  - moon_clearance_complete
```

```
  - iata_approved
```

```
  - company_rating
```

```
  - review_scores_rating
```

What about data - Kedro data catalog!

```
6 def create_pipeline(**kwargs) -> Pipeline:
7     return pipeline(
8         [
9             node(
10                func=preprocess_companies,
11                inputs="companies",
12                outputs="preprocessed_companies",
13                name="preprocess_companies_node",
14            ),
15            node(
16                func=preprocess_reviews,
17                inputs="reviews",
18                outputs="preprocessed_reviews",
19                name="preprocess_reviews_node",
20            ),
21            node(
22                func=create_model_input_table,
23                inputs=["preprocessed_reviews", "p
24                outputs="model_input_table",
25                name="create_model_input_table_nod
26            ),
27        ]
28    )
```

```
42 companies:
43     type: pandas.CSVDataSet
44     filepath: data/01_raw/companies.csv
45
46 reviews:
47     type: pandas.ParquetDataSet
48     filepath: data/01_raw/reviews.parquet
49
50 pictures:
51     type: pillow.ImageDataSet
52     filepath: data/01_raw/images/*.jpg
53
```

- conf
- base
 - parameters
 - ! azure.yml
 - ! catalog.yml
 - ! logging.yml
 - ! parameters.yml
 - local
 - data
 - docs
 - notebooks
 - src

Configuration and code separation with envs!

```
companies:
  type: pandas.CSVDataSet
  filepath: data/01_raw/companies.csv
```

Local catalog.yml

```
reviews:
  type: pandas.ParquetDataSet
  filepath: data/01_raw/reviews.parquet
```

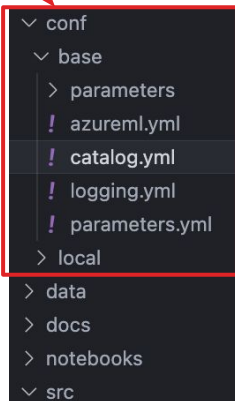
```
pictures:
  type: pillow.ImageDataSet
  filepath: data/01_raw/images/*.jpg
```

```
companies:
  type: pandas.CSVDataSet
  filepath: abfs://my_blob_container/data/01_raw/companies.csv
```

Cloud catalog.yml

```
reviews:
  type: pandas.SQLQueryDataSet
  sql: "select * from reviews;"
  credentials: db_credentials
```

```
pictures:
  type: kedro_azureml.AzureMLFileDataSet
  dataset: my_dataset_from_azureml
  filepath: data/01_raw/images/*.jpg
```



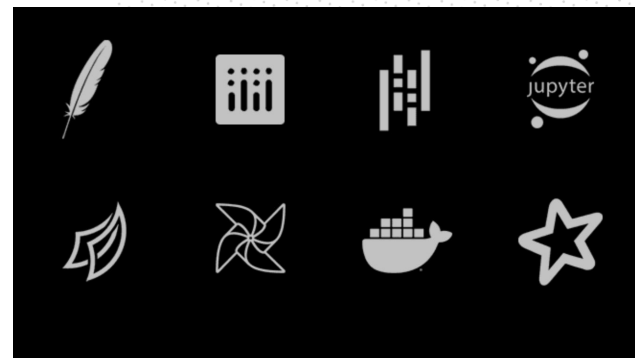
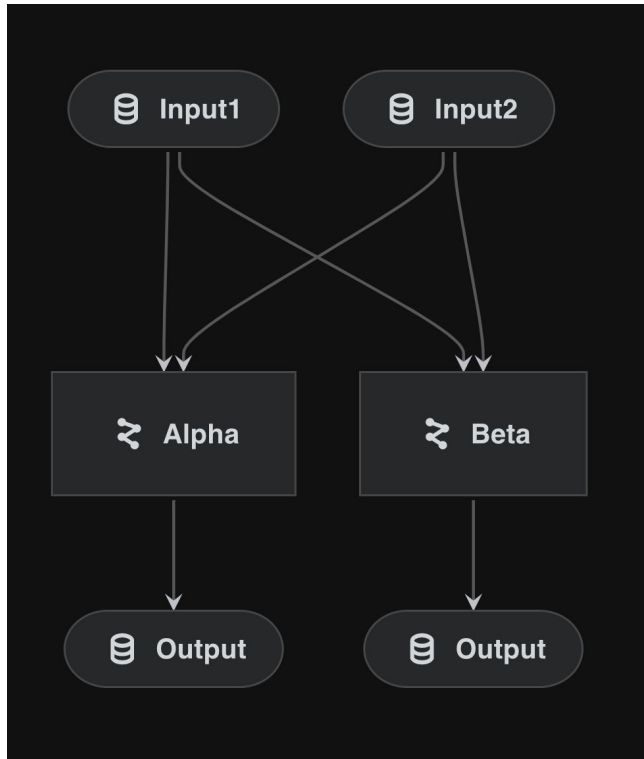
- ▼ conf
 - ▼ base
 - > parameters
 - ! azureml.yml
 - ! catalog.yml
 - ! logging.yml
 - ! parameters.yml
 - > local
- > data
- > docs
- > notebooks
- ▼ src

```
42 companies:
43     type: pandas.CSVDataSet
44     filepath: data/01_raw/companies.csv
45
46 reviews:
47     type: pandas.ParquetDataSet
48     filepath: data/01_raw/reviews.parquet
49
50 pictures:
51     type: pillow.ImageDataSet
52     filepath: data/01_raw/images/*.jpg
53
```

Data Catalog

What other features does Kedro have?

Modular and dynamic pipelines!



Extensibility & Integrations

Kedro can be integrated with multiple industry leading solutions, including: Apache Spark, Pandas, Dask, Matplotlib, Plotly, fsspec, Apache Airflow, Jupyter Notebook and Docker.

Stay tuned for our blog post soon!

Are we done yet?



Building blocks of the GID MLOps



Data Scientist

Experimentation + EDA

Machine Learning frameworks

Portable MLOps framework

Experiment tracking and collaboration

IaC and automation



Kedro

mlflow



Terraform

Cloud Integrations (incl. GID Kedro plugins)

Execution environment

Data



MLOps / ML Engineer

Example technologies:



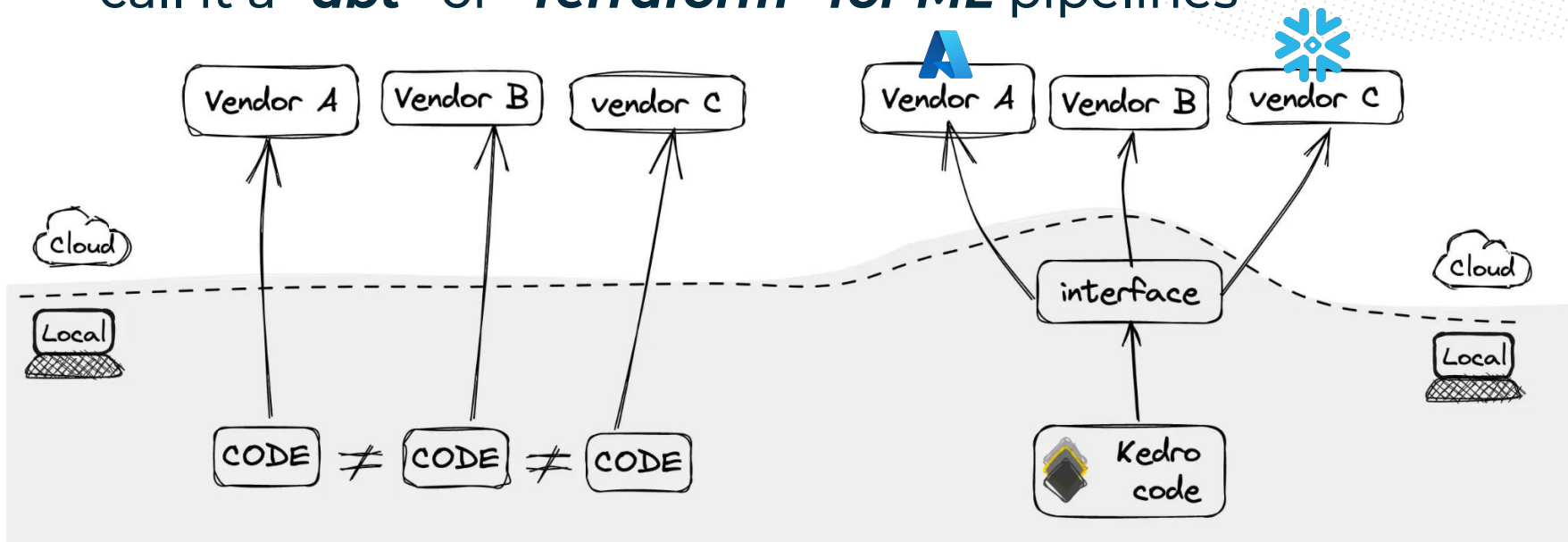
XGBoost

GID MLOps Platform



Kedro plugins to run your ML pipelines, everywhere!

- Kedro is claimed to be a “React” for ML ... but we prefer to call it a “**dbt**” or “**Terraform**” for ML pipelines



Source: [Xebia blog](#)

Write once - run (almost) everywhere



Kedro Vertex AI (GCP)

github.com/getindata/kedro-vertexai



Kedro Sagemaker (AWS)

github.com/getindata/kedro-sagemaker



Kedro Kubeflow (Kubernetes)

github.com/getindata/kedro-kubeflow



