Red Hat Summit

Connect

CI/CD dla aplikacji AI/ML

Red Hat

Red Hat

Artur Pająk

Senior Solution Architect
Red Hat

Red Hat

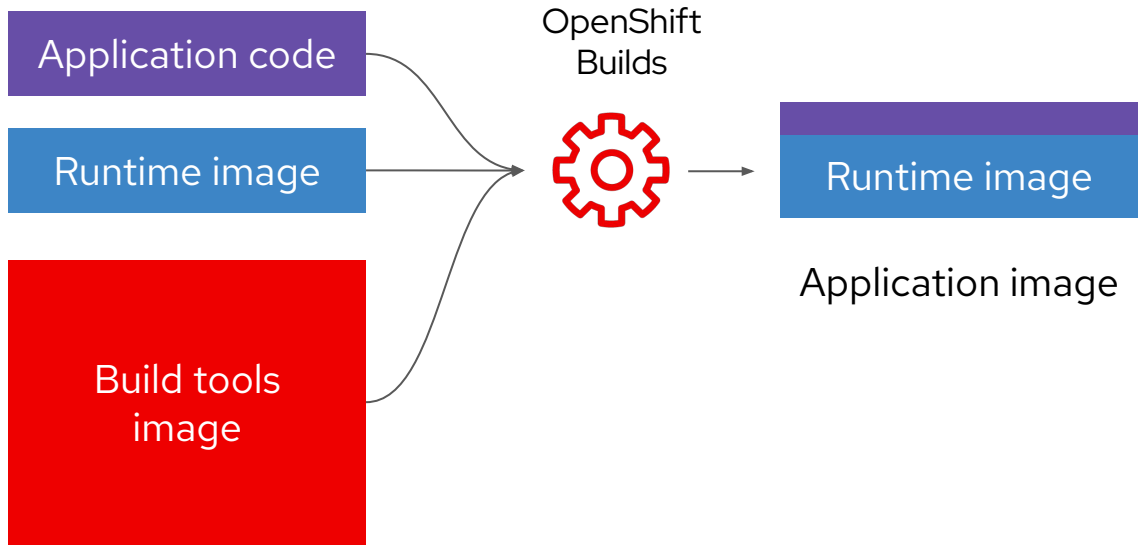
CI/CD for “classic” application.

Red Hat

Build images from code using S2I + other & integrate with Github Actions

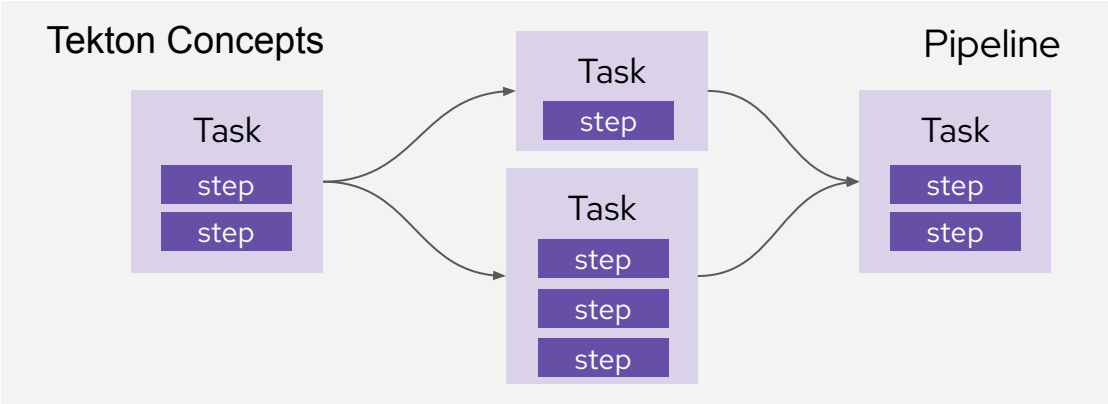
- Build images on OpenShift and Kubernetes
- Use Kubernetes build tools
  - Source-to-Image
  - Buildpacks
  - Buildah
  - Kaniko
  - ...more

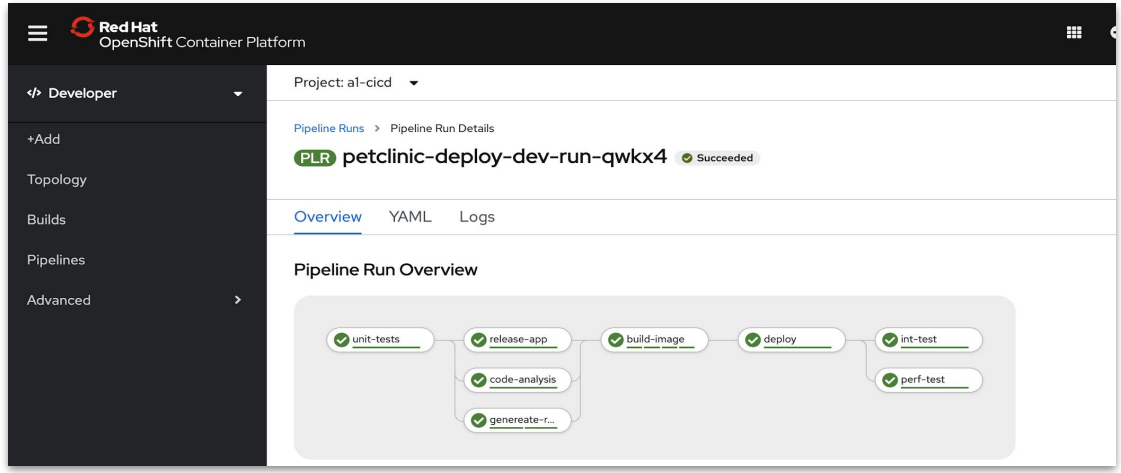
- Create lean application images
- Extend with your own build tools
- Based on Shipwright open-source project



Tekton provides Kubernetes-Native CI/CD pipelines

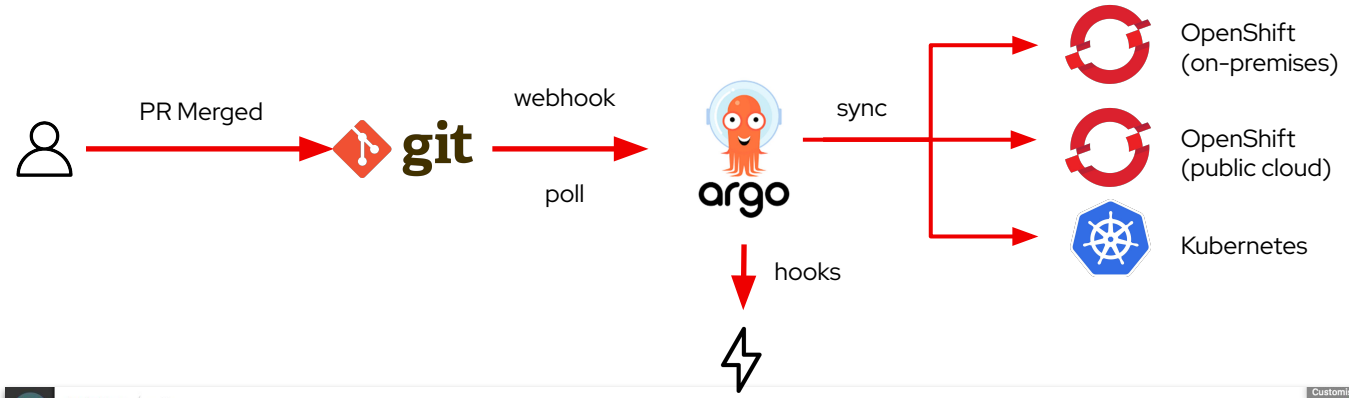
- Based on Tekton Pipelines
- Kubernetes-native declarative CI/CD
- Pipelines run on-demand in isolated containers
- No central server to maintain! No plugin conflicts!
- Task library and integration with Tekton Hub
- Secure pipelines aligned with Kubernetes RBAC
- Visual and IDE-based pipeline authoring
- Pipeline templates when importing apps
- Automated install and upgrades via OperatorHub
- CLI, Web, VS Code and IntelliJ plugins

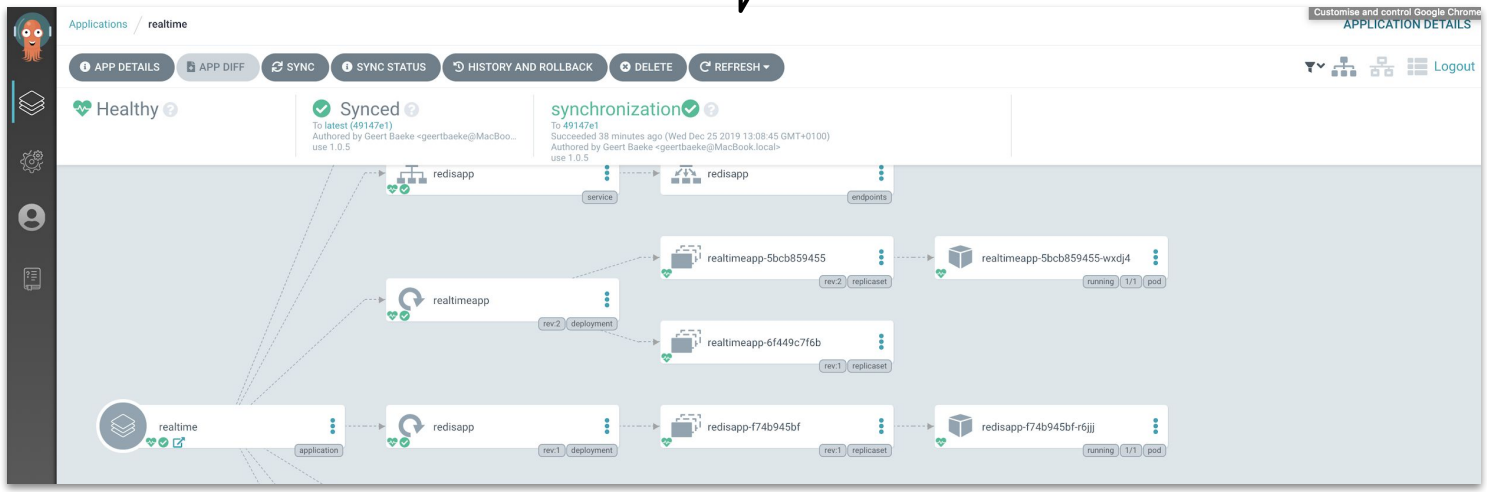




Red Hat

Argo CD for declarative GitOps continuous delivery





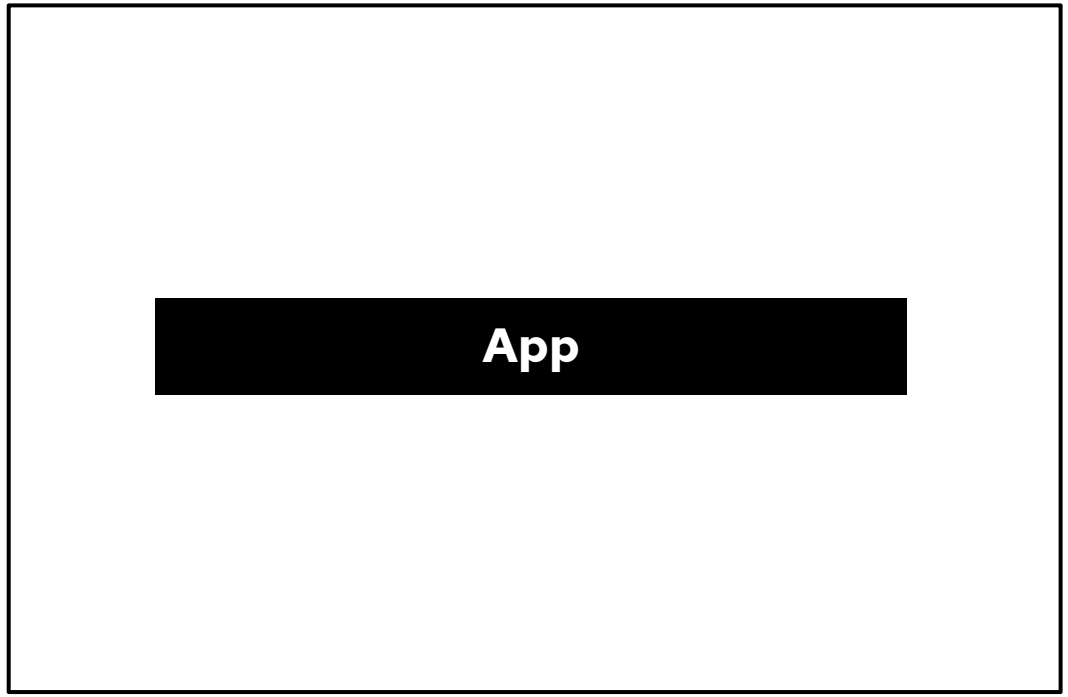
- Configurations versioned in Git
- Automatically syncs configuration from Git
- Drift detection, visualization and correction
- Granular control over sync order
- Rollback and rollforward to any Git commit
- Manifest templating support (Helm, Kustomize, etc)
- Visual insight into sync status

CI/CD challenges in AI/ML application.

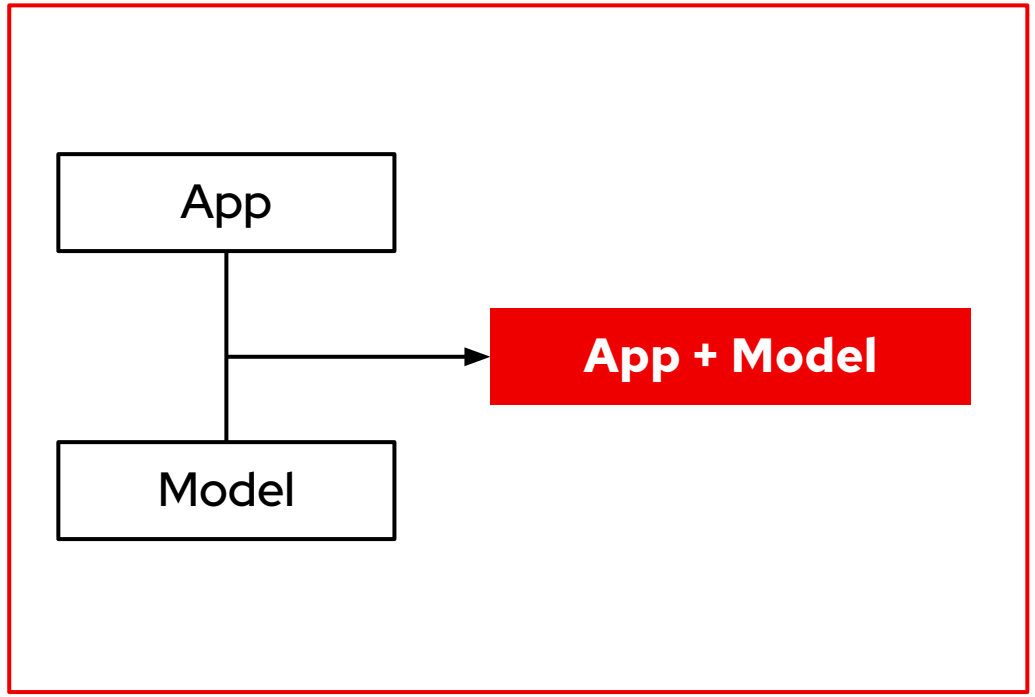
Red Hat

Classical App vs AI App – (Challenge 1)

A.
Insurance calculator app



B.
Insurance app



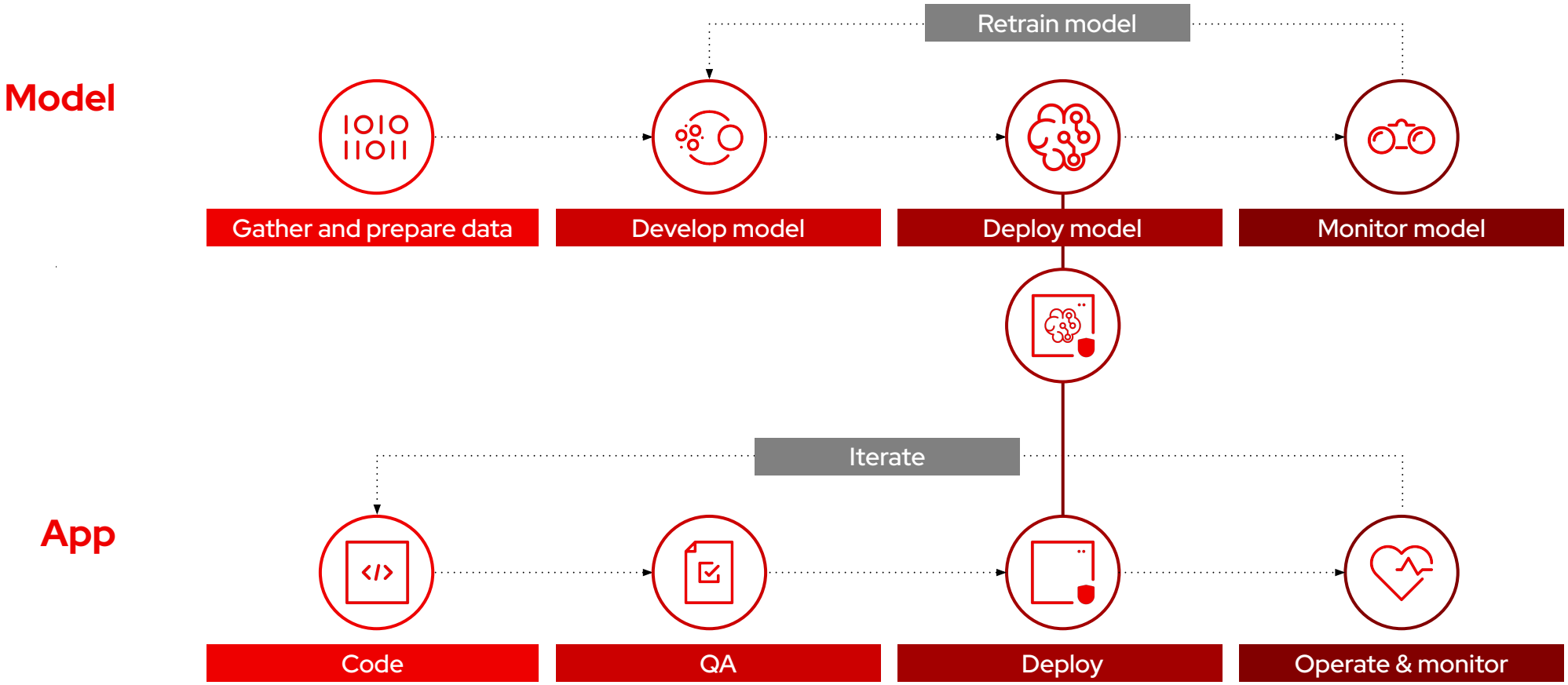
Red Hat

Examples of intelligent applications – (Challenge 2)

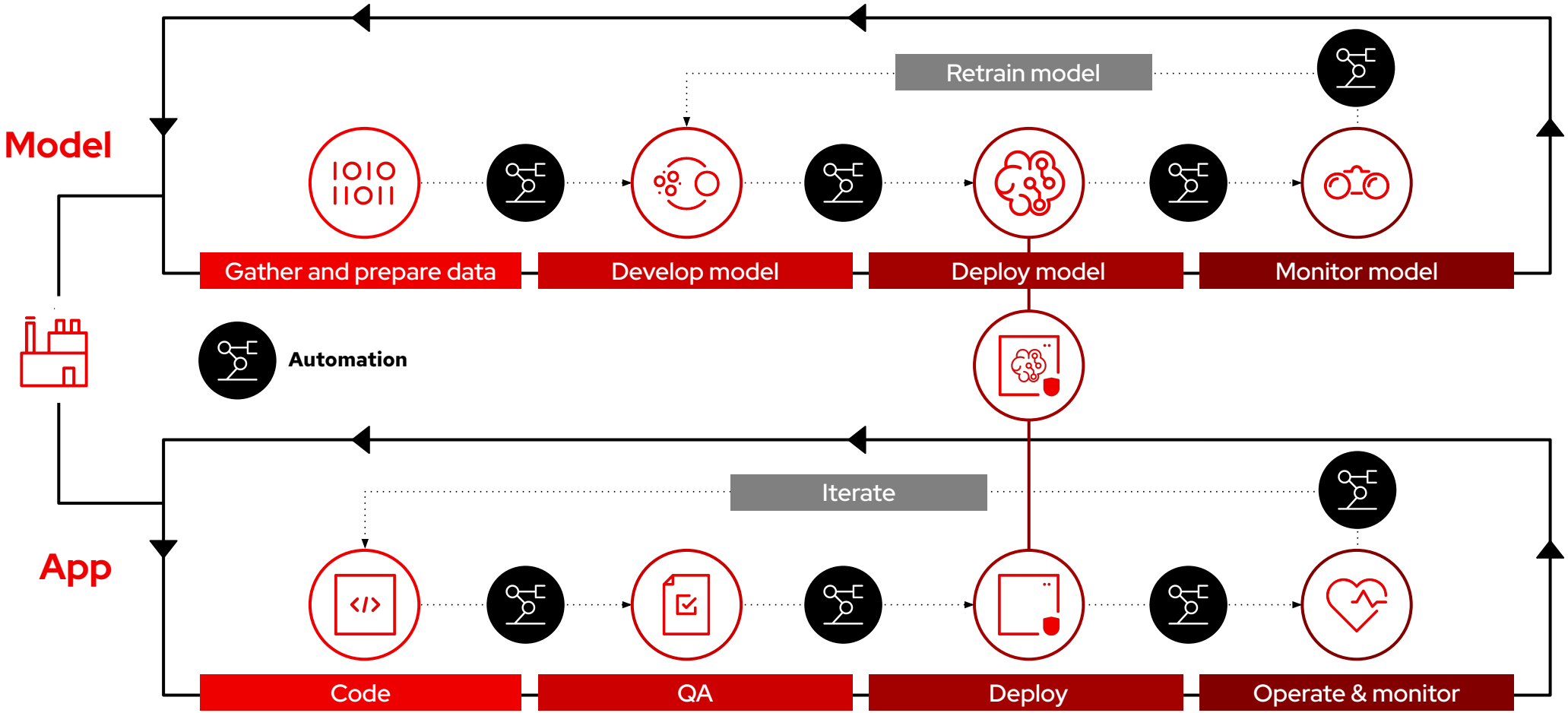
- **Recommendation engines**
  Netflix, Amazon, etc.
- **Virtual assistant**
  Siri, Alexa, etc.
- **Detecting fraudulent activity**
  Money laundering, spam, hacking, insurance
- **Quantifying risks and making smart decisions**
  Insurance, loans
- **Pattern detection**
  Images, videos: how many cars, humans, etc.
- **Analyze specialized data**
  Seismic data for oil and gas
- **Teach AI to play video games**
  AI opponents
- **Text analysis**
  Summarization, accuracy, offensive, plagiarism detection
- **Medical**
  Tumour detection
- **Customer retention**
  Predict who's about to leave

Architecture CI/CD for AI/ML application.

Red Hat


Retrain model
Model
Gather and prepare data
Develop model
Deploy model
Monitor model
Iterate
App
Code
QA
Deploy
Operate & monitor

Red Hat


Model
Retrain model
Gather and prepare data
Develop model
Deploy model
Monitor model
Automation
App
Iterate
Code
QA
Deploy
Operate & monitor

Red Hat

Implementation CI/CD for AI/ML part of application.

Red Hat

Red Hat OpenShift AI

Integrated AI platform

Create and deliver gen AI and predictive models at scale across hybrid cloud environments.

Model development

Bring your own models or customize Granite models to your use case with your data. Supports integration of multiple AI/ML libraries, frameworks, and runtimes.

Model serving and monitoring

Deploy models across any OpenShift footprint and centrally monitor their performance.

Lifecycle management

Expand DevOps practices to MLOps to manage the entire AI/ML lifecycle.

Resource optimization and management

Scale to meet workload demands of gen AI and predictive models. Share resources, projects, and models across environments.

Available as

- Fully managed cloud service
- Traditional software product on-site or in the cloud!

14

Red Hat OpenShift AI

Dashboard Application

- Data Science Projects
- Admin Features
- Model Registry

Model Development, Training & Tuning

Workbenches

- Minimal Python
- PyTorch
- CUDA
- Standard Data Science
- TensorFlow
- VS Code
- RStudio
- TrustyAI

CodeFlare SDK

ISV images

Custom images

Distributed workloads

KubeRay

Kueue

CodeFlare

Models

RHEL AI

Ecosystem models

Data and model Pipelines

Model Serving

Serving Engines

Kserve

ModelMesh

Serving Runtimes

OVMS

vLLM, Caikit/TGIS

Custom

Model Monitoring

Performance metrics

Operations metrics

Quality metrics

Object Storage

OpenShift Operators

- OpenShift GitOps
- OpenShift Pipelines
- OpenShift ServiceMesh
- OpenShift Serverless
- Prometheus

Red Hat OpenShift

Red Hat

Red Hat OpenShift AI

Dashboard Application

- Data Science Projects
- Admin Features
- Model Registry

Model Development, Training & Tuning

Workbenches

- Minimal Python
- PyTorch
- CUDA
- Standard Data Science
- TensorFlow
- VS Code
- RStudio
- TrustyAI

CodeFlare SDK

ISV images

Custom images

Distributed workloads

- KubeRay
- Kueue
- CodeFlare

Models

- RHEL AI
- Ecosystem models

Data and model Pipelines

Model Serving

Serving Engines

- Kserve
- ModelMesh

Serving Runtimes

- OVMS
- vLLM, Caikit/TGIS
- Custom

Model Monitoring

- Performance metrics
- Operations metrics
- Quality metrics

Object Storage

OpenShift Operators

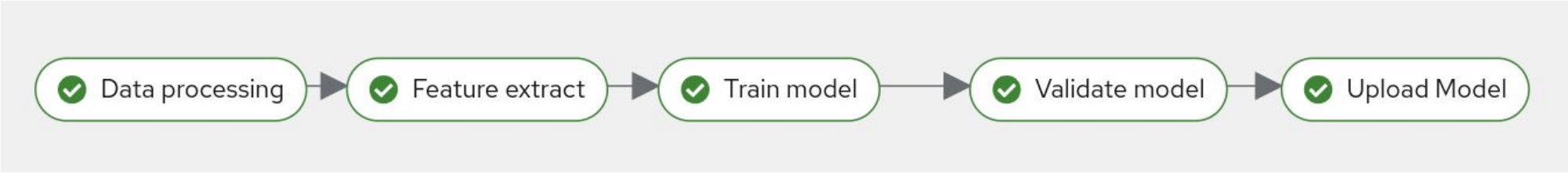
- OpenShift GitOps
- OpenShift Pipelines
- OpenShift ServiceMesh
- OpenShift Serverless
- Prometheus

Red Hat OpenShift

Red Hat

Data Science Pipelines

- Portable ML workflows to automate end-to-end ML tasks.
- Enables continuous integration and deployment of machine learning operations in staging and production.
- Based on **Kubeflow pipelines v2.0**. This internally leverages **Argo Workflows** to run the ML workflows.
- Example:
  - Here is a sample workflow that automates the ML tasks of processing data, extracting features from the data, train the ml model, validate it and upload the model to s3 object store.



Components

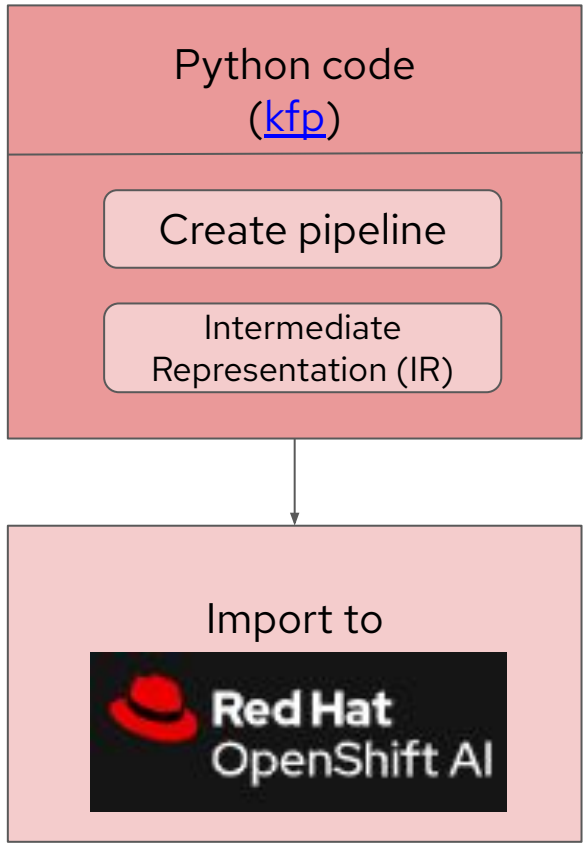
- **Pipeline Server**
  - A server that is attached to your data science project and hosts your data science pipeline.
  - Requires S3-compatible data connection to store your pipeline artifacts.
- **Pipeline**
  - A pipeline defines the configuration of your machine learning workflow and the relationship between each component in the workflow.
    - Pipeline code: A definition of your pipeline in a KubeFlow-formatted YAML file.
    - Pipeline graph (using Elyra GUI): A graphical illustration of the steps executed in a pipeline run and the relationship between them.
- **Pipeline run**: An execution of your pipeline.
  - Triggered run: A previously executed pipeline run.
  - Scheduled run: A pipeline run scheduled to execute at least once.

1. Using Kubeflow Pipelines SDK

ml_train_upload.py

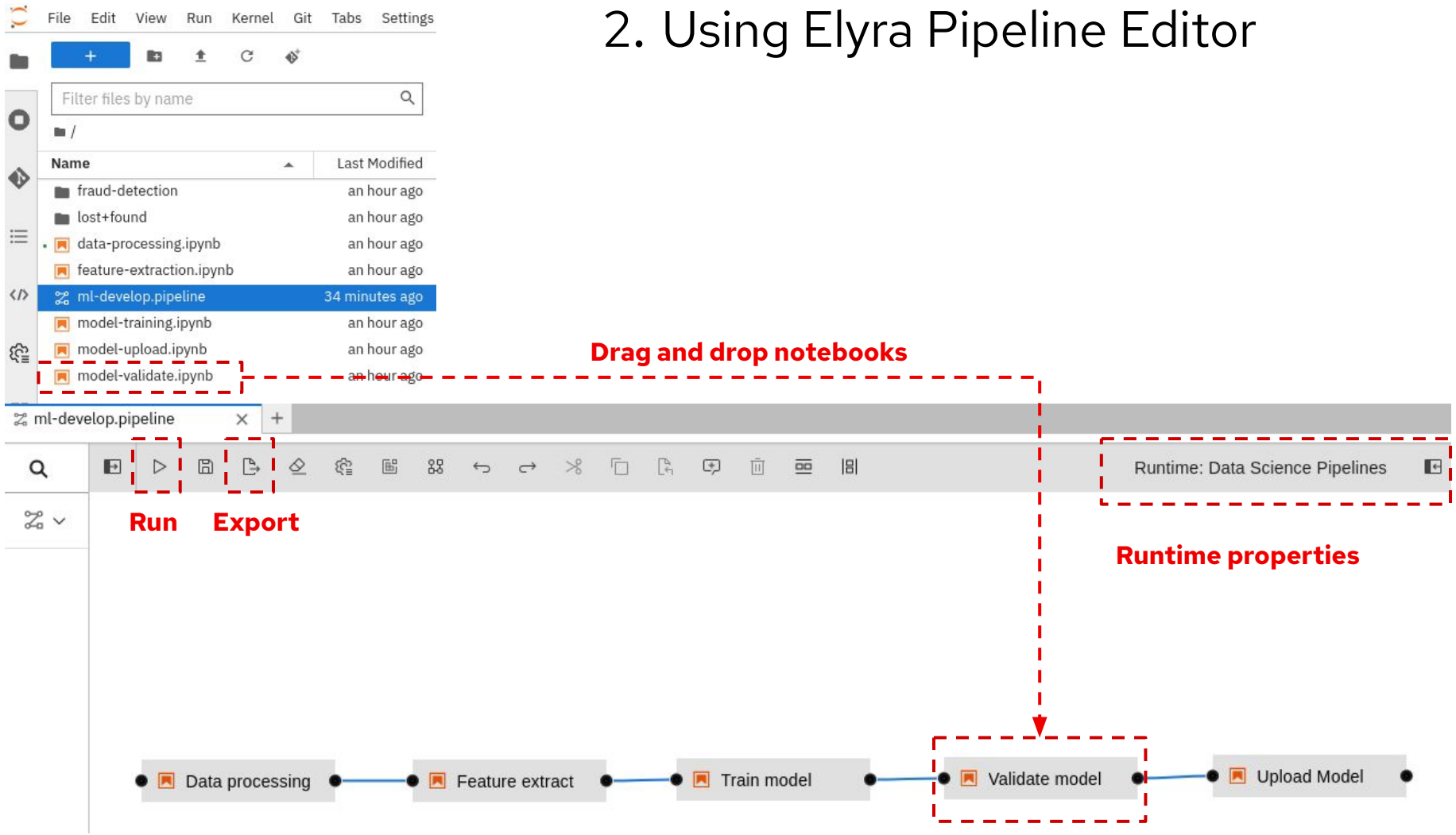
```
import kfp
from kfp.components import create_component_from_func

get_data_component = create_component_from_func(
                        get_data,
                        base_image=...",
                        packages_to_install=[ ]
)
@kfp.dsl.pipeline(name="train_upload_stock_kfp")
def sdk_pipeline():
        get_data_task = get_data_component()
        ...
from kfp_tekton.compiler import TektonCompiler
...
TektonCompiler().compile(sdk_pipeline, __file__.replace(".py", ".yaml"))
```



jupyter Pipeline Editor

2. Using Elyra Pipeline Editor



Red Hat

Distributed Workloads

Red Hat OpenShift AI

Admin

- Applications
- Data Science Projects
- Data Science Pipelines
- Distributed Workload Metrics
- Model Serving
- Resources
- Settings

Distributed Workload Metrics

Monitor the metrics of your active resources.

Project: Insurance Claims

Project metrics | Distributed workload status

Refresh interval: 5 minutes

Top resource-consuming workloads

CPU: 857 cores — Tuning, Accident Seve..., Claims Paymen...

Memory: 6220 GiB — Tuning, Accident Seve..., Claims Paymen...

Accelerator: 90 accelerators — Tuning, Accident Seve..., Claims Paymen..

Accelerator memory: 7461 GiB — Tuning, Accident Seve..., Claims Paymen...

Workload resource metrics

Name | Search by name | 1 - 5 of 24 | 1 of 5

Name	CPU usage (cores)	Memory usage (GiB)	Accelerator usage (GPUs)	Accelerator memory usage (GiB)	Status
Tuning	657 / 768	3720 / 7450	64 / 64	4761 / 4761	Running
Claims Payment M...	100 / 100	1250 / 1250	13 / 13	1350 / 1350	Running

Overview | Jobs | Serve | Cluster | Actors | Metrics | Logs

Cluster utilization

Set up Prometheus and Grafana for better Ray Dashboard experience

Time-series charts are hidden because either Prometheus or Grafana server is not detected. Follow these instructions to set them up and refresh this page.

Recent jobs

raysubmit_GRzZz9316FWqFNuQ (03000000)
python train_tf_cpu.py

View all jobs →

Serve Deployments

No Deployments yet...

View all deployments →

Cluster status and autoscaler

Node count

Set up Prometheus and Grafana for better Ray Dashboard experience

Time-series charts are hidden because either Prometheus or Grafana server is not detected. Follow these instructions to set them up and refresh this page.

Node Status

Active:
1 node_731b9ecca32196eefe3779746e42fea7d2bef09fb431c5ec93d4e12f
1 node_8700da8485c309538d41ba758dd1a593c004c1f5131e64ad3449a85c
1 node_fb554d4537099b7b36b0b0565876b1d24fb6fae4081f8e5391c6fd88
Pending:
(no pending nodes)
Recent failures:
(no failures)

Resource Status

Usage:
0.0/10.0 CPU
0B/14.90GiB memory
0B/4.09GiB object_store_memory
Demands:
(no resource demands)

Overview | Jobs | Serve | Cluster | Actors | Metrics | Logs

Jobs / 03000000

Entrypoint: python train_tf_cpu.py

Status: SUCCEEDED View details

Job ID: 03000000

Submission ID: raysubmit_GRzZz9316FWqFNuQ

Duration: 4m 30s

Started at: 2024/11/14 14:50:35

Ended at: 2024/11/14 14:55:05

Runtime environment: View

User-provided metadata: -

Actions
- Stack Trace (Driver)
- CPU Flame Graph (Driver)
- Memory Profiling (Driver)

Ray Data Overview

Ray Core Overview

Total: 369 | Finished: 369 | Hide finished

369 / 369

Red Hat

Summary

Red Hat

Red Hat Summit

Connect

Thank you

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat