MLOPS: THE MOST IMPORTANT PIECE IN THE ENTERPRISE AI PUZZLE

Francesca Lazzeri, PhD
Principal Cloud Advocate Manager, Microsoft
@frlazzeri

@frlazzeri

When is a ML algorithm becoming AI?

MLOps == How to bring ML to production

Bring together **people**, **process**, and **platform** to automate ML-infused software delivery & provide continuous value to our users.

People

- Blend together the work of individual engineers in a repository.
- Each time you commit, your work is automatically built and tested, and bugs are detected faster.
- Code, data, models and training pipelines are shared to accelerate innovation.

Process

- Provide templates to bootstrap your infrastructure and model development environment, expressed as code.
- Automate the entire process from code commit to production.

Platform

- Safely deliver features to your customers as soon as they're ready.
- Monitor your pipelines, infrastructure and products in production and know when they aren't behaving as expected

How is MLOps different from DevOps?

Data/model versioning != code versioning - how to version data sets as the schema and origin data change

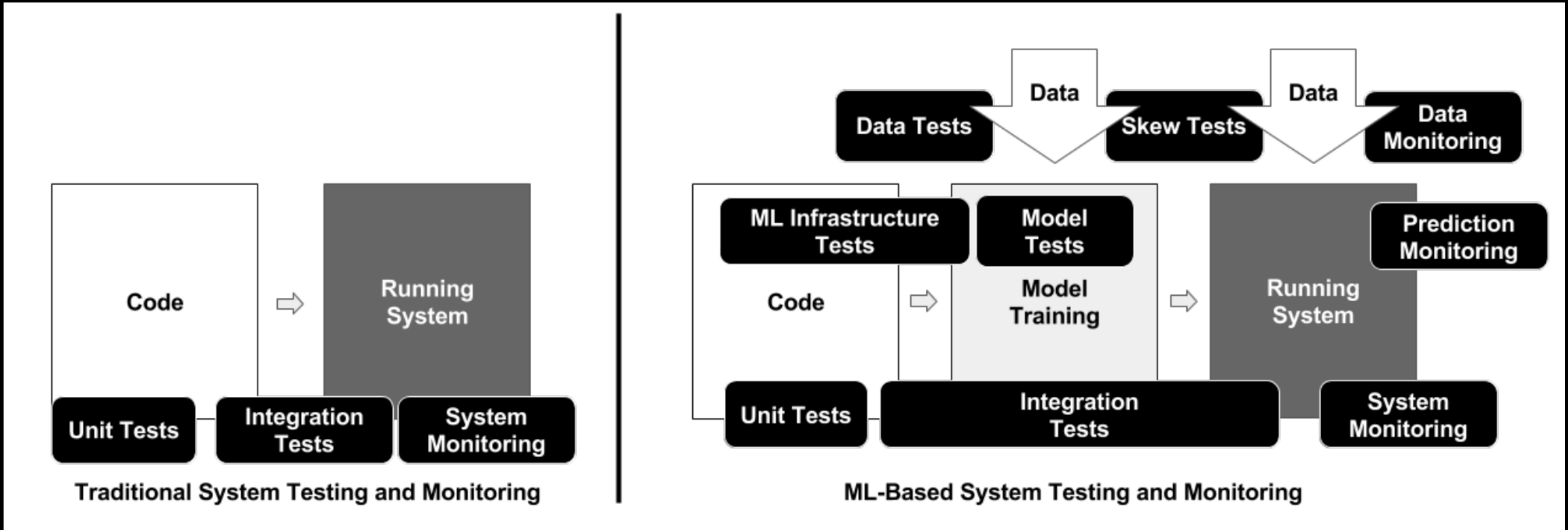
Digital audit trail (lineage) requirements change when dealing with code + data

Model reuse is different than software reuse, as models must be tuned based on input data / scenario. To reuse a model you may need to fine-tune / transfer learn on it (meaning you need the training pipeline)

Model performance tends to decay over time & you need the ability to retrain them on demand to ensure they remain useful in a production context.

@frlazzeri

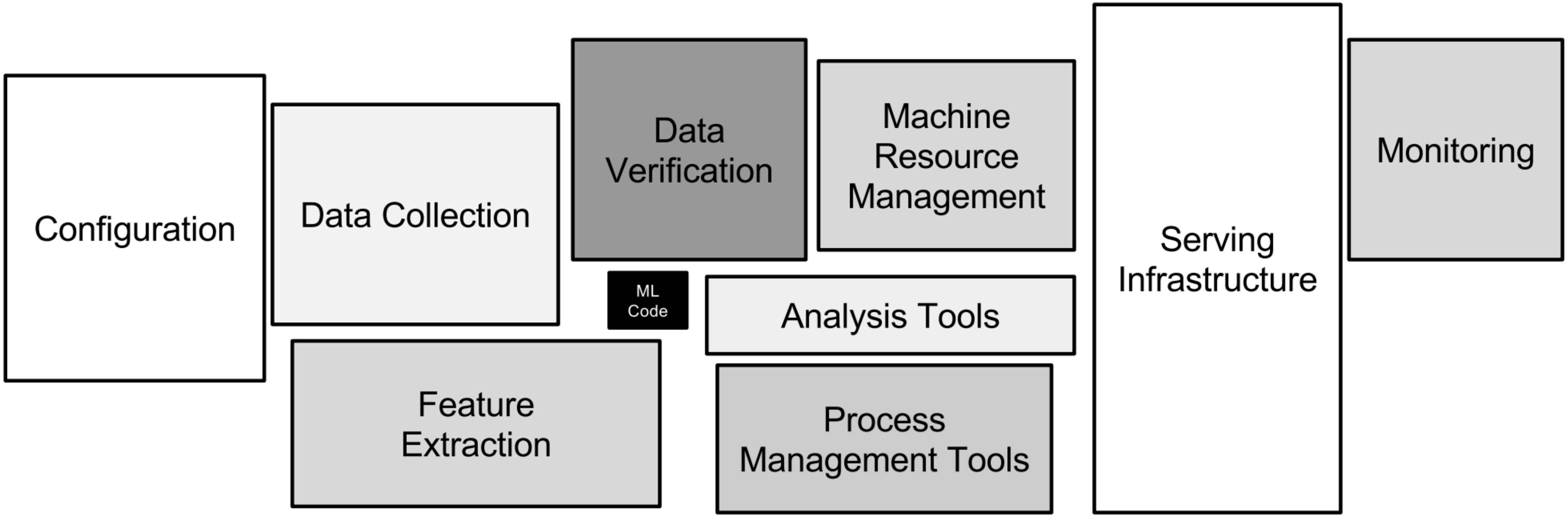
Traditional vs. ML infused systems



ML introduces two new assets into the software development lifecycle – **data** and **models**.

More assets & process to manage

@frlazzeri



Sculley, D.; Holt, Gary; Golovin, Daniel; Davydov, Eugene; Phillips, Todd; Ebner, Dietmar; Chaudhary, Vinay; Young, Michael; Crespo, Jean-Francois; Dennison, Dan (7 December 2015). "Hidden Technical Debt in Machine Learning Systems"

@frlazzeri

Customer pain points

Customer pain	Capability to Address
Hard to deploy a model for inference after I have trained it.	**No-code deployment** for models of common languages and frameworks
Hard to integrate the ML lifecycle into my application lifecycle.	**Production-grade model release** with model validation, multi-stage deployment, controlled rollout
Hard to know how and when to retrain an ML model.	**Model feedback loop with** AB scorecards and drift analysis, integrated with ML pipelines for retraining
Hard to figure out where my model came from and how it's being used.	**Enterprise asset management** with Audit trail, policy + quota management

So… how do we implement MLOps in the real world?

There are many jobs & tools involved in production ML

Data Scientist

- Azure Machine Learning
- GitHub
- TensorFlow, PyTorch, sklearn
- Azure Compute – CPU/GPU/FPGA

IT / Ops

Data Analyst

Business Owner

& many more...

Data Engineer

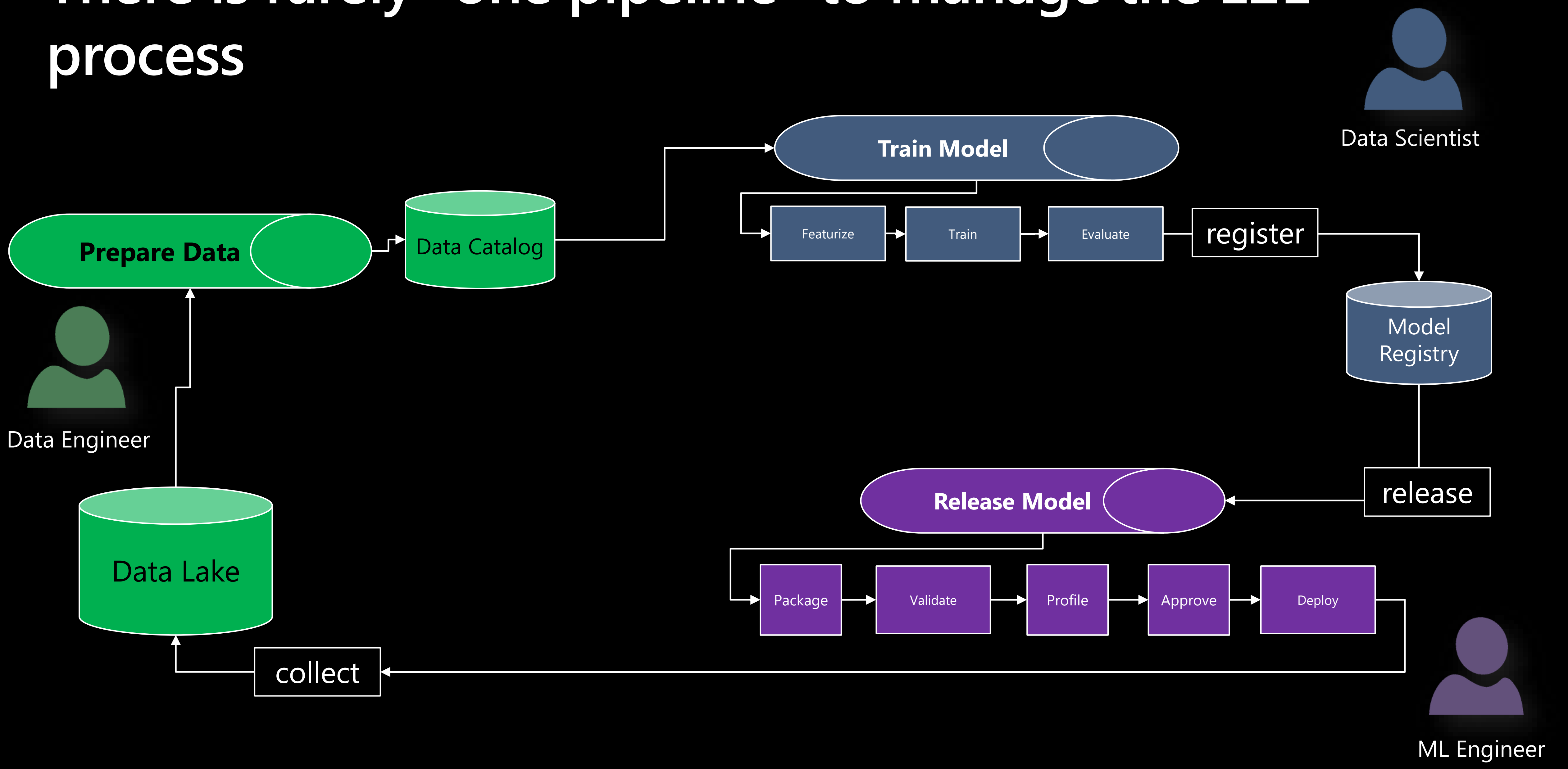
- Azure Data Lake
- Azure Data Factory
- Azure DataBricks
- Azure SQL

ML Engineer

- Azure DevOps
- GitHub
- Azure Kubernetes Service
- Azure IoT Edge
- Azure Monitor

@frlazzeri

There is rarely "one pipeline" to manage the E2E process



MLOps – Process Maturity Model

@frlazzeri

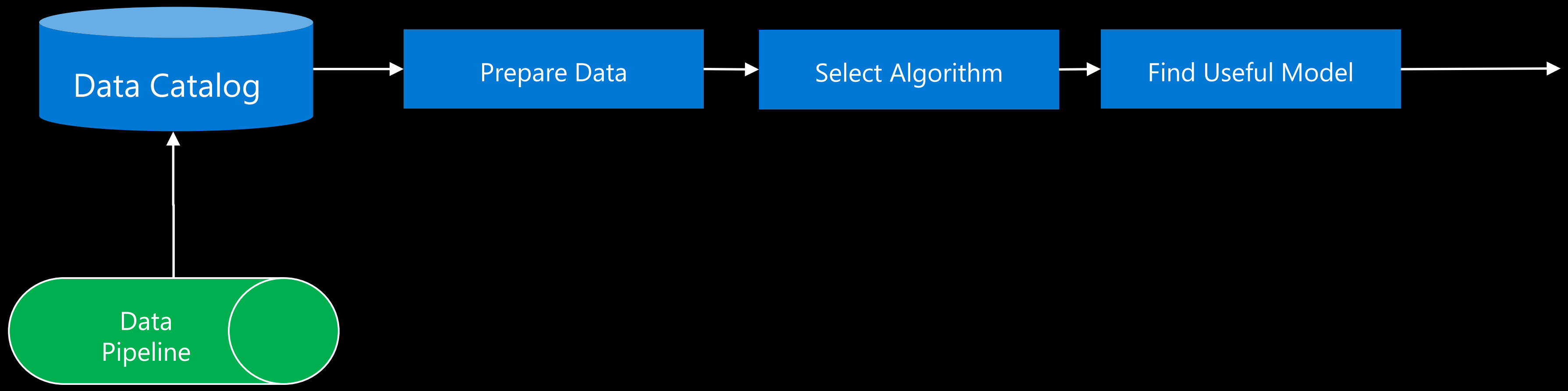
Maturity Level	People	Model Creation	Model Release	Application Integration	Technology
Level 1 - No MLOps	• Data Scientists - silo'd, not in regular comms with larger team • Data Engineers - silo'd (if exists), not in regular comms with larger team • Software Engineers - Silo'd, receive model "over the wall"	• Data pipeline gathers data automatically • Compute may or may not be managed • Experiments are not predictably tracked • End result may be a single file manually handed off (model), with inputs/outputs	• Manual process • Scoring script may be manually created well after experiments, likely version controlled • Is handed off to Software Engineers	• Basic integration tests exist for the model • Heavily reliant on Data Scientist expertise to implement model • Releases are automated • Application code has unit tests	• Automated Builds • Automated Tests for Application code • Manual model training • No centralized tracking of model performance
Level 2 - Automated Training	• Data Scientists - Working directly with Data Engineers to convert experimentation code into repeatable scripts/jobs • Data Engineers - Working with Data Scientists • Software Engineers - Silo'd, receive model "over the wall"	• Data pipeline gathers data automatically • Compute is managed • Experiment results are tracked • Both training code and resulting models are version controlled	• Manual Release • Scoring Script is version controlled with tests • Release is managed by Software engineering team	• Basic integration tests exist for the model • Heavily reliant on Data Scientist expertise to implement model • Application code has unit tests	• Automated Builds • Automated Tests for Application code • Automated model training • Centralized tracking of model training performance • Model Management
Level 3 - Automated Model Deployment	• Data Scientists - Working directly with Data Engineers to convert experimentation code into repeatable scripts/jobs • Data Engineers - Working with Data Scientists and Software Engineers to manage inputs/outputs • Software Engineers - Working with Data Engineers to automate model integration into application code	• Data pipeline gathers data automatically • Compute is managed • Experiment results are tracked • Both training code and resulting models are version controlled	• Automatic Release • Scoring Script is version controlled with tests • Release is managed by CI/CD pipeline	• Unit and Integration tests for each model release • Less reliant on Data Scientist expertise to implement model • Application code has unit/integration tests	• Automated Builds • Integrated A/B testing of model performance for deployment • Automated Tests for All code • Automated model training • Centralized tracking of model training performance • Model Management
Level 4 - Automated Retraining (full MLOps)	• Data Scientists - Working directly with Data Engineers to convert experimentation code into repeatable scripts/jobs. Working with Software Engineers to identify markers for retraining • Data Engineers - Working with Data Scientists and Software Engineers to manage inputs/outputs • Software Engineers - Working with Data Engineers to automate model integration into application code. Implementing metrics gathering post-deployment	• Data pipeline gathers data automatically • Retraining triggered automatically based on production metrics • Compute is managed • Experiment results are tracked • Both training code and resulting models are version controlled	• Automatic Release • Scoring Script is version controlled with tests • Release is managed by CI/CD pipeline	• Unit and Integration tests for each model release • Less reliant on Data Scientist expertise to implement model • Application code has unit/integration tests	• Automated Builds • Integrated A/B testing of model performance for deployment • Automated Tests for All code • Automated model training and testing • Centralized tracking of model training performance • Model Management • Verbose, centralized metrics from deployed model

@frlazzeri

Level 1 – No MLOps

Interactive, exploratory, get to something useful.

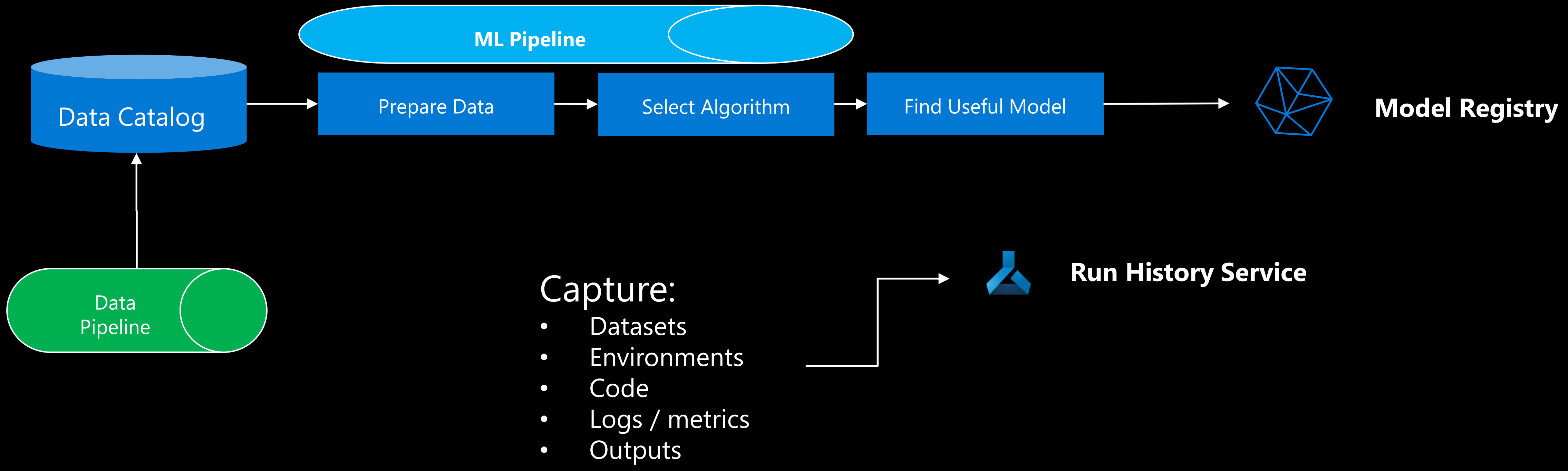
Data Scientist



What’s next?

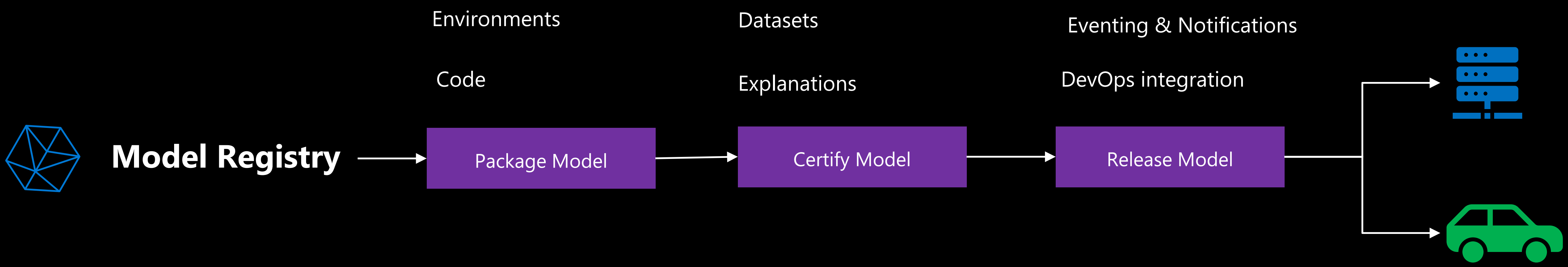
Level 2 – Reproducible Model Training

Version code, data, ensure model can be recreated.



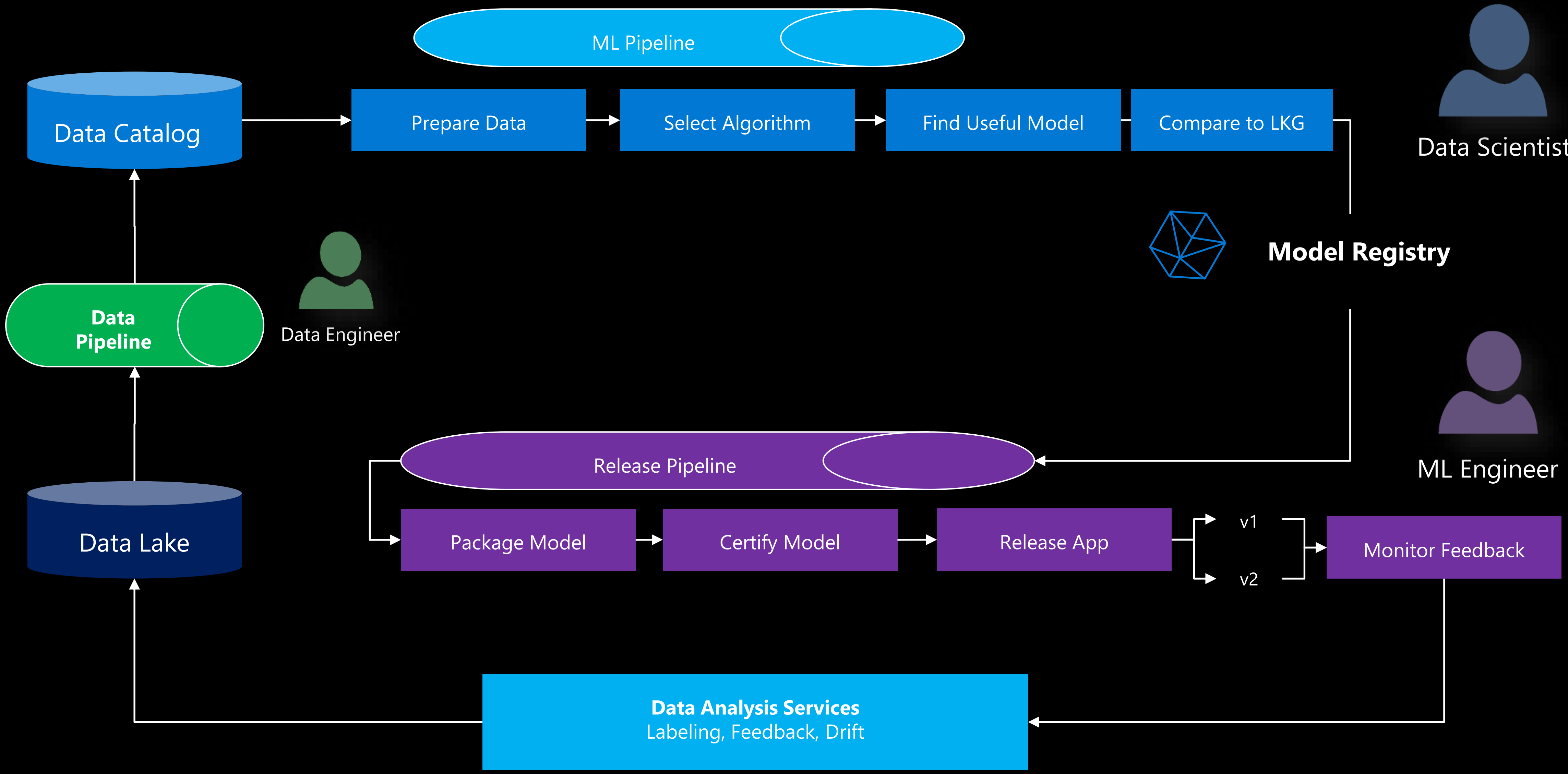
Level 3 – Automated Model Deployment

Package, certify, deploy



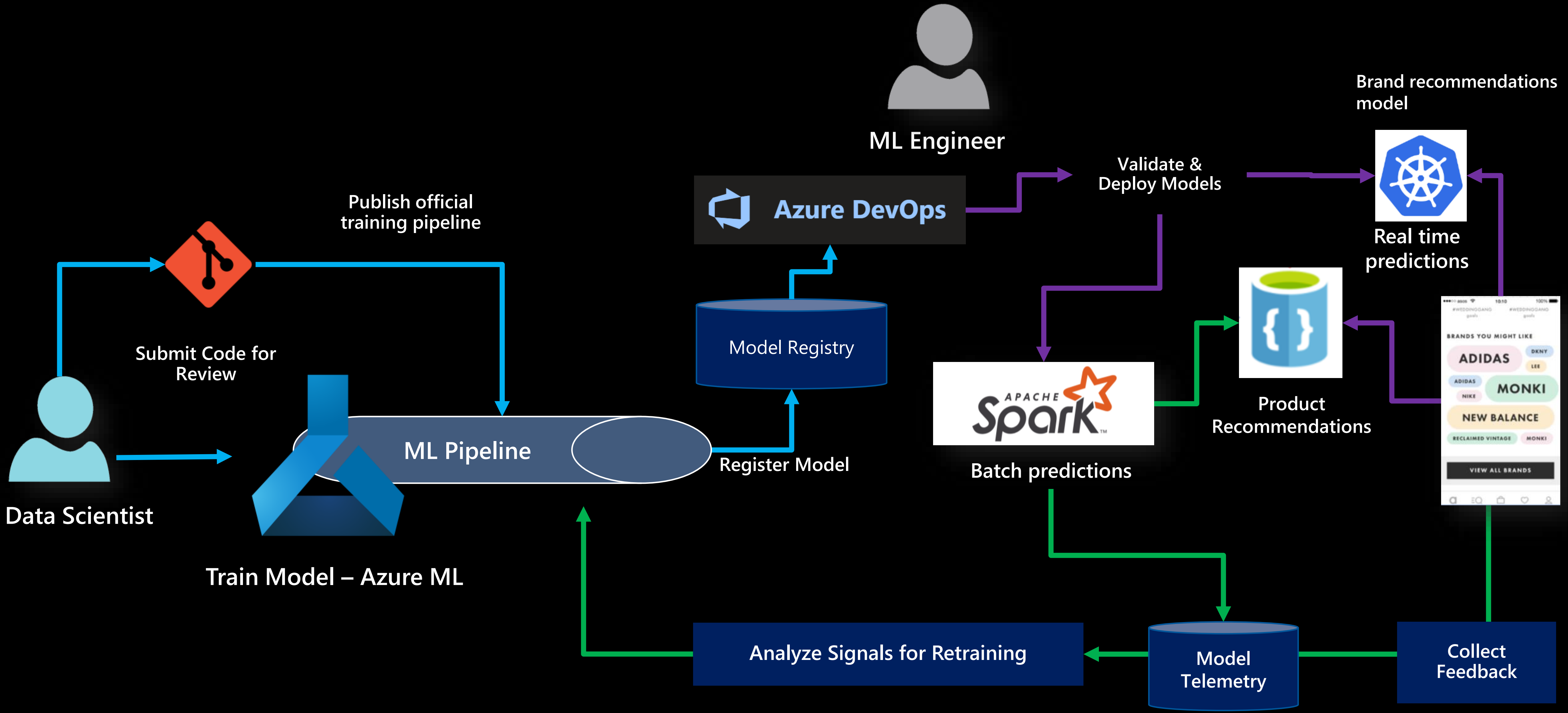
Level 4 – Automated E2E ML Lifecycle

@frlazzeri



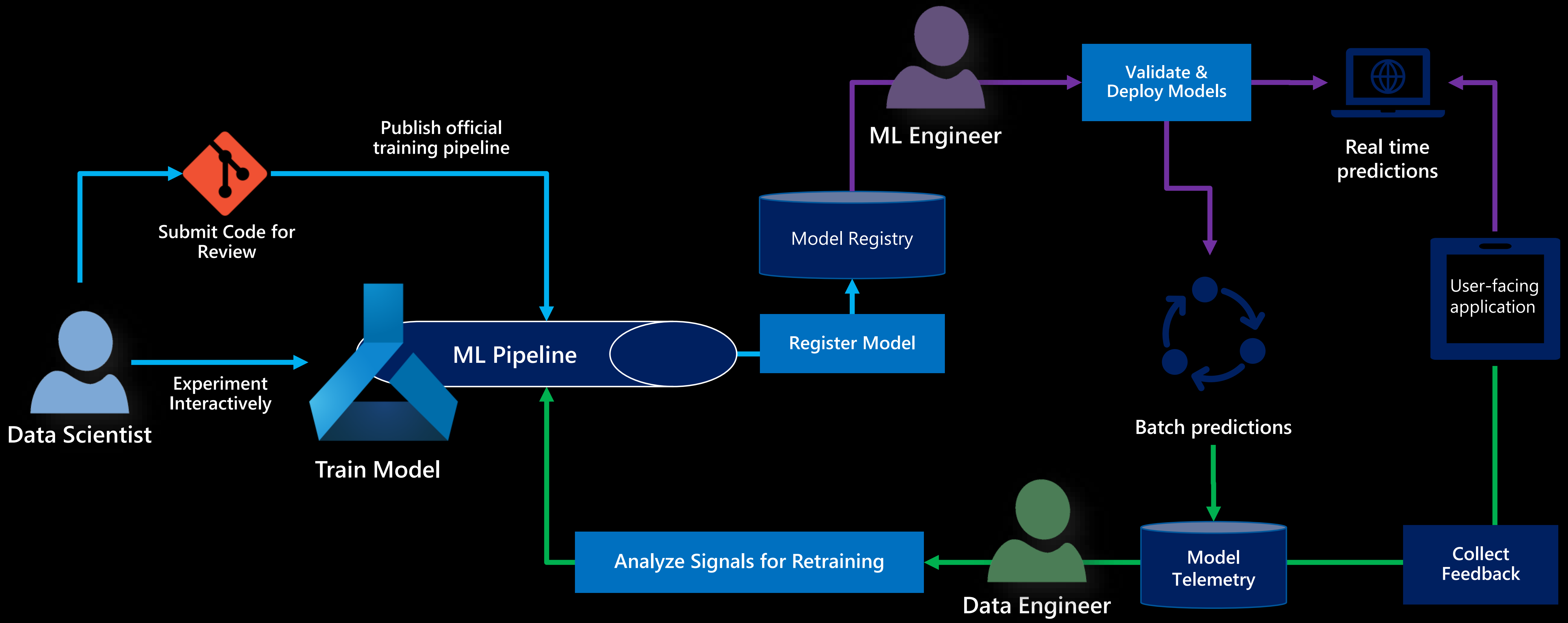
Real world Examples

Leveraging MLOps to ship recommender systems.



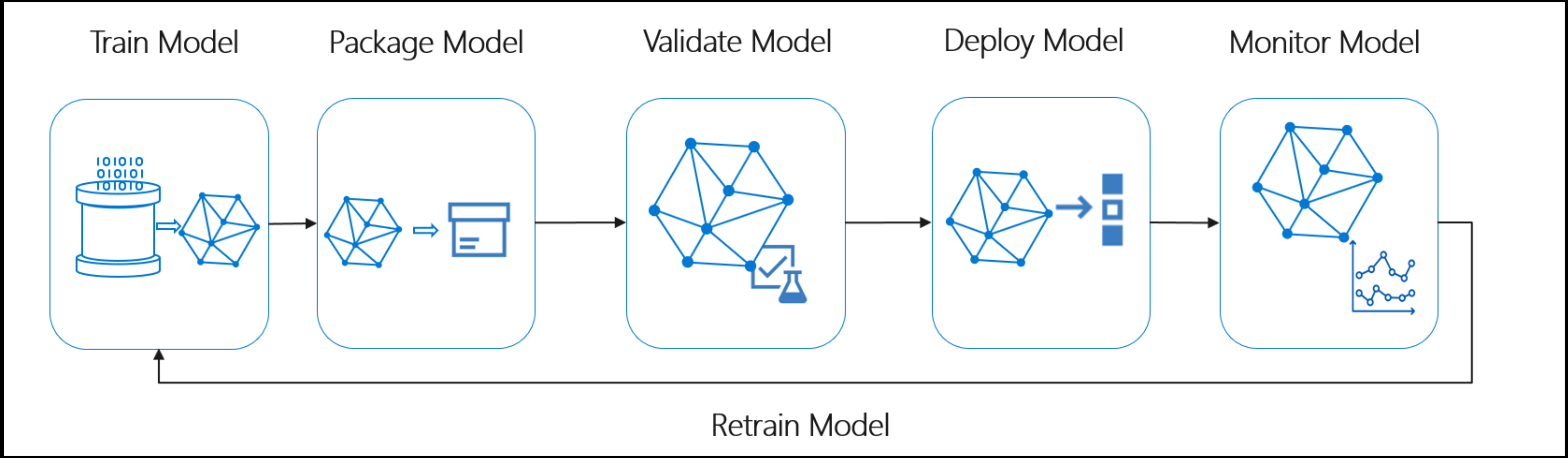
https://github.com/Microsoft/Recommenders

Generalized MLOps process



Azure Machine Learning MLOps Features

How does Azure ML help with MLOps?



@frlazzeri

Azure Machine Learning

Experience
SDK, Notebooks, Drag-n-drop, Wizard

MLOps
Reproducible, Reusable, Automatable, Git, CLI, REST

Dataset Management
Profiling, Drift, Labeling

Model Training
Experiments, Runs

Model Management
Models

Model Serving
Batch, Realtime

Compute
Jobs, Clusters, Instances

Orchestration
Security, Mgmt, Deployment

Cloud
CPU, GPU, FPGA

Edge
CPU, GPU, NPU

Dataset management & versioning

Track tabular data and file data

Easily import / export across language boundaries

Sample usage

```python
from azureml.core import Workspace, Dataset

subscription_id = '15ae9cb6-95c1-483d-a0e3-b1a1a3b06324'
resource_group = 'ignite'
workspace_name = 'ignite'

workspace = Workspace(subscription_id, resource_group, workspace_name)

dataset = Dataset.get_by_name(workspace, name='NYC_Taxi')
dataset.to_pandas_dataframe()
```

Datasets

Registered datasets | Dataset monitors

+ Create dataset | Refresh

Name	Version	Created on	Modified on	Properties	Tags
NYC_Taxi	2	Oct 28, 2019 3:09 PM	Oct 28, 2019 3:12 PM	Tabular, Ti...	opendatasets: nyc_tlc_yellow
ChicagoSafety	1	Oct 6, 2019 6:18 AM	Oct 7, 2019 12:07 PM	Tabular, Ti...	opendatasets: city_safety_chicago
IBM-Employee-Attrition	2	Oct 3, 2019 5:37 AM	Oct 31, 2019 5:06 AM	Tabular	
MNIST-web	1	Oct 2, 2019 9:23 PM	Oct 2, 2019 9:23 PM	File	
MNIST	1	Oct 2, 2019 5:05 AM	Oct 2, 2019 5:05 AM	File	

Prev Next

data4ml-demo > Datasets > florida-weather-2010

florida-weather-2010 Version 1 (latest)

Details | Explore | Models

Refresh | Generate profile | Unregister | New version

Preview | Profile

# of Columns: 23 # of Rows: 47040

Column	Profile	Type	Min	Max	Count	Missing count	Empty count	Error count	Mean	Std dev...
usaf		Integer	13170	999999	47040	0	0	0	710537.81	353958.24
wban		Integer	12894	99999	47040	0	0	0	76299.15	35772.07
datetime	No graph data.	Date	2010-01-01T...	2011-01-01T...	47040	0	0	0		
latitude		Decimal	-34.07	60.38	47040	0	0	0	18.97	31.38
longitude		Decimal	-85.79	5.33	47040	0	0	0	-62.01	33.51
elevation		Decimal	3.00	146.00	47040	0	0	0	45.62	52.90
windAngle		Decimal	0.00	360.00	47040	1771	0	0	159.39	112.90

Declarative ML pipelines

Define training pipeline declaratively

Easy to diff / compare

16 lines (15 sloc) | 548 Bytes

```yaml
pipeline:
    name: SamplePipelineForTraining
    steps:
        TrainStep:
            python_script_step:
                name: "PythonScriptStep"
                script_name: "train_explain.py"
                allow_reuse: True
                source_directory: "."
            runconfig: 'aml_config/train.runconfig'
            outputs:
                result:
                    destination: Output
                    datastore: workspaceblobstore
                    type: mount
```

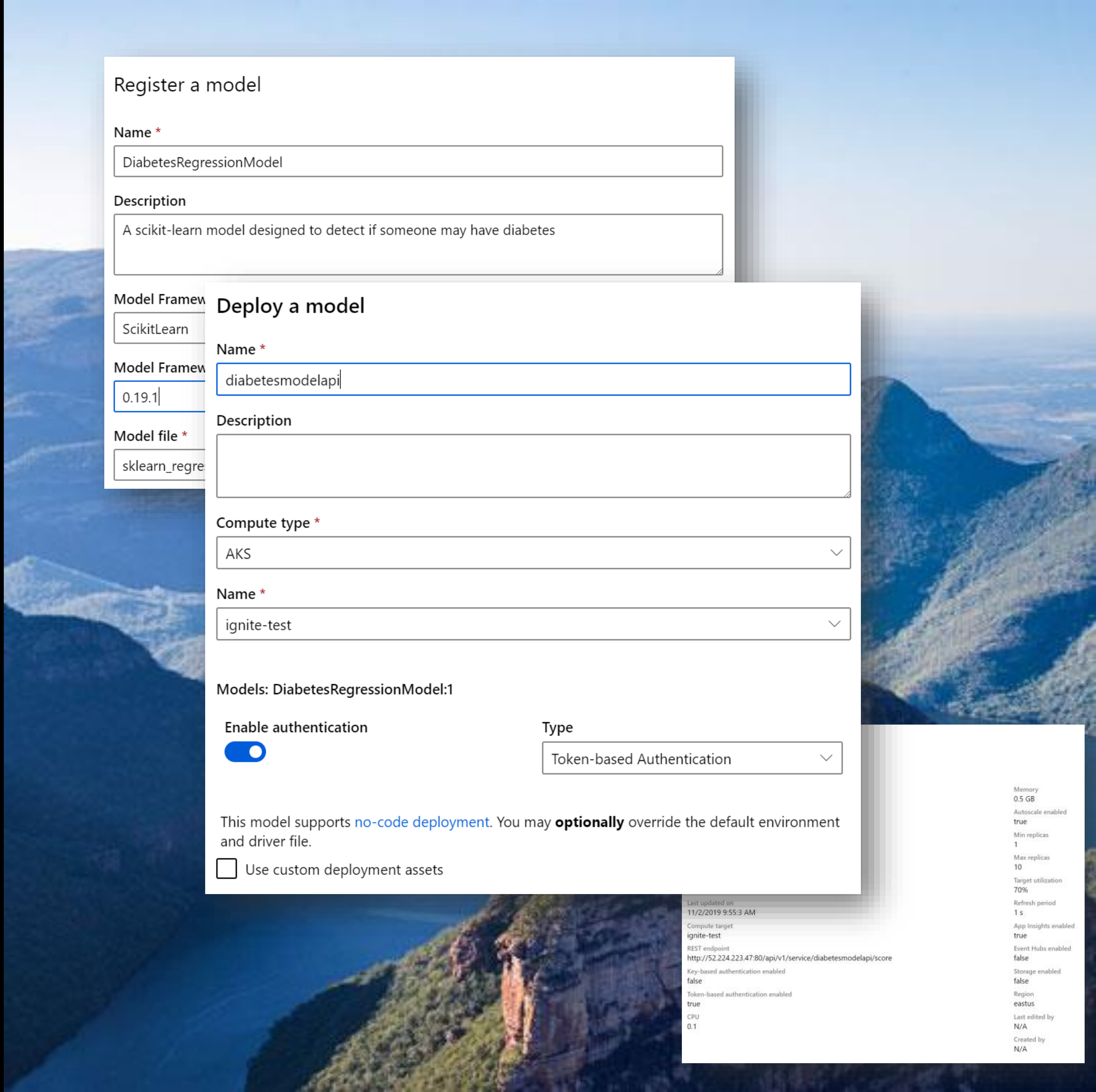
Model management, packaging & deployment

Capture framework / version / resource requirements

Supports no-code deployment

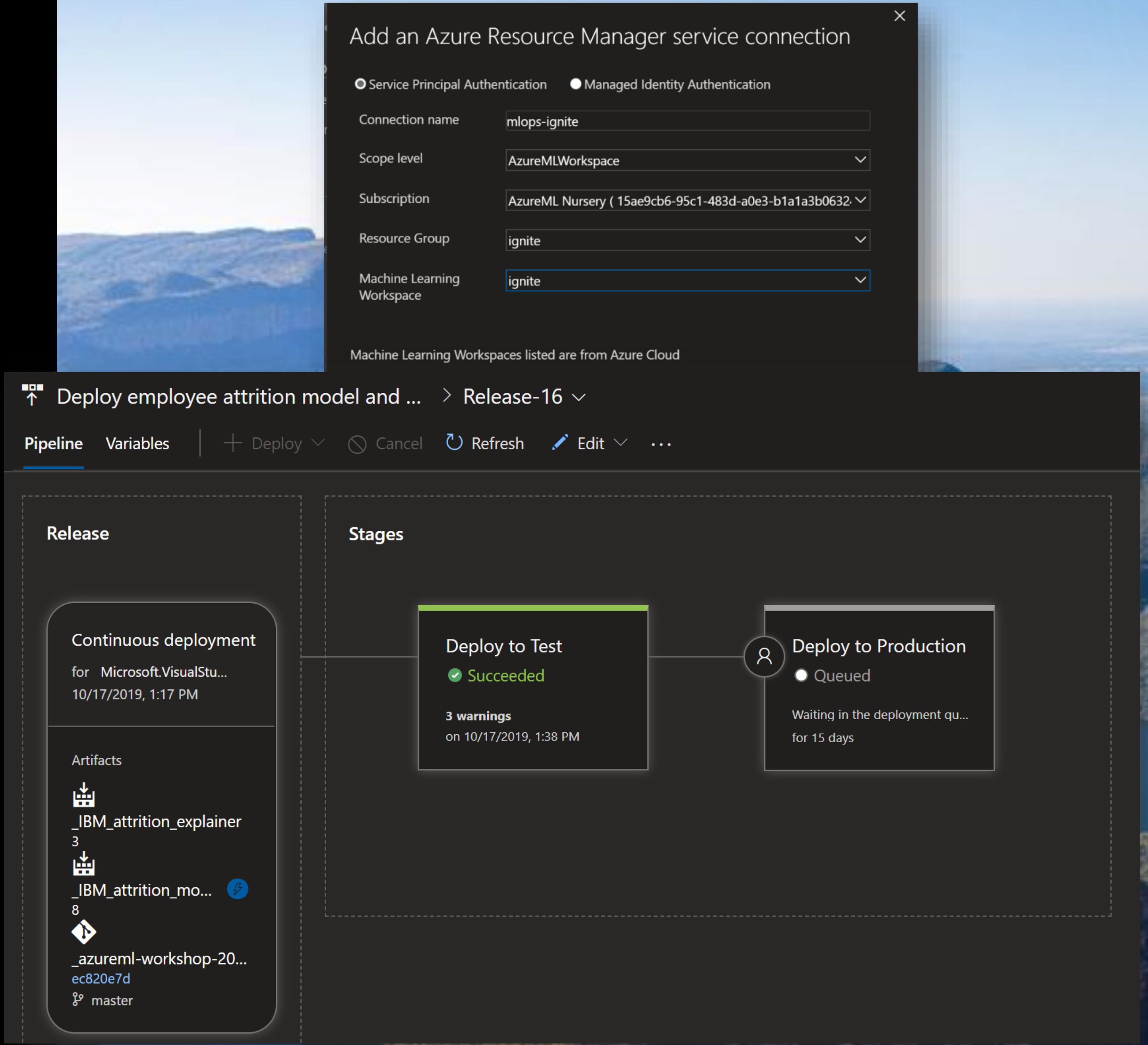
Supported frameworks:

- scikit-learn
- TensorFlow (SavedModel)
- ONNX (all models)



Azure DevOps integration

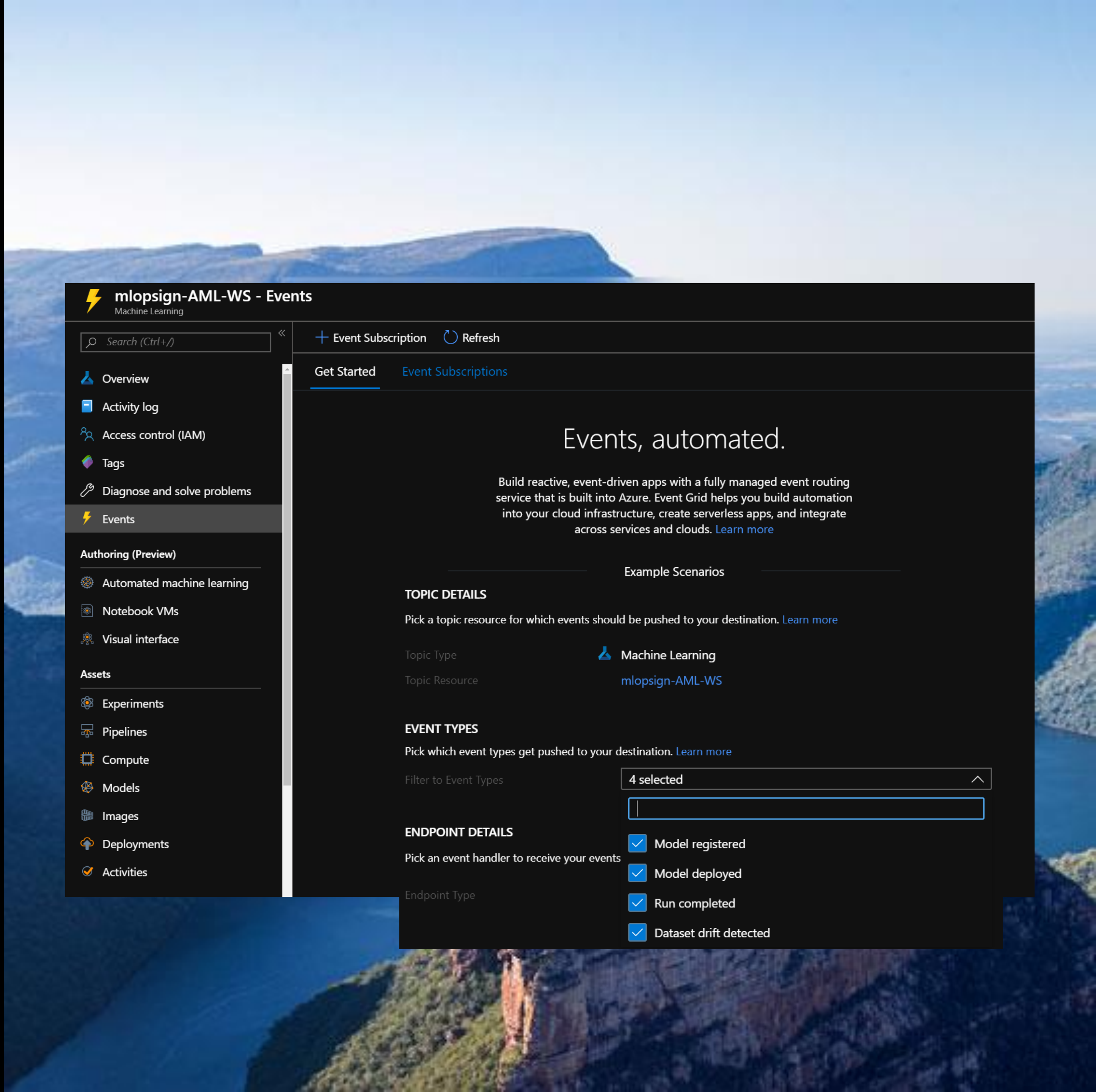
Automate training & deployment into existing release management processes



