



**OpenShift
Commons**

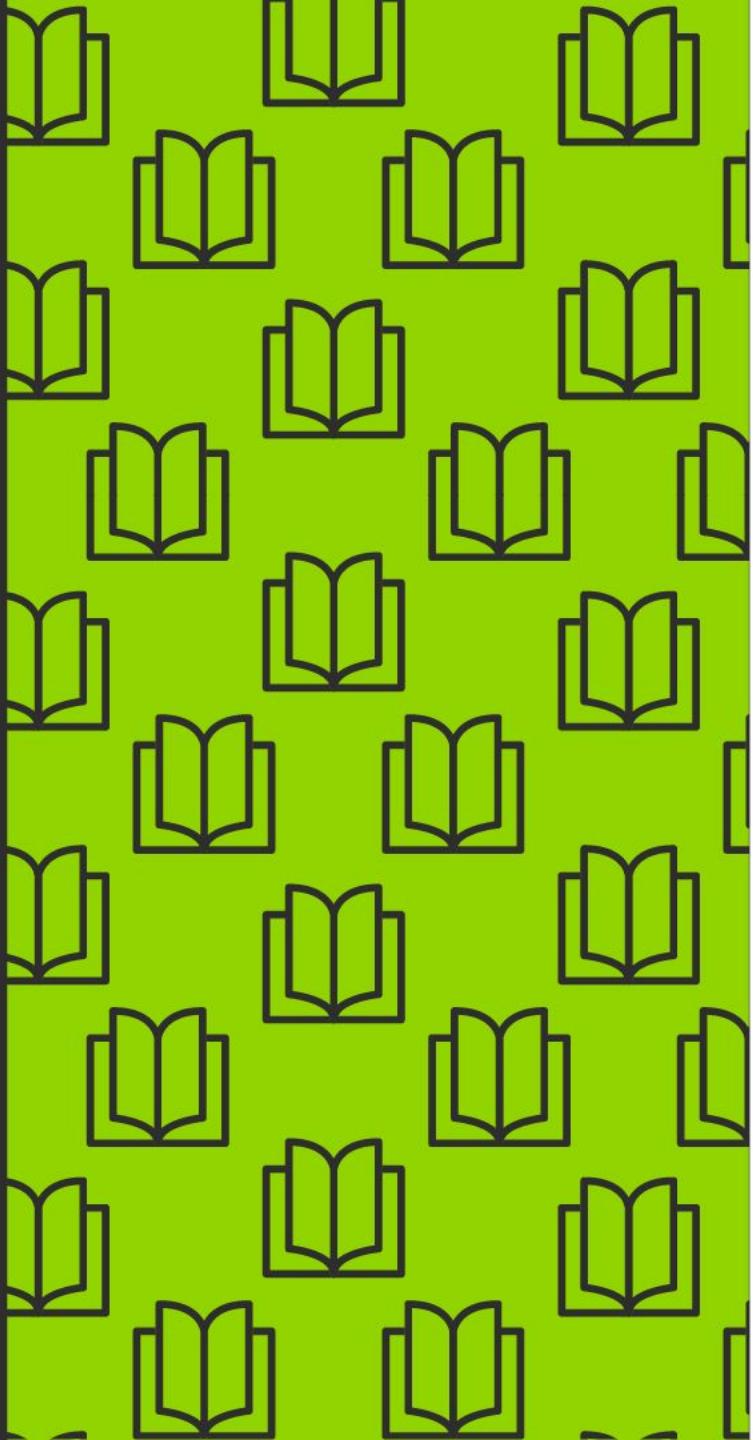
What's the deal with Managed Services & Model Delivery?

Audrey Reznik
Sr. Principal Software Engineer
areznik@redhat.com



What we'll discuss today

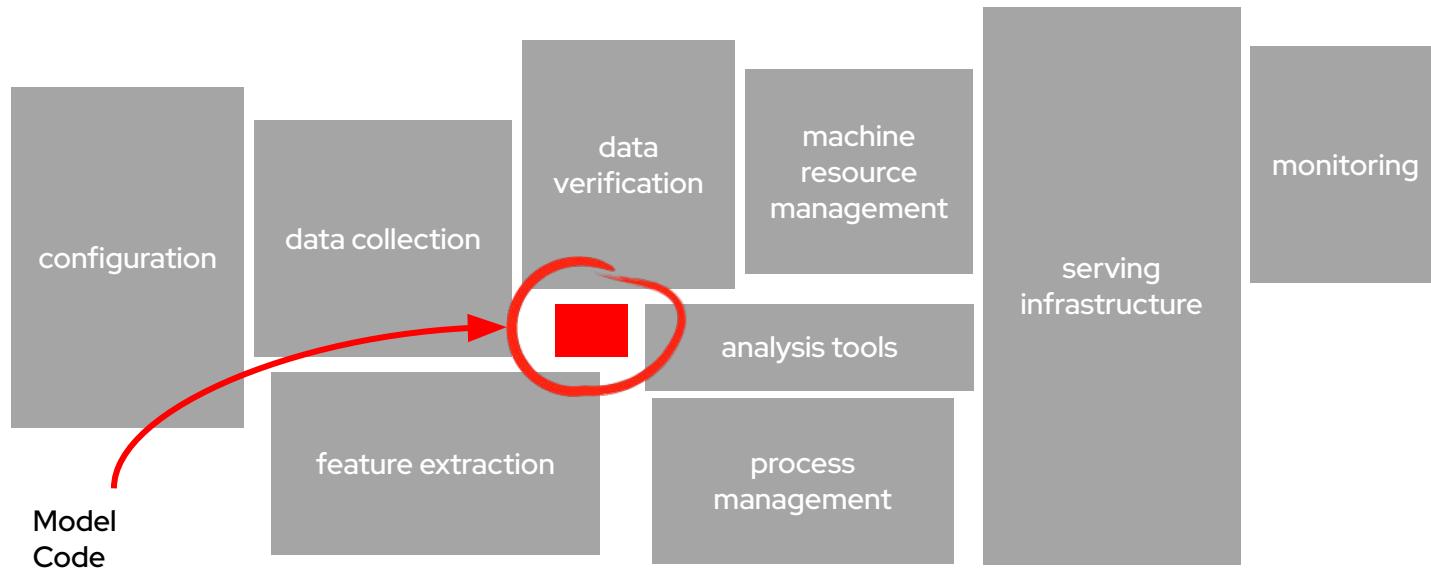
- ▶ A Model's role in an Intelligent Application
- ▶ What are Managed Services?
 - Who uses Managed Services? Hint... not only Data Scientists
- ▶ What do Managed Services have to do with Model Delivery?
- ▶ Where do I find Managed Services?
- ▶ Demo of using Red Hat OpenShift Data Science platform and it's Managed Services.

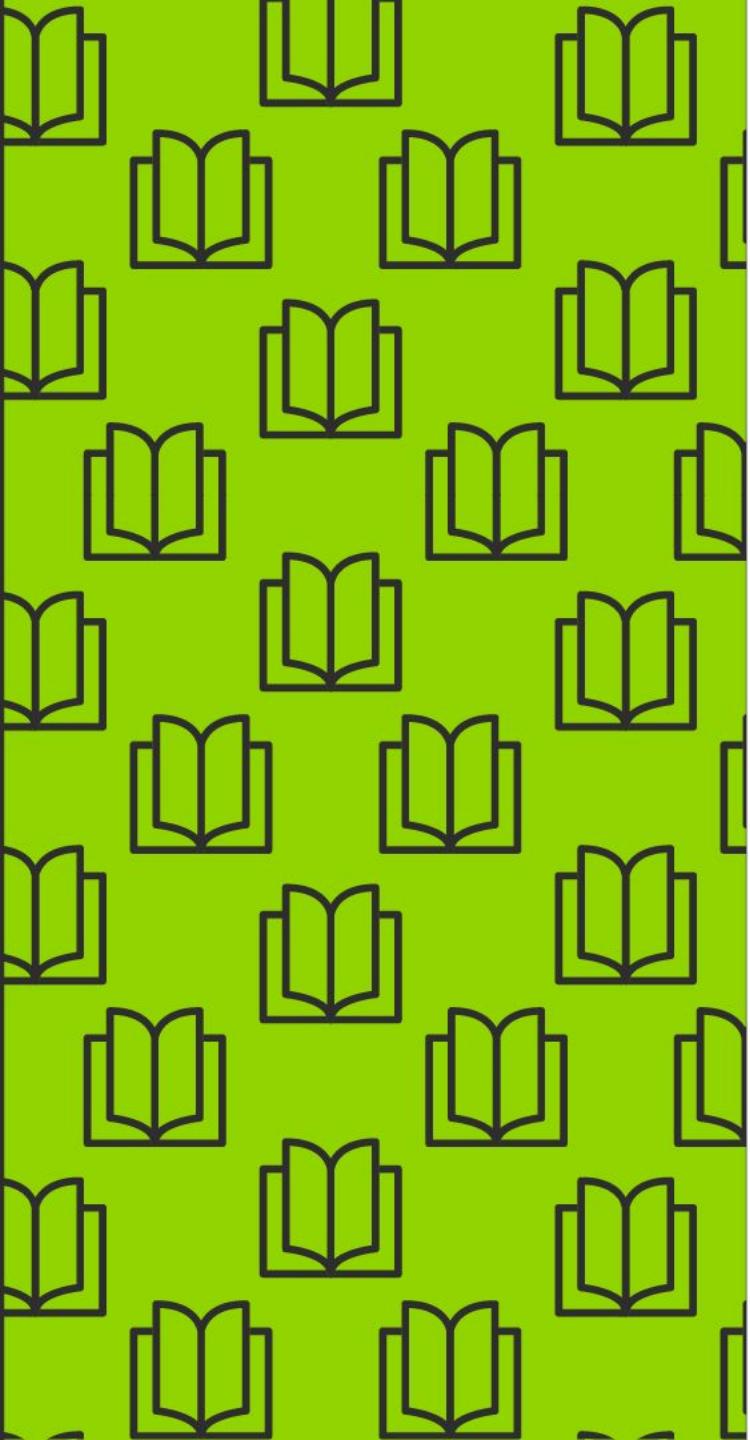


A Model's role in an Intelligent Application

A Model's role in Intelligent Applications

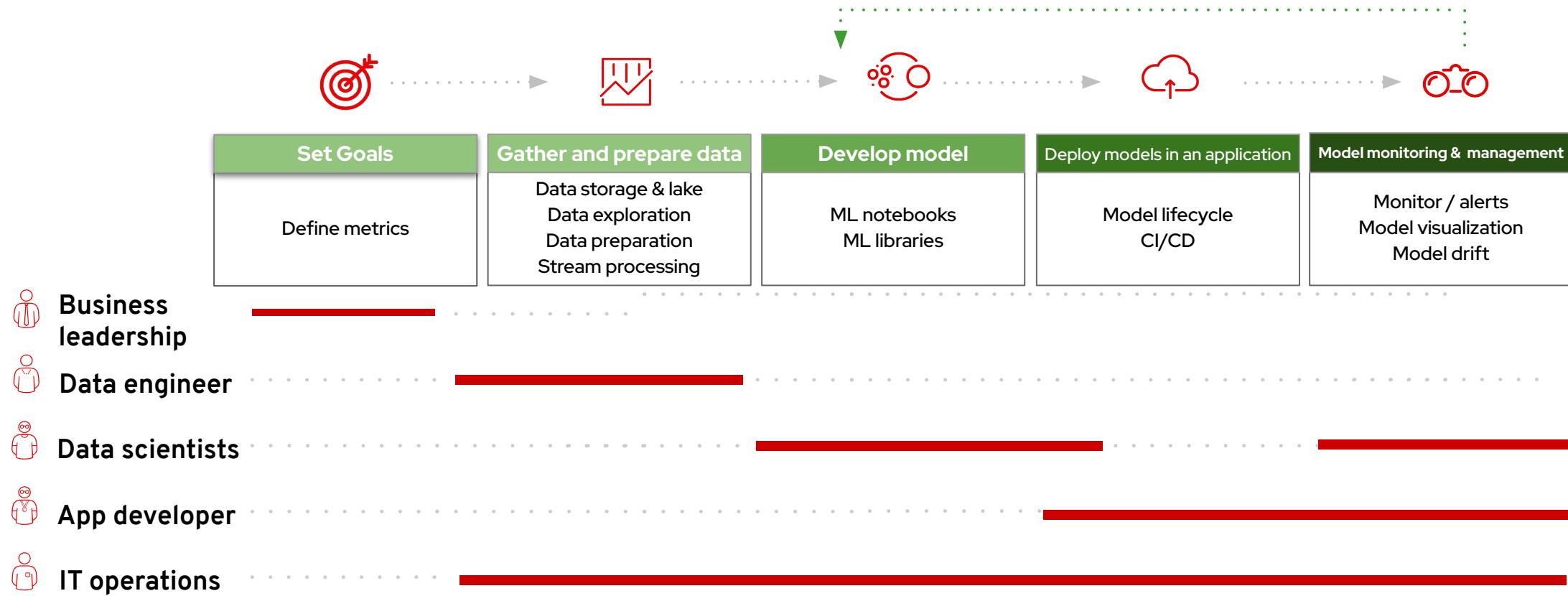
The model is a tiny but essential part ...

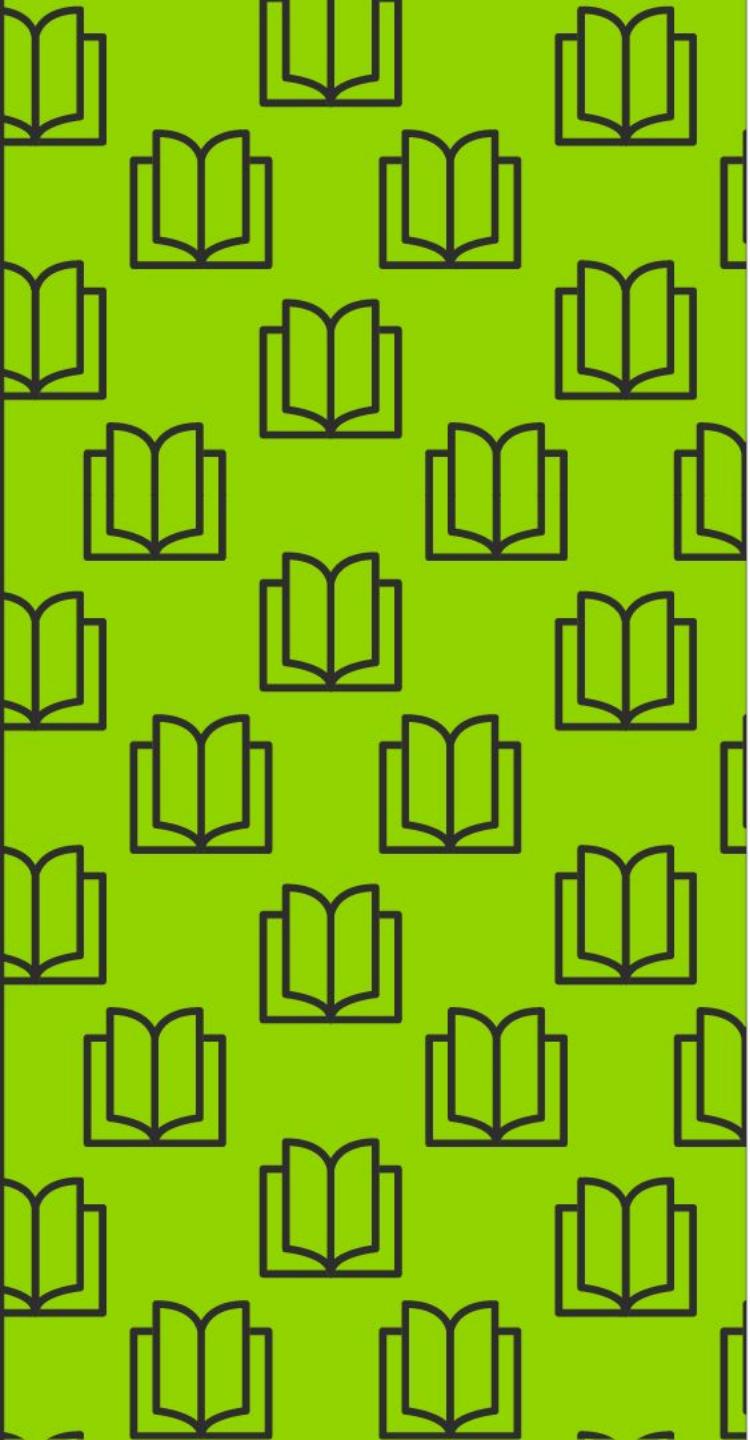




What are Managed
Services ... and
who uses them?

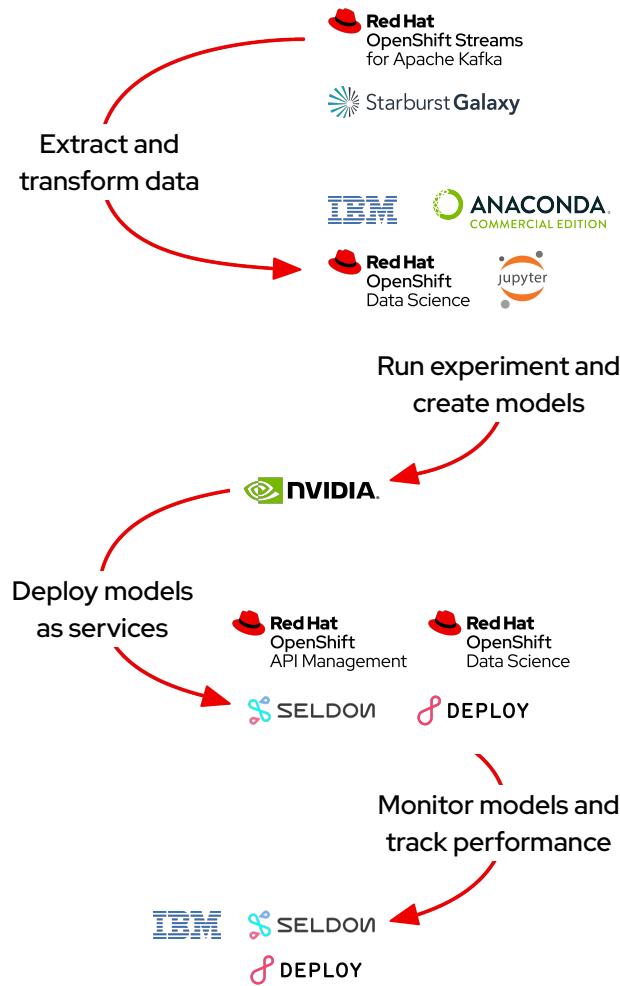
Managed Services (and Personas) for a Model Lifecycle





What do Managed Services have to do with Model Delivery?

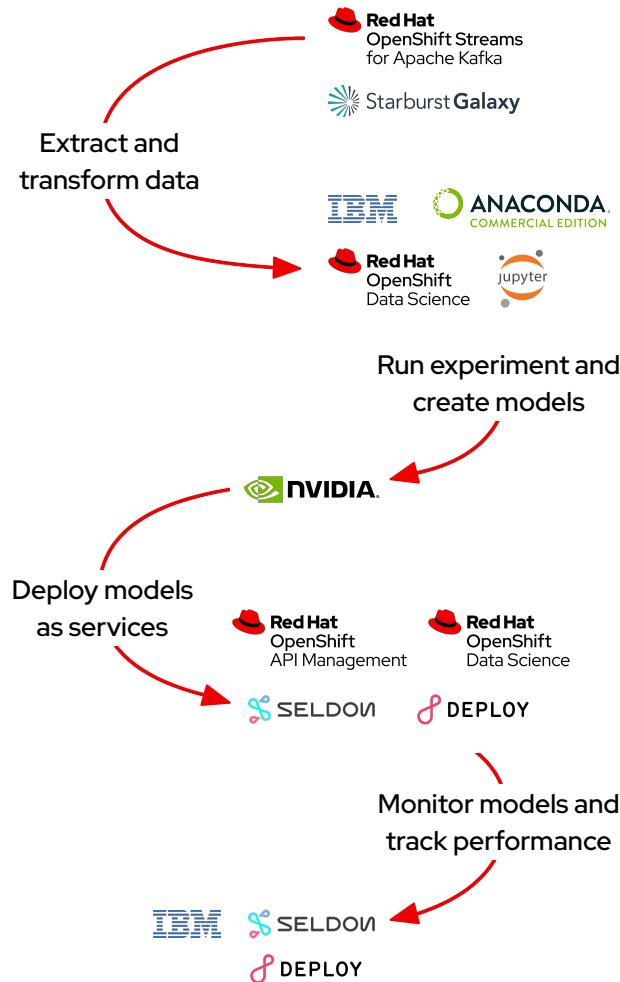
Supporting the model (operationalization) life cycle



Extract & Transform the Data - Starburst Galaxy

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

Supporting the model (operationalization) life cycle



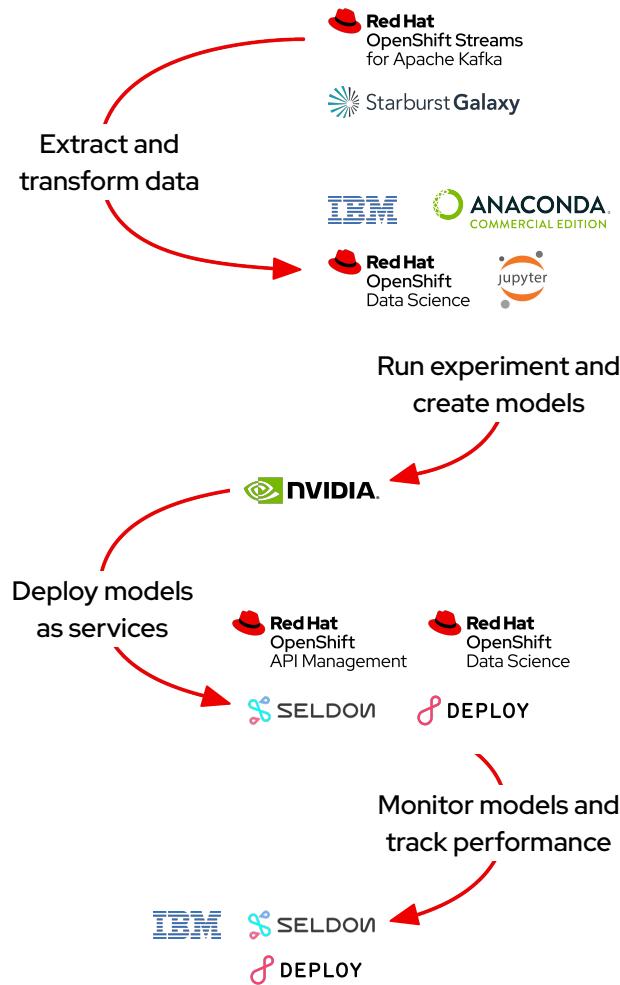
Extract & Transform the Data - Starburst Galaxy

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

Create Models - Anaconda Commercial Edition

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

Supporting the model (operationalization) life cycle



Extract & Transform the Data - Starburst Galaxy

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

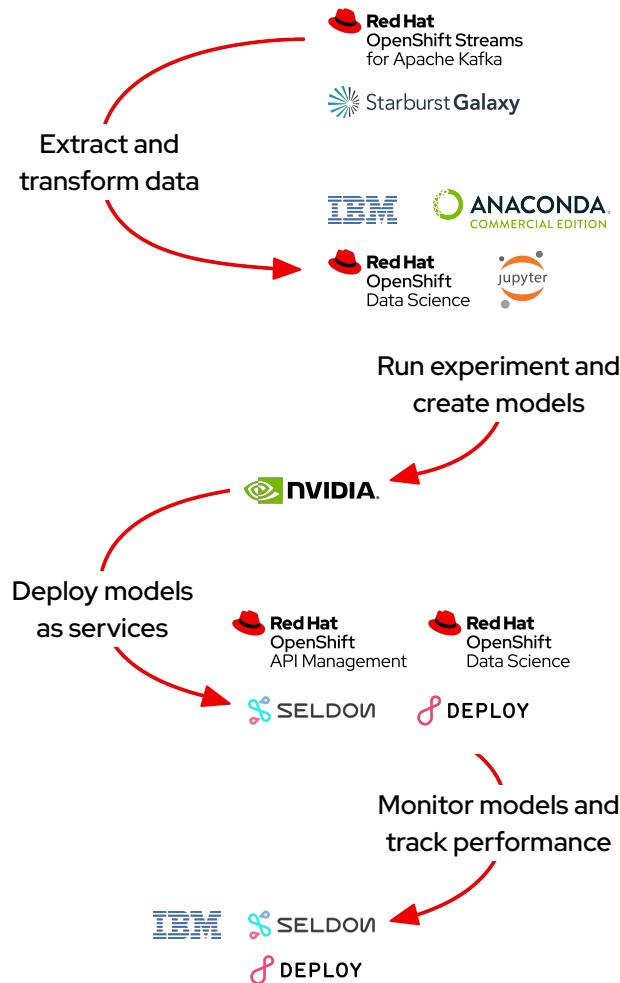
Create Models - Anaconda Commercial Edition

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

Run Experiments - IBM Watson Studio

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

Supporting the model (operationalization) life cycle



Extract & Transform the Data - Starburst Galaxy

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

Create Models - Anaconda Commercial Edition

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

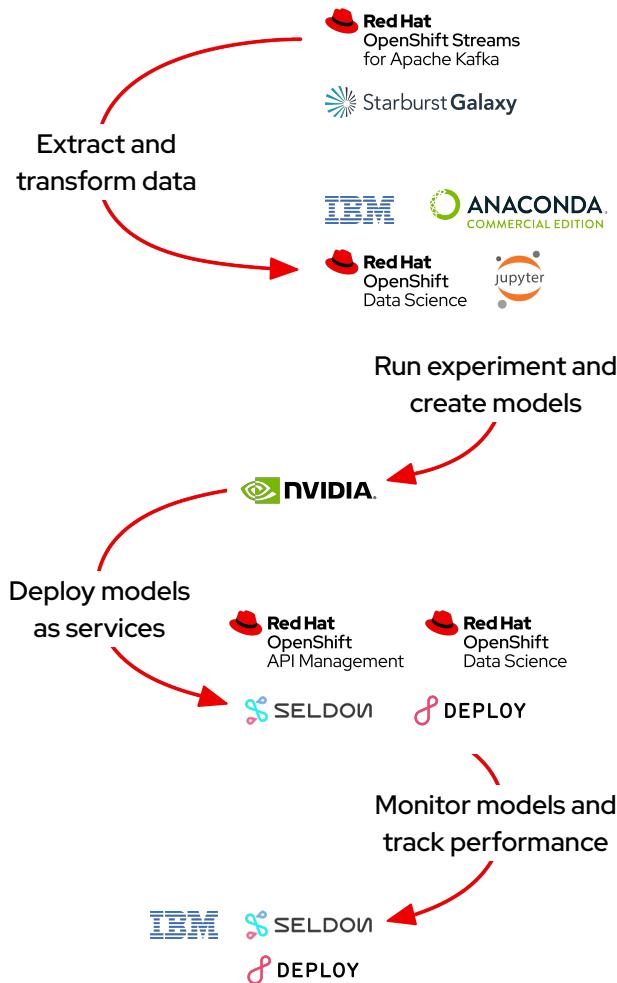
Run Experiments - IBM Watson Studio

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

Deploy models as services - Seldon Deploy

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

Supporting the model (operationalization) life cycle



Extract & Transform the Data - Starburst Galaxy

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

Create models - Anaconda Commercial Edition

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

Run experiments - IBM Watson Studio

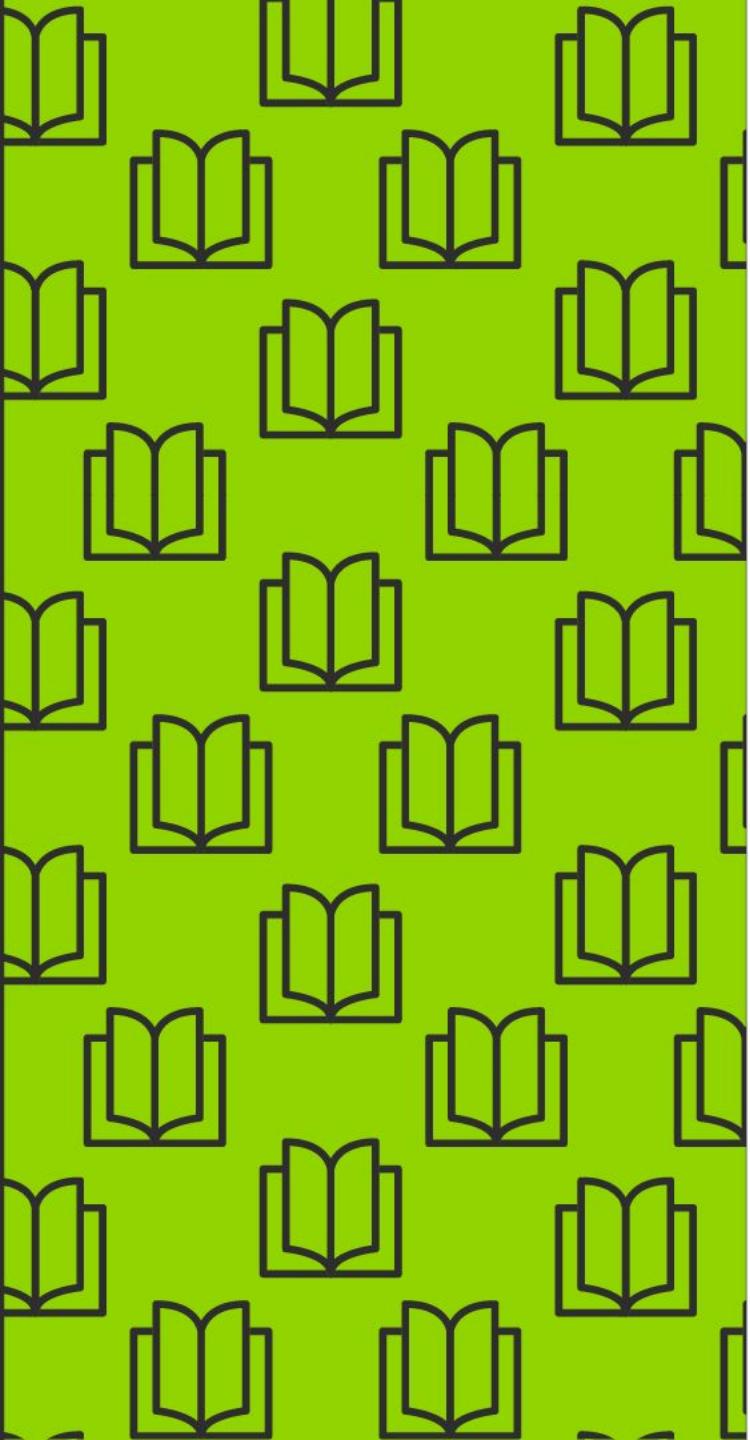
Build, run, and manage AI models at scale with Watson Machine Learning and Watson OpenScale.

Deploy models as services - Seldon Deploy

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

Monitor models & track performance - Seldon Deploy

Monitor model performance and gleam meaningful analytics from the model



Where can I find
Managed
Services?

Red Hat Managed Cloud Services

Red Hat managed
cloud platform

Open hybrid cloud platform with self service capabilities

Accelerators (NVIDIA GPUs)

Cloud infrastructure



powered by aws

Red Hat OpenShift Service on
Amazon Web Services



Red Hat Managed Cloud Services

Red Hat managed
cloud services



Red Hat OpenShift Data Science



Source to image



Red Hat managed
cloud platform

Open hybrid cloud platform with self service capabilities

Accelerators (NVIDIA GPUs)

Cloud infrastructure



Red Hat OpenShift Service on
Amazon Web Services

powered by aws



Red Hat Managed Cloud Services

ISV managed cloud services



Red Hat managed cloud services



Source to image



Red Hat managed cloud platform

Open hybrid cloud platform with self service capabilities



Red Hat OpenShift Service on Amazon Web Services

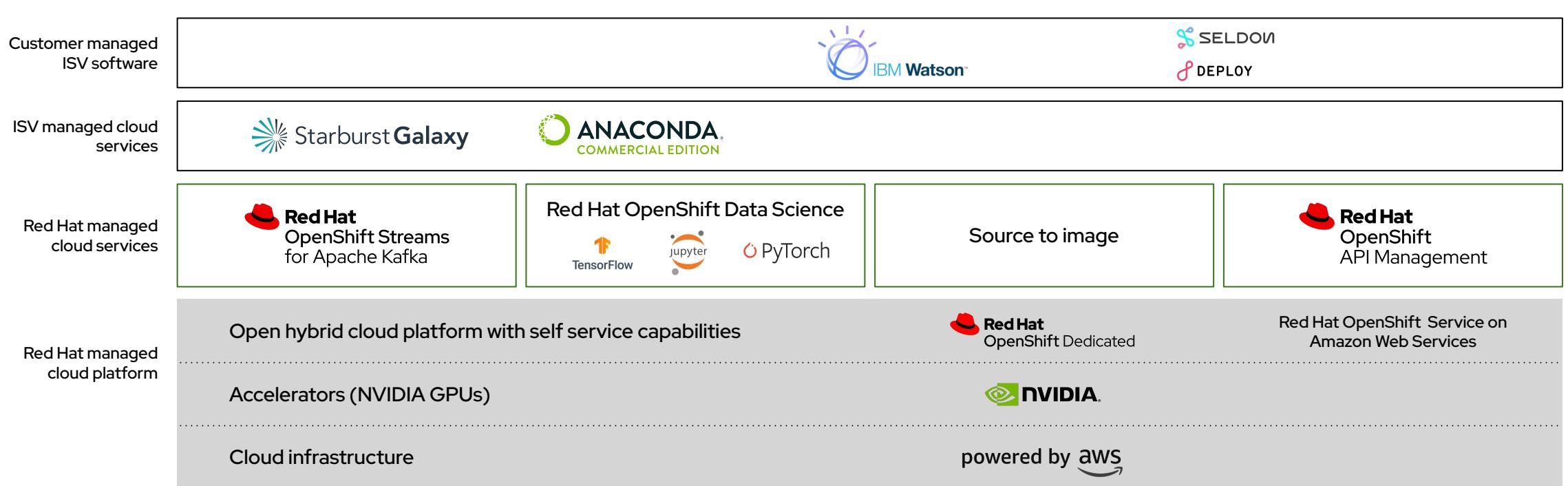
Accelerators (NVIDIA GPUs)



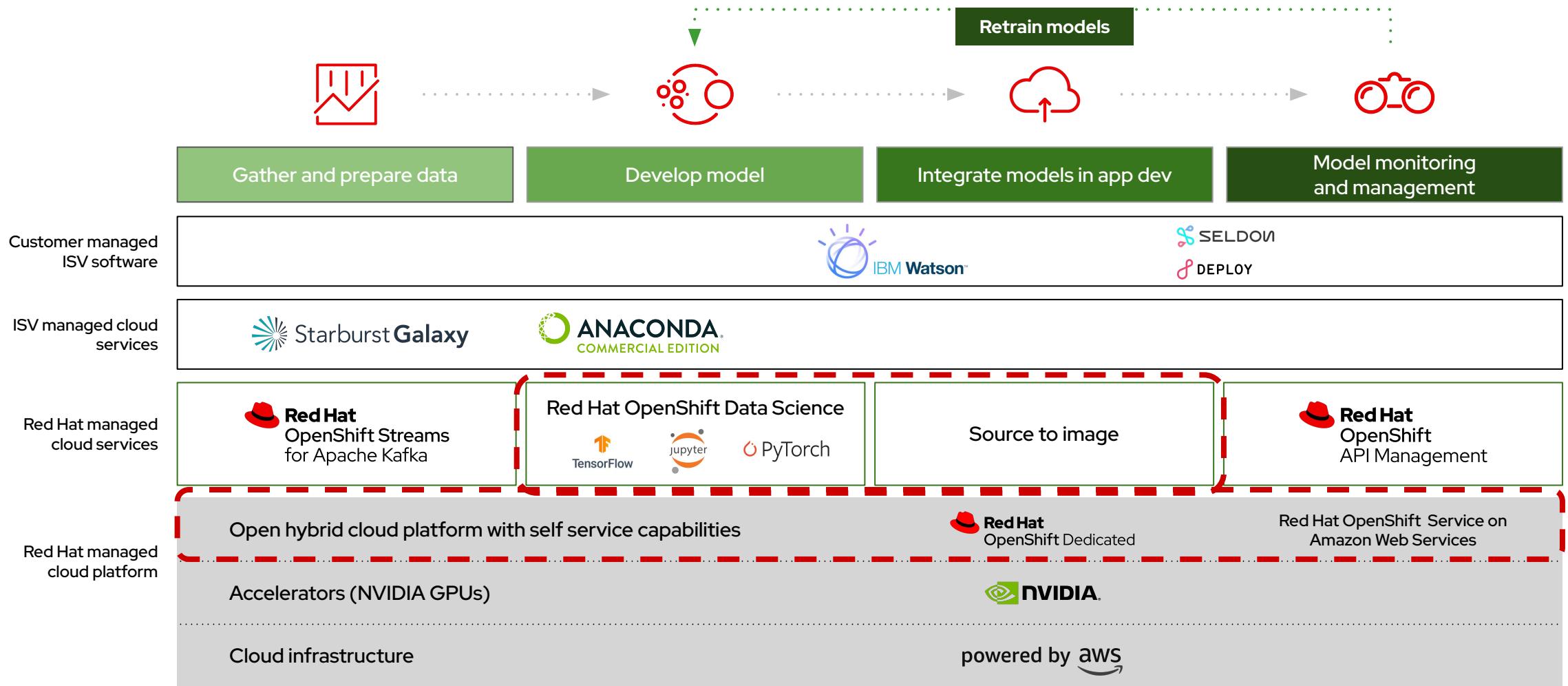
Cloud infrastructure

powered by aws

Red Hat Managed Cloud Services

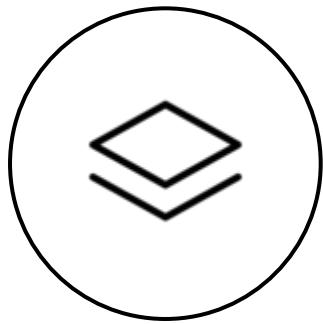


Red Hat Managed Cloud Services



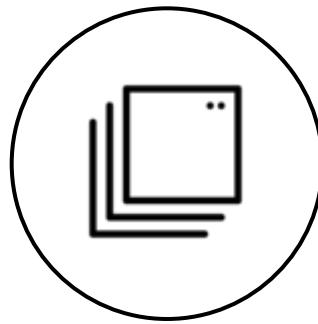
Depth and scale without lock-in

Capabilities delivered through the combination of Red Hat and partner ecosystem



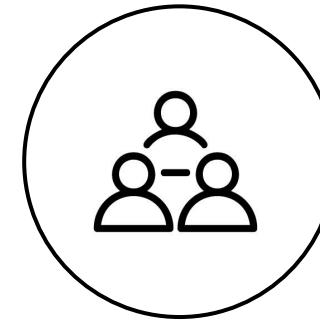
Managed cloud platform

Deployed on Red Hat OpenShift and managed on Amazon Web Services providing access to compute and accelerators based on your workload



Red Hat portfolio and services

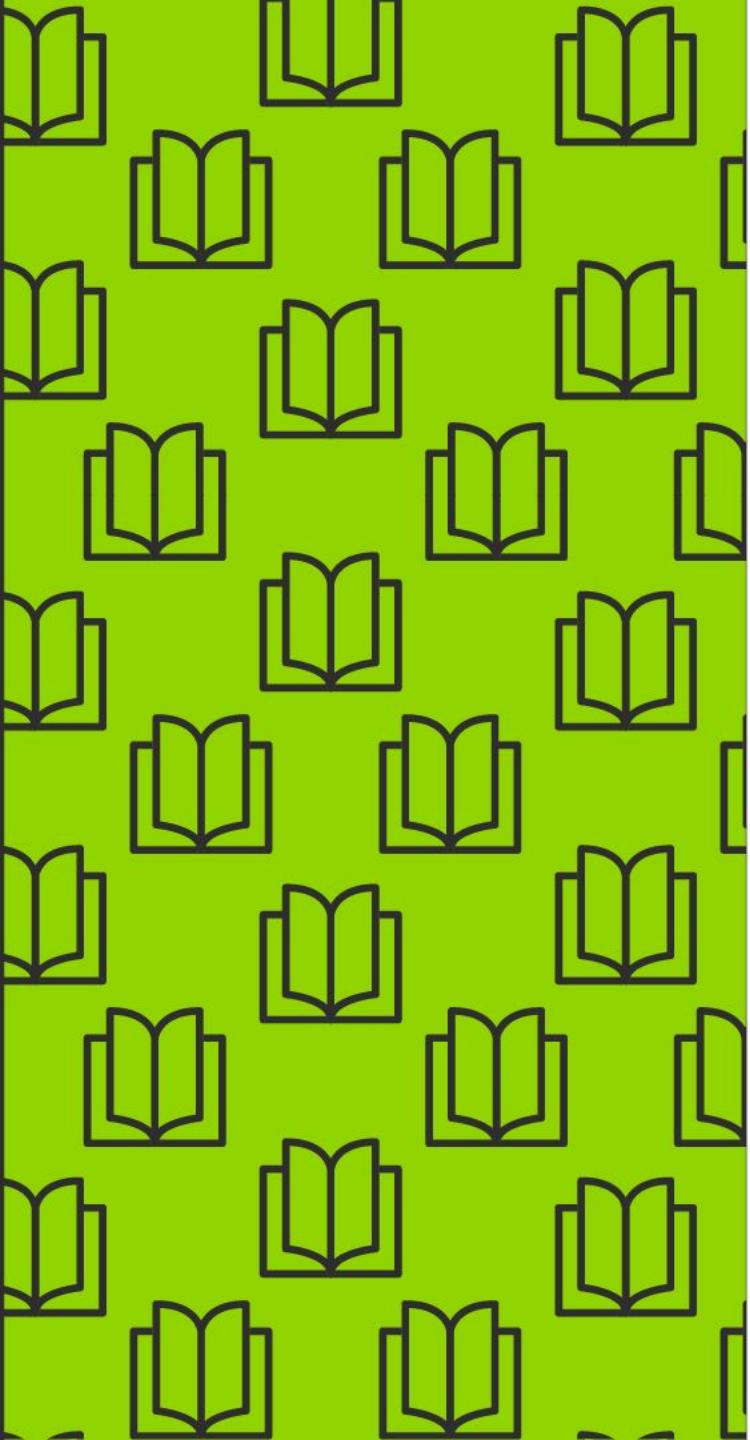
Complement common data science tools in Red Hat OpenShift Data Science with other Red Hat products and cloud services



Partner ecosystem

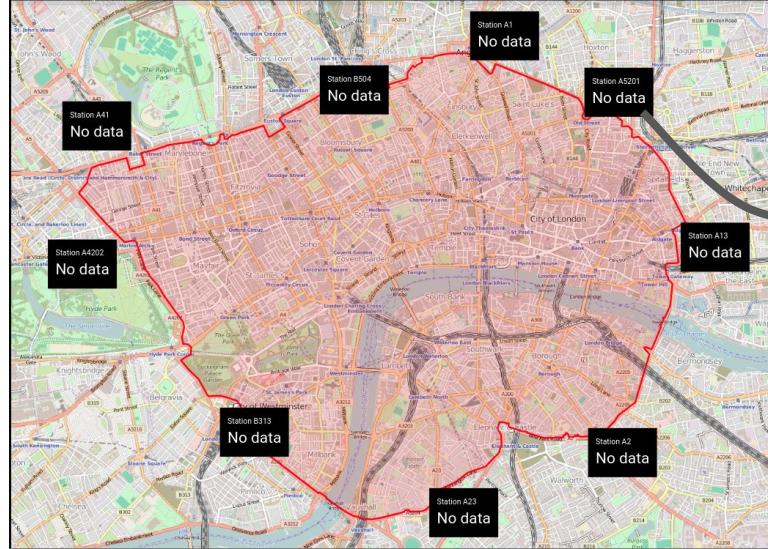
Access specialized capabilities by adding certified ISV ecosystem products and services from Red Hat Marketplace



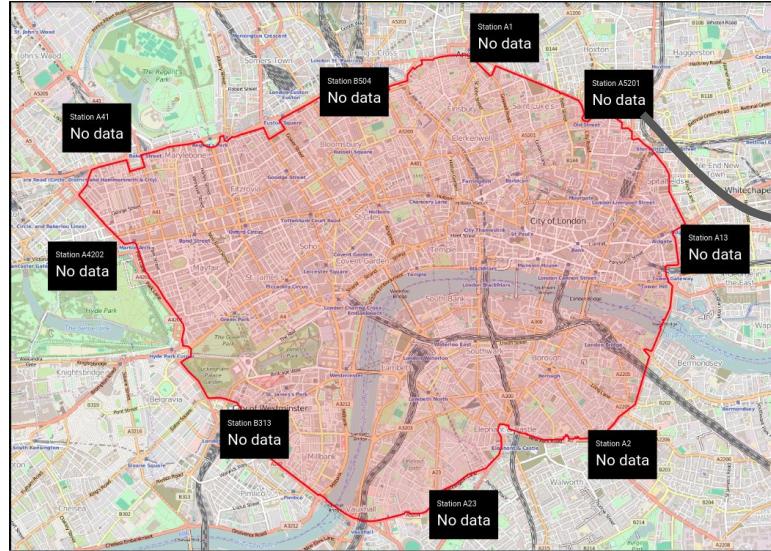


Demo - Red Hat OpenShift Data Science platform & its Managed Services

License Plate Detection

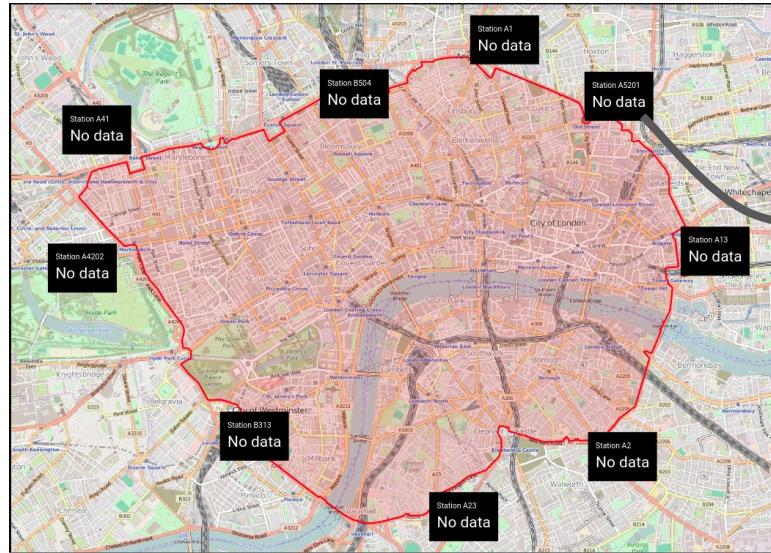


License Plate Detection



AKI8BZL

License Plate Detection



AKI8BZL

Kafka

MirrorMaker

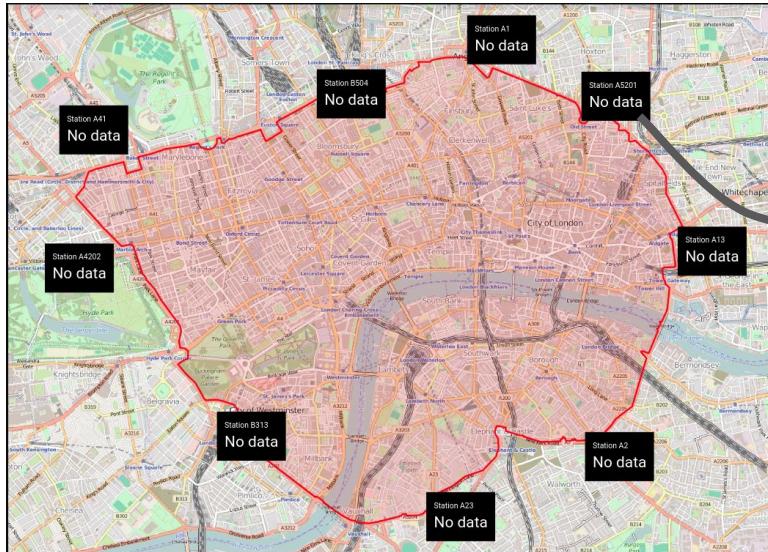
Kafka

AMBER
ALERT

shift
Commons

Red Hat
OpenShift

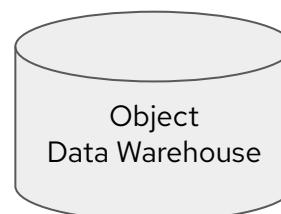
License Plate Detection



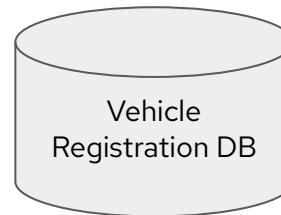
AKI8BZL

Kafka

MirrorMaker



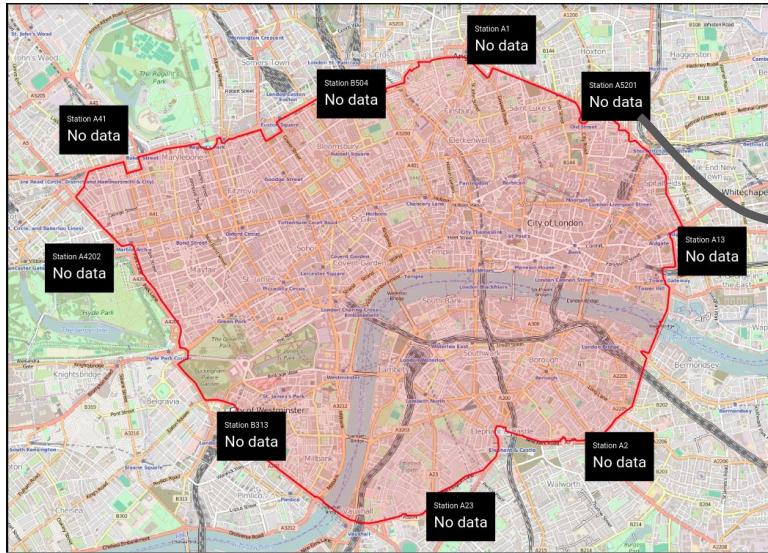
LPR Event Consumer



AMBER
ALERT

Red Hat
OpenShift

License Plate Detection

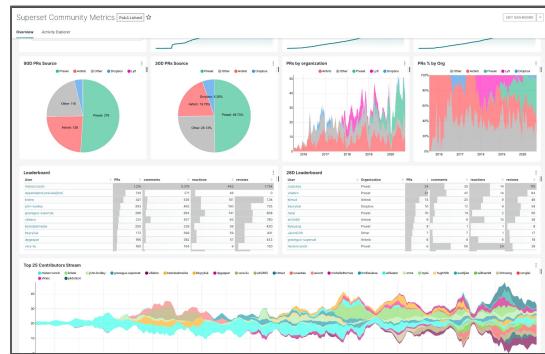


AKI8BZL

Kafka

MirrorMaker

Kafka



BI +
Analytics
Tools

Object
Data Warehouse

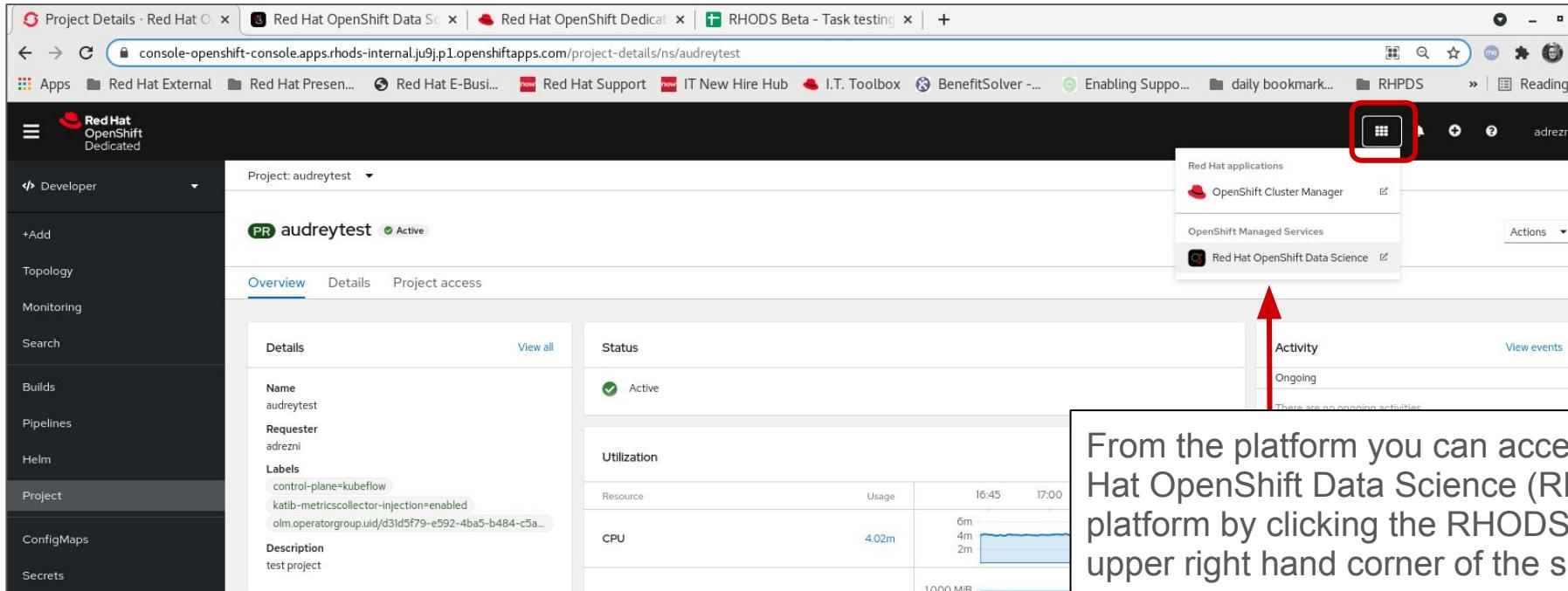
Vehicle
Registration DB

LPR Event
Consumer

**AMBER
ALERT**

Red Hat
OpenShift

Log into the Red Hat OpenShift Dedicated platform



Red Hat OpenShift Data Science Managed Services

The screenshot shows the Red Hat OpenShift Data Science Managed Services interface. The top navigation bar includes the Red Hat logo and the title "Red Hat OpenShift Data Science". The left sidebar has a "Resources" section selected. The main content area is titled "Resources" and displays a grid of 17 items out of 43. Each item card includes a logo, a title, a brief description, a "Quick start" button, and a "Documentation" button. The items are:

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

Red Hat OpenShift Data Science - Dashboard (user interface)

The screenshot displays the Red Hat OpenShift Data Science user interface. On the left, a sidebar menu includes 'Enabled' (selected), 'Explore', and 'Resources'. The main area shows two tabs: 'Enabled' and 'Explore'.

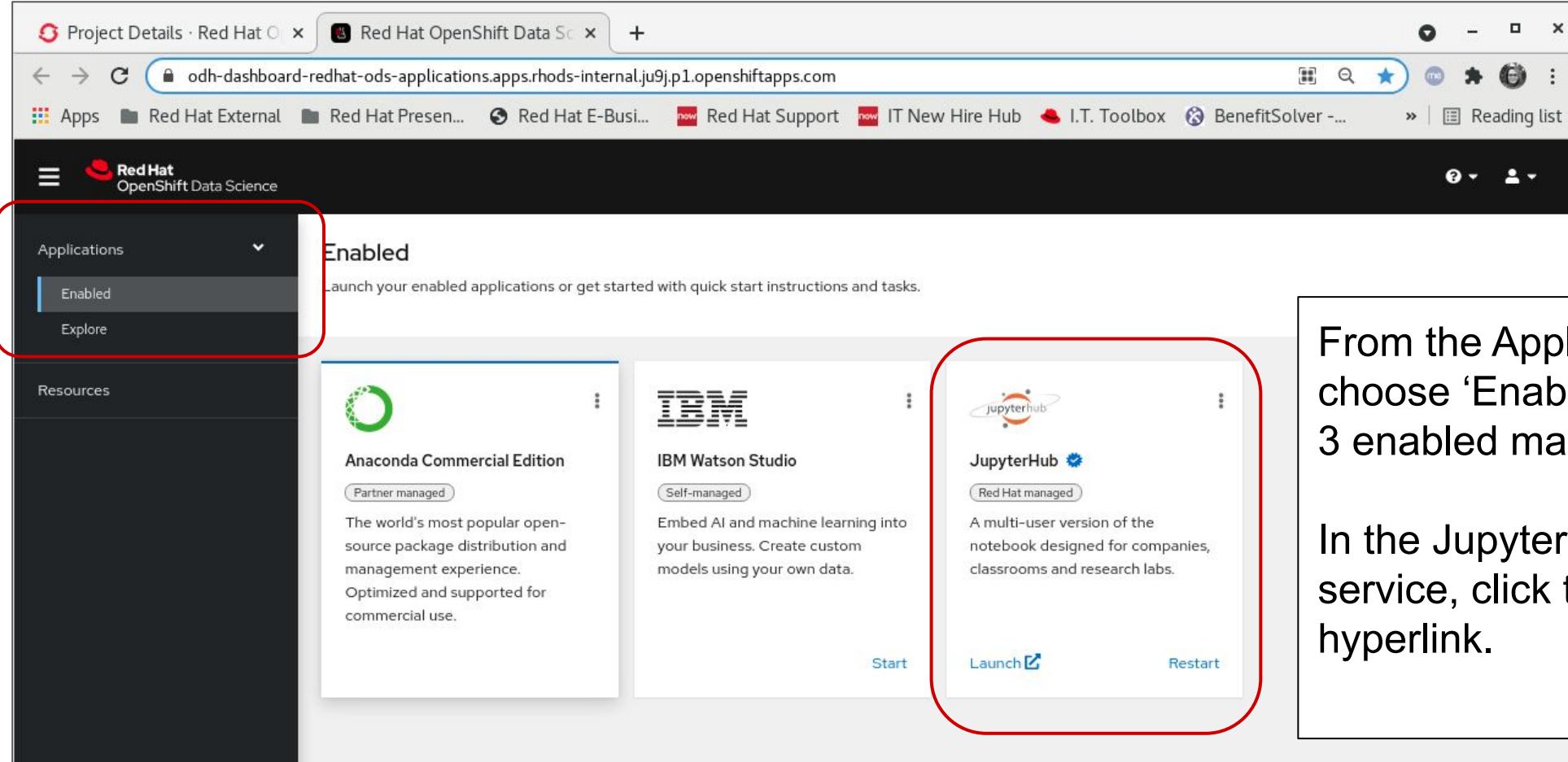
Enabled Tab:

- JupyterHub** (Red Hat managed): A multi-user version of the notebook designed for companies, classrooms and research labs. Buttons: 'Launch' and 'Close tour'.
- A modal window titled 'Creating a Jupyter notebook' is open, showing a quick start guide: '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.'

Explore Tab:

- 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.
- 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.

Launch (Enable) the JupyterHub Managed Service



The screenshot shows the Red Hat OpenShift Data Science dashboard. The top navigation bar includes links for Project Details, Red Hat OpenShift Data Science, Apps, Red Hat External, Red Hat Presentations, Red Hat E-Business, Red Hat Support, IT New Hire Hub, I.T. Toolbox, BenefitSolver, and a Reading list. The main header says "Red Hat OpenShift Data Science". On the left, a sidebar menu has "Applications" expanded, with "Enabled" selected (highlighted by a red box). Below this are "Explore" and "Resources". The main content area is titled "Enabled" and contains three service cards: "Anaconda Commercial Edition" (Partner managed), "IBM Watson Studio" (Self-managed), and "JupyterHub" (Red Hat managed). The "JupyterHub" card is highlighted with a red box. It has a "Launch" button at the bottom.

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 interface for JupyterHub. At the top, there's a navigation bar with links for Home, Token, Admin, and Services. Below the navigation, the title 'Start a notebook server' is displayed, followed by the sub-instruction 'Select options for your notebook server.'

Notebook image: This section contains five options, each with a radio button and a detailed description:

- 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: This section includes two dropdown menus:

- Container size:** Large
- Number of GPUs:** 1

Environment variables: This section shows a 'Custom variable' dropdown set to 'Variable name' and a text input field containing 'JUPYTER_ENABLE_LAB'. Below it is a 'Variable value' input field with the value '1' and a 'Secret' checkbox.

Buttons: At the bottom left are 'Add more variables' and 'Start server' buttons.

Available notebook images

The screenshot shows the JupyterHub interface for starting a notebook server. At the top, there's a navigation bar with links for Home, Token, Admin, and Services. Below that, a section titled "Start a notebook server" prompts the user to "Select options for your notebook server". A horizontal line separates this from the "Notebook image" selection area. In this area, six options are listed as radio buttons:

- 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

The screenshot shows a dropdown menu for 'Container size' with the following options:

- X Large (selected)
- 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)

The 'Large' option is highlighted with a red rectangle.

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](#)

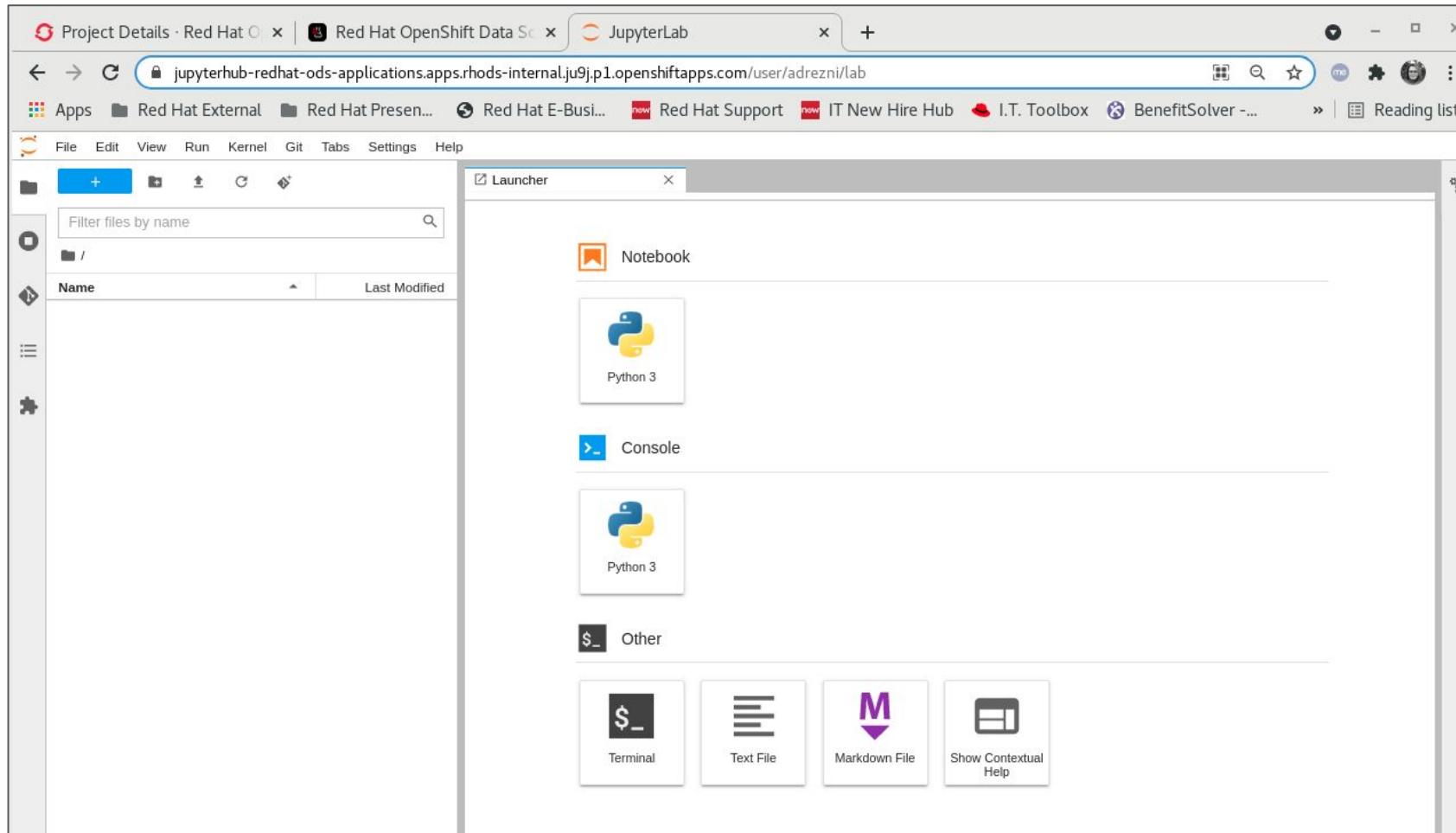
Start server

Server Start up...

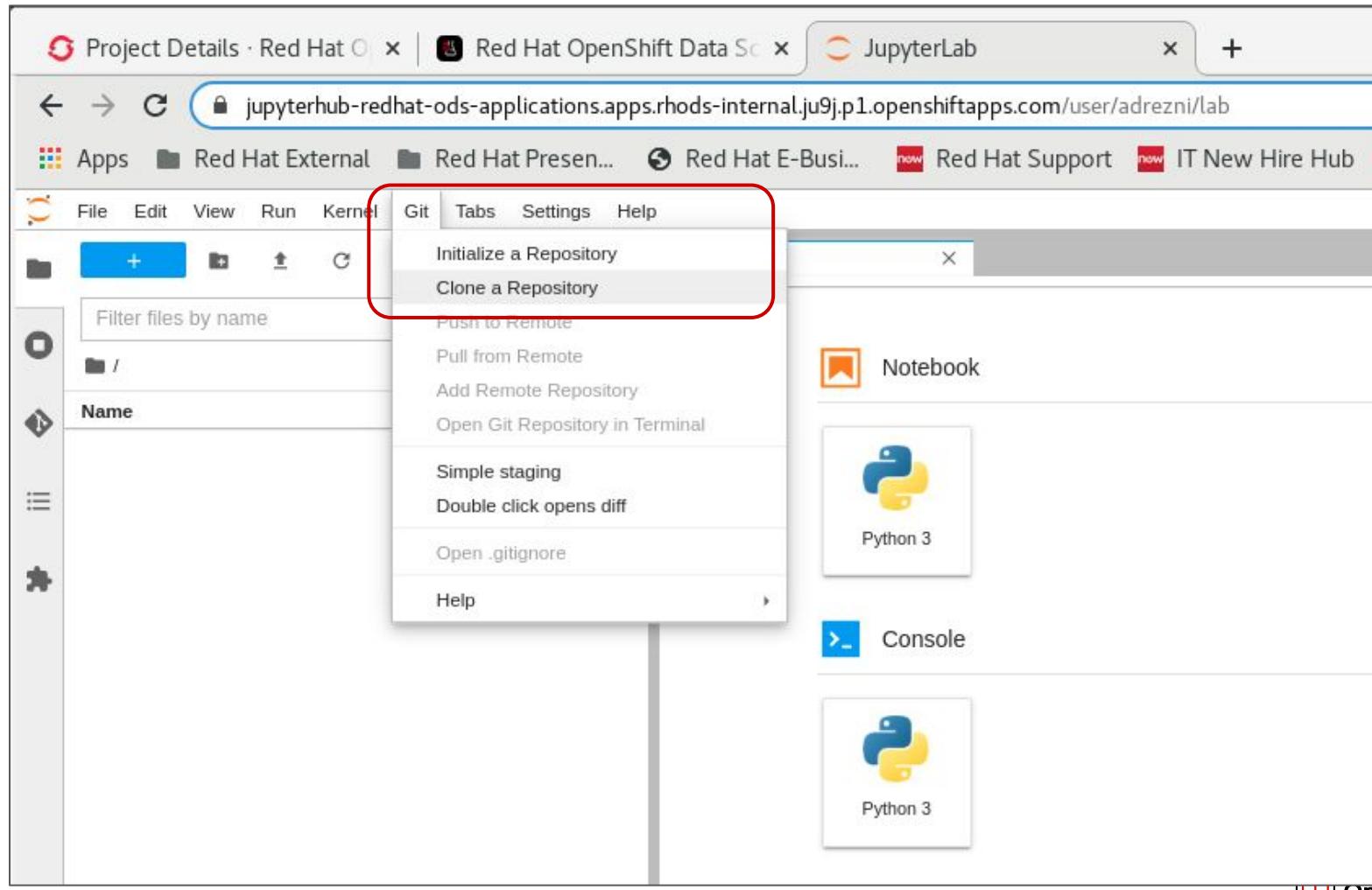
The screenshot shows a web browser window with three tabs open: "Project Details · