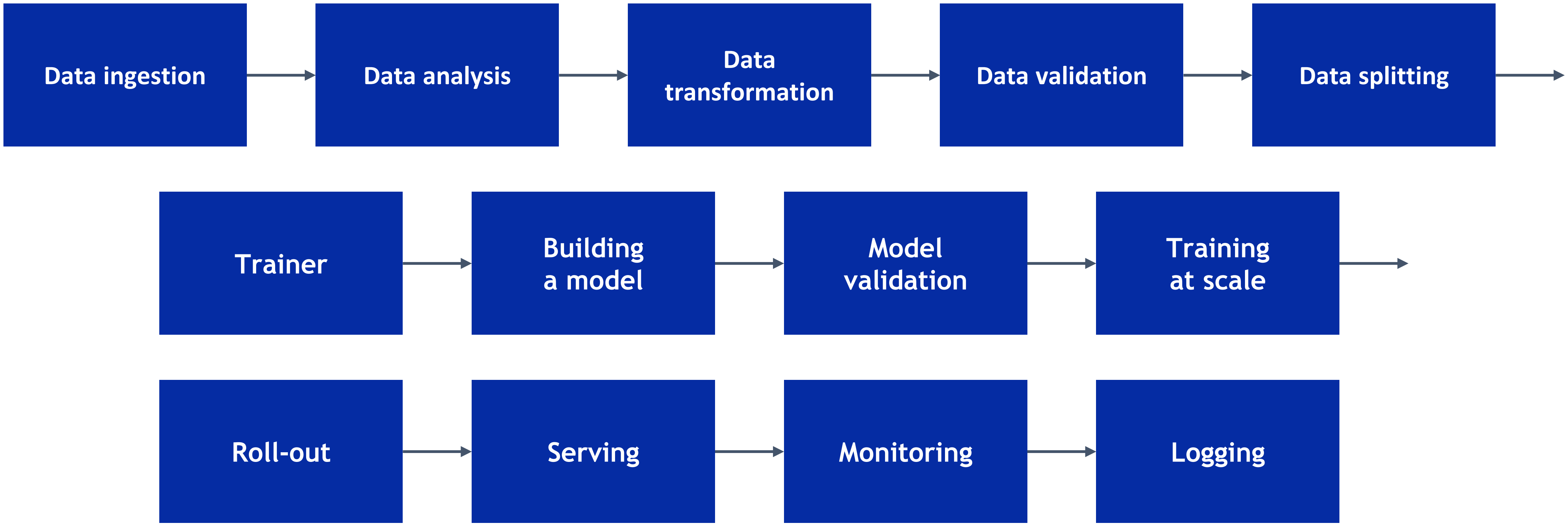
Microsoft

DevOps for ML ("MLOps")

Priyanshi Singh
Cloud Solution Architect
Artificial Intelligence & Machine Learning

Building a model


Data ingestion
Data analysis
Data transformation
Data validation
Data splitting
Trainer
Building a model
Model validation
Training at scale
Roll-out
Serving
Monitoring
Logging

MLOps Best Practices

1. Naming Convention
2. Code Quality Checks (Unit Testing?)
3. Experiments – Yes Track your Experiments!
4. Data Validation
5. Model Validation across Segments
6. Resource Utilization : yes Experiments cost Money
7. Monitor Predictive Service Performance
8. Open Communication Lines are Important

Source: https://neptune.ai/blog/mlops-10-best-practices

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

101010
010101
101010

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

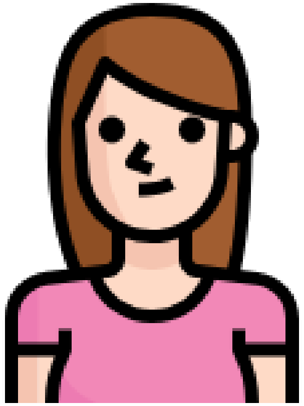
Platform

- 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

Ok, but, like, I’m
a data scientist. ~~IDGAF~~
I don’t care
about all that.

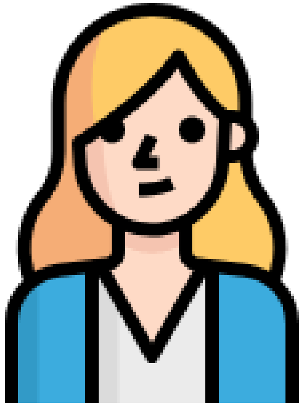
Yes You Do!

Cowboys and Ranchers Can Be Friends!



Data Scientist

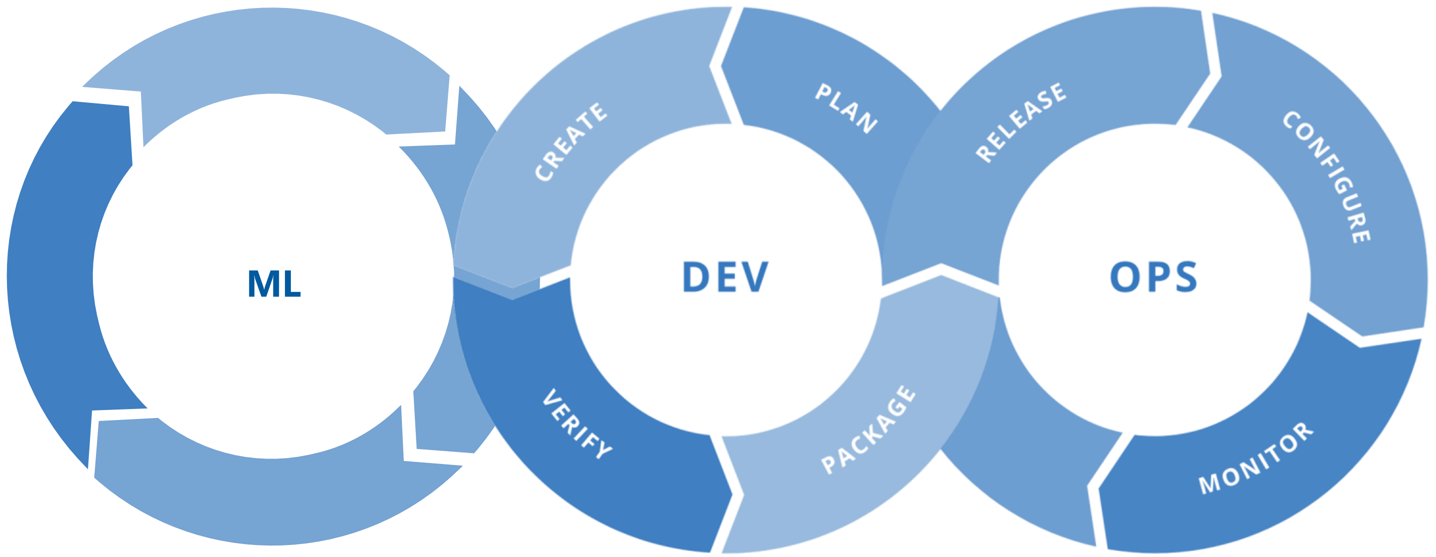
- Quick iteration
- Frameworks they understand
- Best of breed tools
- No management headaches
- Unlimited scale



SRE/ML Engineers

- Reuse of tooling and platforms
- Corporate compliance
- Observability
- Uptime

MLOps = ML + DEV + OPS



Experiment
Data Acquisition
Business Understanding
Initial Modeling

Develop
Modeling + Testing
Continuous Integration
Continuous Deployment

Operate
Continuous Delivery
Data Feedback Loop
System + Model Monitoring

MLOps Benefits

Automation / Observability

- Code drives **generation** and **deployments**
- Pipelines are **reproducible** and **verifiable**
- All artifacts can be **tagged** and **audited**

Validation

- SWE best practices for quality control
- Offline comparisons of model **quality**
- Minimize **bias** and enable **explainability**

Reproducibility /Auditability

- Controlled rollout capabilities
- Live comparison of predicted vs. expected performance
- Results fed back to watch for drift and improve model

== VELOCITY and SECURITY (For ML)

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…

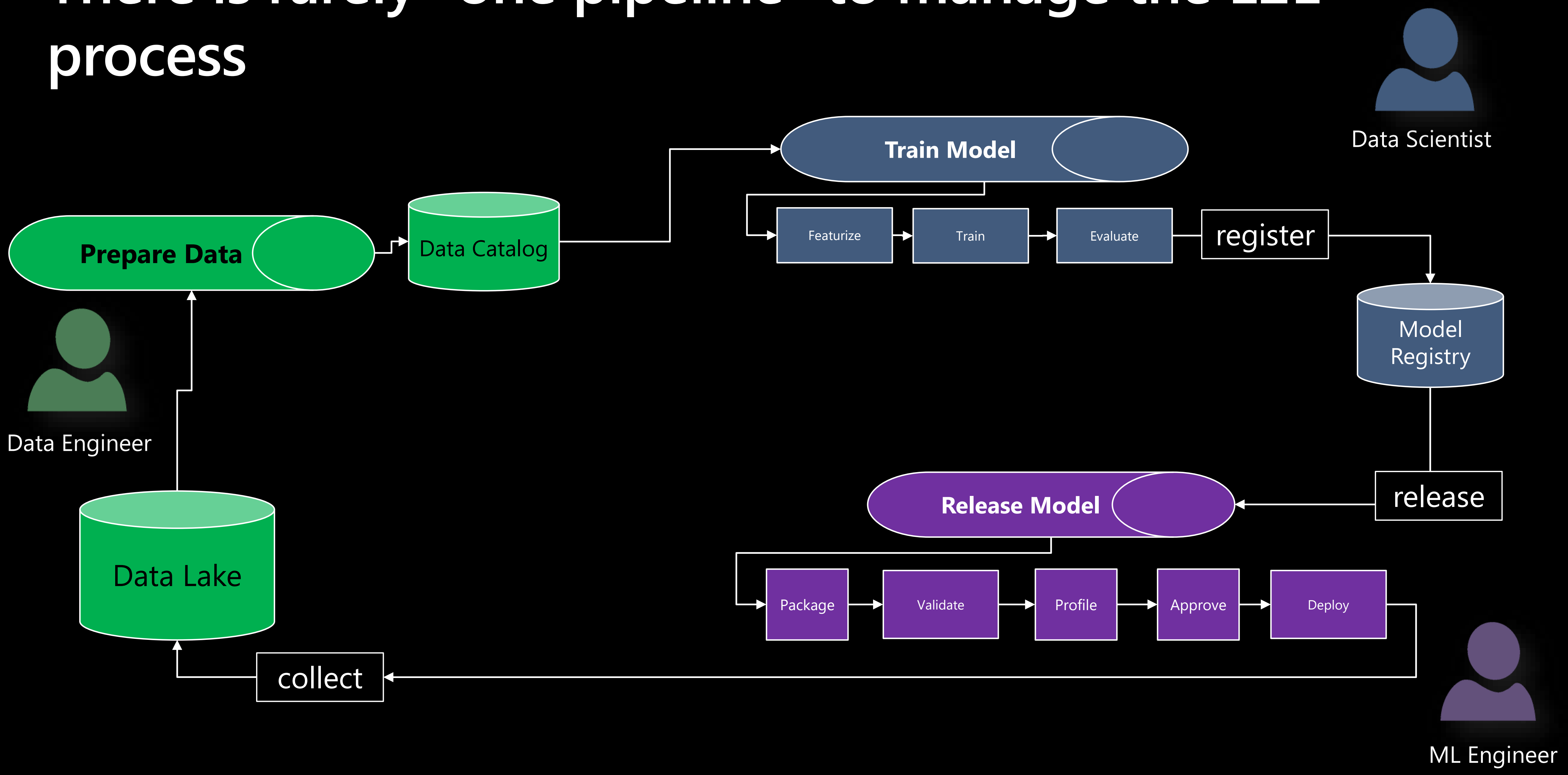
Data Engineer

Azure Data Lake
Azure Data Factory
Azure DataBricks
Azure SQL

Azure DevOps
GitHub
Azure Kubernetes Service
Azure IoT Edge
Azure Monitor

ML Engineer

There is rarely “one pipeline” to manage the E2E process

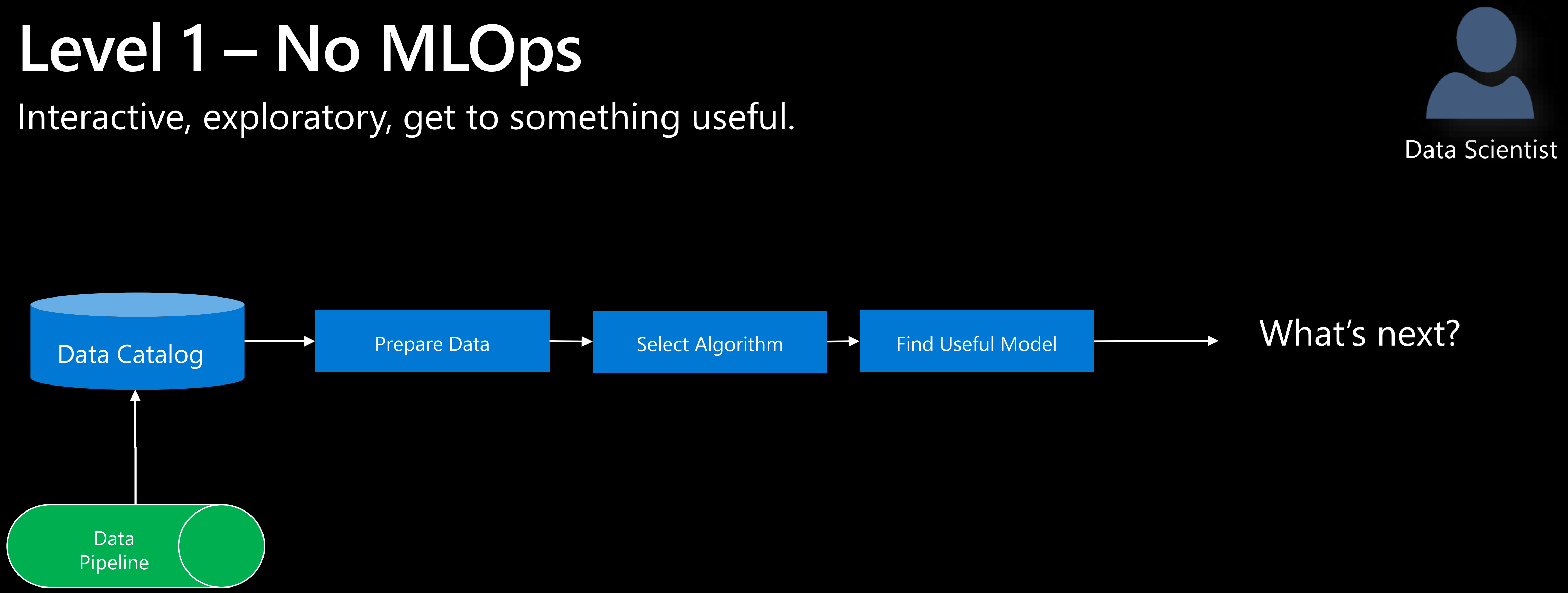


MLOps – Process Maturity Model

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

Level 1 – No MLOps

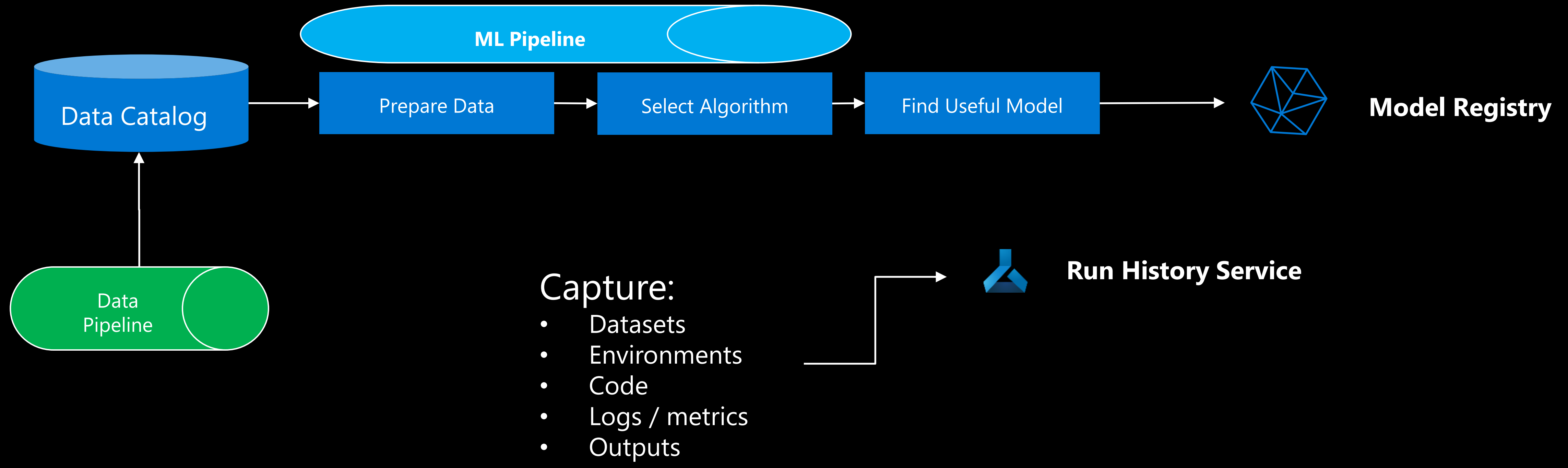
Interactive, exploratory, get to something useful.



Level 2 – Reproducible Model Training

Version code, data, ensure model can be recreated.

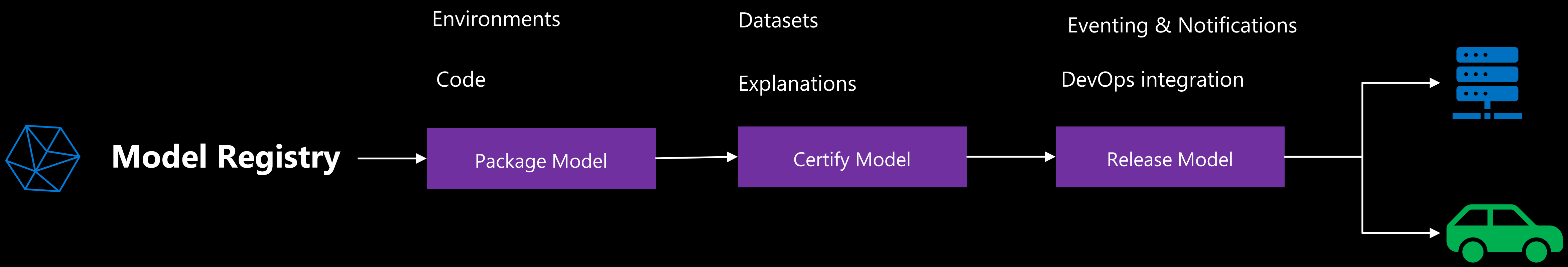
Data Scientist



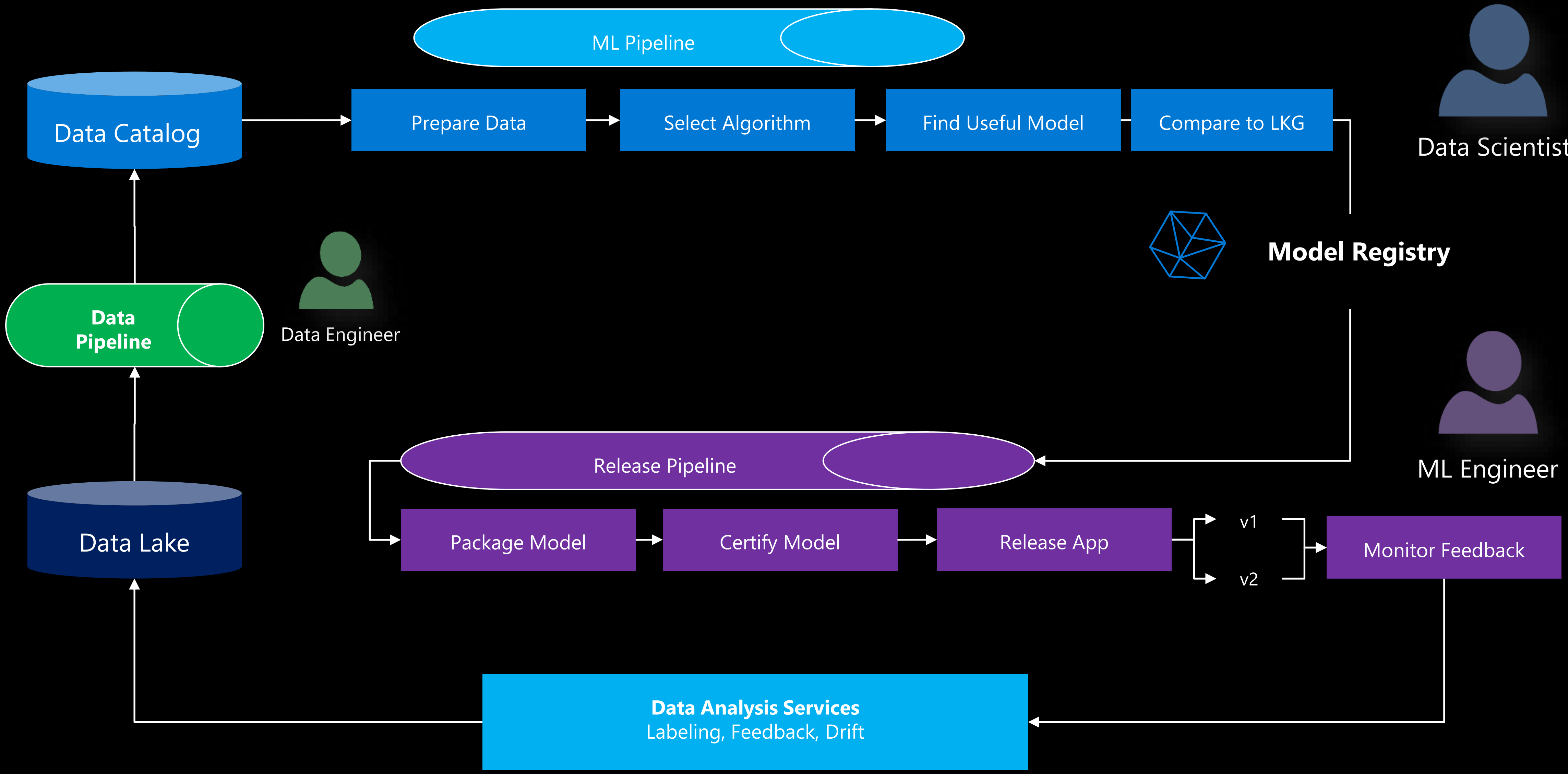
Level 3 – Managed Model Operationalization

Package, certify, deploy

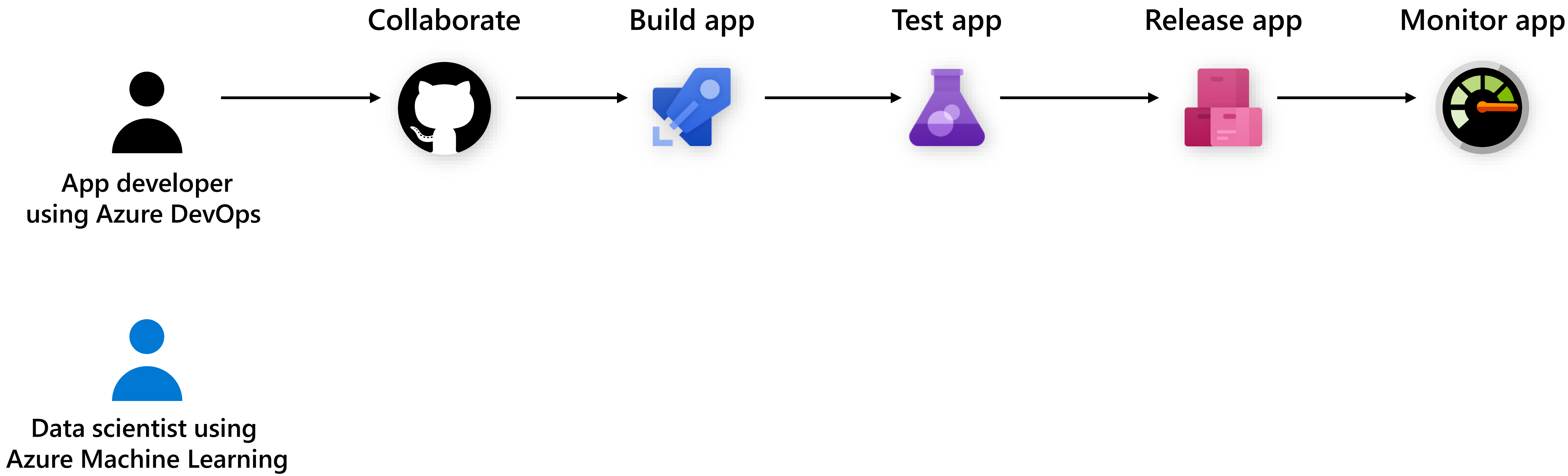
ML Engineer



Level 4 – Automated E2E ML Lifecycle



MLOps Workflow



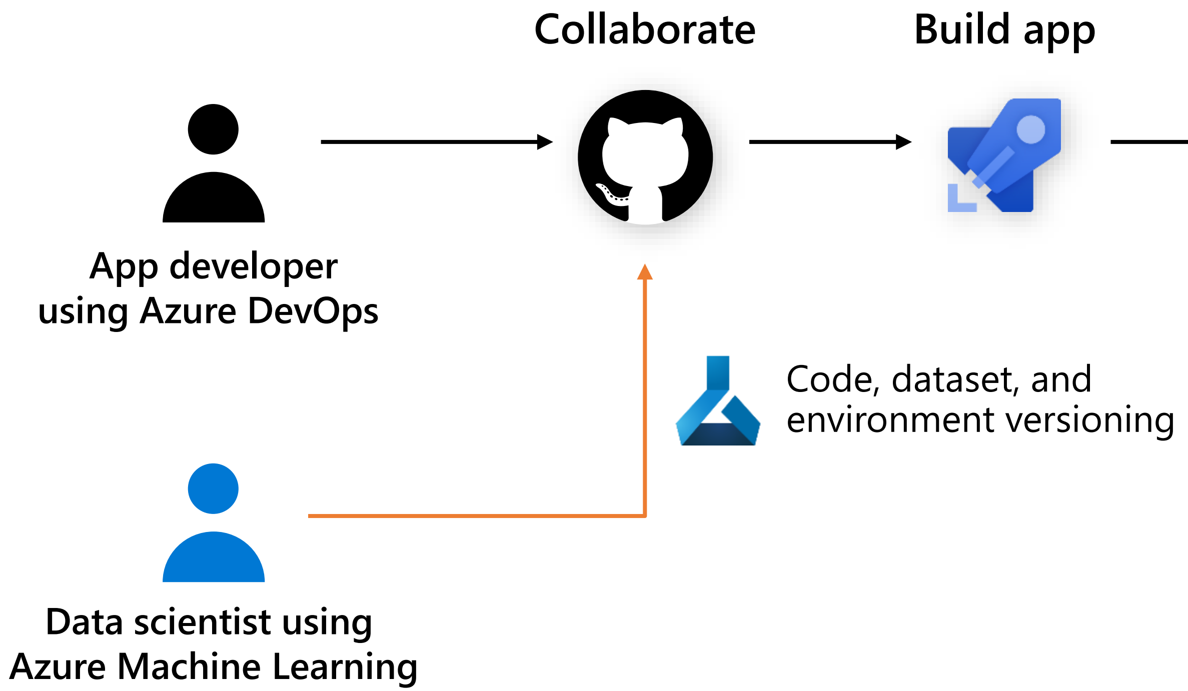
Model reproducibility

Model validation

Model deployment

Model retraining

MLOps Workflow



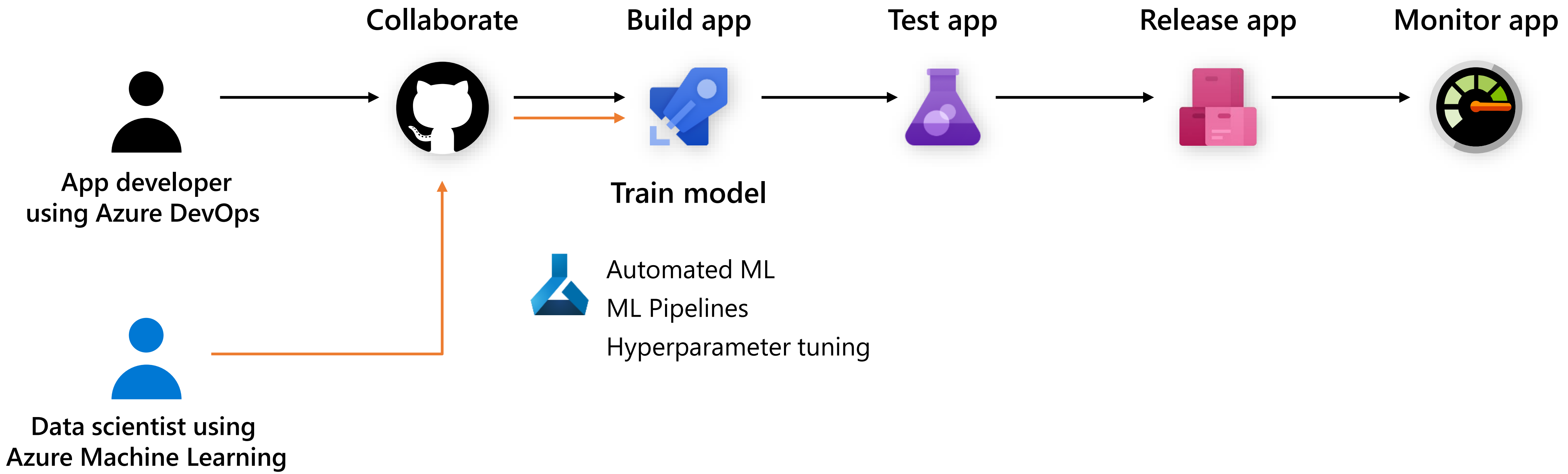
Model reproducibility

Model validation

Model deployment

Model retraining

MLOps Workflow



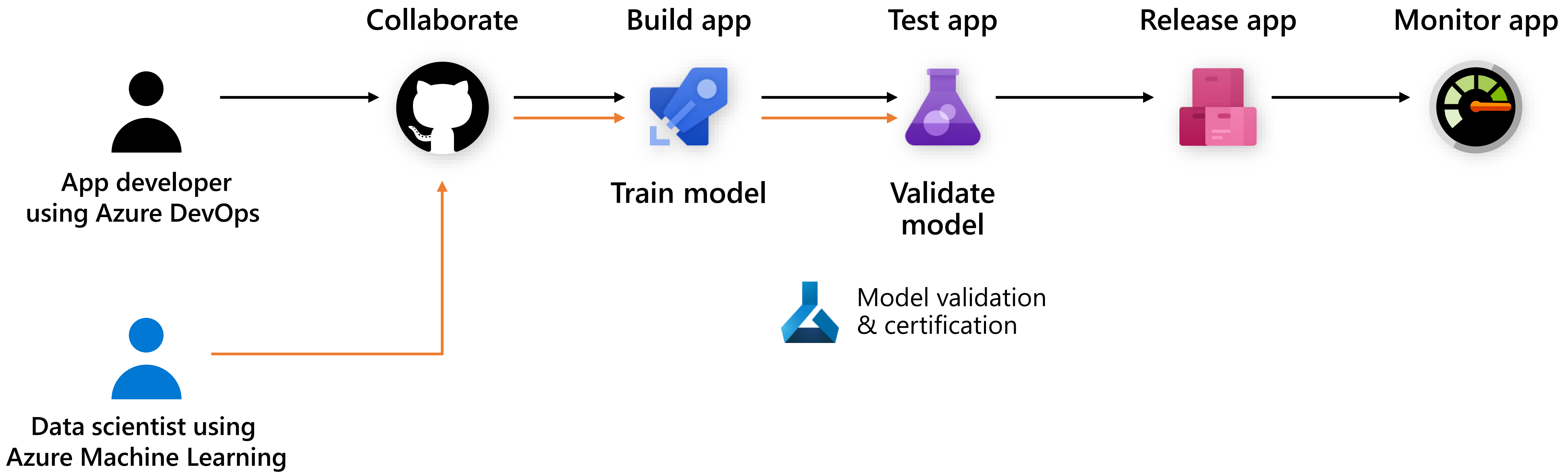
✓ Model reproducibility

✗ Model validation

✗ Model deployment

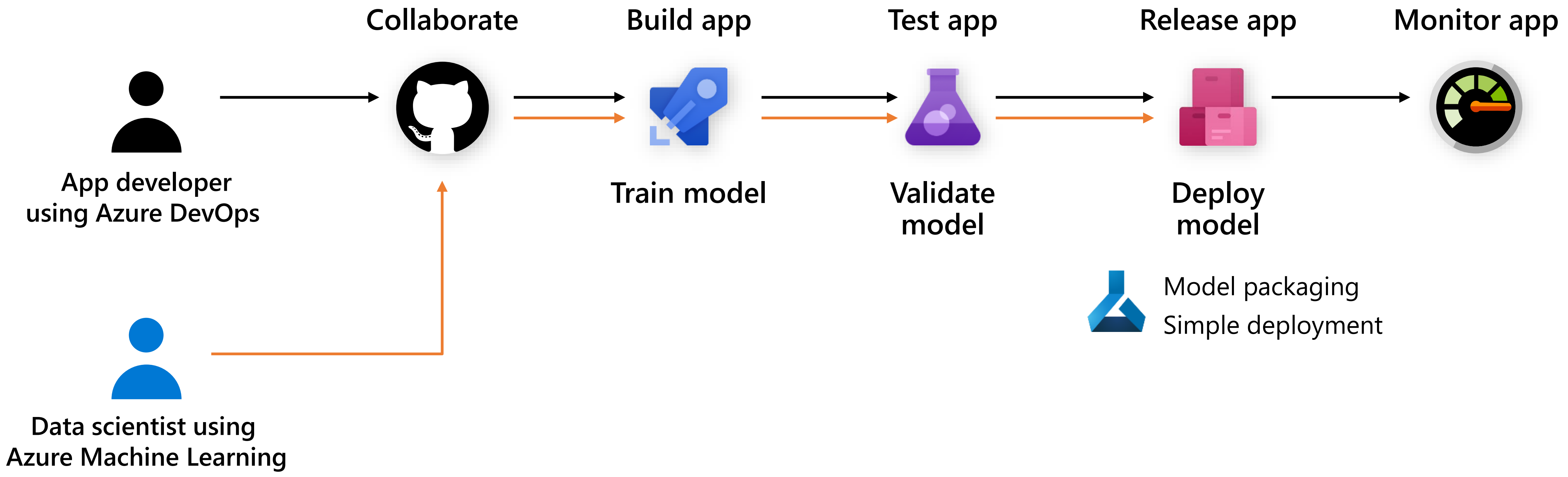
✗ Model retraining

MLOps Workflow



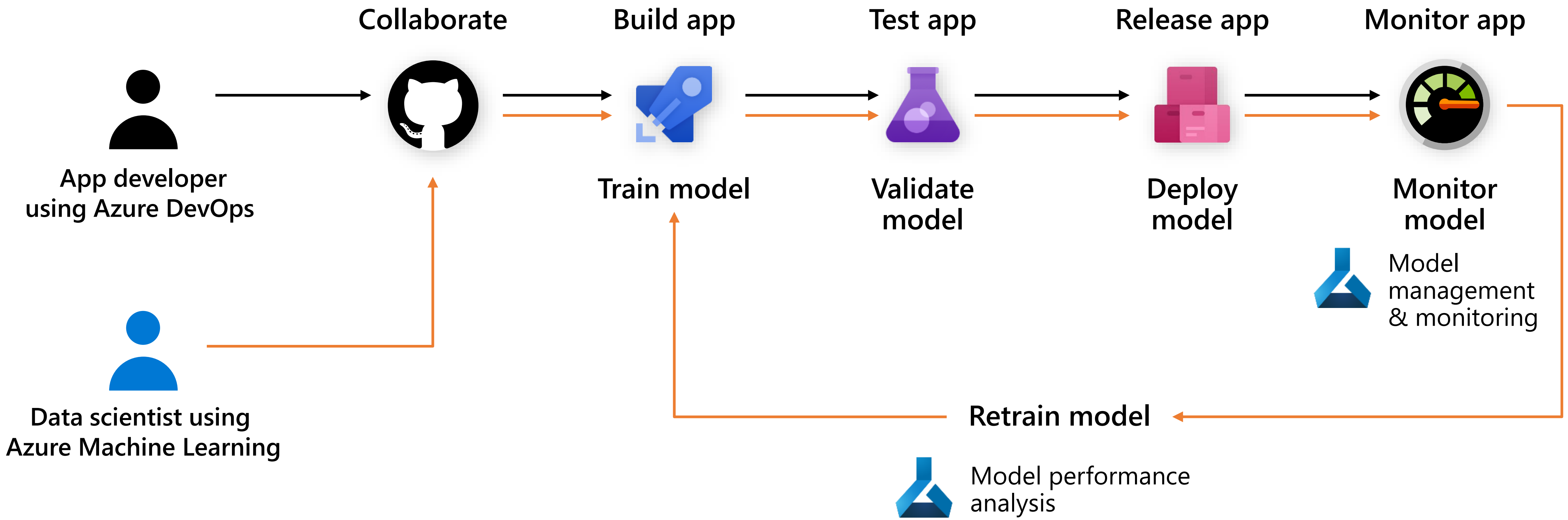
✓ Model reproducibility ✓ Model validation ✗ Model deployment ✗ Model retraining

MLOps Workflow



✓ Model reproducibility ✓ Model validation ✓ Model deployment ✗ Model retraining

MLOps Workflow

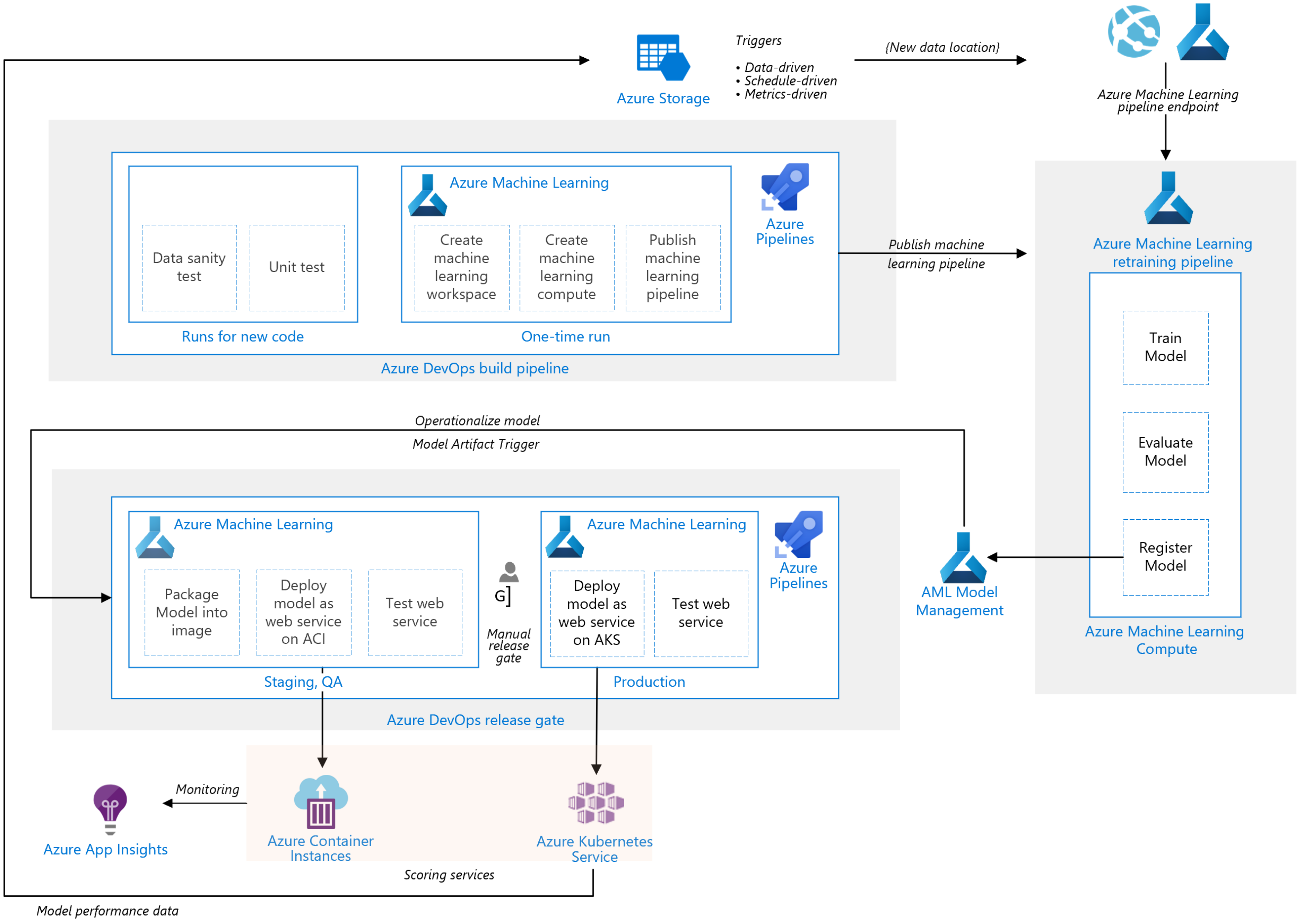


Model reproducibility

Model validation

Model deployment

Model retraining



Azure Storage
Triggers
• Data-driven
• Schedule-driven
• Metrics-driven
{New data location}
Azure Machine Learning pipeline endpoint
Data sanity test
Unit test
Runs for new code
Azure Machine Learning
Create machine learning workspace
Create machine learning compute
Publish machine learning pipeline
One-time run
Azure Pipelines
Azure DevOps build pipeline
Publish machine learning pipeline
Azure Machine Learning retraining pipeline
Train Model
Evaluate Model
Register Model
Azure Machine Learning Compute
Operationalize model
Model Artifact Trigger
Azure Machine Learning
Package Model into image
Deploy model as web service on ACI
Test web service
Staging, QA
Manual release gate
Azure Machine Learning
Deploy model as web service on AKS
Test web service
Production
Azure Pipelines
Azure DevOps release gate
AML Model Management
Azure App Insights
Monitoring
Azure Container Instances
Azure Kubernetes Service
Scoring services
Model performance data

MLOps Questions

- Define MLOps and how it is different from Data Science ?
- What is a model Registry? What does the pipeline look like ?
- Difference between MLOps and DevOps?
- What are the Benefits of MLOps?
- Explain Model/Concept Drift?
- What is a ML pipeline? Can you draw and explain it ?
- Is Model deployment end of ML Lifecycle ?

Reference Links

1. https://github.com/microsoft/MCW-ML-Ops
2. https://github.com/microsoft/MLOpsPython
3. https://azure.microsoft.com/en-us/services/machine-learning/mlops/
4. https://www.amazon.com/Data-Science-Solutions-Azure-Techniques/dp/1484264045
5. https://docs.microsoft.com/en-us/samples/microsoft/mlopspython/mlops-with-azure-ml/