Azure ML Event Grid integration

Fully managed event routing for all activities in the ML lifecycle

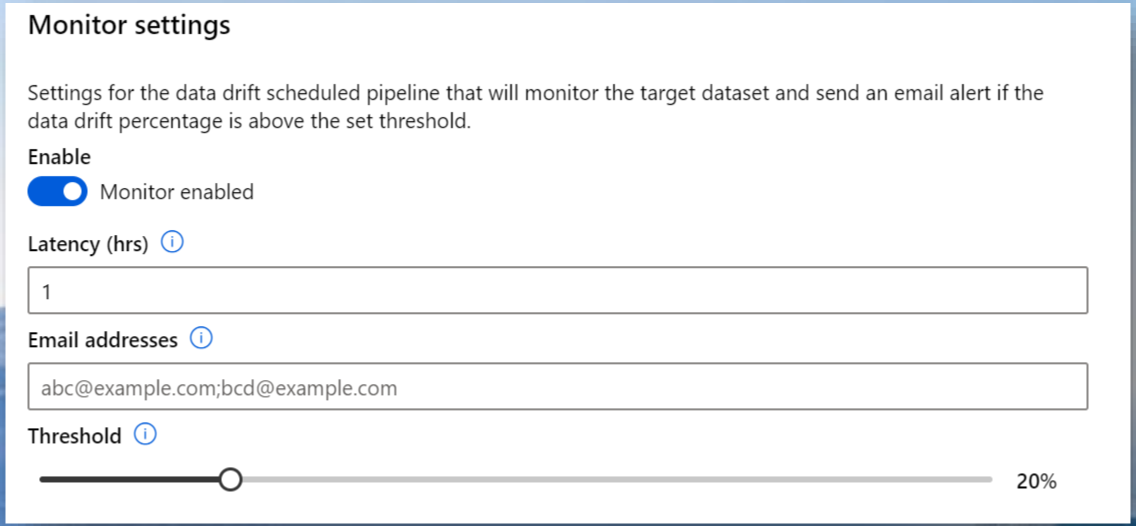
Let's look at some examples…

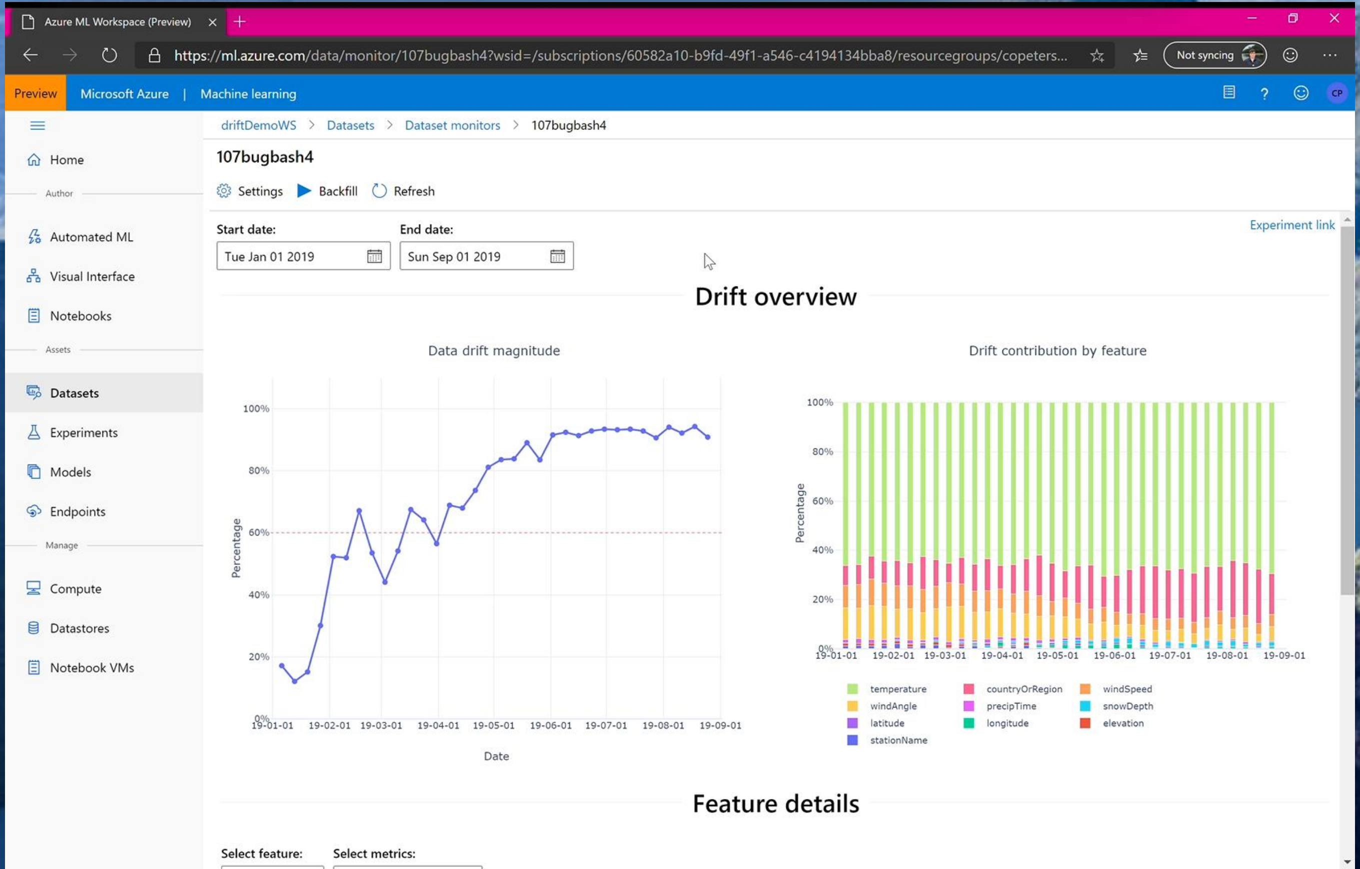


Set up a data drift monitor…

Compare datasets over time

Determine when to take a closer look





Key Takeaways

Better together: ML + DevOps mindset

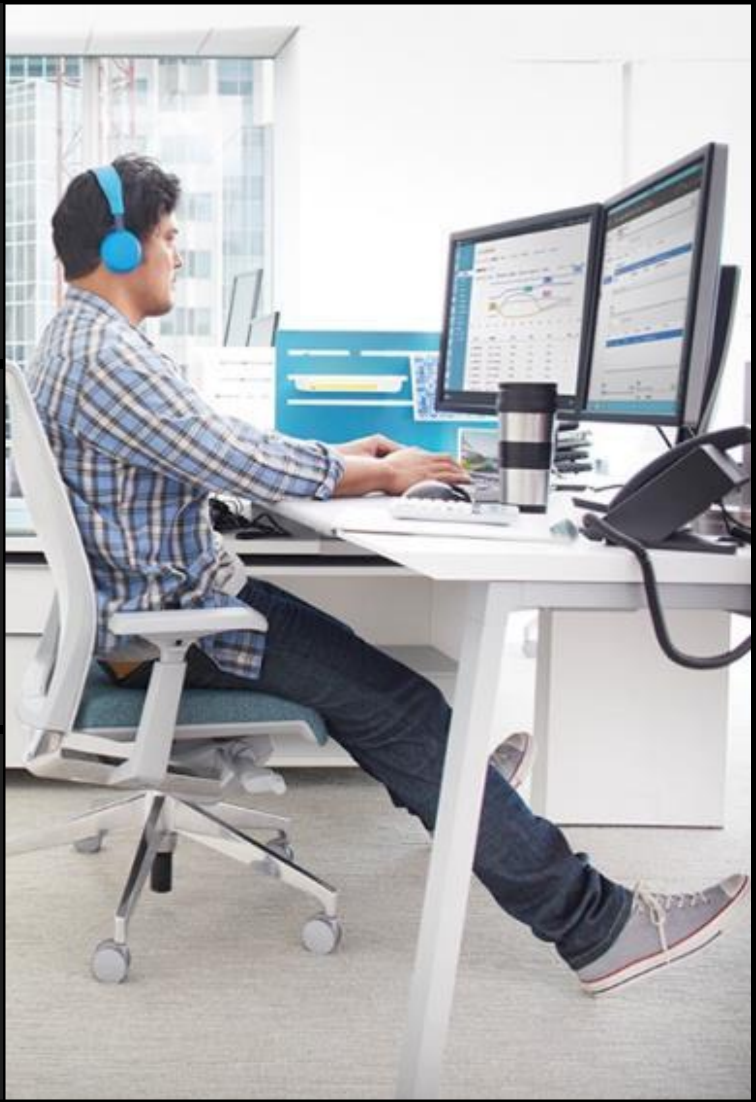
MLOps provides structure for building, deploying and managing and an enterprise-ready AI application lifecycle

MLOps enhances delivery

Adoption will increase the agility, quality and delivery of AI project teams.

More than technology

MLOps is a conversation about people, process and technology
AI principles and practices need to be understood by all roles



@frlazzeri

Learn More

Start Free

Build, train, and deploy models with an Azure free account

https://azure.microsoft.com/free

Documentation

Dig into our technical documentation

https://aka.ms/AzureMLDocs

https://github.com/microsoft/MLOps

Give feedback

Tell us what you think, ask for a feature

https://aka.ms/AzureML_feedback

THANK YOU!

Francesca Lazzeri, PhD
Principal Cloud Advocate Manager, Microsoft
@frlazzeri