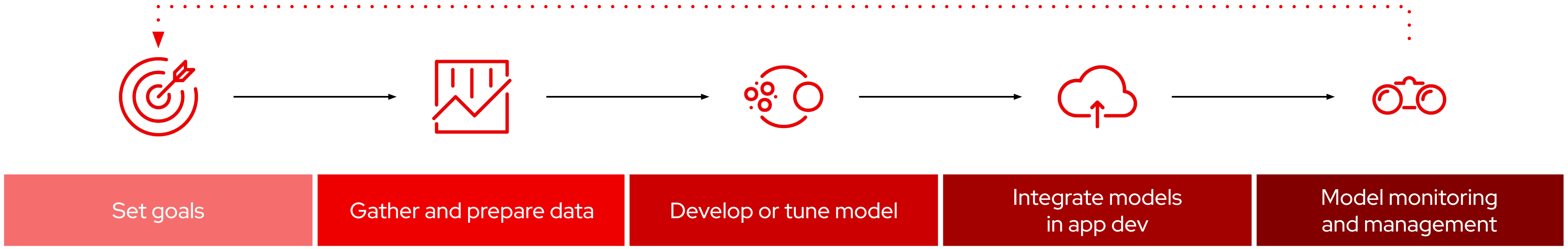
# Un puente entre MLOps y DevOps con OpenShift AI

Juan Vicente Herrera
@jvicenteherrera

# Move models from experimentation to production faster

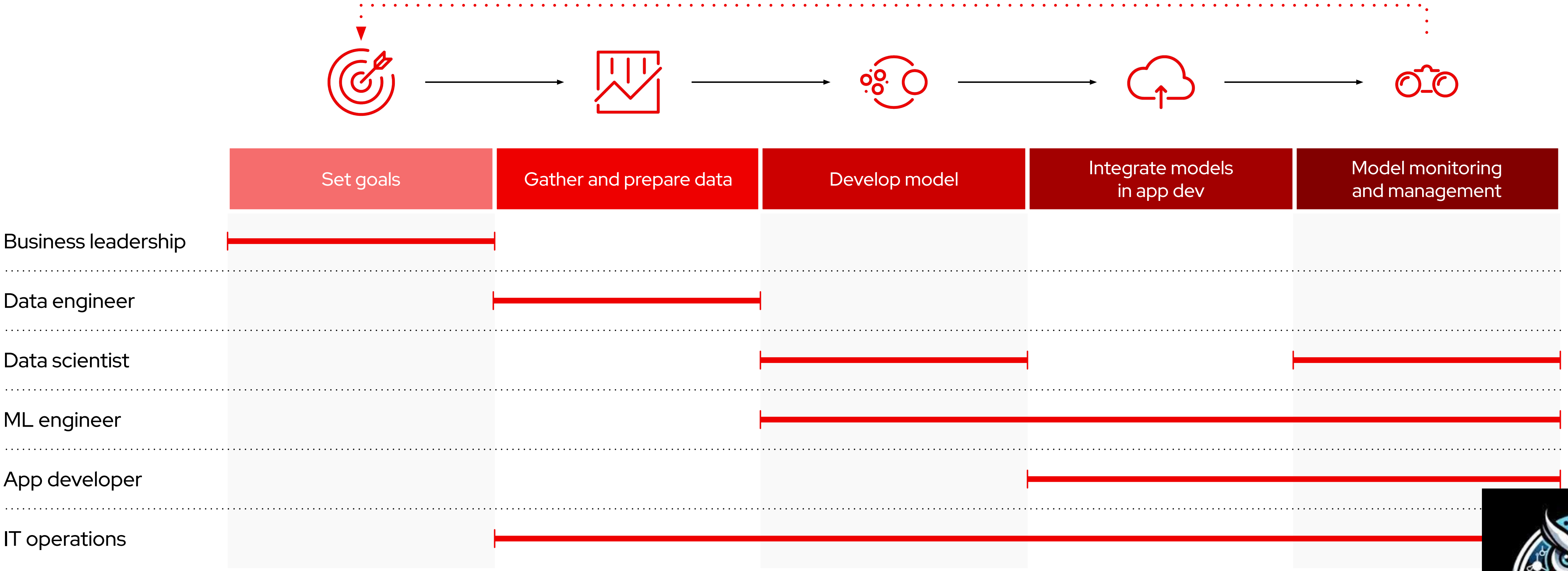
Operationalize AI is the catalyst for incorporating AI into practical applications




**Operationalize AI** is the process of integrating AI capabilities into the day-to-day operations of an organization. It involves taking your models from experimentation to production while contributing to the overall goals of the organization.

# Operationalizing AI/ML requires collaboration

## Every member of your team plays a critical role in a complex process



# Conceptual machine learning architecture

**Gather and prepare data**

Data storage
Data lake
Data exploration
Data preparation
Stream processing

**Develop model**

ML notebooks
ML libraries

**Deploy models in an application**

Model lifecycle
CI/CD

**Model monitoring and management**

Monitor / alerts
Model visualization
Model drift

**Hybrid, multi cloud platform with self service capabilities**

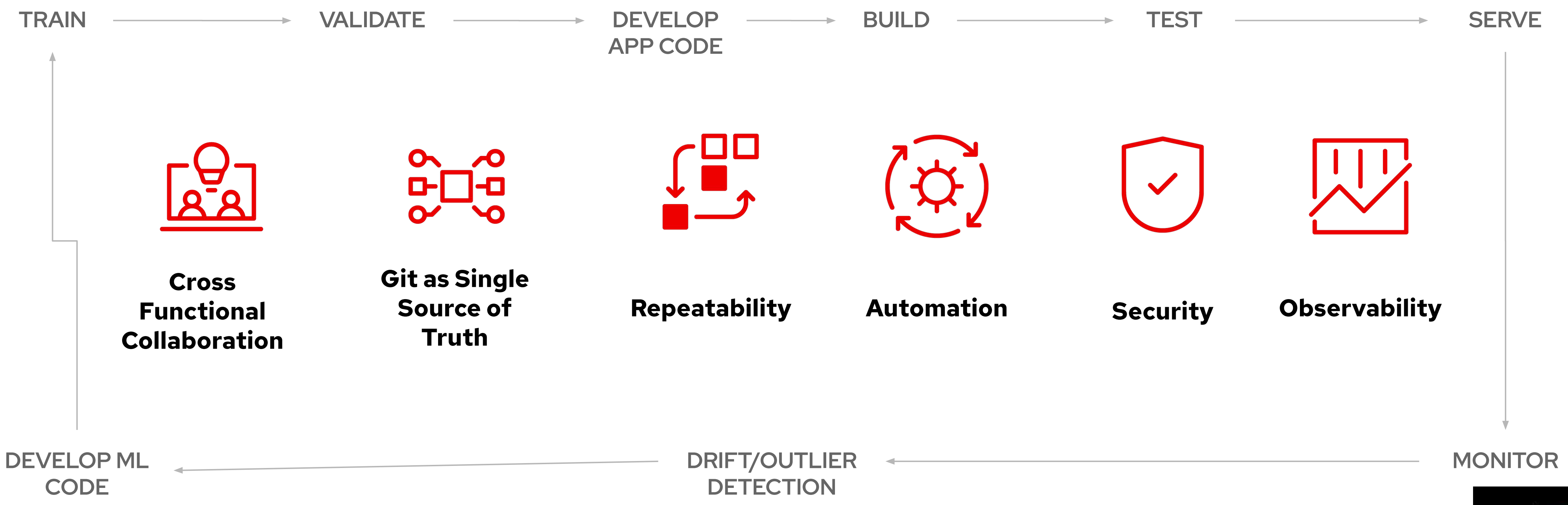
**Compute acceleration**

## Infrastructure

- Physical
- Virtual
- Private cloud
- Public cloud
- Edge

# Enter MLOps

MLOps incorporates DevOps and GitOps to improve the lifecycle management of the ML application



# Just like DevOps, MLOps requires changes.

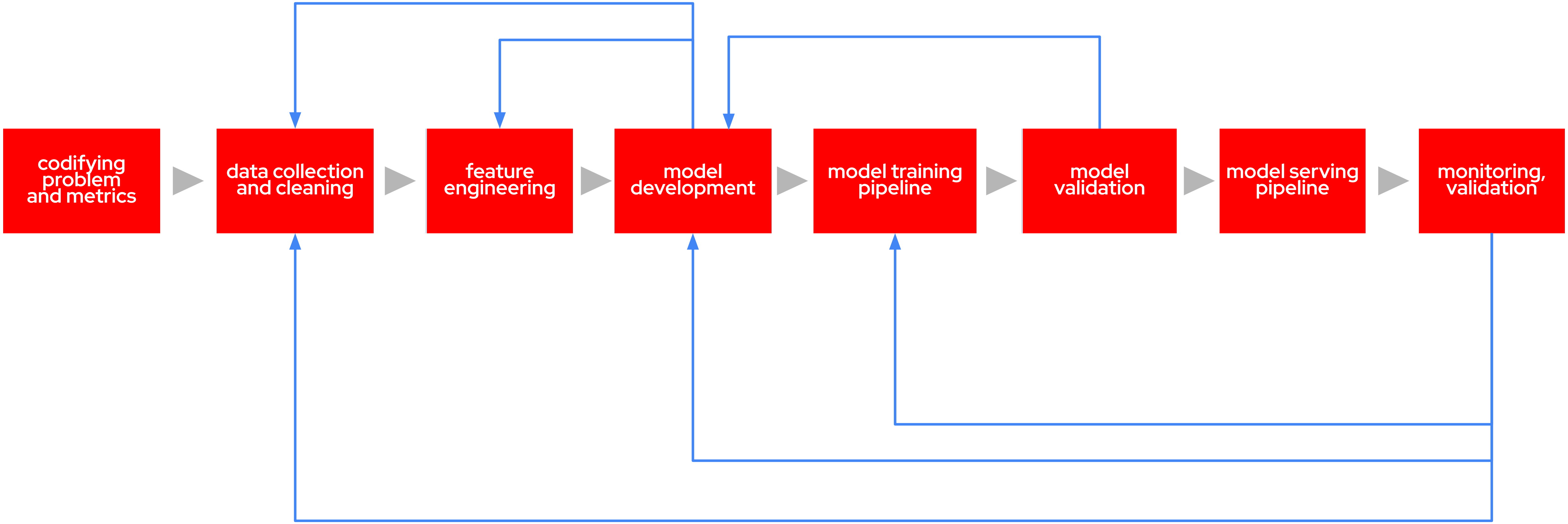
**Multi-disciplinary teams**
Cross-train on the basics.

**Automation**
Automate everything that can be automated.

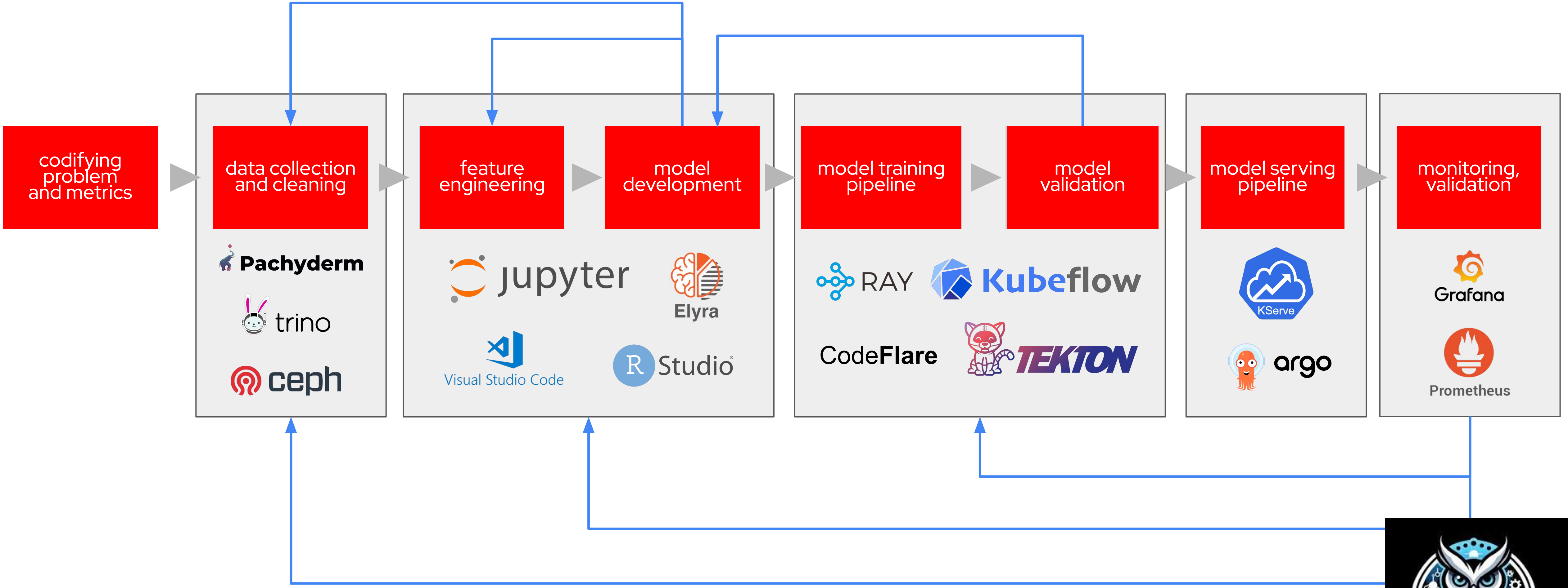
**Patience**
It's a gradual process, so it won't happen overnight.

**Metrics**
Pair measuring and tracking with transparency.

# The MLOps workflow



# MLOps with open source



# Open Data Hub

## An open source MLOps suite

- Multi-tenant data science platform
- Self-service workbenches

- Preinstalled machine learning libraries
- Custom stack can be integrated

- Distributed model training
- Parallelize workloads across nodes and GPUs

**Elyra**

- AI pipeline editor
- Define workflows through Jupyter

**Kubeflow Pipelines**

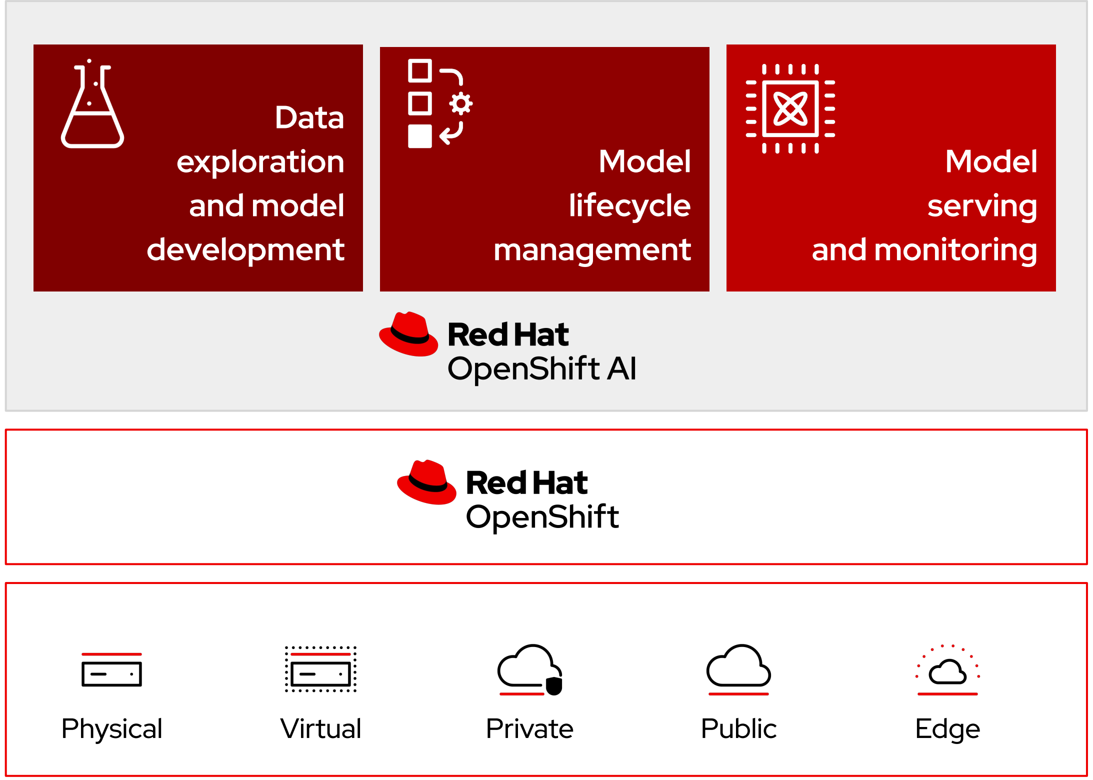
- Machine learning workflow orchestration
- Experiment tracking

**Kserve ModelMesh**

- Deploying machine learning models as micro-services
- Pre-built inference servers

# Understanding the building blocks

A common platform to bring IT, data science, and app dev teams together




## Model development

Conduct exploratory data science in JupyterLab with access to core AI / ML libraries and frameworks including TensorFlow and PyTorch using our notebook images or your own.

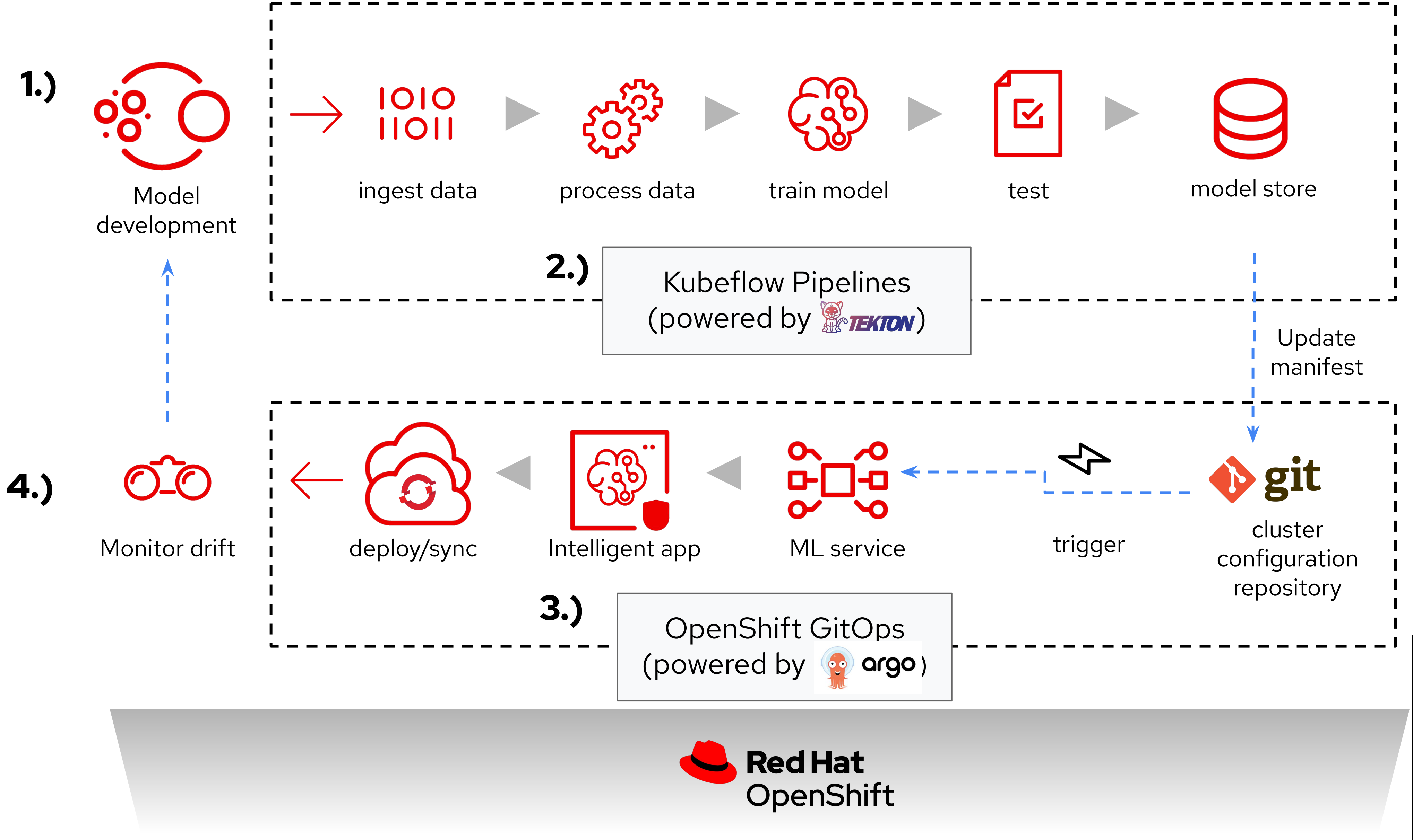
## Lifecycle Management

Create repeatable data science pipelines for model training and validation and integrate them with devops pipelines for delivery of models across your enterprise.

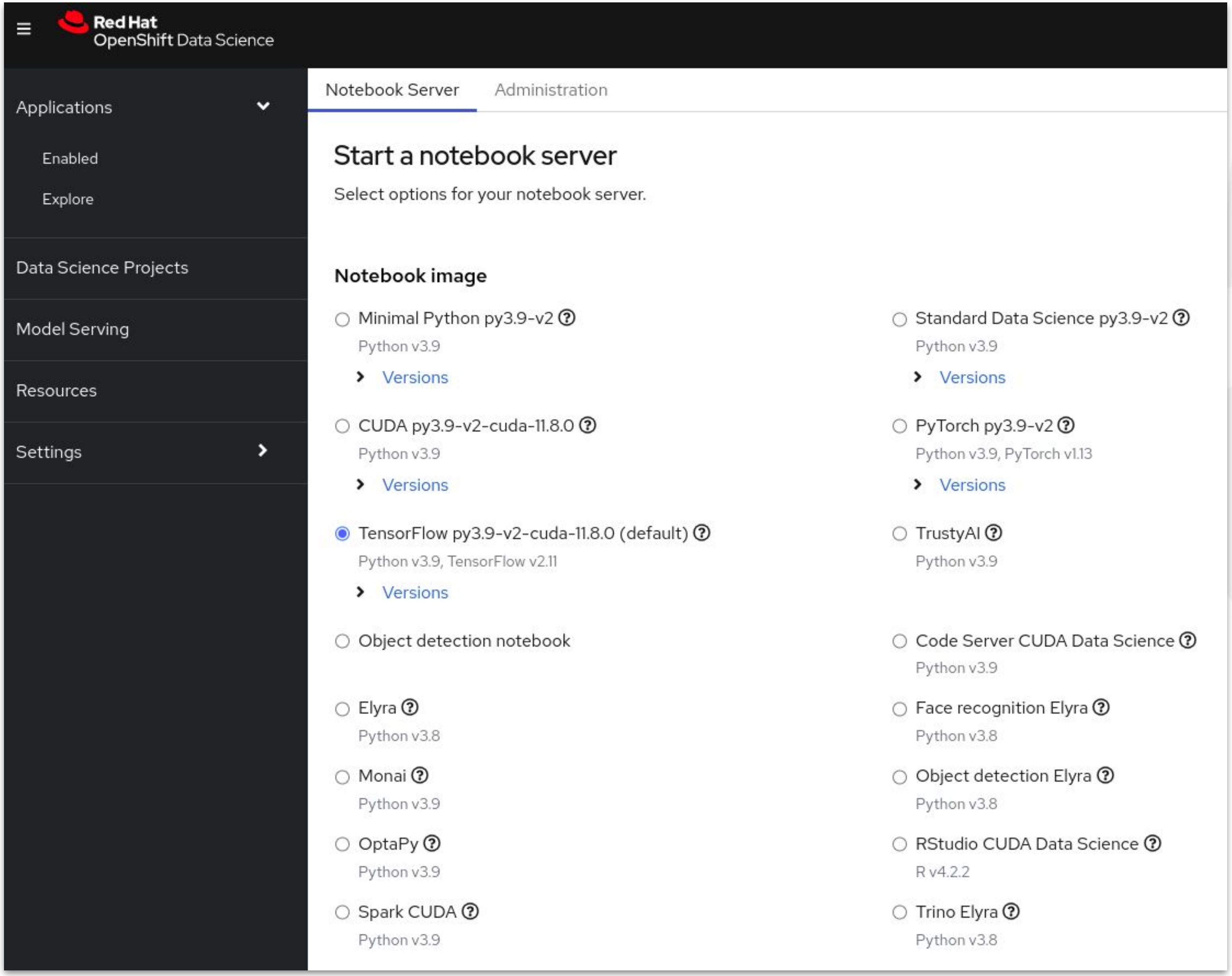
## Model serving & monitoring

Deploy models across any cloud, fully managed, and self-managed OpenShift footprint and centrally monitor their performance.

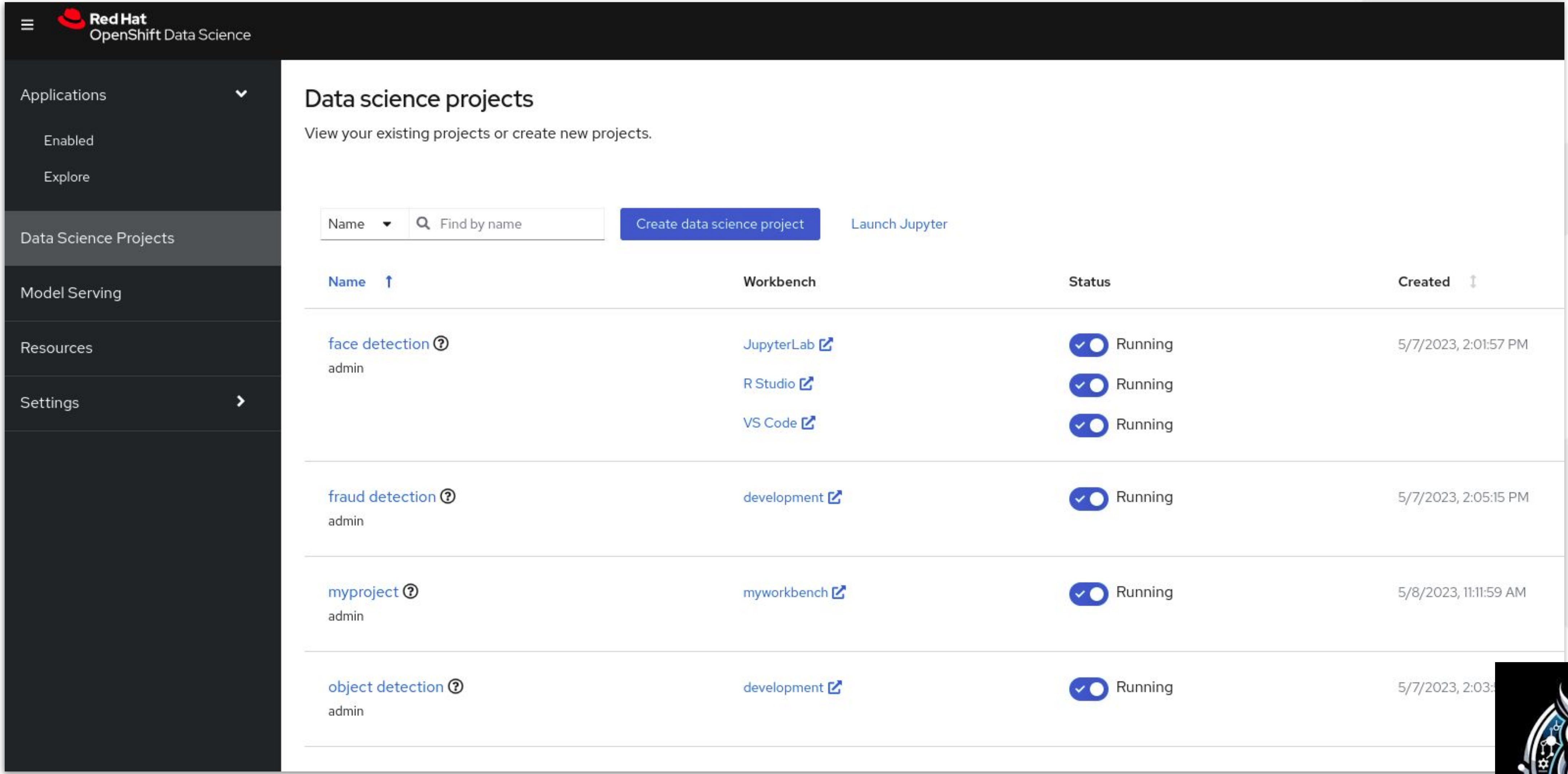
# MLOps with Red Hat OpenShift



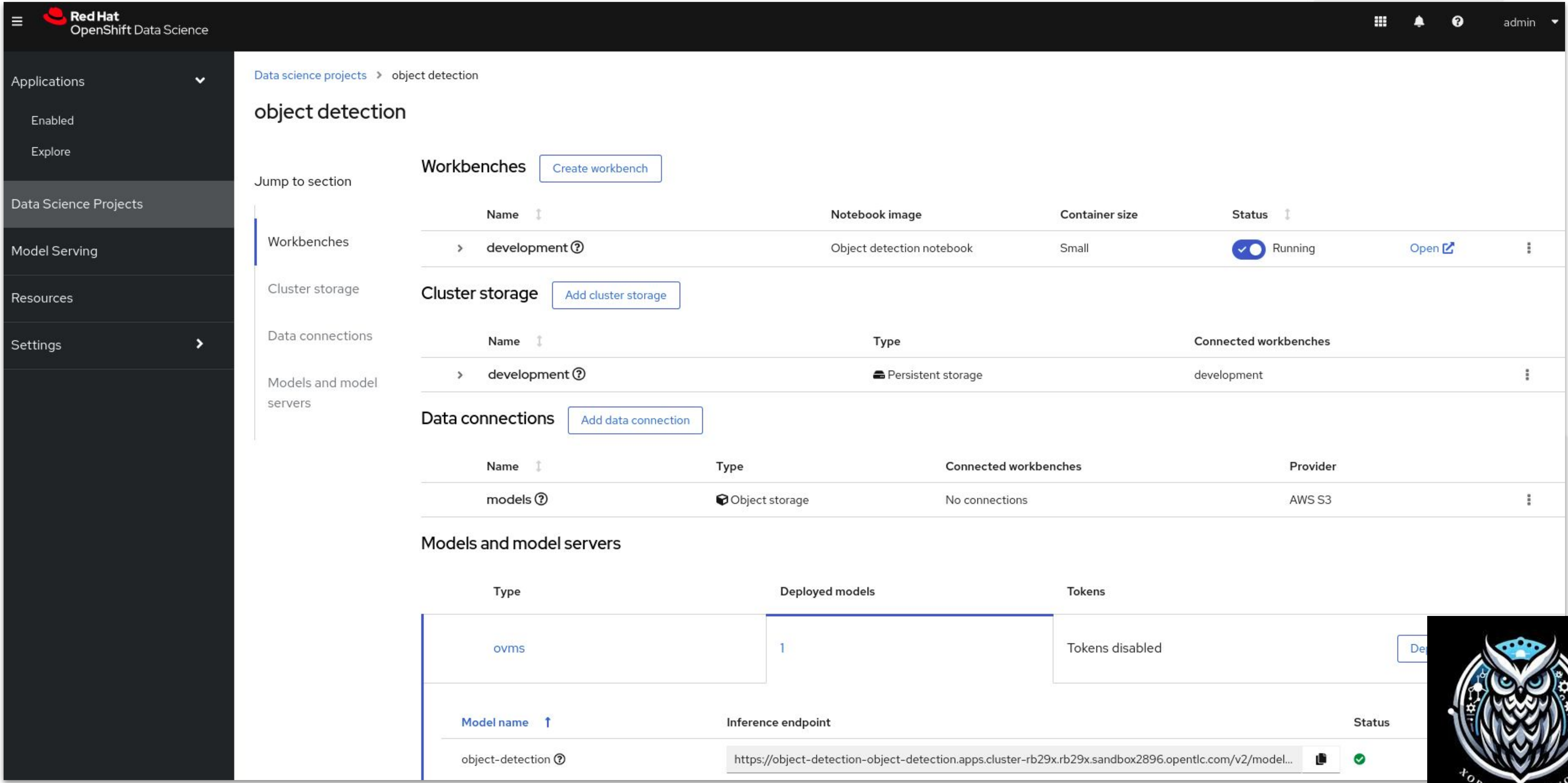
# Workbench images



# Data Science Projects



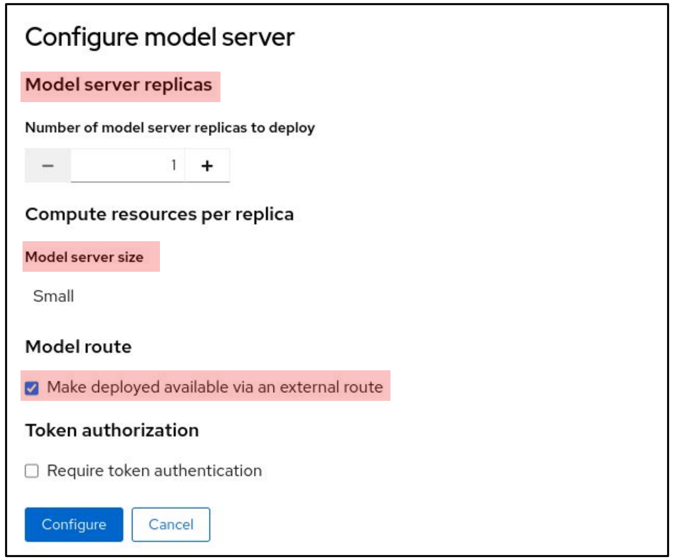
# Project Resources

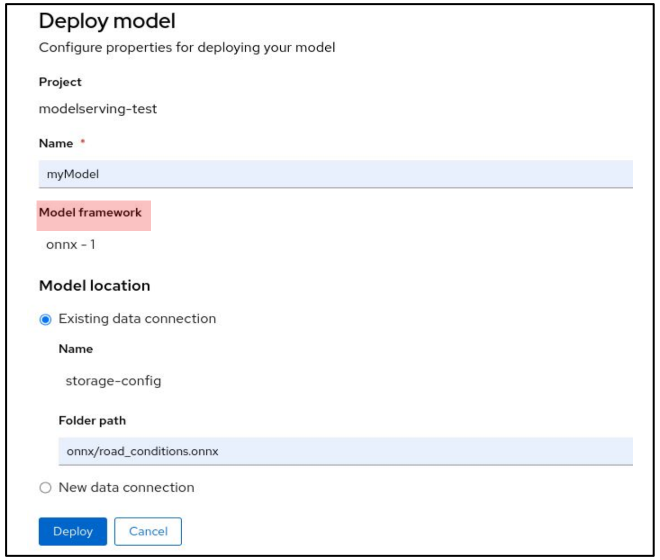


# Serve, scale, and monitor your models

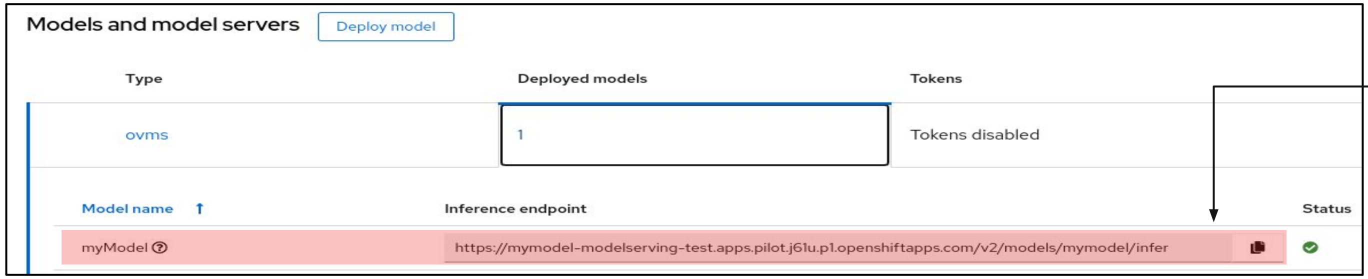
Select the required resources and scale model serving as needed

Make your model public and secure



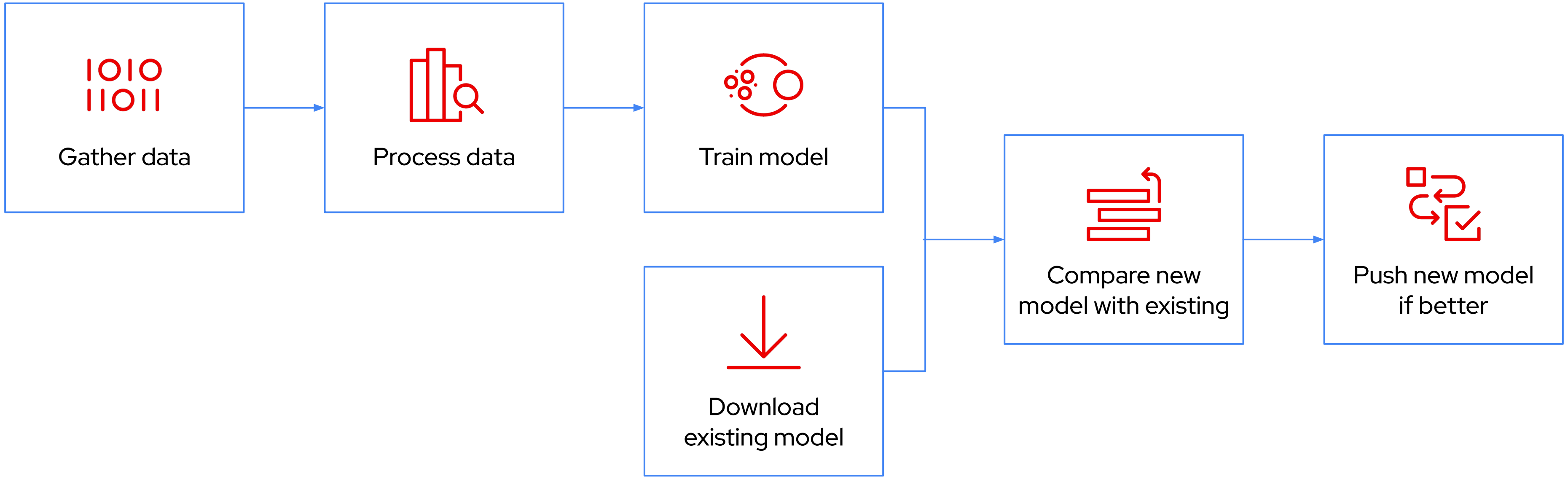




Select your model framework




View your deployed model fleet endp

# Data science pipelines component




- Continuously deliver and test models in production
- Schedule, track, and manage pipeline runs
- Easily build pipelines using graphical front end
- Orchestrate data science tasks into pipelines
- Chain together processes like data prep, build models, and serve models

# Creating Pipelines with Kubeflow Pipelines

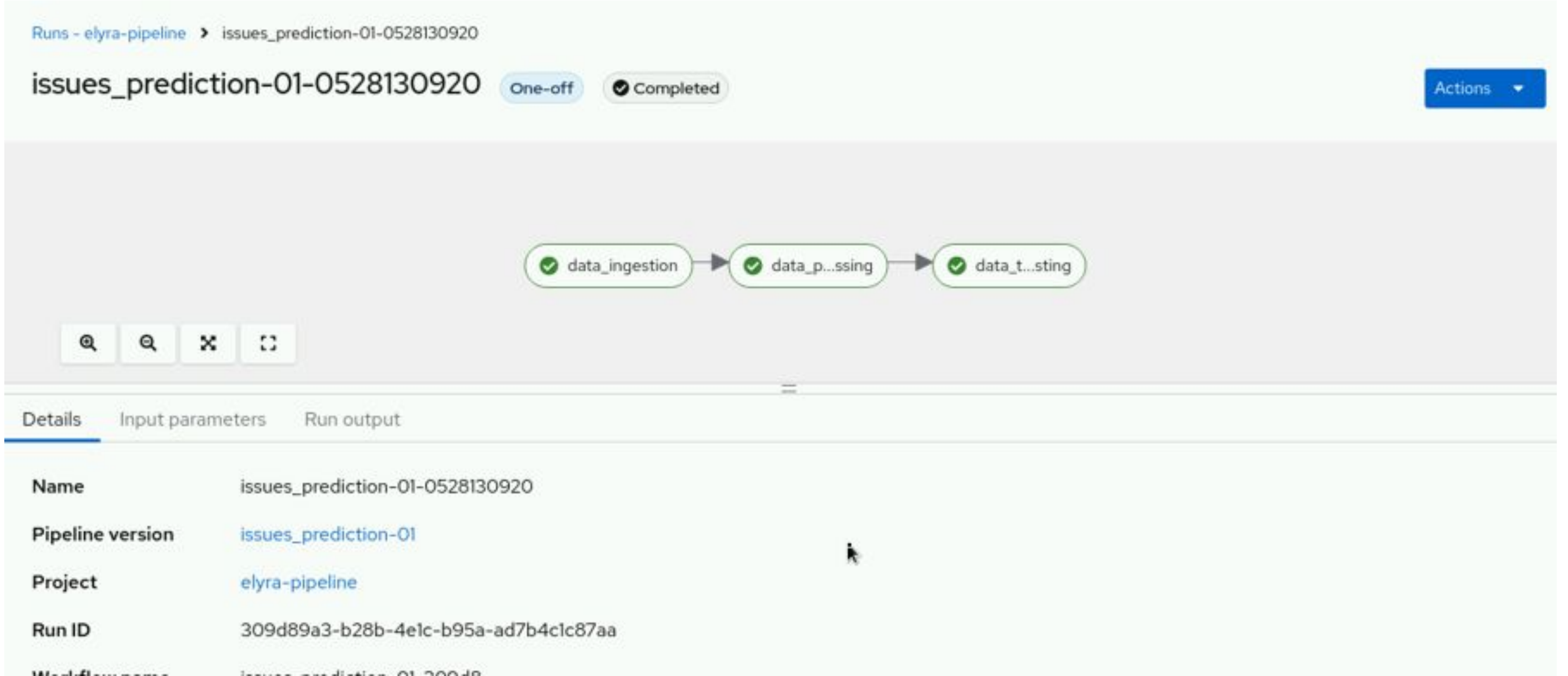
- Data Science Pipelines (DSP) allows data scientists **to track progress as they iterate over development** of ML models. With DSP, a data scientist
- They can create and **track experiments to arrive at the best version of of training data, model hyperparameters, model code,** etc., and repeatably rerun these experiments.


Demo time

# Creating Pipelines with Kubeflow Pipelines

## Flip coin example

1. Define a pipeline in Python.
2. Compile the Python file into a Tekton resource definition.
3. Import the pipeline.
4. Execute the pipeline.



# Creating and Using Model Servers

1. Create  a custom Scikit-learn model server by creating a Python container and a RHOAI serving runtime.
2. Deploy a version of the diabetes model trained with Scikit-learn by using the RHOAI console.


ENJOY
XOPS
CONFERENCE
XOPS
XOPS
XOPS
XOPS