Kedro AzureML (Azure)

github.com/getindata/kedro-azureml



Kedro Snowflake (all clouds)

github.com/getindata/kedro-snowflake



Kedro



In progress: **Kedro DBX**

Read more on our blog: [Running Kedro... everywhere? Machine Learning Pipelines on Kubeflow, Vertex AI, Azure and Airflow](#)

Write once - run (almost) everywhere



Kedro Vertex AI (GCP)

github.com/getindata/kedro-vertexai



Kedro Sagemaker (AWS)

github.com/getindata/kedro-sagemaker



Kedro Kubeflow (Kubernetes)

github.com/getindata/kedro-kubeflow



Kedro AzureML (Azure)

github.com/getindata/kedro-azureml



Kedro Snowflake (all clouds)

github.com/getindata/kedro-snowflake



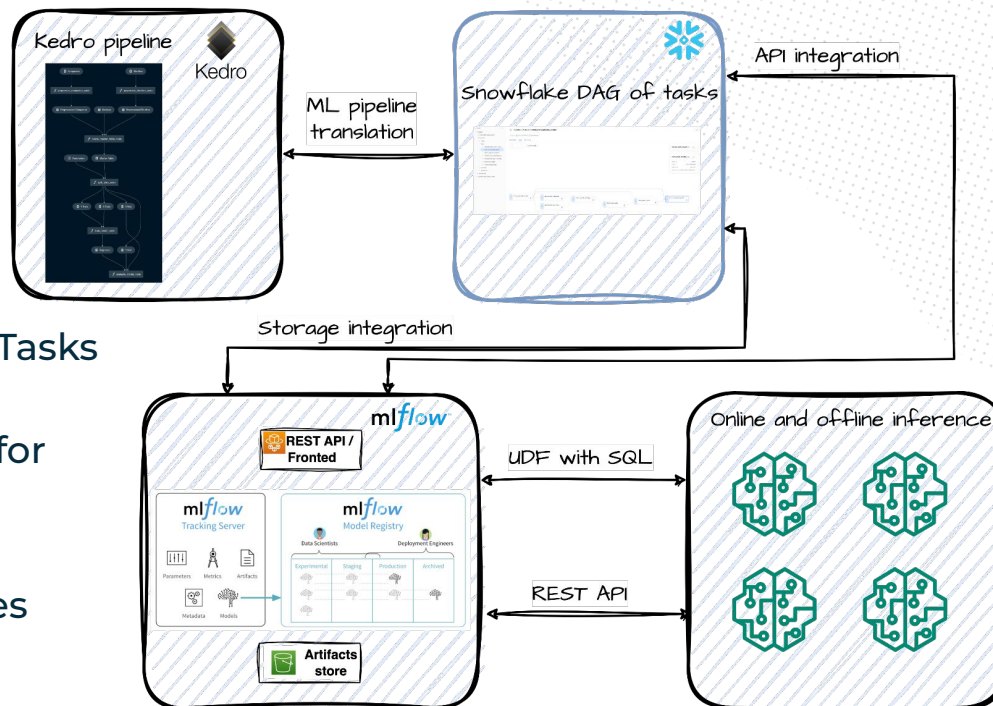
Kedro



In progress: **Kedro DBX**

Read more on our blog: [Running Kedro... everywhere? Machine Learning Pipelines on Kubeflow, Vertex AI, Azure and Airflow](#)

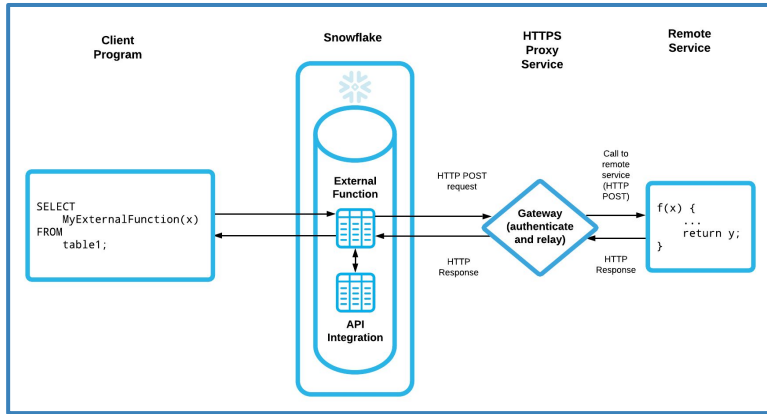
Kedro-Snowflake - putting it all together...



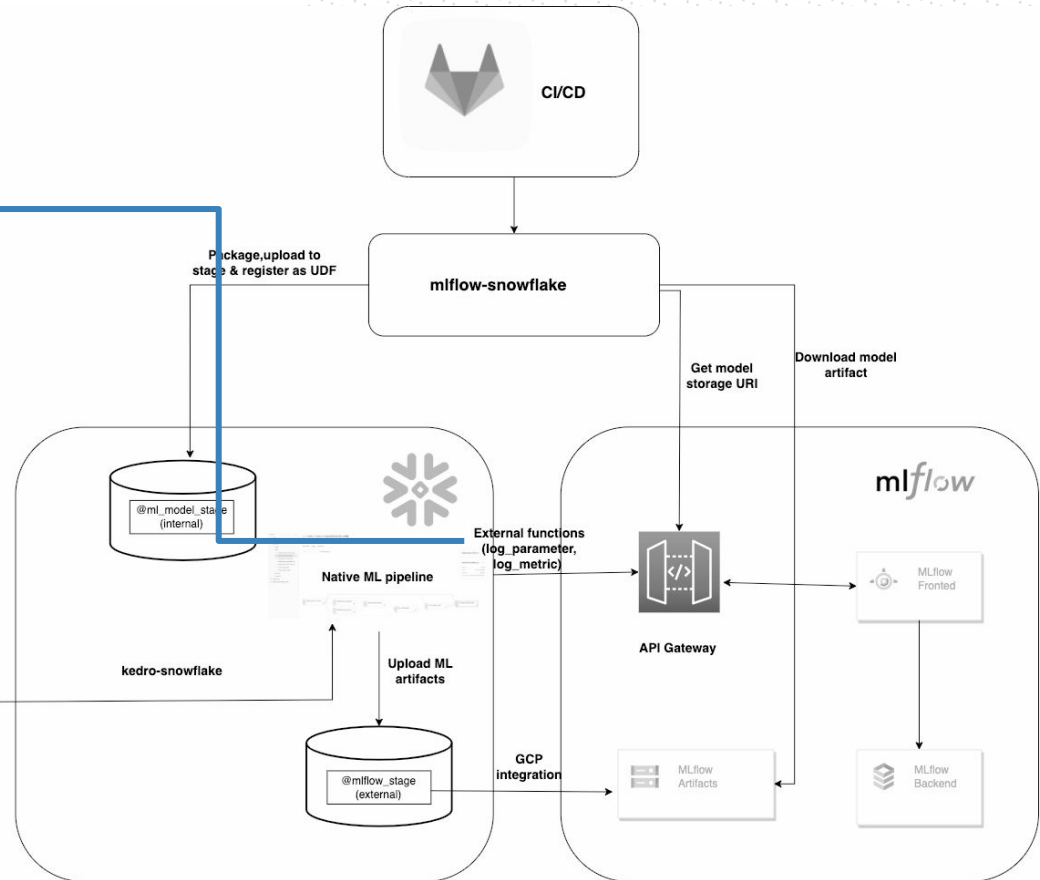
- Native **Snowpark ML** and Tasks integration
- **MLflow Snowflake** plugin for deployment as *UDF*
- MLflow **Sagemaker** - *REST*
- Set of **Terraform** of modules
- Built-in Kedro **starter**



MLOps Platform for Snowflake - GCP



- **Snowflake external functions and API Gateway**





MLOps Platform for Snowflake- GCP

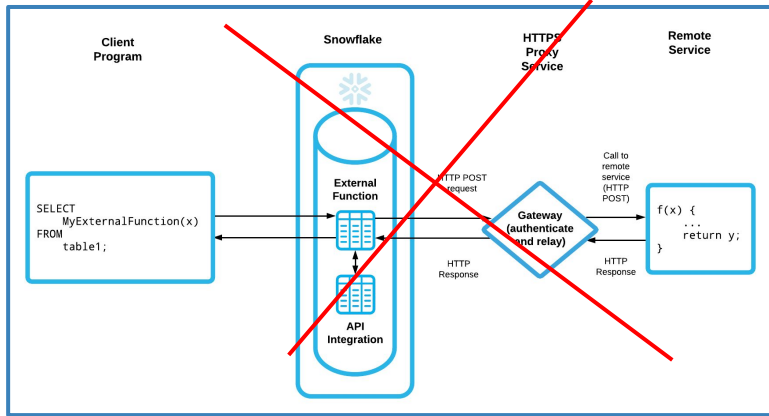
- Glue code for requests/responses to MLflow API
- [PR](#) to the Snowflake provider

```
34
35 resource "snowflake_external_function" "mlflow_run_create" {
36     name      = upper("mlflow_run_create")
37     database  = var.database_name
38     schema   = var.schema_name
39     arg {
40         name = "experiment_id"
41         type = "varchar"
42     }
43     return_type      = "OBJECT"
44     return_behavior  = "VOLATILE"
45     api_integration  = snowflake_api_integration.mlflow_gcp.name
46     request_translator = "${var.database_name}.${var.schema_name}.${snowflake_function.mlflow_run_create_req.name}"
47     response_translator = "${var.database_name}.${var.schema_name}.${snowflake_function.mlflow_generic_res.name}"
48     url_of_proxy_and_resource = "${var.api_gateway_url}/api/2.0/mlflow/runs/create"
49 }
```

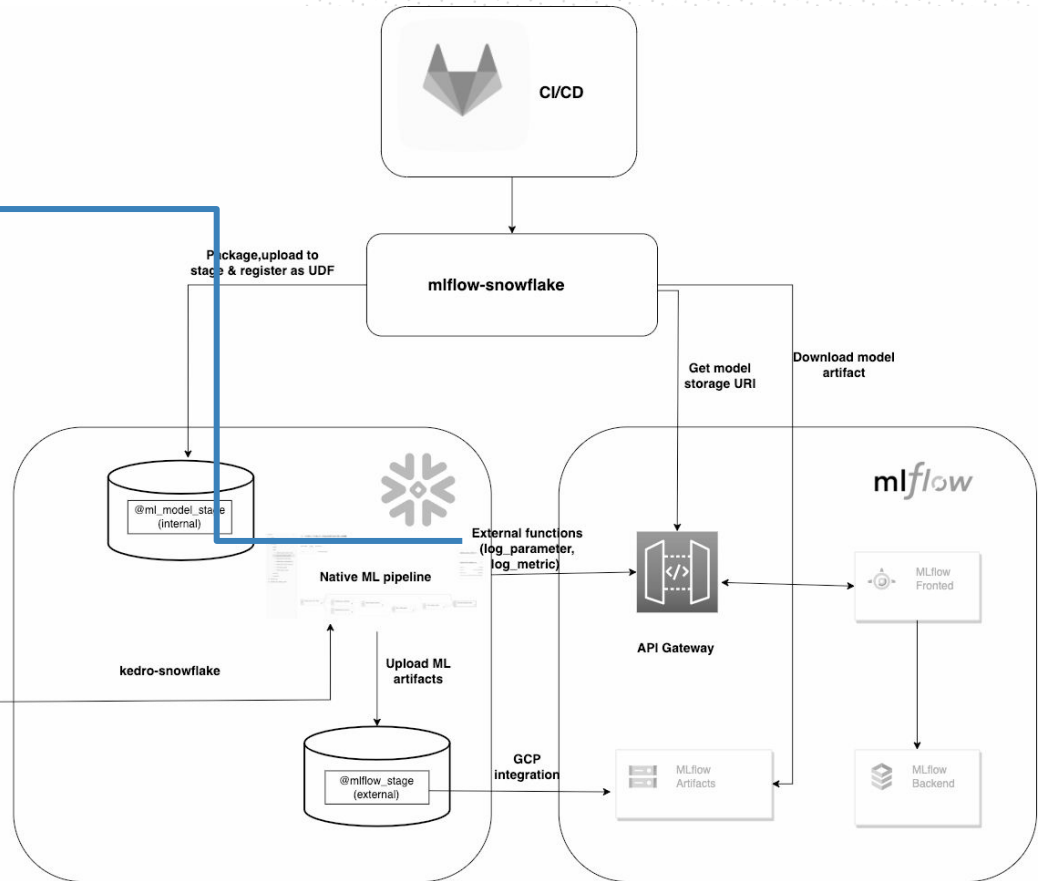
```
36
37 resource "snowflake_function" "mlflow_run_create_req" {
38     name      = upper("mlflow_run_create_req")
39     database  = snowflake_database.db.name
40     schema   = snowflake_schema.schema.name
41     arguments {
42         name = "event"
43         type = "OBJECT"
44     }
45     comment      = "Request translator for MLflow create run"
46     return_type  = "OBJECT"
47     language     = "javascript"
48     statement    = <<EOH
49     let experimentId = EVENT.body.data[0][1]
50     let timestamp = new Date().getTime();
51     return { "body": { "experiment_id": experimentId, start_time: timestamp }}
52     EOH
53 }
```



MLOps Platform for Snowflake - AWS/Azure



- Snowflake external **integrations** and **network** rules
- **No** need for API Gateway

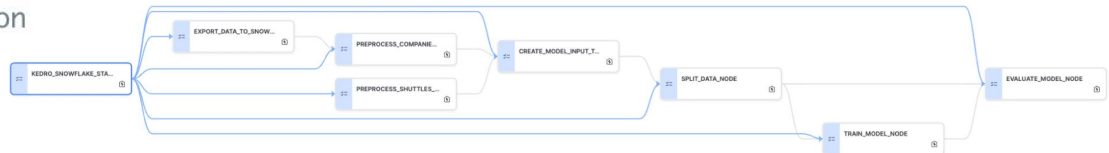




MLOps Platform for Snowflake

The screenshot displays the MLOps platform interface. On the left, a workflow graph shows the following nodes: 'Preprocess Companies Node' and 'Preprocess Shuttles Node' (both functions) feed into 'Create Model Input Table Node' (function). This node feeds into 'Split Data Node' (function), which then feeds into 'Train Model Node' (function). Finally, 'Train Model Node' and 'Split Data Node' feed into 'Evaluate Model Node' (function). The central pane shows a search results view for 'KEDRO / PUBLIC / TRAIN_MODEL_NODE', listing various tasks and procedures. The right pane shows the 'Run History' for 'KEDRO / PUBLIC / TRAIN_MODEL_NODE', indicating one successful task run on May 13, 2023, at 2:09:44 PM, with a duration of 1m 11s.

kedro snowflake run --wait-for-completion

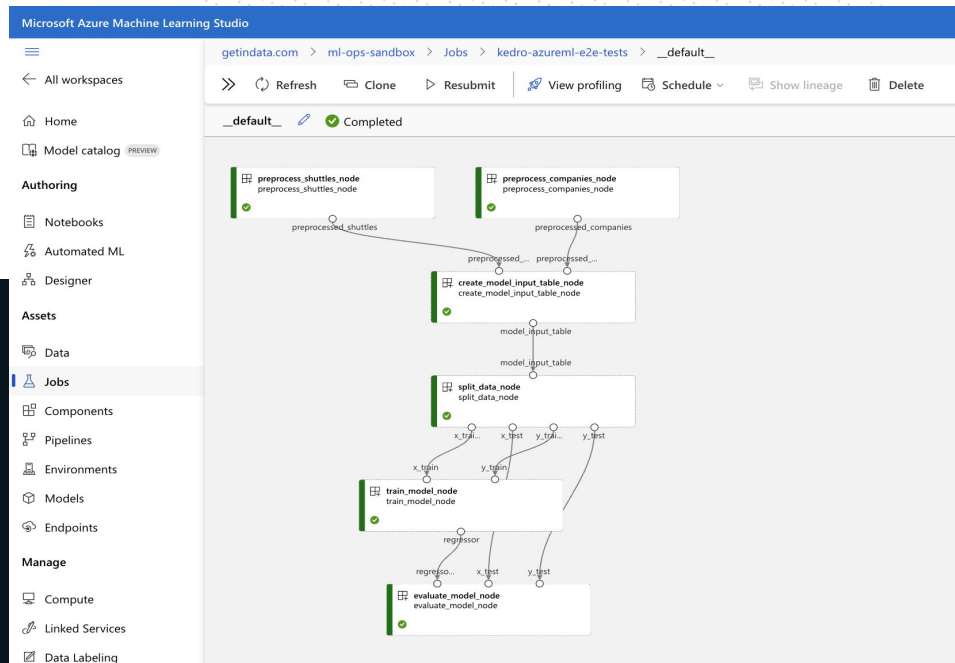
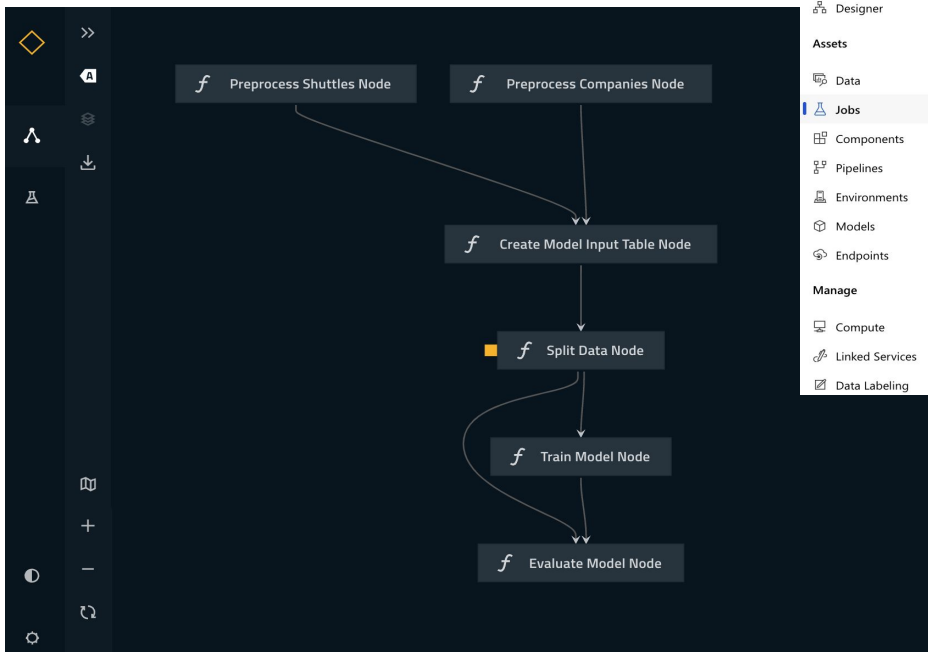




- Support for native Snowflake Tables and Stages in Kedro Data catalog

```
46 companies_snowflake:
47     type: kedro_datasets.snowflake.SnowparkTableDataSet
48     table_name: companies_snowflights_starter
49     credentials: snowflake
50     save_args:
51         mode: overwrite
52
53
54 preprocessed_shuttles:
55     type: kedro_snowflake.datasets.native.SnowflakeStageFileDataSet
56     stage: "@KEDRO_SNOWFLAKE_TEMP_DATA_STAGE" # <-- Snowflake stage to store data in
57     filepath: data/02_intermediate/preprocessed_shuttles.csv # <-- file path within the stage
58     credentials: snowflake # <-- credentials to connect to Snowflake (the same as for SnowparkTableDataSet)
59     dataset: # <-- dataset key defines the dataset type to use
60     type: pandas.CSVDataSet # <-- specify any params for the nested dataset here
```


- **Kedro** AzureML plugin
- Azure ML builtin “**MLflow**”
- Set of **Terraform** of modules



Kedro-AzureML - datasets

```
54 my_pandas_dataframe_dataset:
55     type: kedro_azureml.datasets.AzureMLPandasDataSet
56     azureml_dataset: my_new_azureml_dataset
57
58     # if version is not provided, the latest dataset version is used
59     azureml_dataset_load_args:
60         version: 1
61
62
63 processed_images:
64     type: kedro_azureml.datasets.AzureMLFileDataSet
65     dataset: pillow.ImageDataSet
66     filename_suffix: '.png'
67     azureml_dataset: processed_images
68     azureml_dataset_save_args:
69         create_new_version: true
70
71     # if version is not provided, the latest dataset version is used
72     azureml_dataset_load_args:
73         version: 1
```

Microsoft Azure Machine Learning Studio

getindata.com > ml-ops-sandbox > Data

Data

Data assets | Datalstores | Dataset monitors | PREVIEW

Data assets are references to your data. You can create data assets from datastores, local files, public URLs, or Open Datasets. Data assets can be versioned. [data assets](#)

[+ Create](#) [Refresh](#) [Archive](#) [View options](#)

Search

Name	Source	Version	Created on ↓	Modified on
e2e_tests_preprocessed_compa...	This workspace	15	Mar 8, 2023 1:17 PM	May 4, 2023 4:41 PM
spaceflights_x_test	This workspace	2	Feb 22, 2023 4:23 PM	Feb 22, 2023 4:43 PM
spaceflights_x_train	This workspace	3	Feb 22, 2023 3:55 PM	Feb 22, 2023 4:43 PM
spaceflights_test_data	This workspace	2	Feb 22, 2023 3:51 PM	Feb 22, 2023 3:55 PM
spaceflights_training_data	This workspace	4	Feb 22, 2023 3:49 PM	Feb 22, 2023 3:55 PM
as	This workspace	1	Jan 27, 2023 1:16 PM	Jan 27, 2023 1:16 PM
Stanford Cars	This workspace	1	Sep 22, 2022 10:46 AM	Sep 22, 2022 10:46 AM
BatchPredict1	This workspace	1	Aug 25, 2022 4:07 PM	Aug 25, 2022 4:07 PM
Test	This workspace	1	Aug 17, 2022 9:42 AM	Aug 17, 2022 9:42 AM

Navigation: All workspaces, Home, Model catalog (PREVIEW), Authoring (Notebooks, Automated ML, Designer), Assets (Data, Jobs, Components, Pipelines, Environments, Models, Endpoints), Manage (Compute, Linked Services)

MLOps Azure DevOps pipeline

Azure DevOps marcin-getindata / gidmlopsv7 / Pipelines / MLOps demo pipeline / 20230510.1

gidmlopsv7 +

- Overview
- Boards
- Repos
- Pipelines**
- Pipelines
- Environments
- Releases
- Library
- Task groups
- Deployment groups

Jobs in run #20230510.1

MLOps demo pipeline

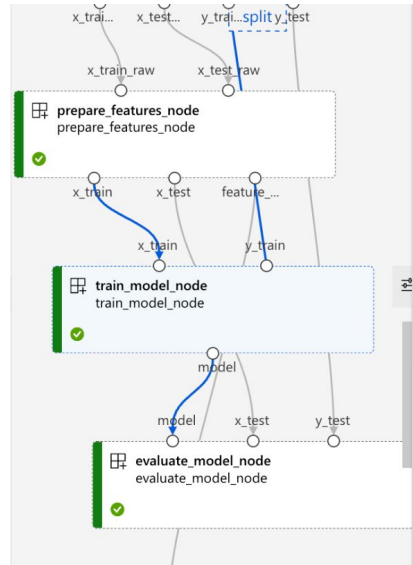
Build & run

▼	MLOps Pipeline	42s
✓	Initialize job	1s
✓	Checkout MLOps by GetInData ...	1s
✓	Login to ACR	<1s
✓	Pull latest image	1s
🔄	Build	39s
○	Push	
○	Run Kedro pipeline in Azure ML	
○	Post-job: Checkout MLOps by Getl...	

Build

```
488 _____ 78.2/78.2 KB 238.3
489 Collecting pytoolconfig[global]>=1.2.2
490   Downloading pytoolconfig-1.2.4-py3-none-any.whl (16 kB)
491   Downloading pytoolconfig-1.2.2-py3-none-any.whl (16 kB)
492 Collecting appdirs>=1.4.4
493   Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
494 Collecting idna<4,>=2.5
495   Downloading idna-3.4-py3-none-any.whl (61 kB)
496 _____ 61.5/61.5 KB 224.5
497 Collecting PySocks!=1.5.7,>=1.5.6
498   Downloading PySocks-1.7.1-py3-none-any.whl (16 kB)
499 Collecting jeepney>=0.6
500   Downloading jeepney-0.8.0-py3-none-any.whl (48 kB)
501 _____ 48.4/48.4 KB 203.5
502 Collecting greenlet!=0.4.17
503   Downloading greenlet-2.0.2-cp39-cp39-manylinux_2_17_x86_64.ma
504 _____ 610.9/610.9 KB 293.6
505 Collecting backports weakref
506   Downloading backports.weakref-1.0.post1-py2.py3-none-any.whl
507 Collecting pyproject_hooks
```

Azure ML "MLflow"



train_model_node

Overview Parameters Outputs + logs Metrics Child jobs Ima

Refresh Create custom chart View as... Current vi

Select metrics

Select to view as visualization or table of the data

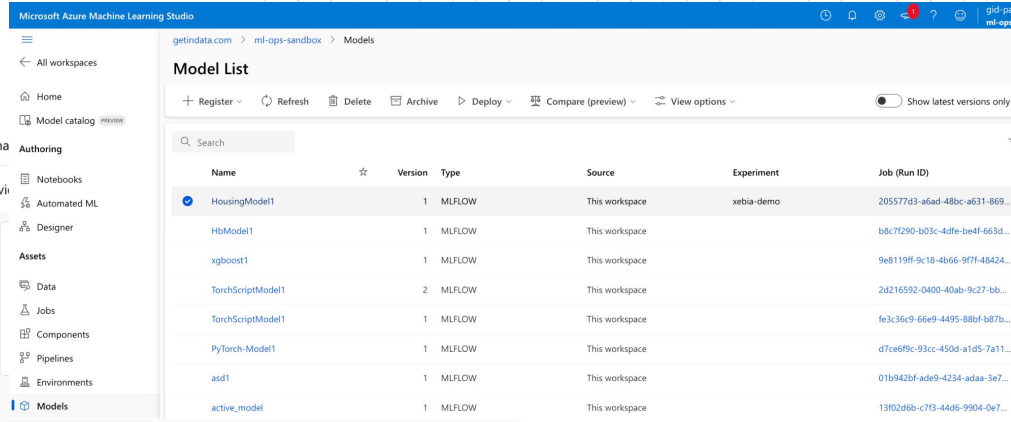
Search

- Select all
- Metrics (5)
 - training_mean_absolute_error
 - training_mean_squared_error
 - training_r2_score
 - training_root_mean_squared_error
 - training_score

training_mean_absolut...
1.7061e+4

training_score
0.9190283

```
60 def train_model(  
61     X_train: pd.DataFrame, y_train: pd.Series, random_state: int, model_params: dict  
62 ):  
63     """Train the model on the training data."""  
64     mlflow.sklearn.autolog(  
65         log_input_examples=True, log_model_signatures=True, log_models=True  
66     )  
67     model = RandomForestRegressor(random_state=random_state, **model_params)  
68     model.fit(X_train, y_train)  
69     return model
```



The screenshot shows the Microsoft Azure Machine Learning Studio interface. The top navigation bar includes "All workspaces", "Home", "Model catalog", and "Authoring". The "Model List" section displays a table of models:

Name	Version	Type	Source	Experiment	Job (Run ID)
HousingModel1	1	MLFLOW	This workspace	xebia-demo	205577d3-a5ad-48bc-a631-869...
HbModel1	1	MLFLOW	This workspace		b8c7f290-b03c-4dfe-ba4f-663d...
xgboost1	1	MLFLOW	This workspace		9e8119ff-9c18-4b66-917f-48424...
TorchScriptModel1	2	MLFLOW	This workspace		2d216592-0400-40ab-9c27-bb...
TorchScriptModel1	1	MLFLOW	This workspace		fe3c36c9-66e9-4495-88bf-b87b...
PyTorch-Model1	1	MLFLOW	This workspace		d7ce6f9c-93cc-450d-a1d5-7a11...
asd1	1	MLFLOW	This workspace		01b942bf-ade9-4234-adaa-3e7...
active_model	1	MLFLOW	This workspace		13f0266b-c7f3-44d6-9904-0e7...

Model deployment - Azure DevOps

The screenshot displays the Azure DevOps web interface. On the left is a navigation sidebar with items like Overview, Boards, Repos, Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area shows the 'MLOps demo pipeline' with a 'Runs' tab selected. A specific run is visible: '#20230510.1 • Update azure-pipelines.yml', manually triggered for the 'master' branch. A modal dialog titled 'Run pipeline' is open in the foreground, allowing manual execution of a pipeline. The dialog includes a dropdown for 'Branch/tag' set to 'master'. Two radio buttons are present: 'Run training pipeline' (unchecked) and 'Deploy model to Online Endpoint' (checked and highlighted with a red box). Below these are several text input fields: 'Name of the model to deploy' (HousingModel:1), 'Name of the endpoint to deploy to' (mlops-getindata-demo-endpoint), 'Description of the endpoint' (Demo endpoint for mlops-getindata-demo), 'Name of the deployment' (mlops-getindata-demo-deployment), 'Instance type for the deployment' (Standard_DS2_v2), 'Authentication method for the endpoint (Key or AMLToken)' (Key), and 'Traffic allocation for the deployment' (100). At the bottom of the dialog, there is an unchecked checkbox for 'Enable system diagnostics' and two buttons: 'Cancel' and 'Run'.

Run pipeline

Select parameters below and manually run the pipeline

Branch/tag
master

Select the branch, commit, or tag

Run training pipeline

Deploy model to Online Endpoint

Name of the model to deploy
HousingModel:1

Name of the endpoint to deploy to
mlops-getindata-demo-endpoint

Description of the endpoint
Demo endpoint for mlops-getindata-demo

Name of the deployment
mlops-getindata-demo-deployment

Instance type for the deployment
Standard_DS2_v2

Authentication method for the endpoint (Key or AMLToken)
Key

Traffic allocation for the deployment
100

Enable system diagnostics

Cancel Run

Model Monitoring

ml-ops-sandbox-fyevv

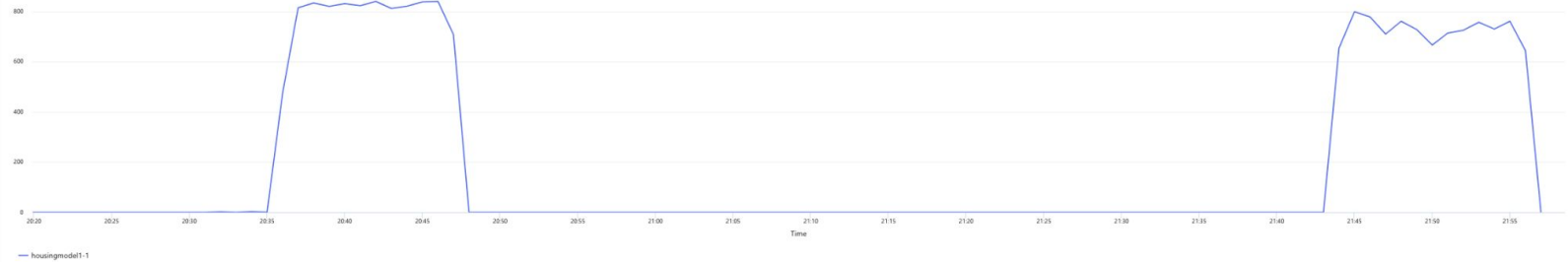
Details Test Consume Monitoring Deployment logs

Time range Wed, May 10, 2023 8:19 PM - Wed, May 10, 2023 9:58 PM Auto refresh

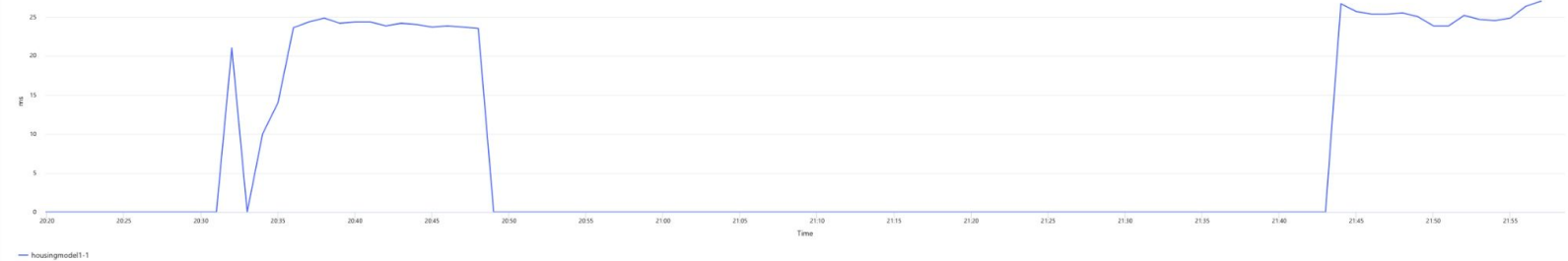
Average Requests per minute
157

Average Request latency
25 ms

Requests per minute



Request latency



3 Take-home messages

- Kedro is one of the best MLOps frameworks to make data scientists more **productive** out-of-the-box
- GetinData contributions to Kedro enable users to extend their Data Platforms with MLOps capabilities **seamlessly**
- **Kedro** together with **MLflow** and **Terraform** are the main building blocks of **our MLOps platform**

Core TEAM



Michał Bryś

Machine Learning Architect



Marek Wiewiórka

Chief Data Architect



Marcin Zabłocki

MLOps Architect



Artur Dobrogowski

MLOps Engineer



Take DATA Pill and join DATA Matrix

- Best content from **Big Data, Cloud, AI/ML** and more
- simple, **condensed formula**
- sent **weekly** every Friday morning

Subscribe now:

