

0 to ML ps platform with Snowflake

Snowflake Poznań Meetup, July 20th 2023





Marek Wiewiórka

Chief Data Architect, GetInData | Part of Xebia | marek@getindata.com
Research Assistant at Warsaw University of Technology



- [Soon to be PhD](#) in bioinformatics
- An open source contributor to [Snowflake Terraform Provider](#), [SeQuiLa](#) and [Kedro](#)
- Personally a keen long distance runner and gravel bike enthusiast



Marcin Zabłocki

MLOPS ARCHITECT at GetInData | Part of Xebia | marcin.zablocki@getindata.com

- DE → DS → MLE → MLOps
- member of Kedro TSC
- LEGO fan
- [ML-Workout.pl](#)



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.**

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. (Retail & consumer goods)

- **Snowflake migration from AWS to Azure**
- Strong governance capabilities and **DataOps focus**

3. Other



SOLUTION AREAS



MLOps & Modern Data Platforms



Data & ML engineering project accelerators

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



Stream processing & real-time analytics

Technologies



GetInData joins Xebia



Our Customers

acast



ING



IAS Integral
Ad Science

Kcell



truecaller

iZettle



BNP PARIBAS



FRESHMAIL

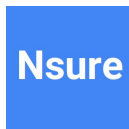
allegro

omio
formerly go^oeuro



databricks

Asales



Bank Polski

LINK 4

Kambi

CSG



KITOPI

NetWorkS



mamava

miinto



VEOLIA



NetSpring

JIMDO

sqoo**ba**



T-Mobile

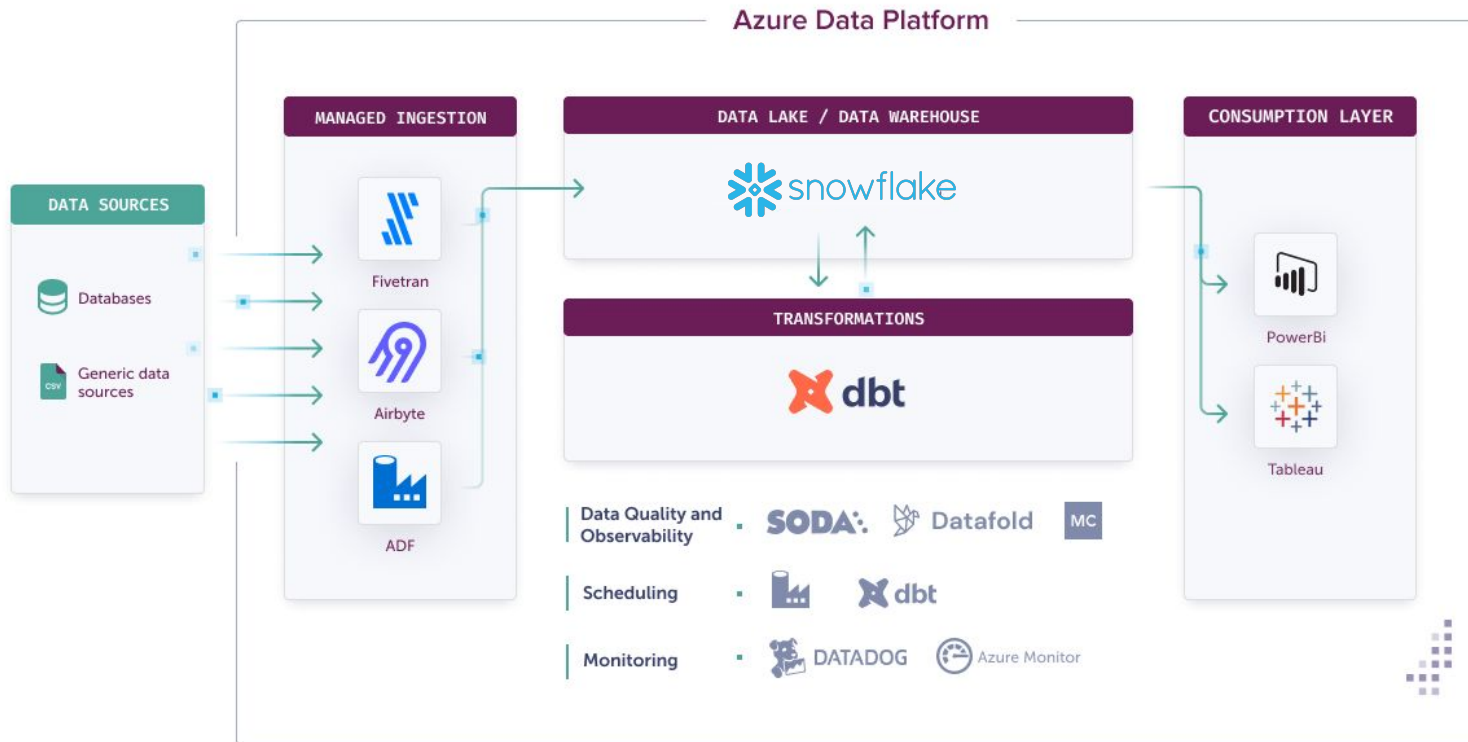


datakin



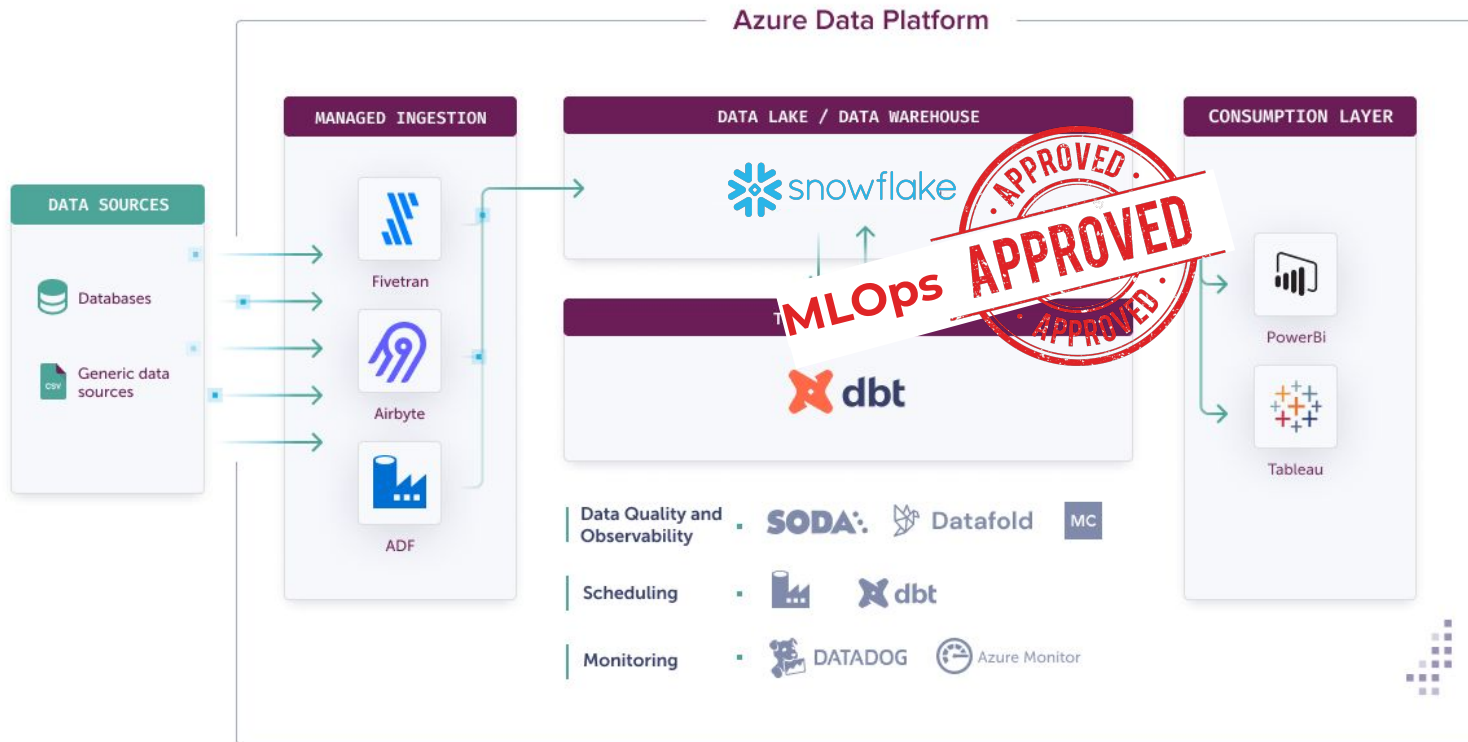
ververica

From MDP¹ to MDP 2.0 (MLOps-enabled Data Platform)



¹MDP - Modern Data Platform

From MDP¹ to MDP 2.0 (MLOps-enabled Data Platform)

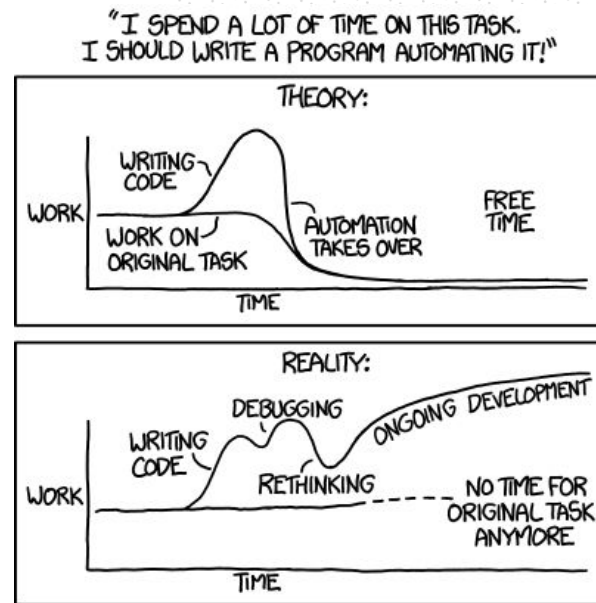


¹MDP - Modern Data Platform

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

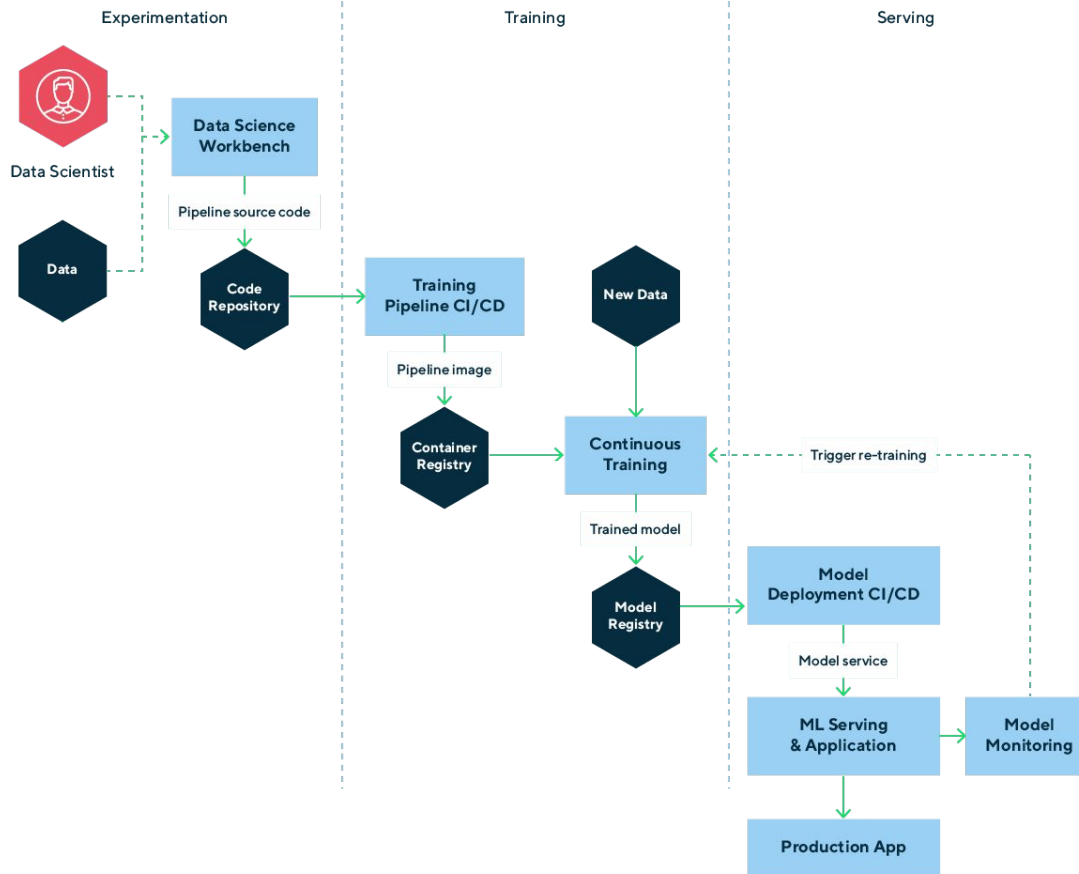
The ultimate goal is **PRODUCTIVITY**



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
- **Open** standards and cloud **agnosticism**
- Short development **feedback loop** (incl. local dev)
- **Fast** new ML projects bootstrapping and **standardization**
- Execution environment **independent** training pipelines
- ... MLOps capabilities provisioned **in days, not months**



MLOps Platform

A process overview



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:



ML projects in layers



Data
Scientist

Experimentation + EDA

Machine Learning frameworks

?

Execution environment

Data



MLOps / ML
Engineer

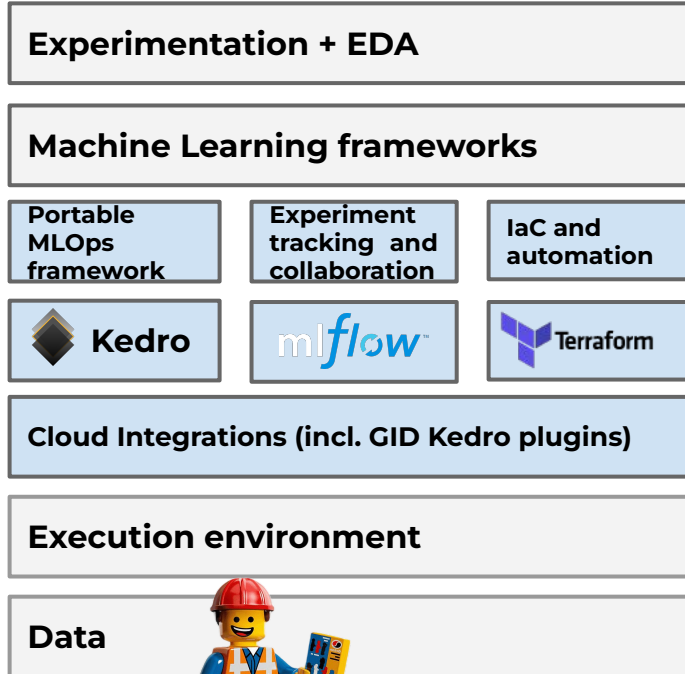
Example technologies:



Building blocks of the GID MLOps



Data Scientist



MLOps / ML Engineer

Example technologies:



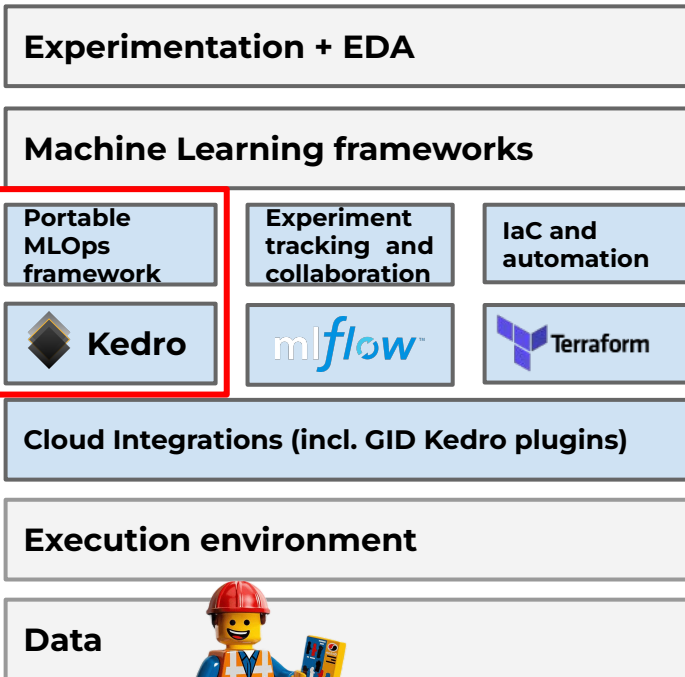
GID MLOps Platform



Building blocks of the GID MLOps



Data Scientist



MLOps / ML Engineer

Example technologies:



XGBoost

GID MLOps Platform



What is Kedro?



Kedro

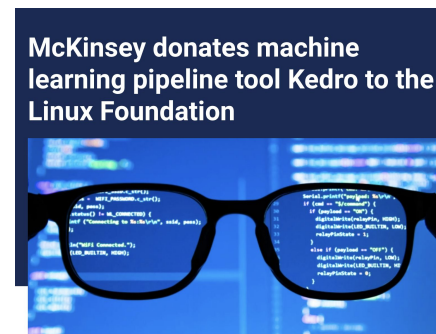
=

Software
Engineering
Principles

+

Data Science

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



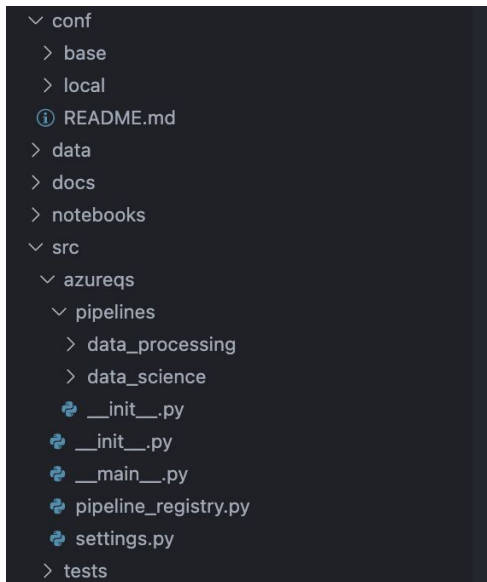
What features does Kedro have? (Part 1)

```

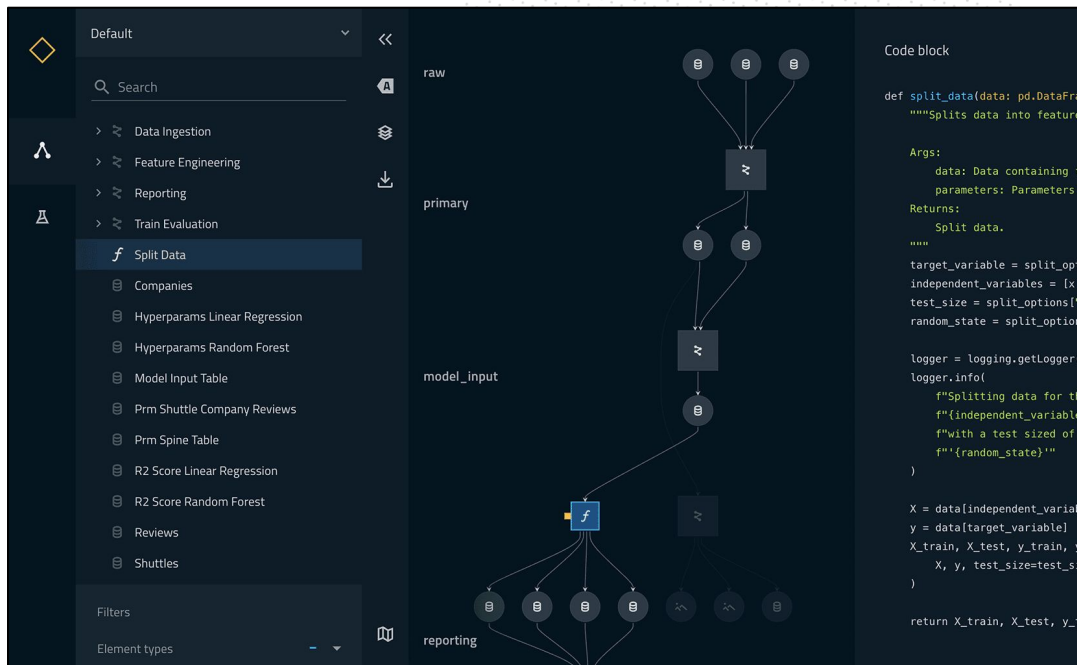
└─ conf
  ├── base
  ├── local
  └─ README.md
└─ data
└─ docs
└─ notebooks
└─ src
  └─ azureqs
    └─ pipelines
      ├── data_processing
      ├── data_science
      ├── __init__.py
      ├── __init__.py
      ├── __main__.py
      ├── pipeline_registry.py
      └── settings.py
  └─ tests
```

**Well defined
project structure
+ project starters**

What features does Kedro have? (Part 1)



Well defined
project structure
+ project starters



Nodes & pipelines
abstractions

Kedro pipeline - data 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", "ratings"],
24                outputs="model_input_table",
25                name="create_model_input_table_node",
26            ),
27        ]
28    )
```

Kedro pipeline - data science

```
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", "params:model_options"],
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", "y_train"],
12                outputs=["X_train", "X_test", "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
```

```

30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```

2
3     model_options:
4         test_size: 0.2
5         random_state: 3
6     features:
7         - engines
8         - passenger_capacity
9         - crew
10        - d_check_complete
11        - moon_clearance_complete
12        - iata_approved
13        - company_rating
14        - review_scores_rating
15
```

What about data?

```
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", "ratings"],
24                outputs="model_input_table",
25                name="create_model_input_table_node",
26            ),
27        ]
28    )
```


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", "preprocessed_companies", "ratings"],
24                outputs="model_input_table",
25                name="create_model_input_table_node",
26            ),
27        ]
28    )
```

```

  41  ✓ conf
  42  ✓ base
  43  > parameters
  44  ! azureml.yml
  45  ! catalog.yml
  46  ! logging.yml
  47  ! parameters.yml
  48  > local
  49  > data
  50  > docs
  51  > notebooks
  52  ✓ 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
```


What features does Kedro have? (Part 2)

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

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

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

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

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
```

Data Catalog / Environments

What features does Kedro have? (Part 2)

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

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

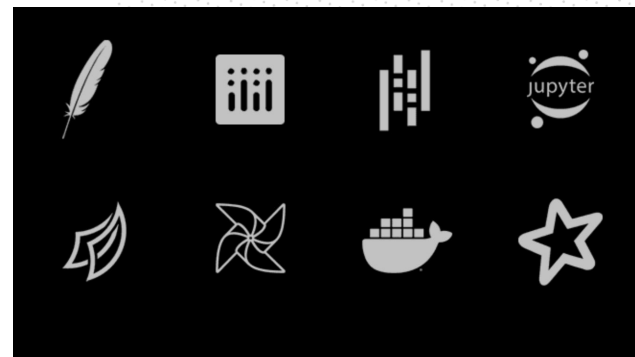
pictures:
  type: pillow.ImageDataSet
  filepath: data/01_raw/images/*.jpg
```

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

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
```

Data Catalog / Environments



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.

ML model?

```
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", "params:model_options"],
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
```



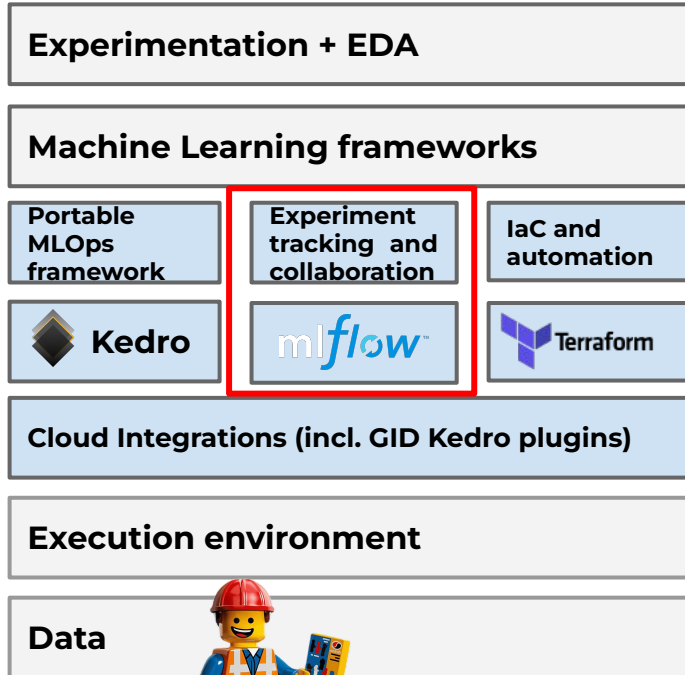
MLflow from Kedro

```
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
```

Building blocks of the GID MLOps



Data Scientist



MLOps / ML Engineer

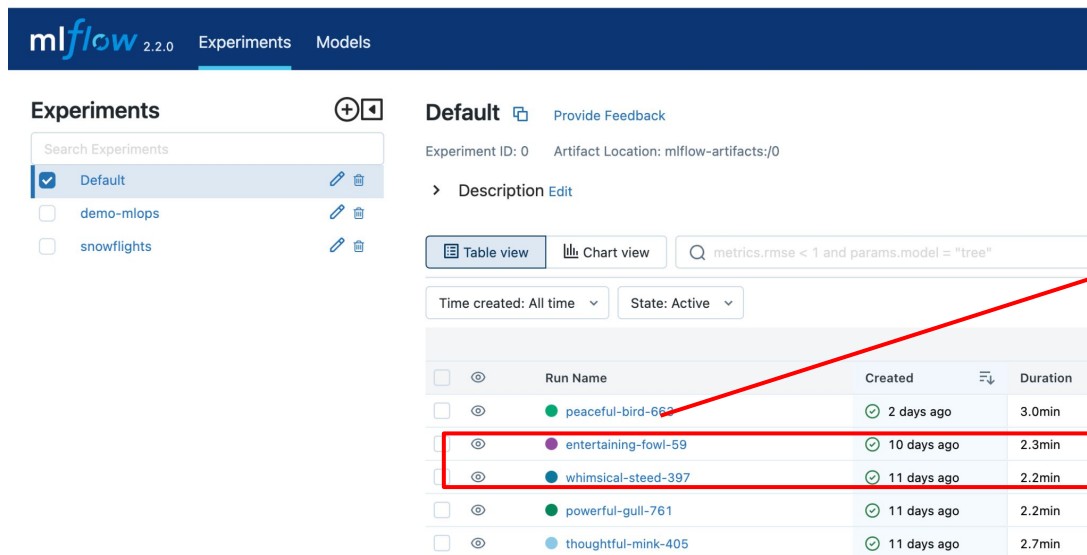
Example technologies:



XGBoost

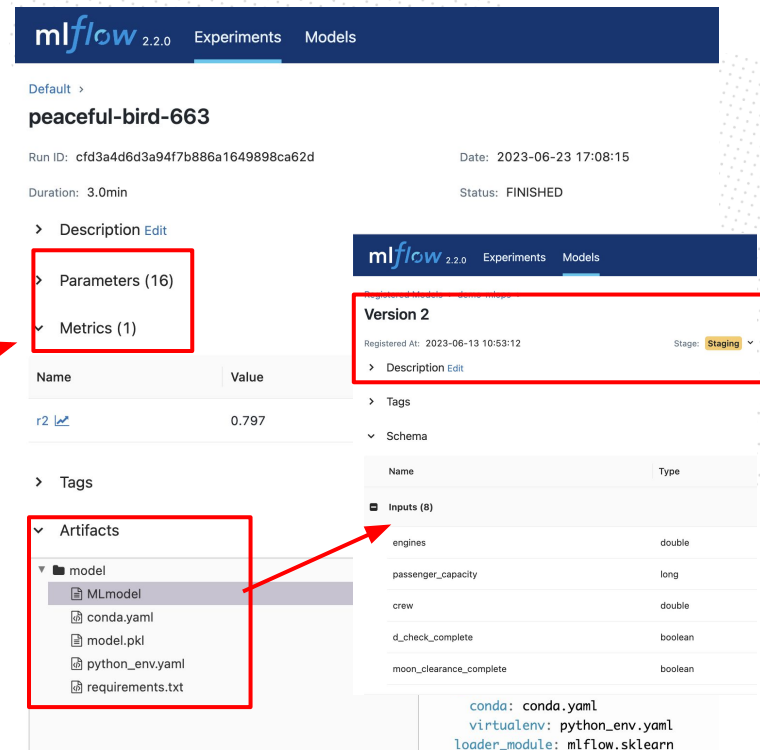
GID MLOps Platform





The screenshot shows the MLflow UI Experiments page. At the top, there's a navigation bar with 'mlflow 2.2.0', 'Experiments', and 'Models'. Below this, a search bar and a list of experiments are visible. The 'Default' experiment is selected. A table lists several runs, with 'entertaining-fowl-59' highlighted in a red box. A red arrow points from this row to the 'Parameters (16)' section in the detailed view on the right.

Run Name	Created	Duration
peaceful-bird-663	2 days ago	3.0min
entertaining-fowl-59	10 days ago	2.3min
whimsical-steed-397	11 days ago	2.2min
powerful-gull-761	11 days ago	2.2min
thoughtful-mink-405	11 days ago	2.7min



The screenshot shows the detailed view of the 'peaceful-bird-663' experiment. The navigation bar at the top shows 'mlflow 2.2.0', 'Experiments', and 'Models'. The experiment name 'peaceful-bird-663' is prominently displayed. Below it, the Run ID, Date, Duration, and Status are shown. The 'Description Edit' section is expanded, showing 'Parameters (16)' and 'Metrics (1)'. A table lists the parameters, with 'r2' having a value of 0.797. The 'Artifacts' section is expanded, showing a list of files including 'MLmodel', 'conda.yaml', 'model.pkl', 'python_env.yaml', and 'requirements.txt'. A red box highlights the 'Artifacts' section, and a red arrow points from it to the 'Inputs (8)' section in the 'Schema' tab. The 'Inputs (8)' section lists various input parameters like 'engines', 'passenger_capacity', 'crew', etc.

Name	Value
r2	0.797

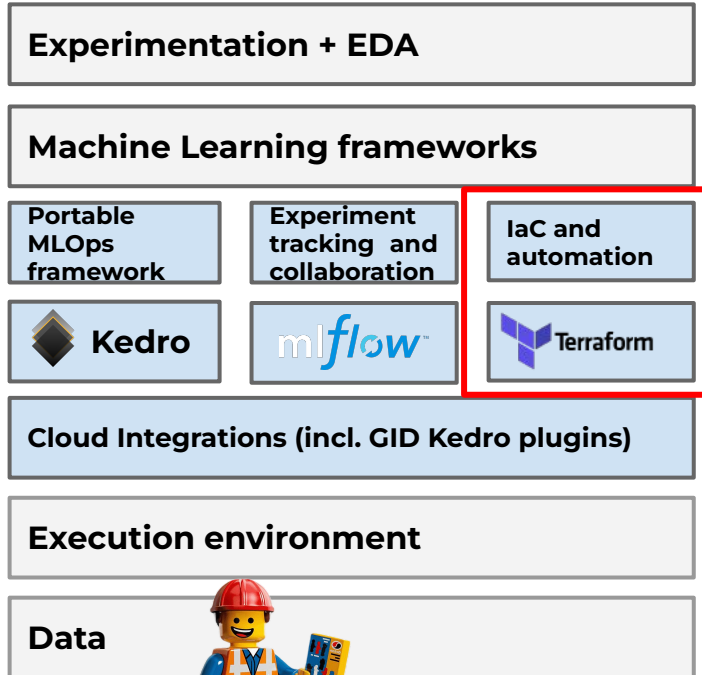
Name	Type
engines	double
passenger_capacity	long
crew	double
d_check_complete	boolean
moon_clearance_complete	boolean

- Experiment tracking
- Model registry
- Model deployments (online and offline) with service plugins

Building blocks of the GID MLOps



Data Scientist



MLOps / ML Engineer

Example technologies:



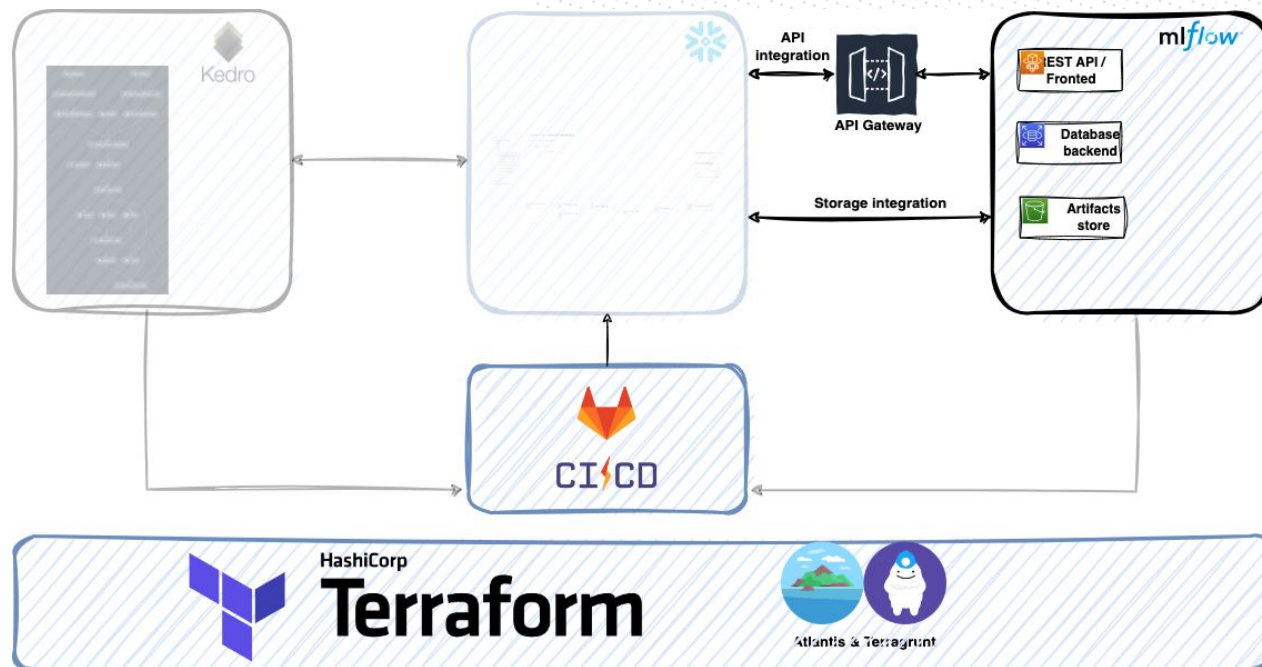
XGBoost

GID MLOps Platform



MLOps Platform provisioned

- Set of **Terraform** modules managed by **Terragrunt**
- Both for Snowflake and specific cloud provider
- **CI/CD** templates
- Available for AWS, Azure and GCP clouds



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:



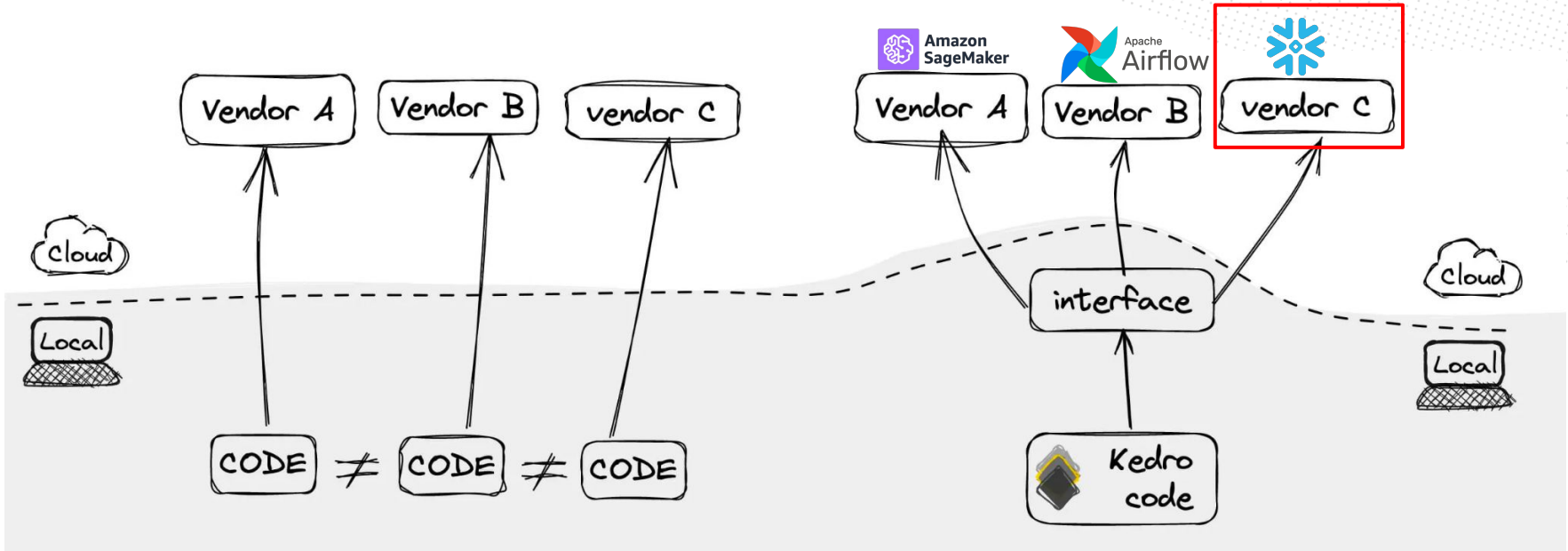
XGBoost

GID MLOps Platform



Why Kedro, again ?!

- 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



Kedro Vertex AI (GCP)

github.com/getindata/kedro-vertexai



Kedro Sagemaker (AWS)

github.com/getindata/kedro-sagemaker



Kedro Airflow (Kubernetes)

github.com/getindata/kedro-airflow-k8s



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

Write once - run (almost) everywhere



Kedro



Kedro Vertex AI (GCP)

github.com/getindata/kedro-vertexai



Kedro Sagemaker (AWS)

github.com/getindata/kedro-sagemaker



Kedro Airflow (Kubernetes)

github.com/getindata/kedro-airflow-k8s



Kedro Kubeflow (Kubernetes)

github.com/getindata/kedro-kubeflow



Kedro AzureML (Azure)

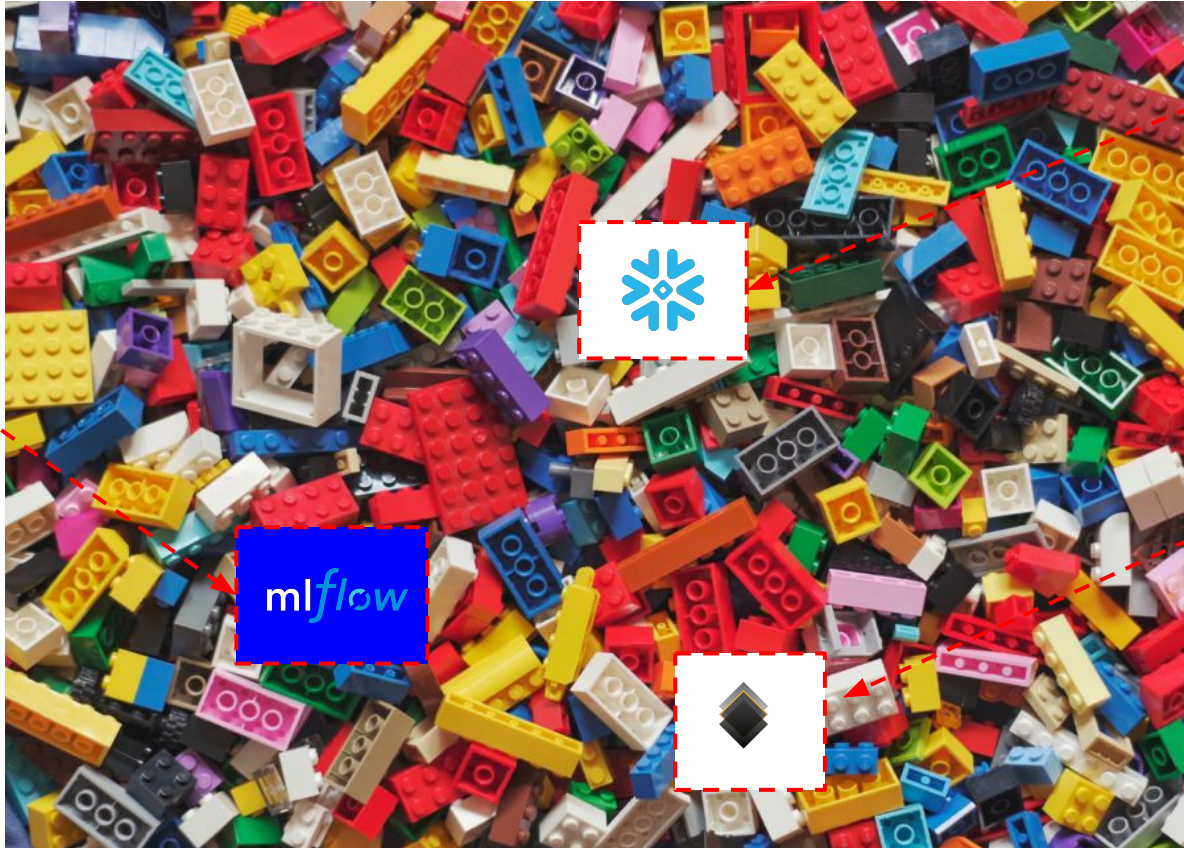
github.com/getindata/kedro-azureml



Kedro Snowflake

github.com/getindata/kedro-snowflake

Putting it all together...



Snowflake

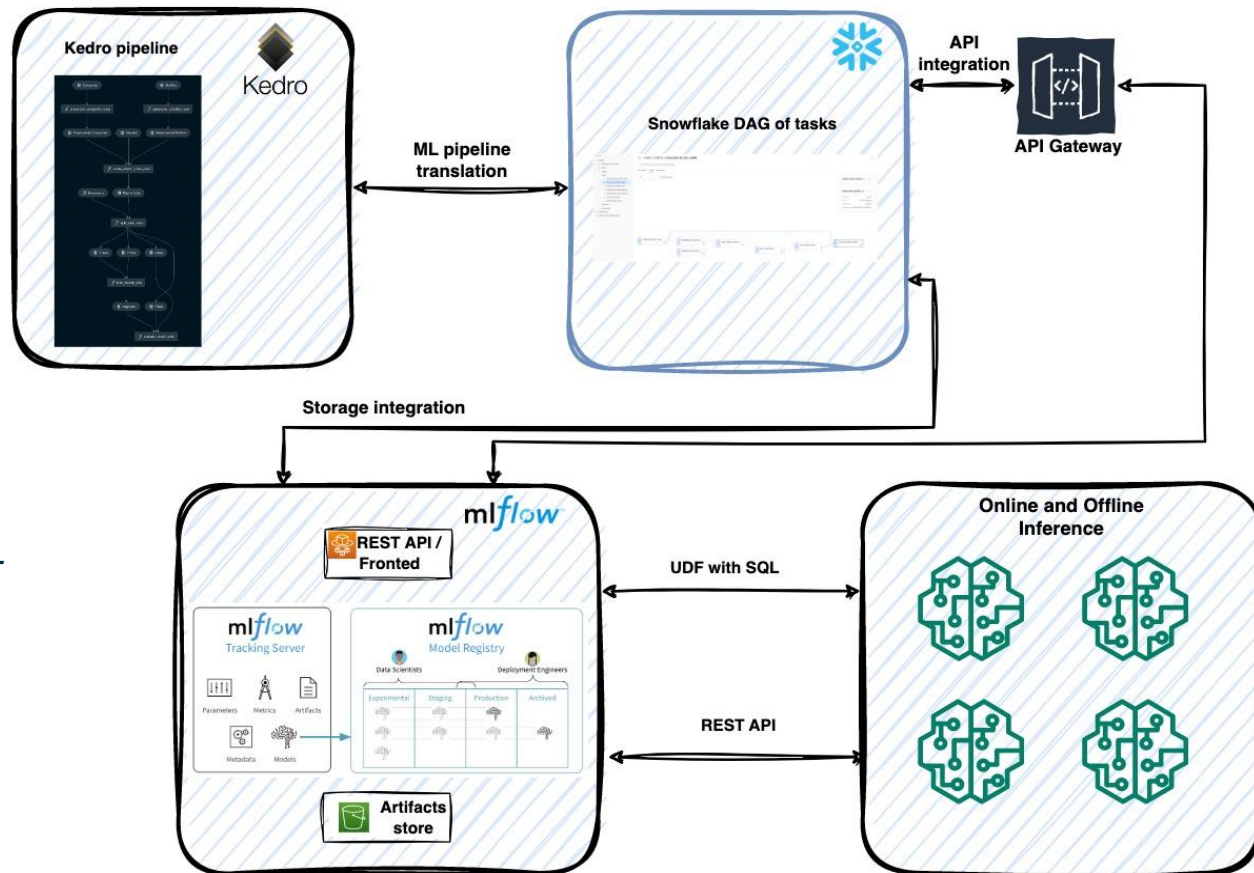
Kedro

MLflow

mlflow

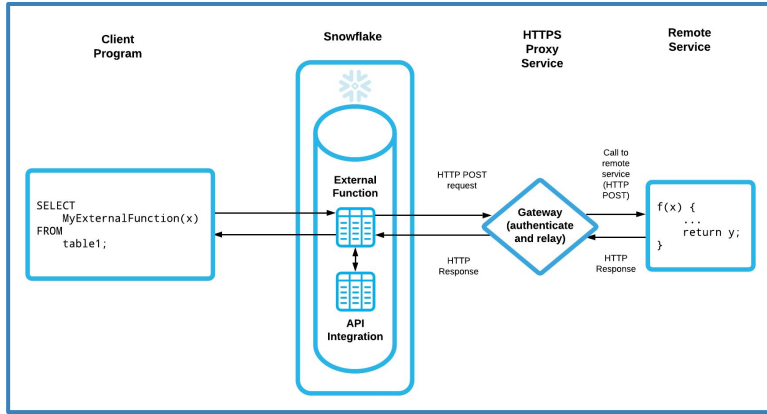
Putting it all together...

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

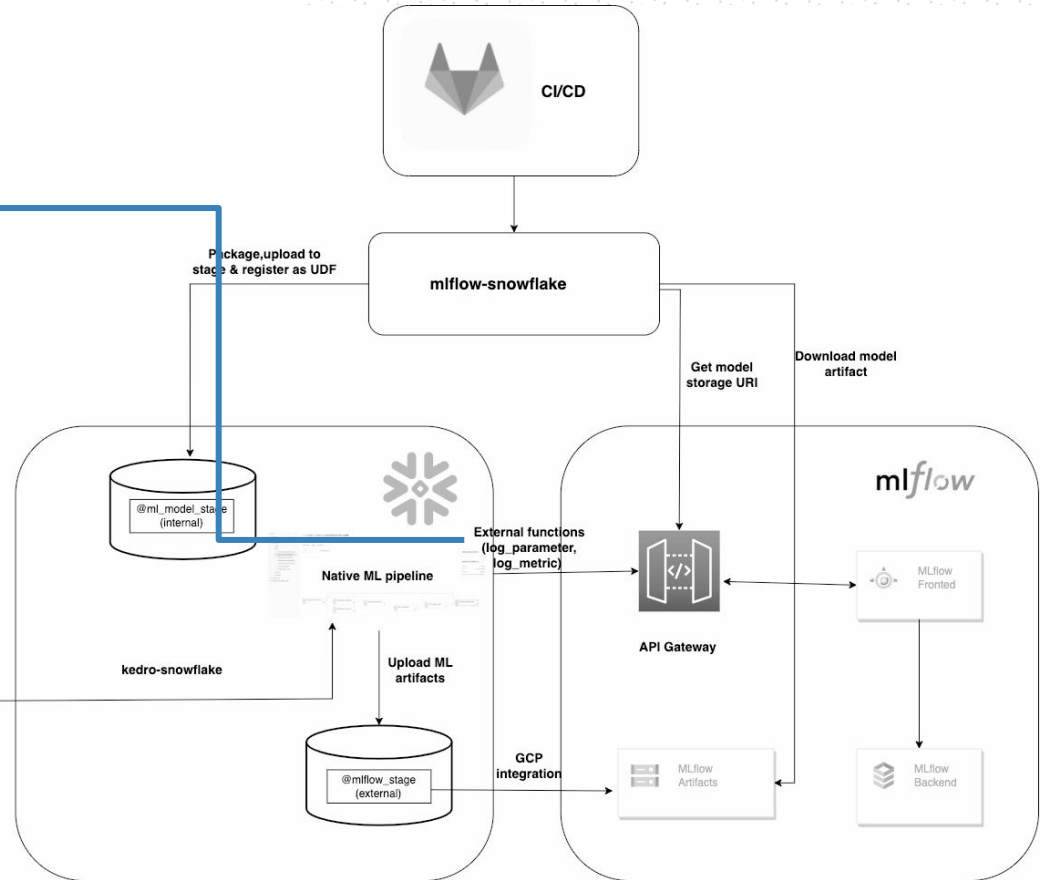




MLOps Platform - MLflow integration



- Snowflake external functions
- Support for AWS, Azure and GCP Gateways
- Snowpark External Access in PuPr this year





External functions wrappers

- 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 }
```



External functions mappings

- Functions mappings
- Kedro hooks

```
77 {% - if cookiecutter.enable_mlflow_integration|lower != "fa
78 # EXPERIMENTAL: Either MLflow experiment name to enable ML
79 # or leave empty
80 mlflow:
81   experiment_name: Default
82   stage: "@MLFLOW_STAGE"
83   # Snowflake external functions needed for calling MLflow instance
84   functions:
85     experiment_get_by_name: {{ cookiecutter.snowflake_database | lower }}.{{ cookiecutter.snowflake_schema | lower }}.mlflow_experiment_get_by_name
86     run_create: {{ cookiecutter.snowflake_database | lower }}.{{ cookiecutter.snowflake_schema | lower }}.mlflow_run_create
87     run_update: {{ cookiecutter.snowflake_database | lower }}.{{ cookiecutter.snowflake_schema | lower }}.mlflow_run_update
88     run_log_metric: {{ cookiecutter.snowflake_database | lower }}.{{ cookiecutter.snowflake_schema | lower }}.mlflow_run_log_metric
89     run_log_parameter: {{ cookiecutter.snowflake_database | lower }}.{{ cookiecutter.snowflake_schema | lower }}.mlflow_run_log_parameter
90 {% else %}
```

```
21 {% if cookiecutter.enable_mlflow_integration|lower != "false" %}
22   @hook_impl
23   def before_node_run(self, node: Node) -> None:
24       if node.name == "export_data_to_snowflake_node":
25           mlflow_helpers.log_params(self.run_params)
26
27   @hook_impl
28   def after_node_run(self, node: Node) -> None:
29       if node.name == "evaluate_model_node":
30           mlflow_helpers.run_update("FINISHED")
31 {% endif %}
```



External functions mappings

- Functions mappings
- Kedro hooks

mlflow 2.2.0 Experiments Models

Default >

peaceful-bird-663

Run ID: cfd3a4d6d3a94f7b886a1649898ca62d Date: 2023-06-23 17:08:15

Duration: 3.0min Status: FINISHED Source: Lifecycle Stage: active

> Description Edit

> Parameters (16)

> Metrics (1)

> Tags

> Artifacts

model

- MLmodel
- conda.yaml
- model.pkl
- python_env.yaml
- requirements.txt

Full Path: mlflow-artifacts/0/cfd3a4d6d3a94f7b886a1649898ca62d/artifacts/model/MLmodel

Size: 1000B

artifact_path: model

flavors:

- python_function:
- env:
- conda: conda.yaml
- virtualenv: python_env.yaml
- loader_module: mlflow.sklearn
- model_module: model.pkl
- predict_fn: predict
- python_version: 3.8.16

sklearn:

- code: null
- pickled_model: model.pkl
- serialization_format: cloudpickle
- sklearn_version: 1.2.2

mlflow_version: 2.0.1

model_uuid: 337fe4299d3d449e5a746019da33ab418

default-dot-oid-mlops-snowflake-platform-aws-r-awsops.com/!experiments/0/3cdd7f447e91c5bc0d51ecac035

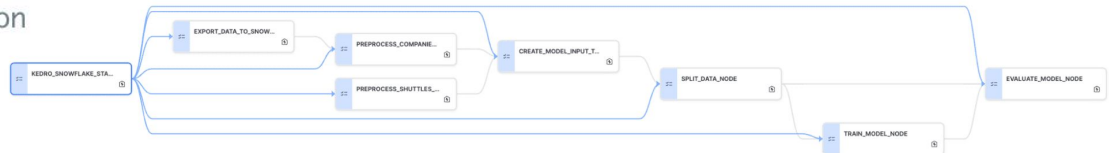
```
21 {% if cookiecutter.enable_mlflow_integration|lower != "false" %}
22 @hook_impl
23 def before_node_run(self, node: Node) -> None:
24     if node.name == "export_data_to_snowflake_node":
25         mlflow_helpers.log_params(self.run_params)
26
27 @hook_impl
28 def after_node_run(self, node: Node) -> None:
29     if node.name == "evaluate_model_node":
30         mlflow_helpers.run_update("FINISHED")
31 {% endif %}
```



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
```



Alternative approaches

Write once - run (almost) everywhere



Kedro



Kedro Vertex AI (GCP)

github.com/getindata/kedro-vertexai



Kedro Sagemaker (AWS)

github.com/getindata/kedro-sagemaker



Kedro Airflow (Kubernetes)

github.com/getindata/kedro-airflow-k8s



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

Amazon SageMaker Studio

File Edit View Run Kernel Git default-1669297364894 / Personal

Launcher Pipelines mlops-housing-demo execution-1672403121591

less than 20 seconds ago

execution-1672403121591

Status **30/12/2022, 13:25** Started time **23m10s** Elapsed time **Stop**

Graph Parameters Settings

Search for step...

```

    graph TD
      A(select_features_node) --> B(preprocess_data_node)
      B --> C(split_data_node)
      C --> D(prepare_features_node)
      C --> E(train_model_node)
      D --> F(evaluate_model_node)
      E --> F
  
```

152%

Simple

execution-16724031

Microsoft Azure Machine Learning

Search within your workspace

getindata.com > mlops-sandbox > Jobs > xebia-demo > _default_

Refresh Clone Resubmit Publish Schedule Show lineage Delete Cancel

default Completed Job overview

```

    graph TD
      A(select_features_node) --> B(preprocess_data_node)
      B --> C(split_data_node)
      C --> D(prepare_features_node)
      C --> E(train_model_node)
      D --> F(evaluate_model_node)
      E --> F
      G(feature_transformer) --> C
  
```

feature_transformer

Google Cloud

gid-mlops-sandbox

mlops-cloud-agnostic-demo-20221230123318

CLONE STOP DELETE

Runtime Graph 7/7 steps completed Expand Artifacts 100%

```

    graph TD
      A(miflow-start-run) --> B(select_features_node)
      A --> C(preprocess_data_node)
      A --> D(split_data_node)
      A --> E(prepare_features_node)
      A --> F(train_model_node)
      A --> G(evaluate_model_node)
      B --> C
      C --> D
      D --> E
      D --> F
      E --> G
      F --> G
  
```

Logs

MLOps orchestration tools in perspective

Kedro-Snowflake vs.

- simpler security setup
- fewer dependencies on external services
- substantially less data transfers
- a unified data and machine learning platform

	Airflow	SageMaker/AzureML/VertexAI	Snowflake/Snowpark
Orchestration	Supported	Supported	Supported
Native data processing	Supported	Supported	Not supported
Docker support	Supported	Supported	Not supported
Native ML capabilities	Supported	Supported	Not supported
Model deployment support (serving)	Supported	Supported	Not supported
Maintenance	High	Low (serverless)	Low / Medium
Extensibility / Customizability	High	Low	Medium
Performance	Depends on setup	Varies	Low to very high
Experiment tracking	External	Built-in	External
GPU support	Supported	Supported	Not supported
Language support	Python + Any (Docker)	Python + Any (Docker)	Python / Java / Scala + SQL
Learning curve	Medium	Medium to High	Low
Unstructured data support	Supported	Supported	Supported
Dataset versioning	Supported	(Azure - yes)	Supported
Open source	Supported	Not supported	Not supported
Dependency management	Docker	Docker	Anaconda / Package upload (Python / Java)
Kedro support	Supported	Supported	Supported
Distributed training support	Not supported	Supported	Not supported

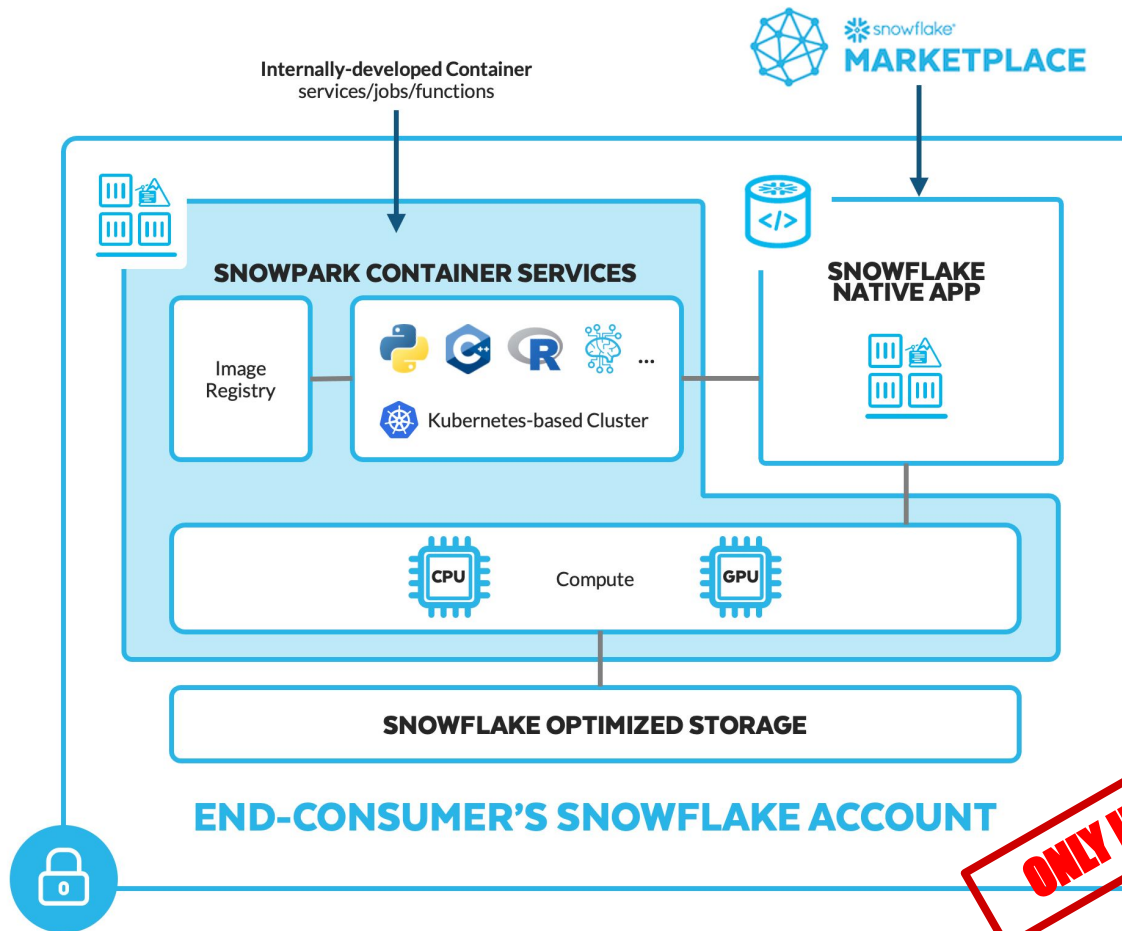
Legend

Supported

Not supported

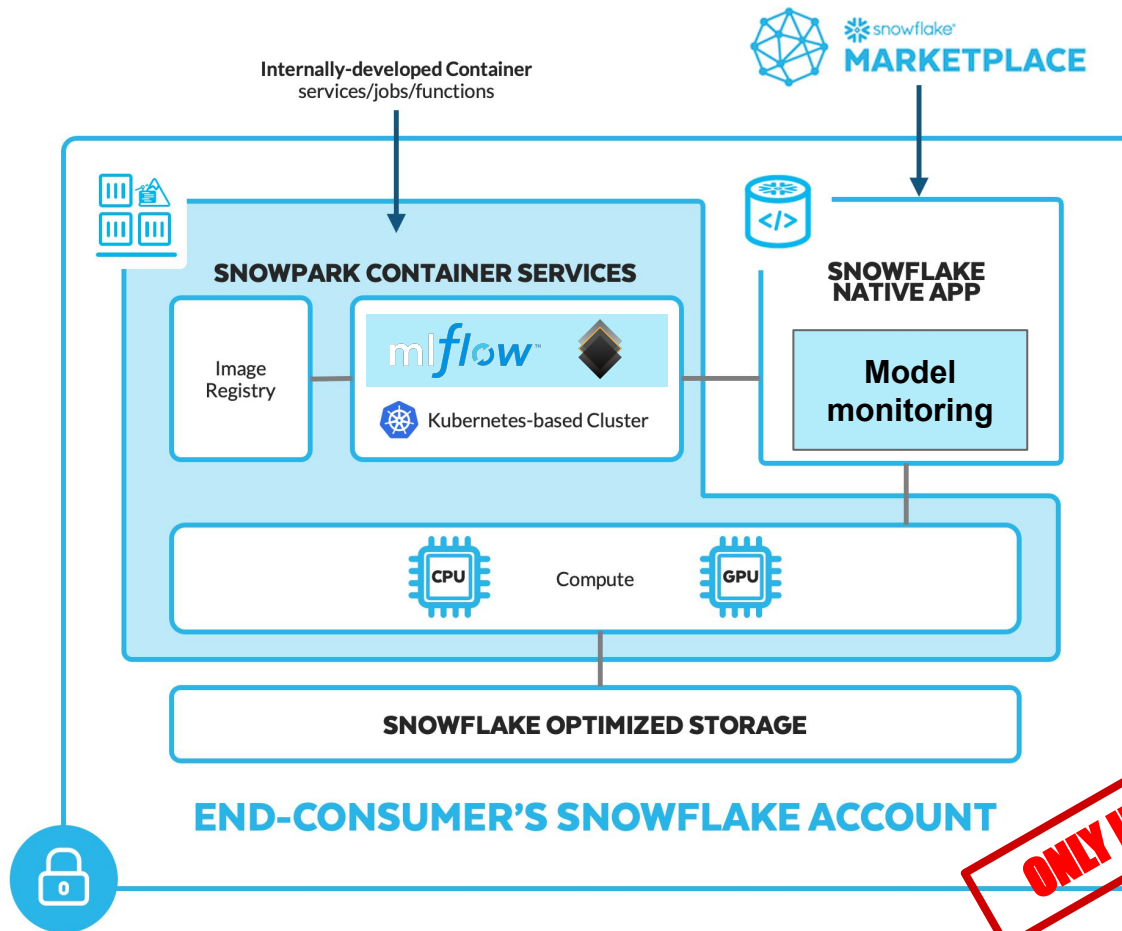
Partially supported

Container services



ONLY IN PRIVATE PREVIEW!

Container services





Demo

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 Snowflake Data Cloud with MLOps capabilities **seamlessly**
- Kedro together with MLflow and Terraform are the main building blocks of **our Snowflake MLOps platform**

- github.com/getindata/kedro-snowflake
- github.com/Snowflake-Labs/mflow-snowflake
- [From 0 to MLOps with ❄️ Snowflake Data Cloud in 3 steps with the Kedro-Snowflake plugin](#)
- [From 0 to MLOps with ❄️ Part 2: Architecting the cloud-agnostic MLOps Platform for Snowflake Data Cloud](#)
- [Running Kedro... everywhere? Machine Learning Pipelines on Kubeflow, Vertex AI, Azure and Airflow](#)

Try it yourself!

1. Install the plugin

```
pip install "kedro-snowflake>=0.1.0"
```

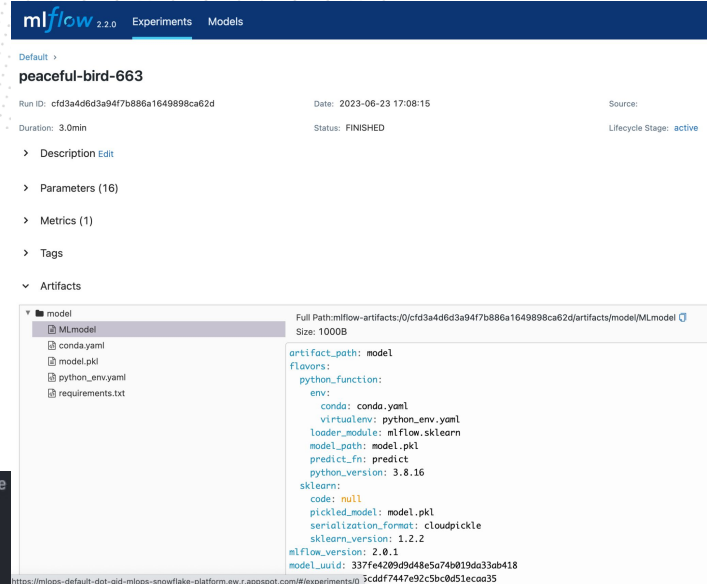
2. Create new project with our Kedro starter :

```
kedro new --starter=snowflights --checkout=master
```

► And answer the interactive prompts  (click to expand)

3. Run the project

```
cd snowflights
kedro snowflake run --wait-for-completion
```



The screenshot shows the mlflow 2.2.0 interface. At the top, there are tabs for 'Experiments' and 'Models'. The main content area displays the details for an experiment named 'peaceful-bird-663'. Key information includes: Run ID: cfd3a4d6d3a94f7b886a1649898ca62d, Date: 2023-06-23 17:08:15, Source: (empty), Duration: 3.0min, Status: FINISHED, and Lifecycle Stage: active. Below this, there are expandable sections for 'Description Edit', 'Parameters (16)', 'Metrics (1)', and 'Tags'. The 'Artifacts' section is expanded, showing a tree view of files under the 'model' directory. The 'MLmodel' file is selected, showing its full path and size (1000B). To the right of the file tree, the artifact's metadata is displayed, including 'artifact_path: model', 'flavors: python_function', 'env: conda: conda.yaml, virtualenv: python_env.yaml', 'loader_module: mlflow.sklearn', 'model_path: model.pkl', 'predict_fn: predict', 'python_version: 3.8.16', 'sklearn: code: null, pickled_model: model.pkl, serialization_format: cloudpickle, sklearn_version: 1.2.2', 'mlflow_version: 2.0.1', and 'model_uuid: 337fe4209d9d48e5a74b019da33ab418'.



Core TEAM



Marek Wiewiórka

Chief Data Architect at Getindata

marek@getindata.com



Marcin Zabłocki

MLOps Architect at Getindata

marcin.zablocki@getindata.com



Michał Bryś

ML Architect at Getindata

michal.brys@getindata.com



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:

