

# What's the deal with Managed Services and Model Delivery?

Technical Overview

Audrey Reznik

Data Scientist

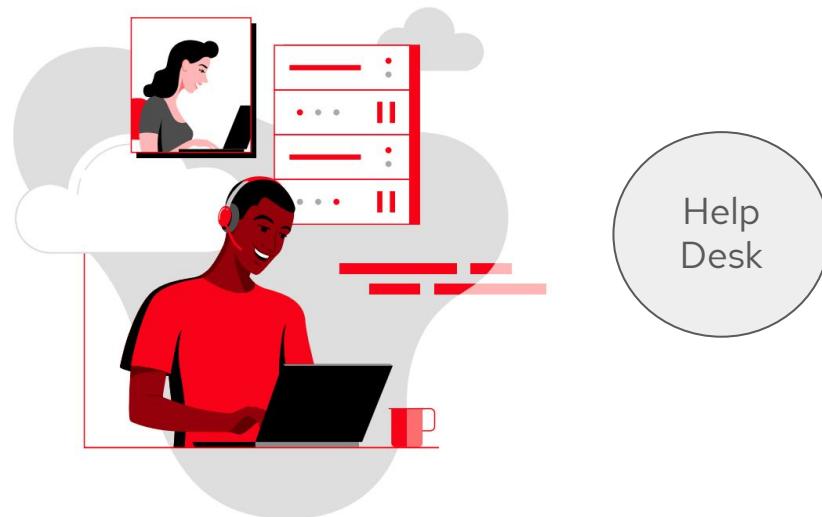
## Agenda:

1. What are Managed Services?
  - a. Who cares about them?
2. Where do I find Managed Services?
3. What do Managed Services have to do with Model Delivery?
  - a. Use case
4. Are Managed Services easy to use?

# Generic IT Managed Services



# Generic IT Managed Services



# Generic IT Managed Services



Help  
Desk

Equipment  
installation

# Generic IT Managed Services

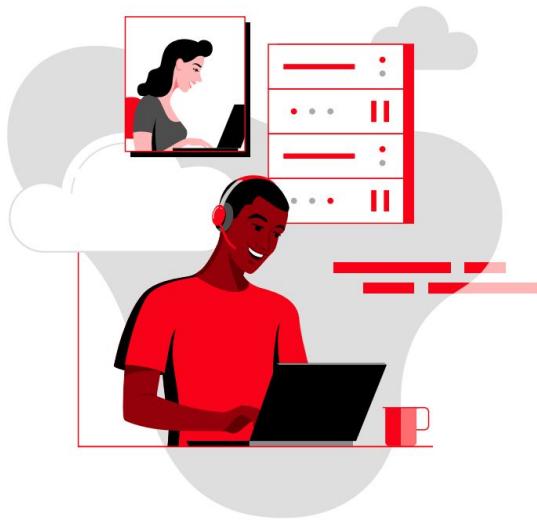


Help  
Desk

Equipment  
installation

Hardware  
maintenance

# Generic IT Managed Services



Help  
Desk

Equipment  
installation

Hardware  
maintenance

Firewall  
Security

# Generic IT Managed Services



Help  
Desk

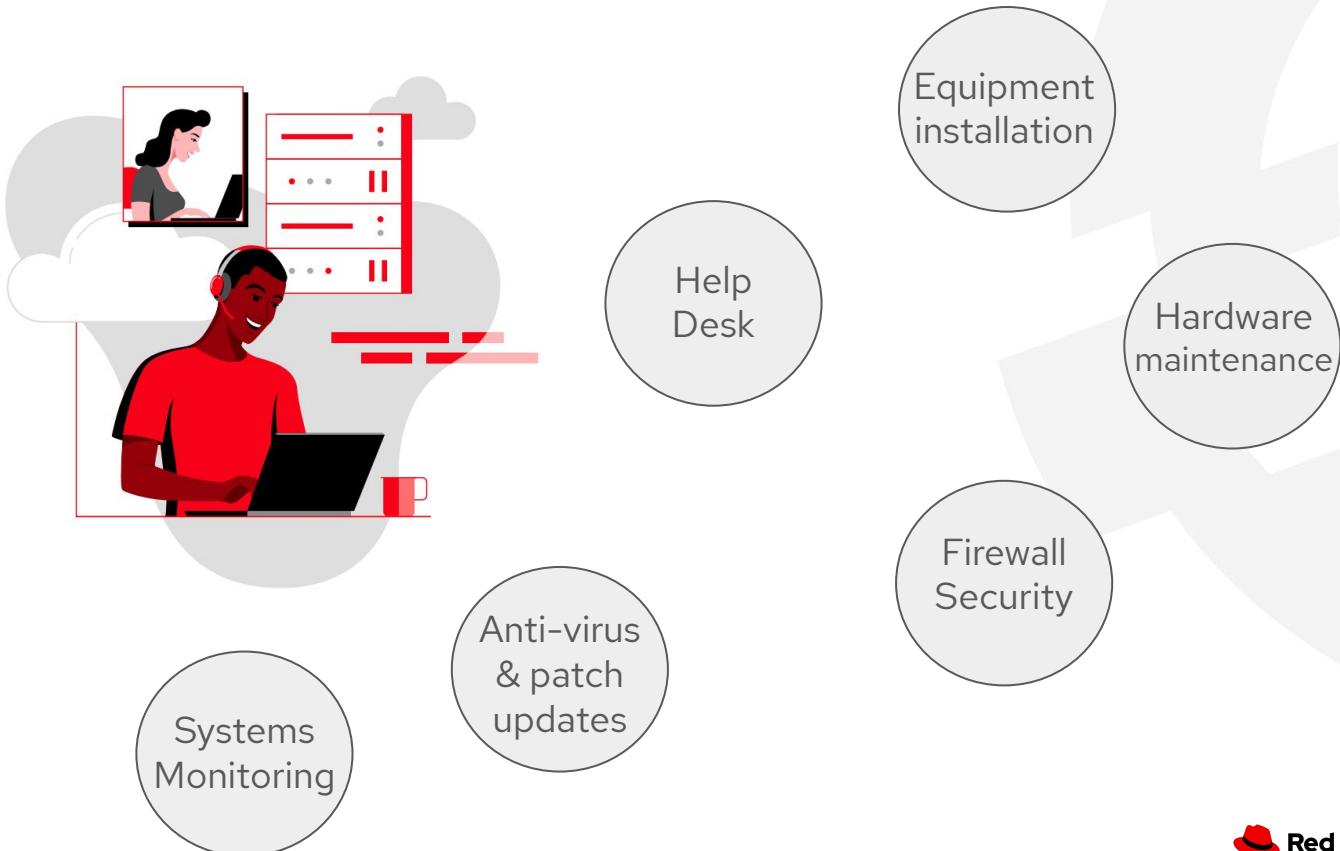
Equipment  
installation

Hardware  
maintenance

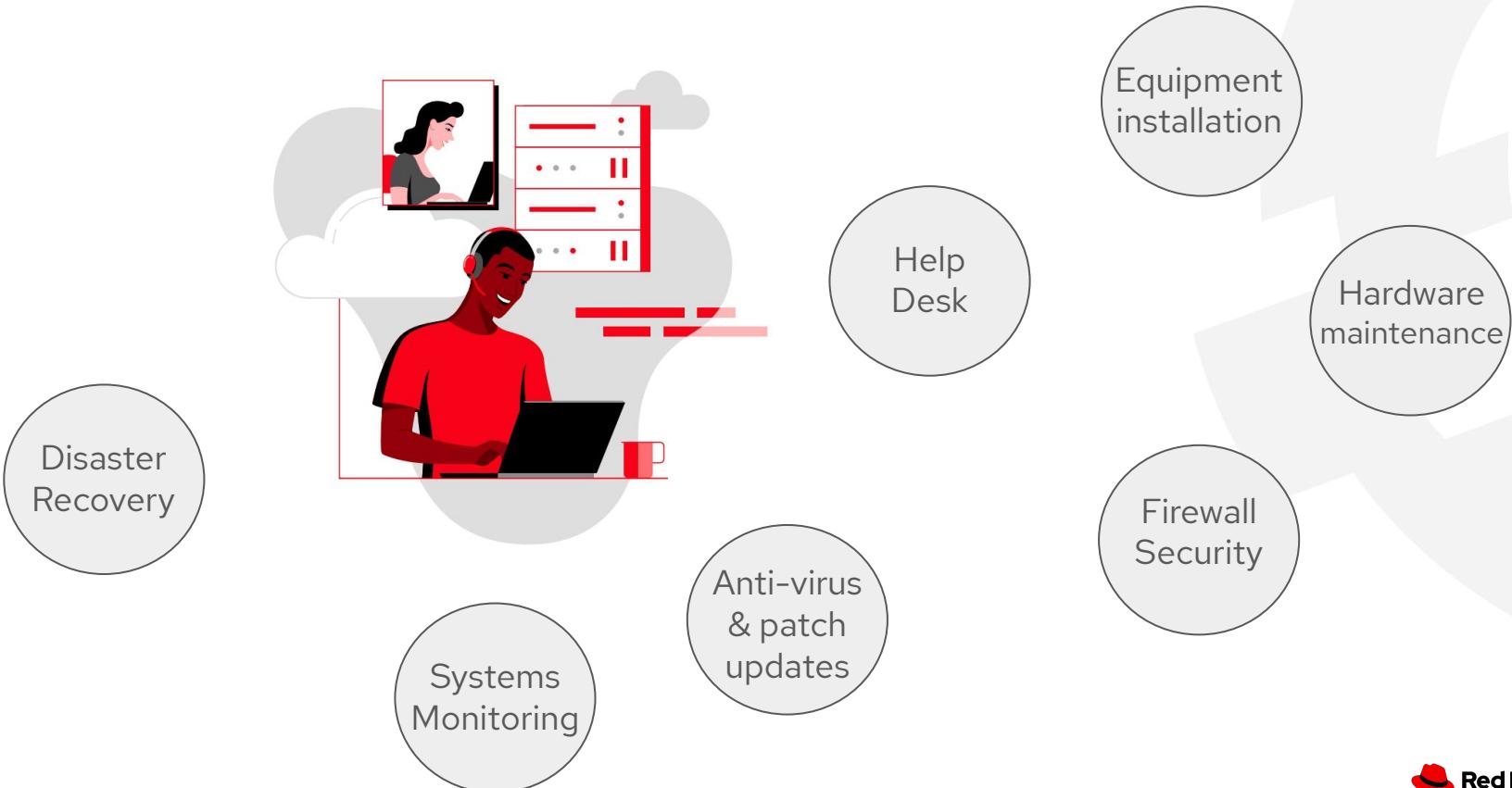
Firewall  
Security

Anti-virus  
& patch  
updates

# Generic IT Managed Services



# Generic IT Managed Services



# Generic IT Managed Services

Managed Backups

Disaster Recovery

Systems Monitoring

Anti-virus & patch updates

Equipment installation

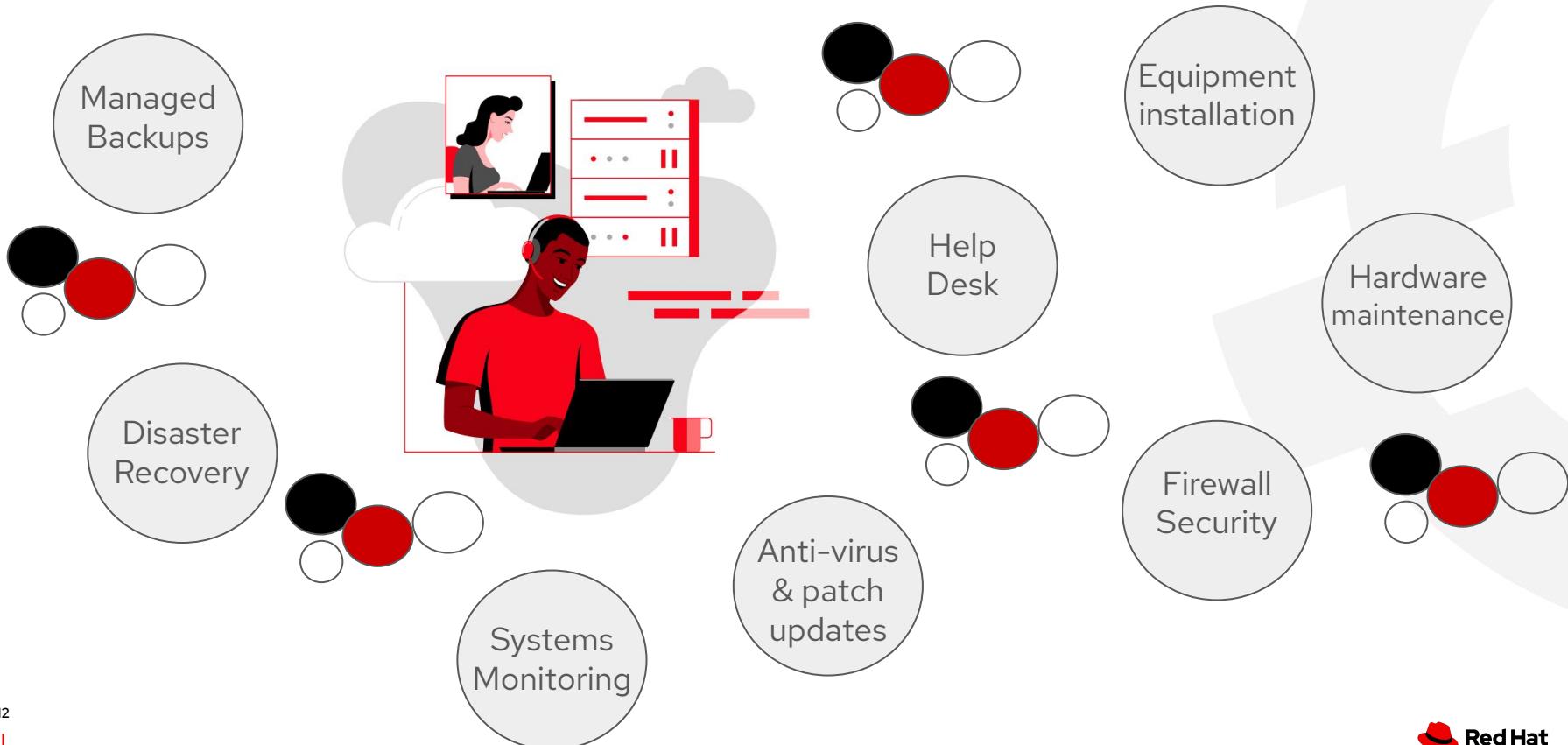
Hardware maintenance

Firewall Security

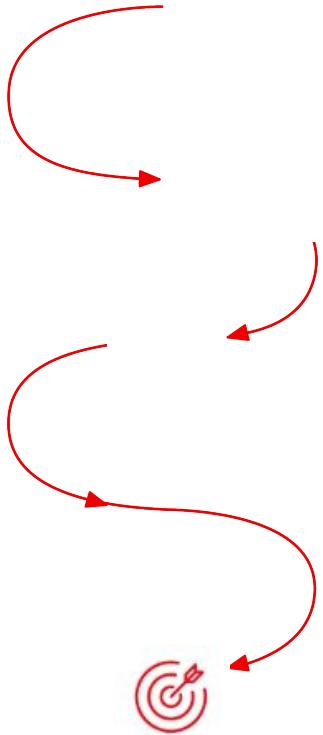


Help Desk

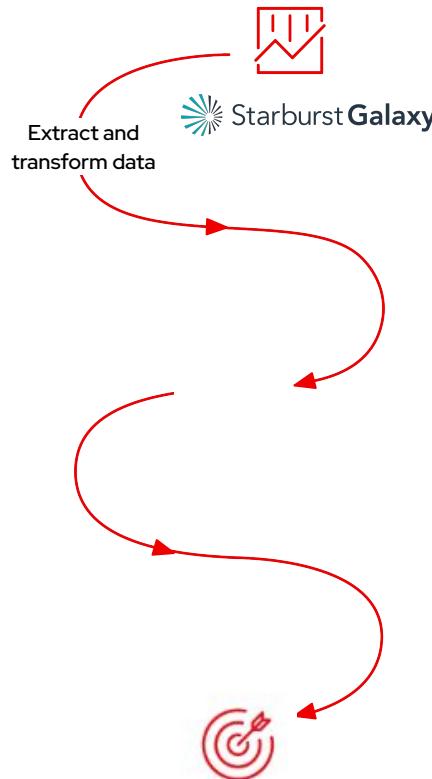
# Generic IT Managed Services



# Managed Services for Data Scientists



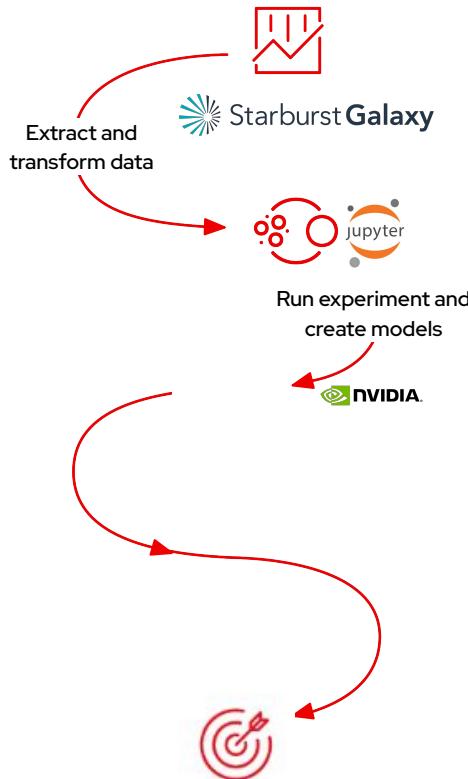
# Managed Services for Data Scientists



## Extract & Transform the Data

Unlock the value of your data by making it fast and easy to access data across hybrid cloud.

# Managed Services for Data Scientists



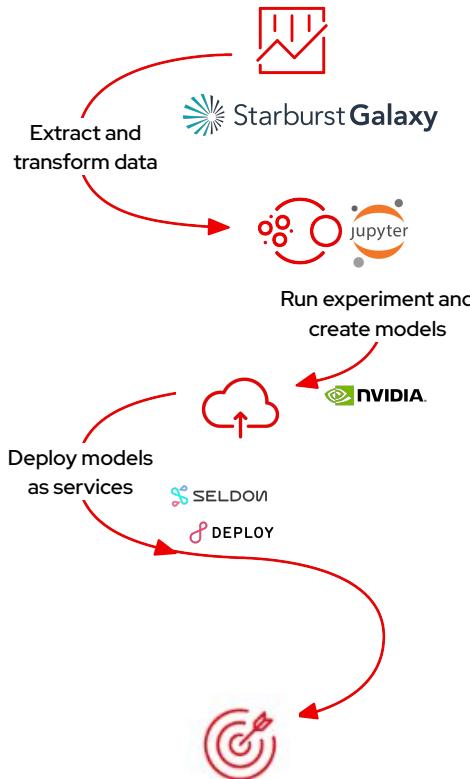
## Extract & Transform the Data

Unlock the value of your data by making it fast and easy to access data across hybrid cloud.

## Run experiments and create models

Would like to run experiments and build / run models using various data science library packages. And... open source products!

# Managed Services for Data Scientists



## Extract & Transform the Data

Unlock the value of your data by making it fast and easy to access data across hybrid cloud.

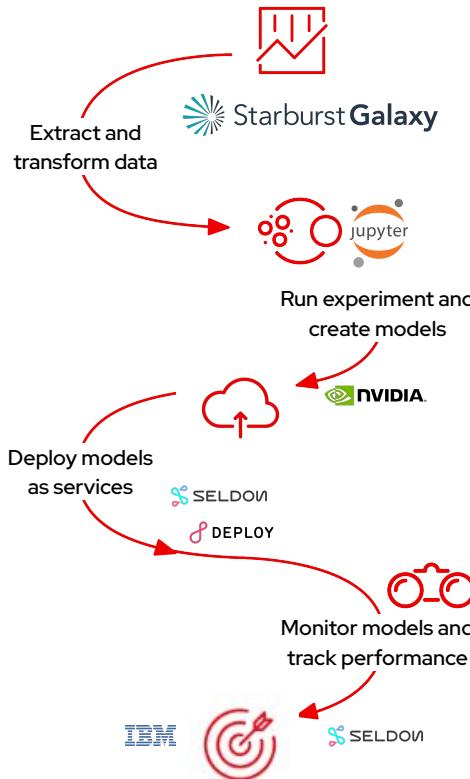
## Run experiments and create models

Would like to run experiments and build / run models using various data science library packages. And... open source products!

## Deploy models as services

Simplify and accelerate the process of deploying and managing your machine learning models.

# Managed Services for Data Scientists



## Extract & Transform the Data

Unlock the value of your data by making it fast and easy to access data across hybrid cloud.

## Run experiments and create models

Would like to run experiments and build / run models using various data science library packages. And... open source products!

## Deploy models as services

Simplify and accelerate the process of deploying and managing your machine learning models.

## Monitor models & track performance

Monitor model performance and gleam meaningful analytics from the model

# Who cares about Data Science Managed Services?

**Data Scientists, Data Engineers and the ITOps that support them.**



# Who Cares about Data Science Managed Services?

**Data Scientists, Data Engineers and the ITOps that support them.**



**What else (besides Managed Services) should they care about?**

1. AIML model operational lifecycle
2. Production Ready Platform (that ITOps feels good about)
3. Flexibility to use available Open Source services
4. Ability to deploy and Portability to move from the platform you initially developed on.

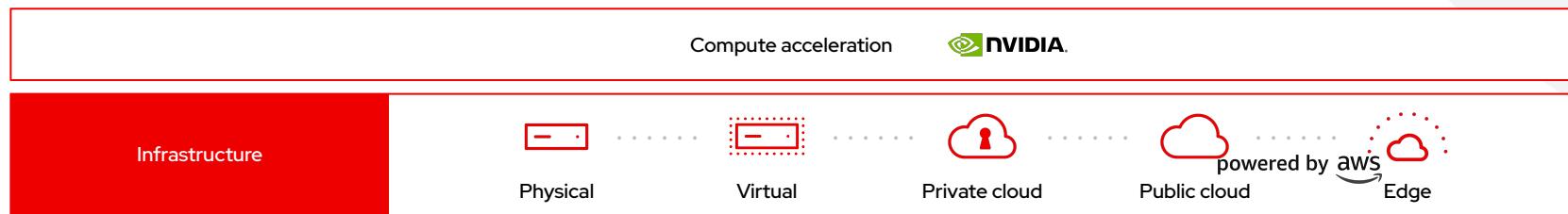
---

Let's build a platform that  
fits our Managed Services  
requirements!

# Infrastructure

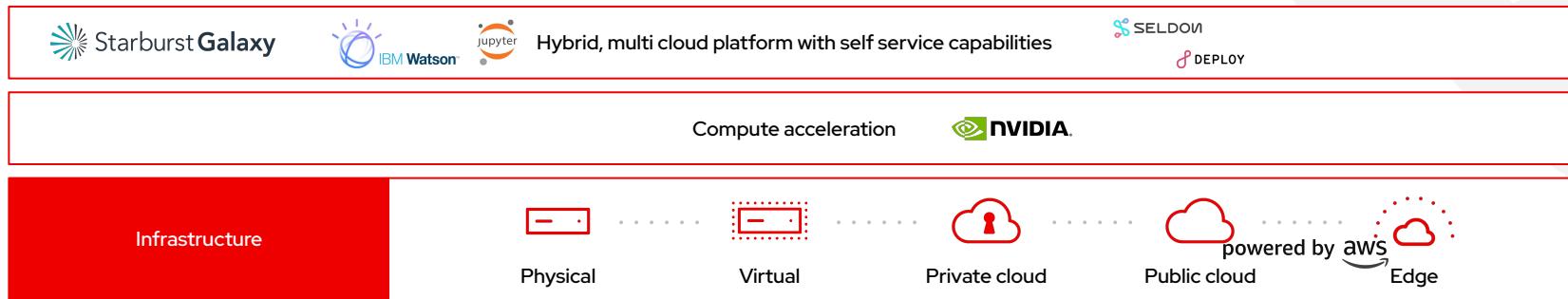


# Compute Acceleration



# Hybrid, multi cloud platform with self (managed) services

23



# Data Science Managed Services and Model Delivery



..... ➤

## Gather and prepare data

- Data storage
- Data lake
- Data exploration
- Data preparation
- Stream processing



Hybrid, multi cloud platform with self service capabilities



24

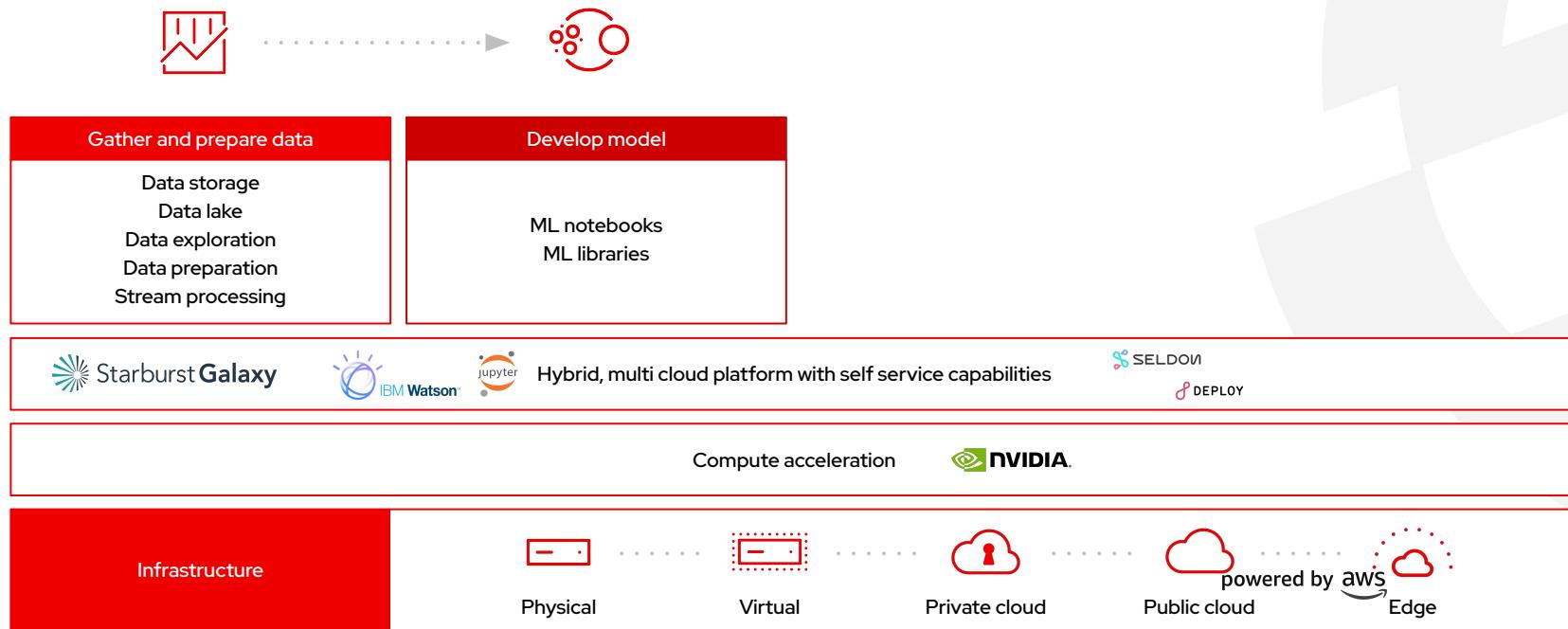
## Compute acceleration



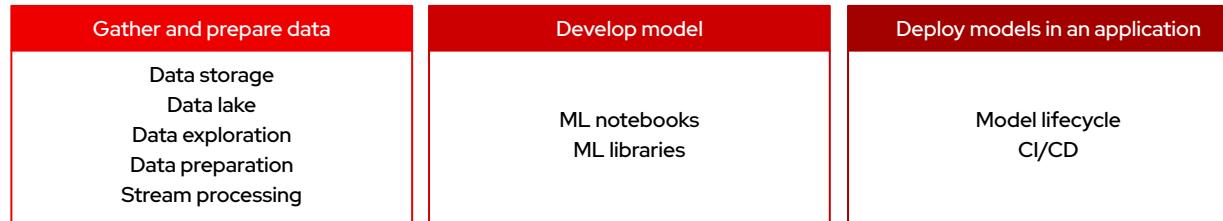
## Infrastructure



# Data Science Managed Services and Model Delivery



# Data Science Managed Services and Model Delivery



Hybrid, multi cloud platform with self service capabilities



26

Compute acceleration



Infrastructure



Physical



Virtual



Private cloud

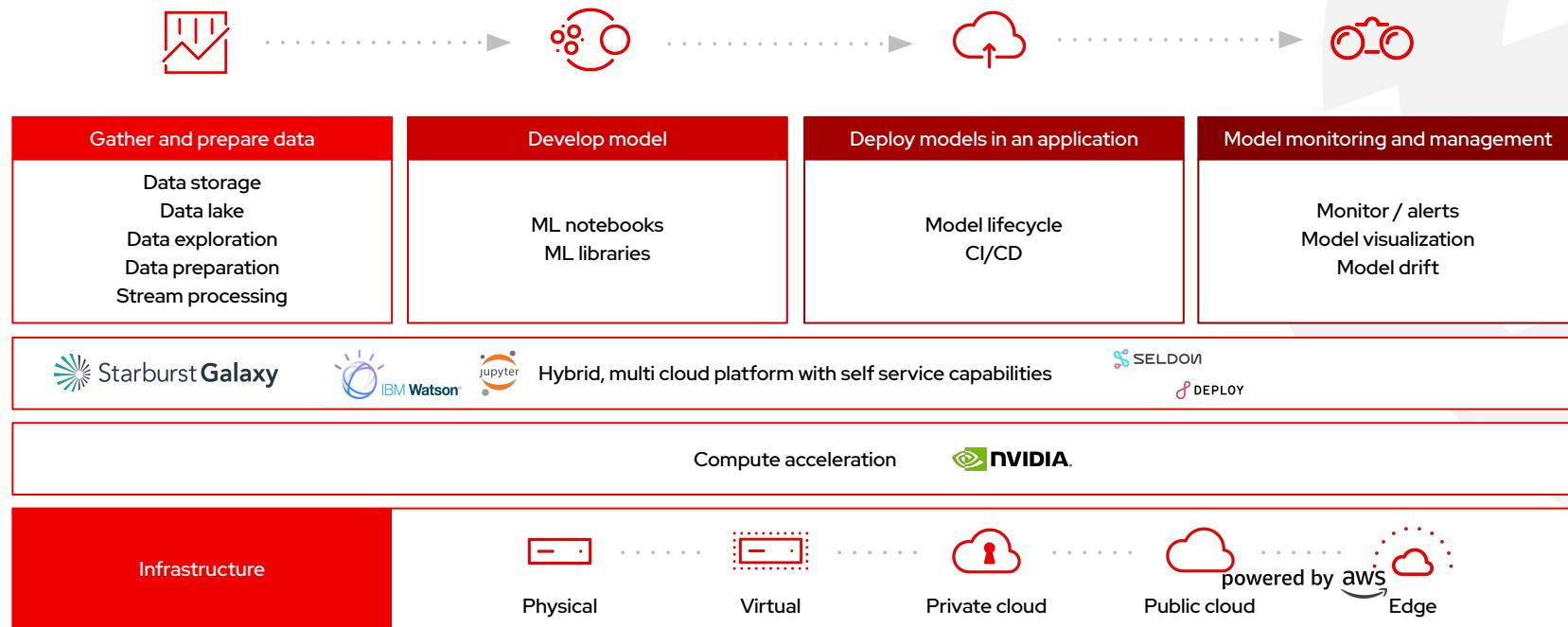


Public cloud

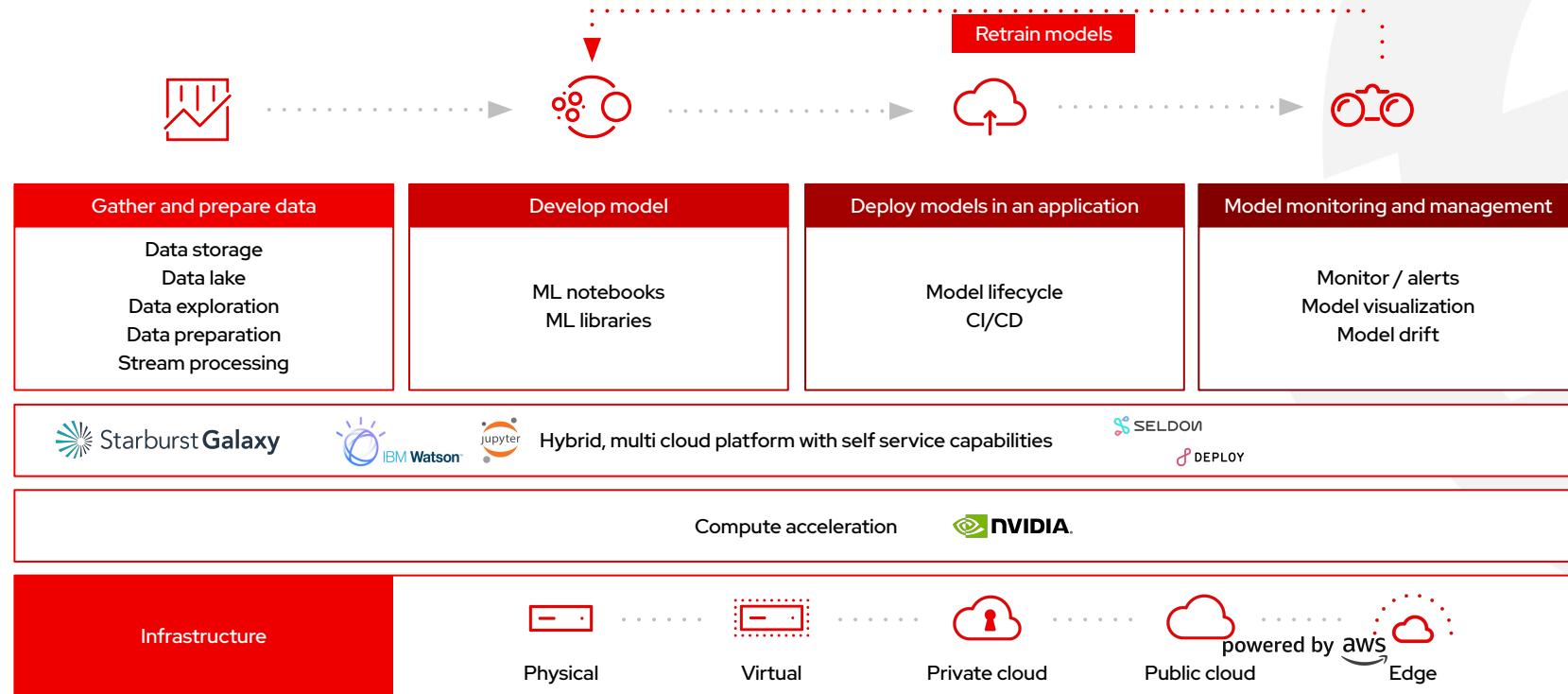


Edge

# Data Science Managed Services and Model Delivery



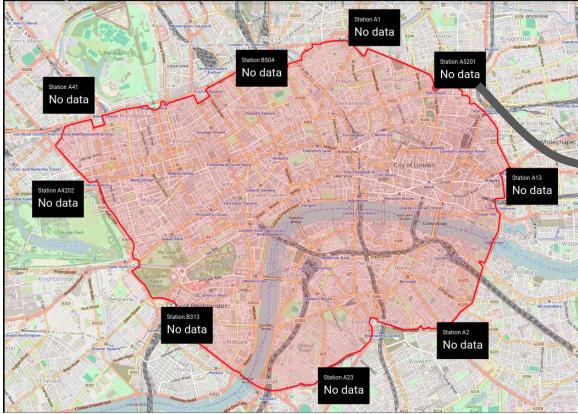
# Data Science Managed Services and Model Delivery



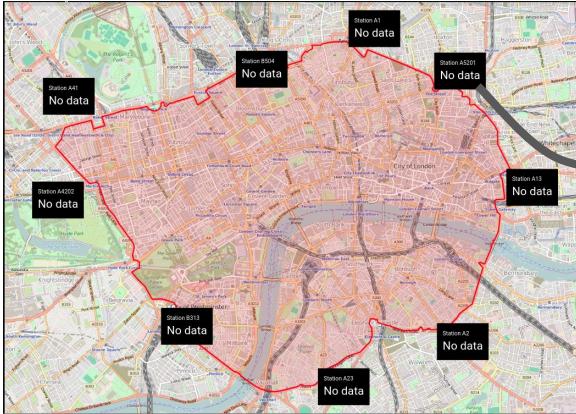
---

# AIML Use case

# License Plate Detection

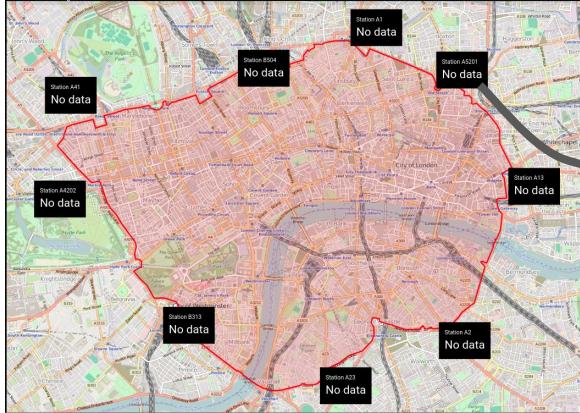


# License Plate Detection



AKI8BZL

# License Plate Detection



**AKI8BZL**

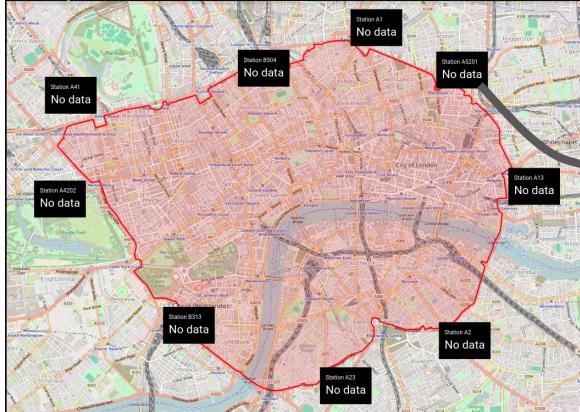
Kafka

MirrorMaker

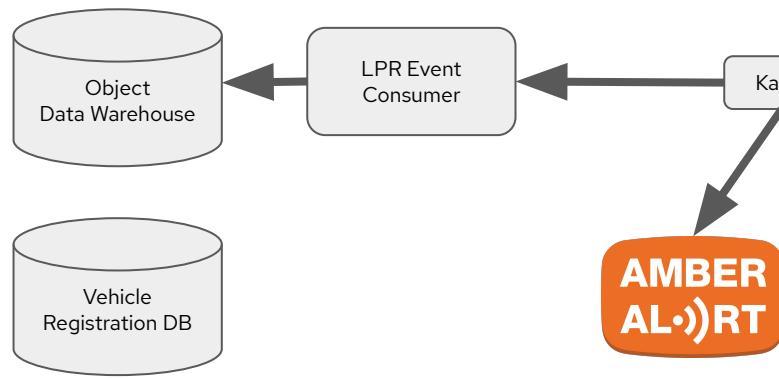
Kafka

**AMBER  
ALERT**

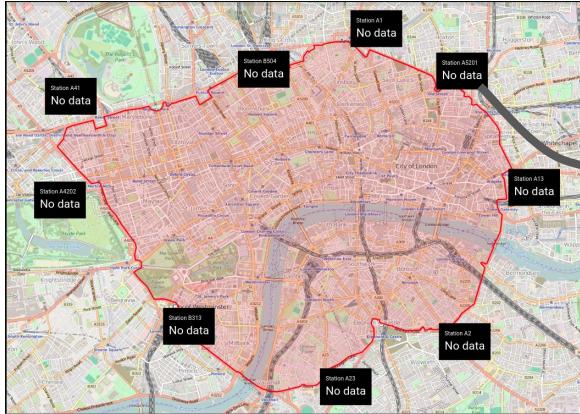
# License Plate Detection



AKI8BZL



# License Plate Detection



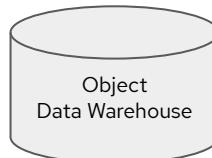
**AKI8BZL**

Kafka

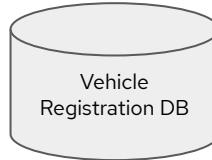
MirrorMaker



BI +  
Analytics  
Tools



Object  
Data Warehouse



Vehicle  
Registration DB



LPR Event  
Consumer

**AMBER  
ALERT**

Kafka

# Confession Time

# Red Hat Data Science Managed Services - Resources

The screenshot shows the Red Hat OpenShift Data Science interface with the 'Resources' tab selected. The page displays a grid of 17 items out of 43, each representing a learning path or quick start guide. The items include:

- Anaconda Commercial Edition**: Documentation. Quick start (10 minutes). Description: The world's most popular open-source package distribution and management experience. Optimized and supported for commercial use.
- Connecting to Red Hat OpenShift Streams for Apache Kafka**: Documentation. Quick start (10 minutes). Description: This quick start will walk you through connecting to Red Hat Streams for Apache Kafka using Jupyter Notebooks.
- Creating a Jupyter notebook**: Documentation. Quick start (5 minutes). Description: This quick start will walk you through creating a Jupyter notebook.
- Creating an Anaconda-enabled Jupyter notebook**: Documentation. Quick start (5 minutes). Description: This quick start will walk you through creating an Anaconda-enabled Jupyter notebook.
- Deploying a Model with Watson Studio**: Documentation. Quick start (15 minutes). Description: This quick start walks you through importing a Notebook in Watson Studio, building a model with AutoAI, and deploying a model.
- Deploying a sample Python application using Flask and OpenShift**: Documentation. Quick start (10 minutes). Description: How to deploy a Python model using Flask and OpenShift.
- IBM Watson Studio**: Documentation. Quick start (10 minutes). Description: Embed AI and machine learning into your business. Create custom models using your own data.
- JupyterHub**: Documentation. Quick start (10 minutes). Description: A multi-user version of the notebook designed for companies, classrooms and research labs.
- Launch a SKLearn model and update model by canarying**: Documentation. Quick start (10 minutes). Description: How to perform a canary promotion of a Scikit-Learn model.
- Monitor drift for deployed model**: Documentation. Quick start (10 minutes). Description: Monitor drift for deployed image classifier model.

# Red Hat OpenShift Data Science - Dashboard (user interface)

The screenshot displays two main sections of the Red Hat OpenShift Data Science dashboard:

- Enabled:** Shows a single application entry for "JupyterHub" (Red Hat managed). It is described as a multi-user version of the notebook designed for companies, classrooms, and research labs. Buttons for "Launch" and "Close tour" are present.
- Explore:** Shows a grid of optional programs:
  - Anaconda Commercial Edition:** Partner managed, self-managed. Described as the world's most popular open-source package distribution and management experience. Optimized and supported for commercial use.
  - IBM Watson Studio:** Self-managed. Embed AI and machine learning into your business. Create custom models using your own data.
  - JupyterHub:** Red Hat managed. A multi-user version of the notebook designed for companies, classrooms and research labs.
  - OpenShift Streams for Apache Kafka:** Red Hat managed. A managed cloud service for streaming data that reduces the operational cost and complexity of delivering real-time applications across hybrid-cloud environments.
  - Pachyderm:** Coming soon. An enterprise-grade, open source data science platform that makes explainable, repeatable, and scalable ML/AI a reality.
  - PerceptiLabs:** Coming soon. Accelerates machine learning by streamlining the workflow and advancing explainability of the models.
  - Seldon Deploy:** Self-managed. A specialist set of tools designed to simplify and accelerate the process of deploying and managing your machine learning models.
  - Starburst Enterprise:** Partner managed. Unlocks the value of all data by making it fast and easy to access anywhere.

A modal window titled "Creating a Jupyter notebook" is open, providing quick start instructions for creating a Jupyter notebook. It states: "This quick start shows you how to create a Jupyter notebook. Red Hat® OpenShift® Data Scientist lets you run Jupyter notebooks on our Red Hat® OpenShift Dedicated environment."

## Launch (Enable) the JupyterHub Managed Service

The screenshot shows the Red Hat OpenShift Data Science project dashboard. The left sidebar has a red box around the 'Enabled' tab under the 'Applications' section. The main content area shows three enabled managed services: Anaconda Commercial Edition (partner managed), IBM Watson Studio (self-managed), and JupyterHub (Red Hat managed). The JupyterHub card is also highlighted with a red box and has a 'Launch' button at the bottom.

Enabled

Launch your enabled applications or get started with quick start instructions and tasks.

Anaconda Commercial Edition  
Partner managed

The world's most popular open-source package distribution and management experience. Optimized and supported for commercial use.

IBM Watson Studio  
Self-managed

Embed AI and machine learning into your business. Create custom models using your own data.

JupyterHub Red Hat managed

A multi-user version of the notebook designed for companies, classrooms and research labs.

Start Launch 38 Restart

From the Applications menu, choose 'Enabled'. You will see 3 enabled managed services.

In the JupyterHub managed service, click the 'Launch' hyperlink.

# Create a JupyterHub notebook image

The screenshot shows the 'Start a notebook server' configuration page on the jupyterhub service. It includes sections for selecting a notebook image, specifying deployment size, defining environment variables, and starting the server.

**Notebook image:**

- CUDA ⓘ Python v3.8.3, CUDA 11.0.3
- Standard Data Science ⓘ Python v3.8.3
- Anaconda-enabled notebook. Requires Anaconda license key. ⓘ Anaconda-Python v3.8.5
- PyTorch ⓘ Python v3.8.3, PyTorch 1.8.1, CUDA 11.0.3
- Minimal Python ⓘ Python v3.8.3
- TensorFlow ⓘ Python v3.8.3, TensorFlow 2.4.1, CUDA 11.0.3

**Deployment size:**

Container size: Large

Number of GPUs: 1

**Environment variables:**

Custom variable: JUPYTER\_ENABLE\_LAB

Variable value: 1

Secret:

[Add more variables](#)

[Start server](#)

# Available notebook images

jupyterhub Home Token Admin Services ▾

Start a notebook server

Select options for your notebook server.

Notebook image

- CUDA ⓘ  
Python v3.8.3, CUDA 11.0.3
- Standard Data Science ⓘ  
Python v3.8.3
- Anaconda-enabled notebook. Requires Anaconda license key. ⓘ  
Anaconda-Python v3.8.5
- PyTorch ⓘ  
Python v3.8.3, PyTorch 1.8.1, CUDA 11.0.3
- Minimal Python ⓘ  
Python v3.8.3
- TensorFlow ⓘ  
Python v3.8.3, TensorFlow 2.4.1, CUDA 11.0.3

# Deployment size: Choosing a Container size

Deployment size

**Container size**

X Large

**Default**  
Resources set based on administrator configurations

**Small**  
Limits: 2 CPU, 8Gi Memory Requests: 1 CPU, 8Gi Memory

**Medium**  
Limits: 6 CPU, 24Gi Memory Requests: 3 CPU, 24Gi Memory

**Large**  
Limits: 14 CPU, 56Gi Memory Requests: 7 CPU, 56Gi Memory

X Large

Limits: 30 CPU, 120Gi Memory Requests: 15 CPU, 120Gi Memory

...

## Deployment size: adding a GPU

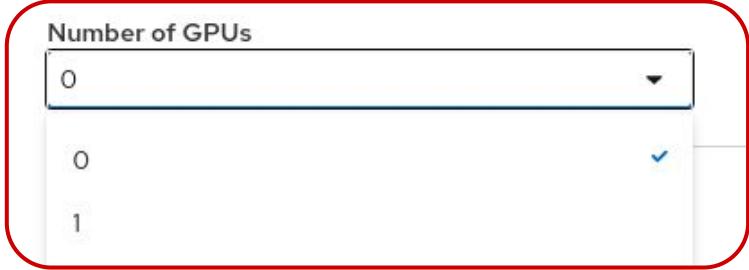
Deployment size

Container size

Large

Number of GPUs

0	▼
0	✓
1	



# Environment Variables

Environment variables

Custom variable -

Variable name S3\_ACCESS\_KEY\_ID

Variable value .....  Secret

Custom variable -

Variable name S3\_SECRET\_ACCESS\_KEY

Variable value .....  Secret

[+ Add more variables](#)

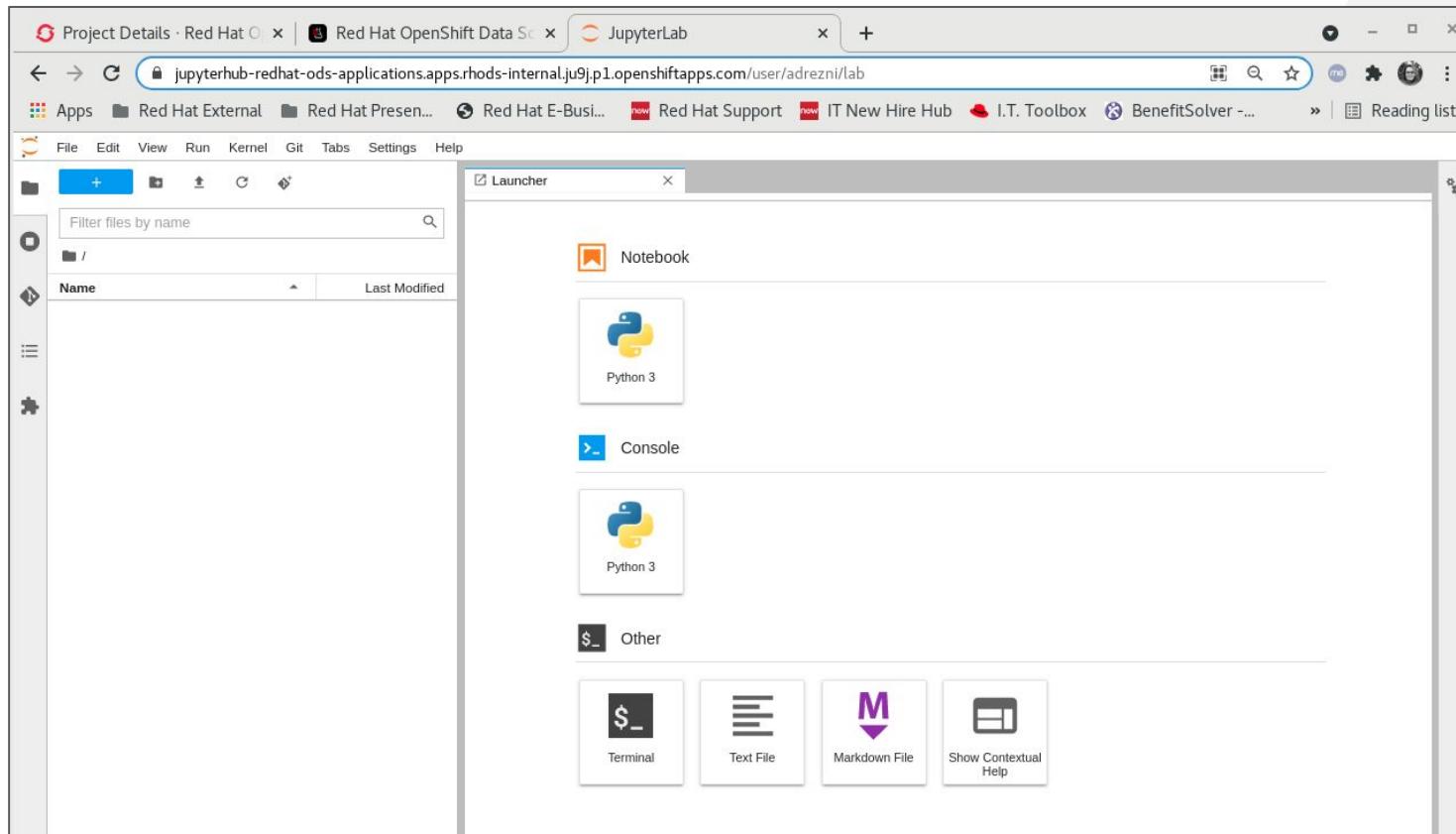
Start server

# Server Start up...

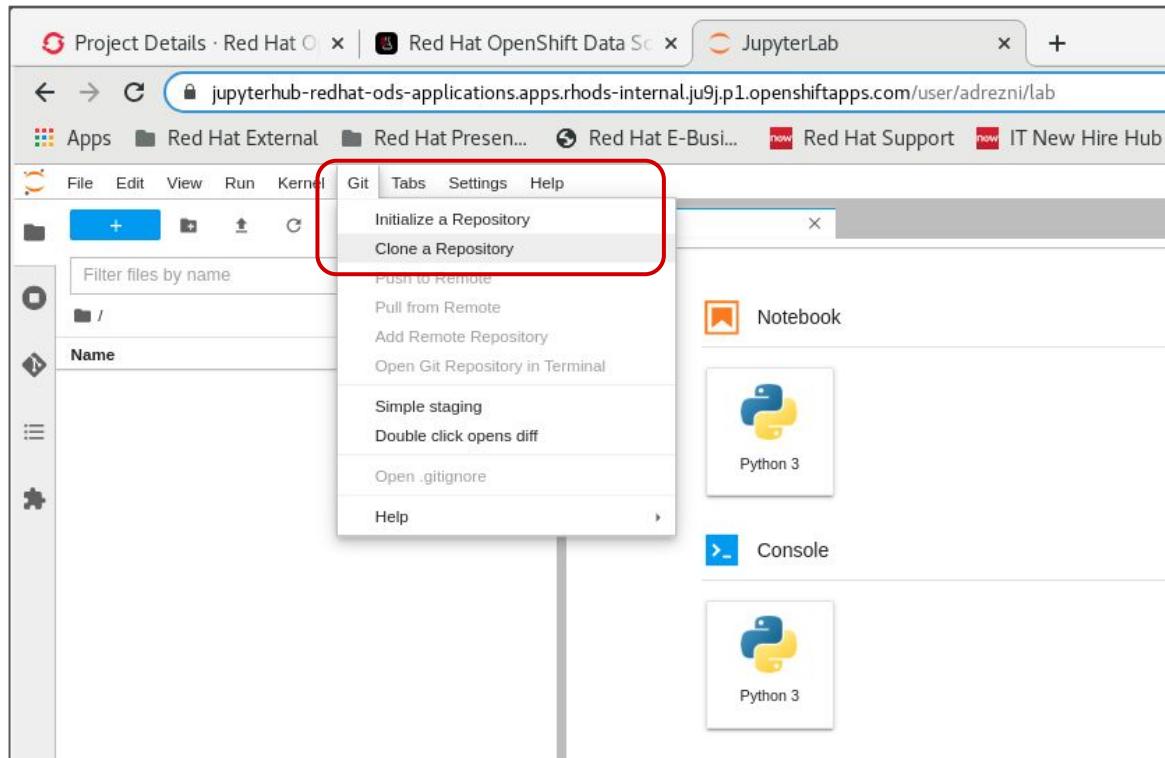
The screenshot shows a web browser window with the following details:

- Tab Bar:** Project Details · Red Hat, Red Hat OpenShift Data Sc..., JupyterHub.
- Address Bar:** jupyterhub-redhat-ods-applications.apps.rhods-internal.ju9.p1.openshiftapps.com/hub/spawn-pending/adrezni
- Header:** Apps, Red Hat External, Red Hat Presentations, Red Hat E-Business, Red Hat Support, IT New Hire Hub, I.T. Toolbox, BenefitSolver, Enabling Support, daily bookmark...
- Page Content:**
  - Header: Your server is starting up.
  - Text: You will be redirected automatically when it's ready for you.
  - Event log section:
    - 2021-05-10T03:25:49Z [Normal] AttachVolume.Attach succeeded for volume "pvc-63351e1a-8cac-4dab-8e34-1eb2610c5c40"
    - Server requested
    - 2021-05-10T03:25:46.161262Z [Normal] Successfully assigned redhat-ods-applications/jupyterhub-nb-adrezni to ip-10-0-207-219.ec2.internal
    - 2021-05-10T03:25:49Z [Normal] AttachVolume.Attach succeeded for volume "pvc-63351e1a-8cac-4dab-8e34-1eb2610c5c40"

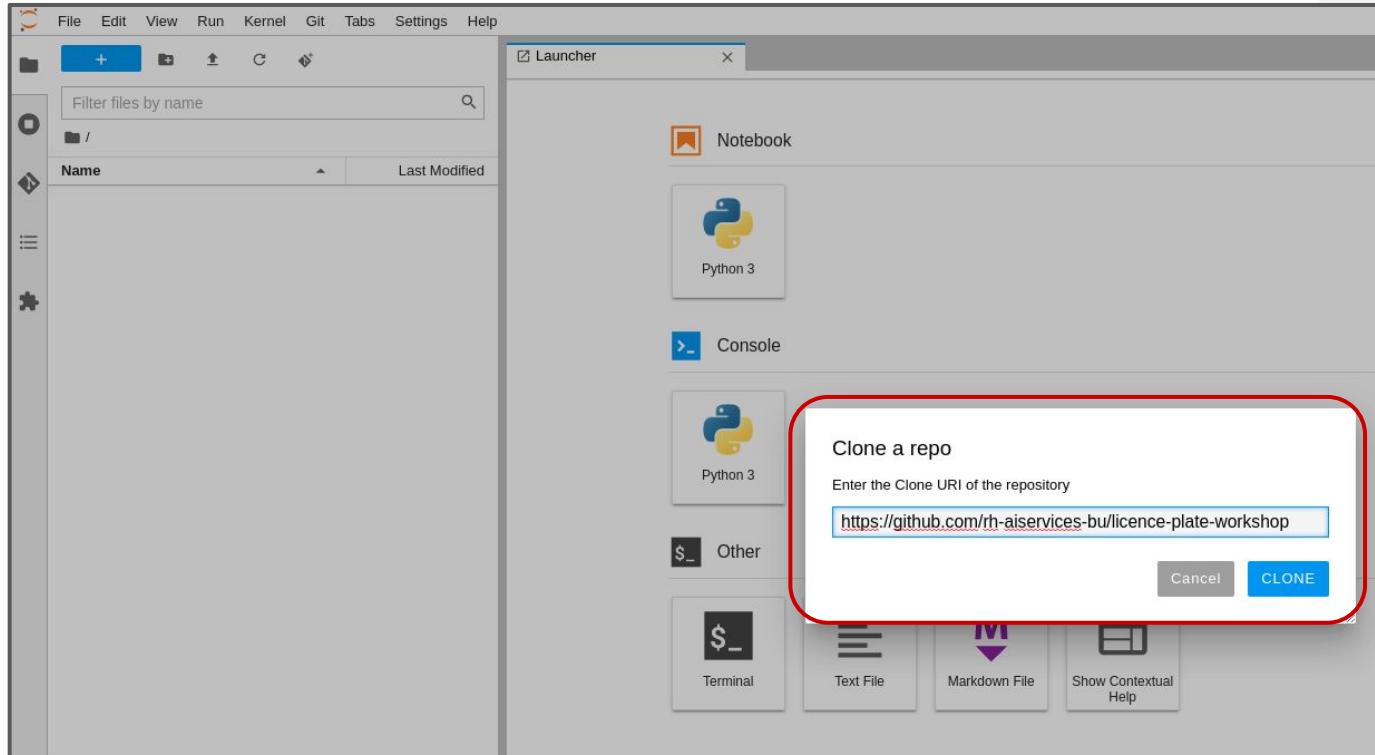
# Welcome to JupyterLab!



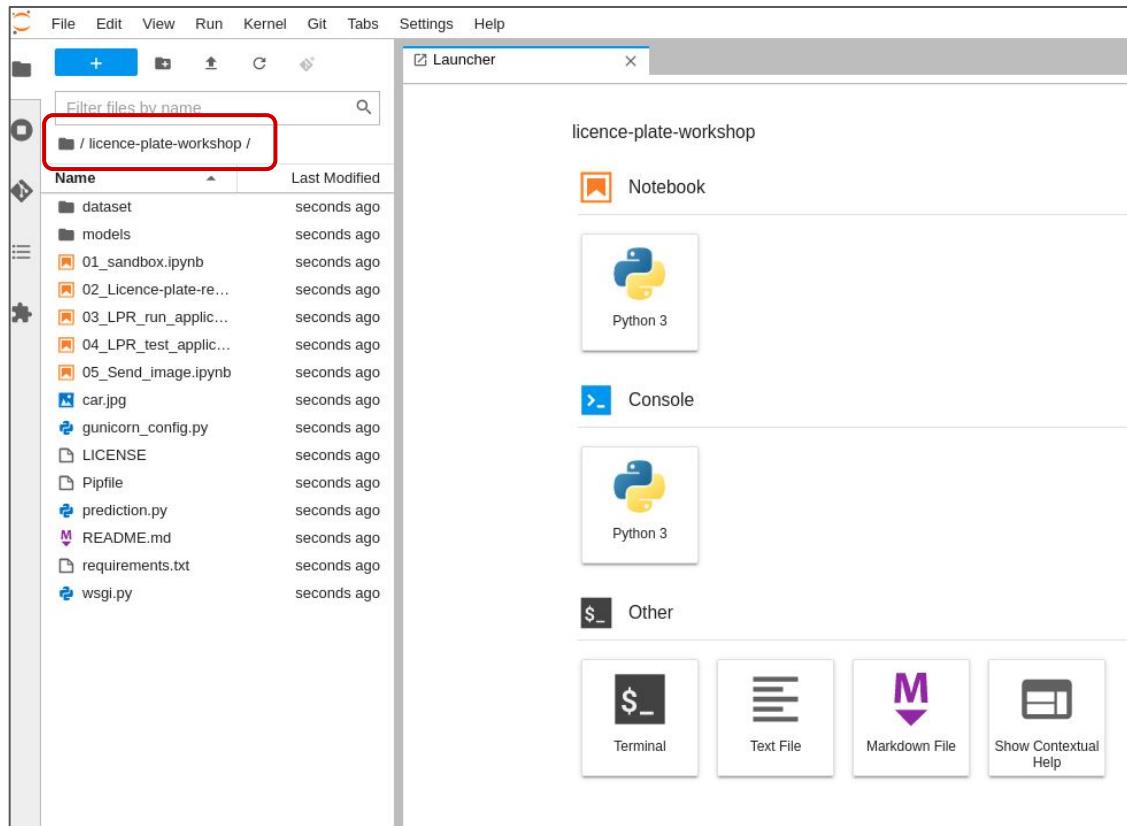
## Choose “Clone a Repository” from the “Git” menu



# Clone license plate GIT repo



# License plate repo



The '**license-plate-workshop**' repo files will appear under the "**Name**" pane.

# Ready to Experiment using a Jupyter Notebook

The screenshot shows a Jupyter Notebook environment with the following details:

- File Explorer:** On the left, it lists files in the directory `/licence-plate-workshop/`. The files include `dataset`, `models`, `01_sandbox.ipynb`, `02_Licence-plate-recognition.ipynb` (selected), `03_LPR_run_application.ipynb`, `04_LPR_test_application.ipynb`, `05_Send_Image.ipynb`, `car.jpg`, `unicorn_config.py`, `LICENSE`, `Pipfile`, `prediction.py`, `README.md`, `requirements.txt`, and `wsgi.py`. Most files were modified 4 hours ago.
- Code Cell:** The selected cell (number 9) contains Python code for license plate recognition using matplotlib and OpenCV:

```
[9]: %matplotlib inline
images_path = 'dataset/images'
for filename in os.listdir(images_path):
    if filename.endswith('.jpg') or filename.endswith('.jpeg') or filename.endswith('.png'):
        vehicle, Lping, license_plate_string = lpr_process(os.path.join(images_path, filename))
        # Display the result
        fig = plt.figure(figsize=(12,6))
        fig.add_subplot(1,2,1)
        plt.axis(False)
        plt.imshow(vehicle)
        fig.add_subplot(1,2,2)
        plt.axis(False)
        plt.imshow(Lping[0]);
        plt.show()
        print('Detected plate number: ' + license_plate_string)
        print('-----')
```
- Output:** The notebook displays two images and their corresponding detected license plate numbers.
  - The first image is a red car with the license plate `M666 YOB`.
  - The second image is a yellow car with the license plate `CRAIG`.
- Bottom Status Bar:** Shows "Simple" mode, Git: Idle, and Python 3 | Idle. It also indicates "Saving completed".

# Package the Model as an API

The screenshot shows a Jupyter Notebook interface with the title bar '03\_LPR\_run\_application.ipynb'. The notebook contains the following content:

## Flask

Our API will be served directly from our container using Flask, a popular Python Web Server. The Flask application, which will call our prediction function, is defined in the `wsgi.py` file.

As always, we'll run first some imports to make sure all our requirements are there:

```
[1]: import sys  
!{sys.executable} -m pip install -r requirements.txt
```

Now that we have everything in place, we can launch the Flask application.  
(Please ignore the CUDA errors or warning if you don't have any GPU).

This cell will be in a **permanent running state**. That's normal as the webserver process will keep running. When you are finished with the test you can just select the cell, and click on the **Stop button** (next to Run).

```
[1]: !FLASK_ENV=development FLASK_APP=wsgi.py flask run
```

Once the models have been loaded, our server is ready to take requests. Leave this notebook running, and open `84_LPR_test_application.ipynb`.

# Package the Model as an API



The screenshot shows a Jupyter Notebook interface with a tab titled "03\_LPR\_run\_application.ipynb". The notebook content includes a section titled "Flask" with the following text:

Our API will be served directly from our container using Flask, a popular Python Web Server. The Flask application, which will call our prediction function, is defined in the `wsgi.py` file.

As always, we'll run first some imports to make sure all our requirements are there:

You'll first launch the Server:

```
!FLASK_ENV=development FLASK_APP=wsgi.py flask run
* Serving Flask app "wsgi.py" (lazy loading)
* Environment: development
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
```

# Package the Model as an API



Then query the API:

## Test Flask App

From the other notebook, `03_LPR_run_application.ipynb`, the flask app is running and ready to serve requests using python http or `curl` statements.

We can start by testing that the server is indeed running:

```
[1]: !curl http://localhost:5000/status
{
    "status": "ok"
}
```

Now we can use curl to send the name of an image, and wait for the result:

```
[2]: !curl -X POST -H "Content-Type: application/json" --data '{"data": "Cars374.png"}' http://localhost:5000/predictions
{
    "prediction": "S7J0V"
}
```

# Build the application inside OpenShift

The screenshot shows the Red Hat OpenShift Data Science project details page. The URL is `console-openshift-console.apps.rhods-internal.ju9jp1.openshiftapps.com/project-details/ns/audreytest`. The left sidebar shows navigation options like Developer, Topology, Monitoring, Search, Builds, Pipelines, Helm, Project (which is selected), ConfigMaps, and Secrets. The main content area displays the project `audreytest` which is `Active`. It includes sections for Details (Name: audreytest, Requester: adrezn1, Labels: control-plane=kubeflow, katib-metricscollector-injection=enabled, olm.operatorgroup.uid/d31d5f79-e592-4ba5-b484-c5a...), Status (Active), Utilization (CPU usage over 1 hour showing a peak around 4.02m), and Activity (no ongoing or recent events).

Project Details - Red Hat | Red Hat OpenShift Data Sc | Red Hat OpenShift Dedicated | RHDS Beta - Task testing | +

console-openshift-console.apps.rhods-internal.ju9jp1.openshiftapps.com/project-details/ns/audreytest

Apps Red Hat External Red Hat Presentations Red Hat E-Business Red Hat Support IT New Hire Hub I.T. Toolbox BenefitSolver ... Enabling Support daily bookmark... RHPDS Reading

Red Hat OpenShift Dedicated

Developer

+Add

Topology

Monitoring

Search

Builds

Pipelines

Helm

Project

ConfigMaps

Secrets

Project: audreytest

PR audreytest Active

Overview Details Project access

Details View all

Name audreytest

Requester adrezn1

Labels control-plane=kubeflow, katib-metricscollector-injection=enabled, olm.operatorgroup.uid/d31d5f79-e592-4ba5-b484-c5a...

Description test project

Status

Active

Utilization

Resource Usage 16:45 17:00 17:15 17:30

Resource	Usage	16:45	17:00	17:15	17:30
CPU	4.02m	6m	4m	2m	

Activity View events

Ongoing

There are no ongoing activities.

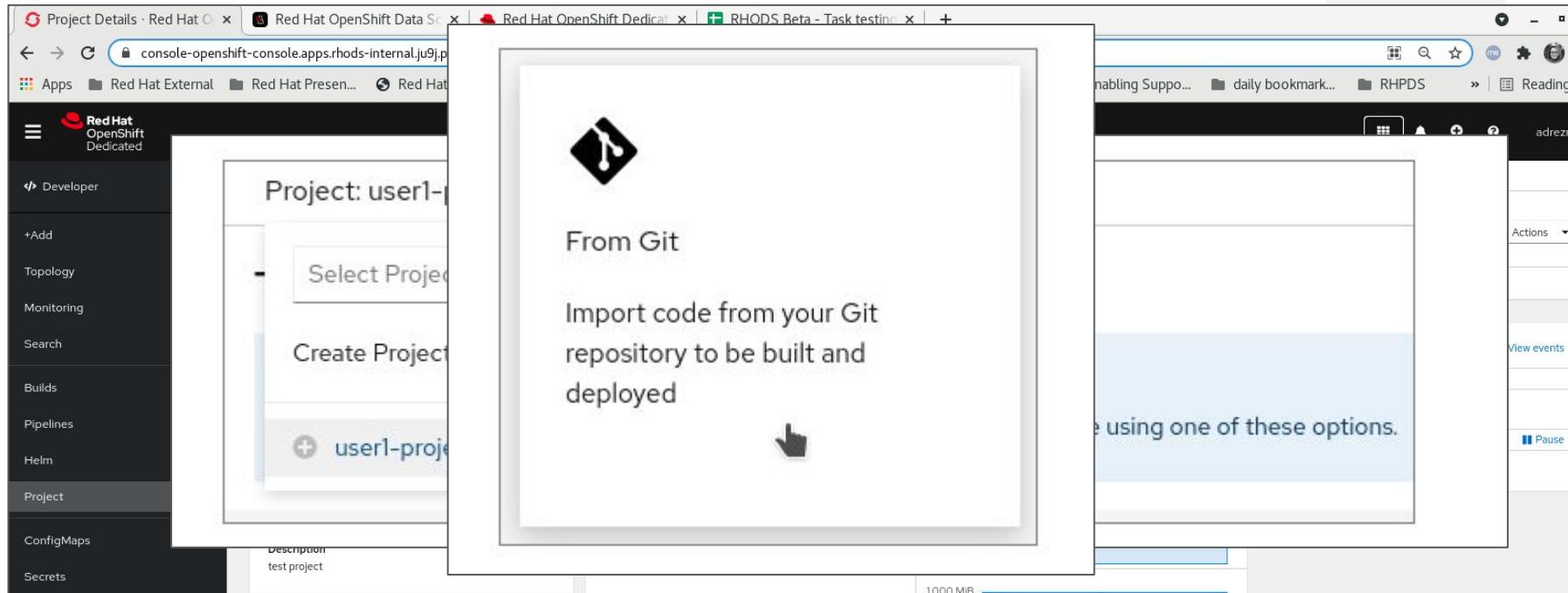
Recent events Pause

There are no recent events.

# Build the application inside OpenShift

The screenshot shows the Red Hat OpenShift Data Service console interface. The top navigation bar includes tabs for 'Project Details - Red Hat', 'Red Hat OpenShift Data Sc...', 'Red Hat OpenShift Dedicated', 'RHODS Beta - Task testing', and a '+' button. The URL in the address bar is `console-openshift-console.apps.rhods-internal.ju9jp1.openshiftapps.com/project-details/ns/audreytest`. The left sidebar has a 'Developer' section with links for '+Add', 'Topology', 'Monitoring', 'Search', 'Builds', 'Pipelines', 'Helm', 'Project' (which is selected), 'ConfigMaps', and 'Secrets'. The main content area displays project details for 'user1-project'. It shows a dropdown for 'Project: user1-project' and 'Application: all applications'. A large button labeled 'Select Project...' is present. Below it is a 'Create Project' section with a '+ user1-project' button. A tooltip or message box is overlaid on the right side of the screen, partially obscuring the UI. The message box contains the text: '...on, component or service using one of these options.' At the bottom of the main content area, there is a table with columns for 'Description' (containing 'test project') and 'Size' (with a progress bar from 0 to 1,000 MIB).

# Build the application inside OpenShift



**Application name**

licence-plate-workshop-git-app

A unique name given to the Application grouping to label your resources.

**Name \***

licence-plate-workshop-git

A unique name given to the component that will be used to name associated resources.

### Resources

Select the resource type to generate

Deployment  
apps/Deployment  
A Deployment enables declarative updates for Pods and ReplicaSets.

DeploymentConfig  
apps.openshift.io/DeploymentConfig  
A DeploymentConfig defines the template for a Pod and manages deploying new Images or configuration changes.

### Advanced options

Create a Route to the Application  
Exposes your Application at a public URL

Click on the names to access advanced options for [Routing](#), [Health checks](#), [Build configuration](#), [Deployment](#), [Scaling](#), [Resource limits](#) and [Labels](#).

**Create** **Cancel**

# Build the application inside OpenShift

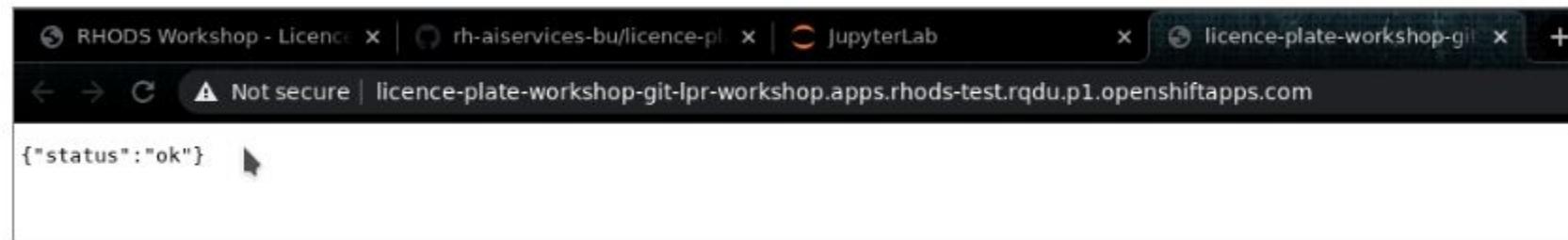
## Routes

RT licence-plate-workshop-git

Location:

<http://licence-plate-workshop-git-lpr-workshop.apps.rhods-test.rqdu.p1.openshiftapps.com> ↗

## Test our Deployed AIML Application



# Test our Deployed AIML Application

## CURL on Linux or Mac with bash/zsh

From anywhere you have an example image like `car.jpg` (replace with the right name in the command, as well as the Route with `/predictions` at the end):

```
(echo -n '{"image": ""'; base64 car.jpg; echo '}') | curl -H "Content-Type: application/json" -
```

## Invoke-WebRequest on Windows with Powershell

From anywhere you have an example image like `car.jpg` (replace with the complete path and name in the command, as well as the Route with `/predictions` at the end):

```
Write-Output ('{"image": "' + ([Convert]::ToString([IO.File]::ReadAllBytes('C:\Users\Guil
```

# Test our Deployed AIML Application

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help.
- Toolbar:** Includes icons for file operations like New, Open, Save, and a search bar.
- File Explorer:** Shows the directory structure of the workspace:
  - / licence-plate-workshop /
  - Name      Last Modified
  - dataset      38 minutes ago
  - models      38 minutes ago
  - 02\_Licenc...      27 minutes ago
  - 03\_LPR\_r...      15 minutes ago
  - 04\_LPR\_t...      15 minutes ago
  - 05\_Send\_I...      seconds ago
  - car.jpg      2 minutes ago
  - gunicorn\_c...      38 minutes ago
  - LICENSE      38 minutes ago
  - Pipfile      38 minutes ago
  - prediction.py      38 minutes ago
  - README....      38 minutes ago
  - requireme...      38 minutes ago
  - wsgi.py      38 minutes ago
- Launcher Tab:** Shows the current tab is "05\_Send\_image.ipynb".
- Code Cell:** Displays Python code to send an image file to a deployed application.

```
[1]: my_image = 'car.jpg'
my_route = 'http://licence-plate-workshop-git-user1-project.apps.rhods-workshop.j61u.p1.openshiftapps.com/'

[2]: import base64
import requests
from json import dumps

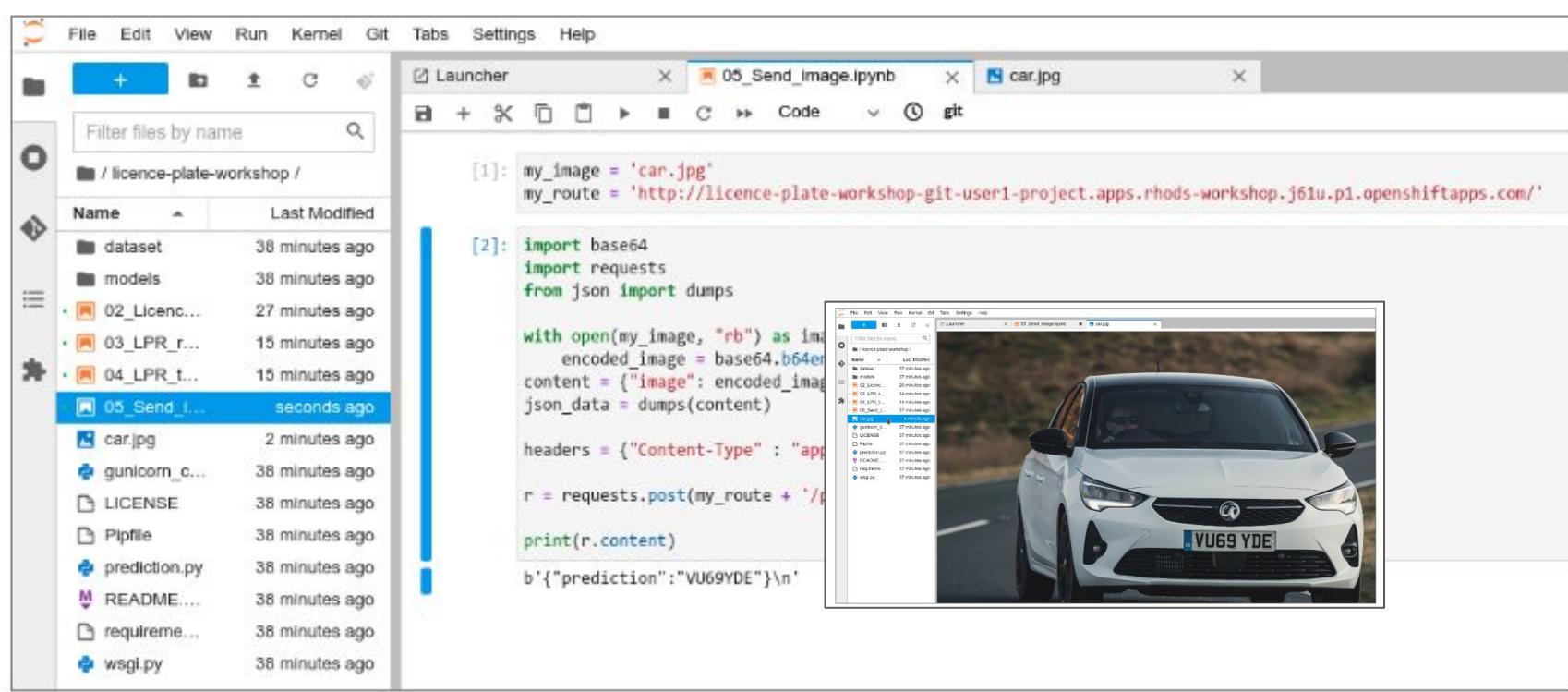
with open(my_image, "rb") as image_file:
    encoded_image = base64.b64encode(image_file.read()).decode('utf-8')
content = {"image": encoded_image}
json_data = dumps(content)

headers = {"Content-Type": "application/json"}

r = requests.post(my_route + '/predictions', data=json_data, headers=headers)

print(r.content)
b'{"prediction": "VU69YDE"}\n'
```

# Test our Deployed AIML Application



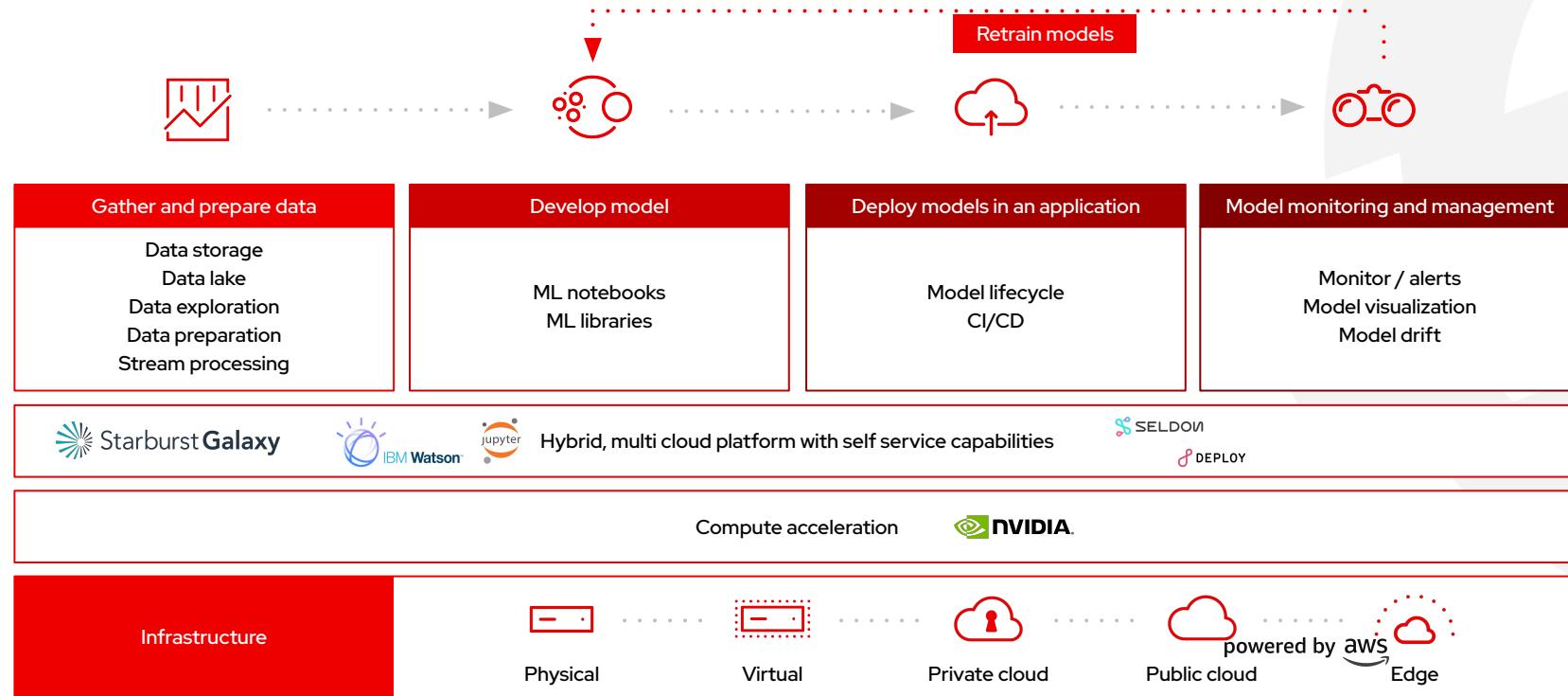
The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** On the left, it shows a file tree for the directory `/ licence-plate-workshop /`. The files listed include `dataset`, `models`, `02_Licenc...`, `03_LPR_r...`, `04_LPR_t...`, `05_Send_I...` (which is selected), `car.jpg`, `gunicorn_c...`, `LICENSE`, `Pipfile`, `prediction.py`, `README....`, `requireme...`, and `wsgi.py`. All files were modified 38 minutes ago except for `05_Send_I...` which was modified "seconds ago".
- Code Cell:** The main area contains two code cells:
  - [1]:

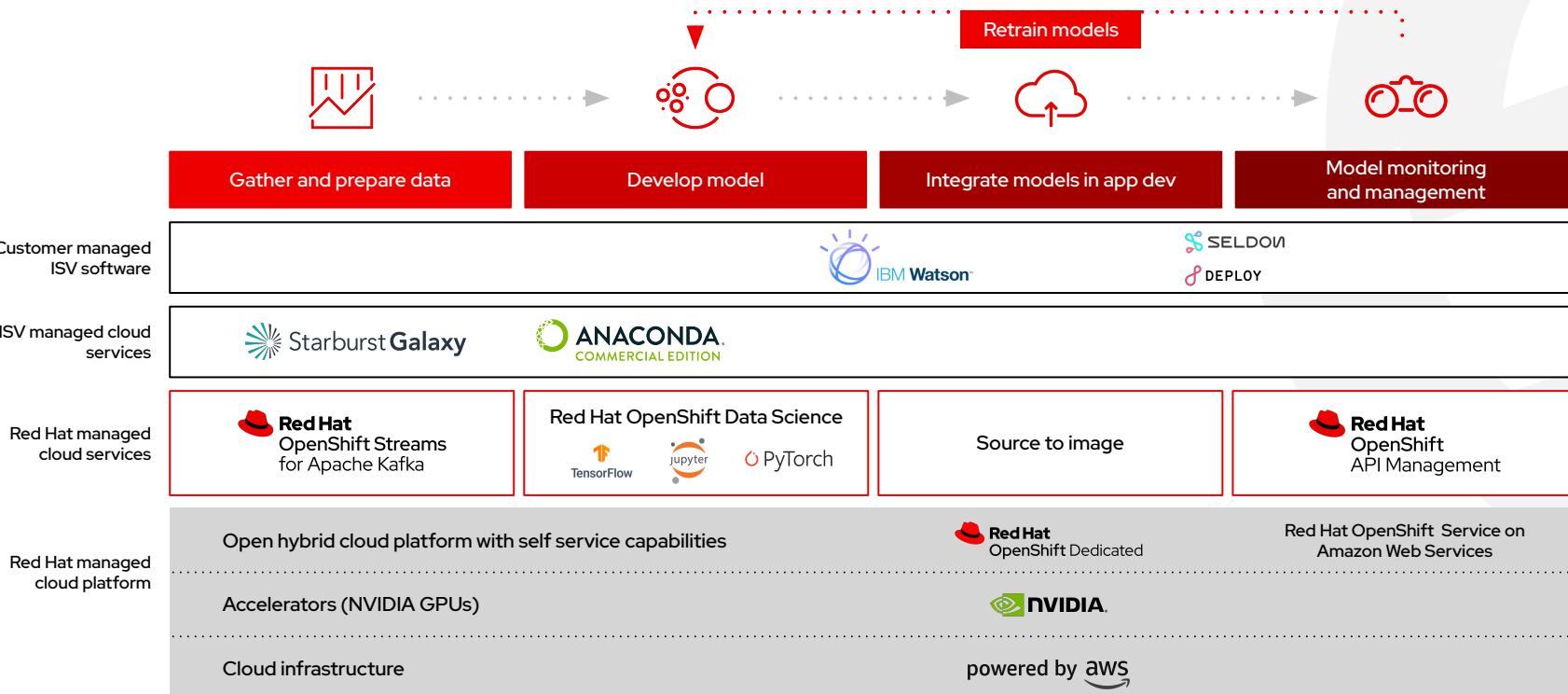
```
my_image = 'car.jpg'  
my_route = 'http://licence-plate-workshop-git-user1-project.apps.rhods-workshop.j61u.p1.openshiftapps.com/'
```
  - [2]:

```
import base64  
import requests  
from json import dumps  
  
with open(my_image, "rb") as im:  
    encoded_image = base64.b64encode(im.read())  
    content = {"image": encoded_image}  
    json_data = dumps(content)  
  
    headers = {"Content-Type": "application/json"}  
  
    r = requests.post(my_route + '/predict',  
                      json=json_data,  
                      headers=headers)  
  
    print(r.content)
```
- Output:** To the right of the code cells, there is a preview image of a white car with a license plate that reads `VU69 YDE`.

# Conceptual Architecture for Managed Services and Model Delivery



# Red Hat OpenShift Data Science Platform



## What did we learn today?

1. What are Managed Services?
  - a. Who cares about them?
2. Where do I find Managed Services?
  - a. Public Cloud, Hybrid Cloud, onPrem
3. What do Managed Services have to do with Model Delivery?
  - a. Use case
4. Are Managed Services easy to use?

# Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)

[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)

[facebook.com/redhatinc](https://www.facebook.com/redhatinc)

[twitter.com/RedHat](https://twitter.com/RedHat)

## Backup Slides

# Log into our Platform

The screenshot shows the 'Project Details' page for the 'audreytest' project in the 'Red Hat OpenShift Data Science' namespace. The URL in the browser is `console-openshift-console.apps.rhods-internal.ju9j.p1.openshiftapps.com/project-details/ns/audreytest`. The top navigation bar includes links for 'Red Hat OpenShift Dedicated' and 'RHODS Beta - Task testing'. The left sidebar has sections for 'Developer', '+Add', 'Topology', 'Monitoring', 'Search', 'Builds', 'Pipelines', 'Helm', 'Project' (which is selected), 'ConfigMaps', and 'Secrets'. The main content area displays the 'audreytest' project details, including its name, requester, labels, and description. It also shows the project's status as 'Active', utilization metrics for CPU and memory, and activity logs. A red circle highlights the 'More' icon (three dots) in the top right corner of the header.

# Say hello to Bob and his AIML model!

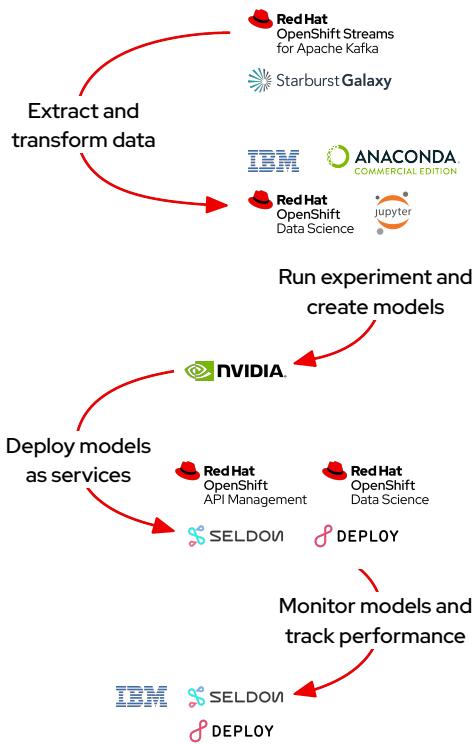
Ok I have my model working perfectly on my laptop. I just need to put this into production by 8am tomorrow for my customer demo!



\$#@! %#\* !\*@#!!!



# ... and the model operationalization life cycle using RHODS



## Starburst Galaxy

Unlock the value of your data by making it fast and easy to access data across hybrid cloud.

## Anaconda Commercial Edition

Curated access to an extensive set of data science packages to be used in your Jupyter projects.

## IBM Watson Studio

Build, run, and manage AI models at scale with Watson Machine Learning and Watson OpenScale.

## NVIDIA

GPU-enabled hardware makes it easier for customers to stand up resource-intensive environments and accelerate their data science experiments.

## Seldon Deploy

Simplify and accelerate the process of deploying and managing your machine learning models.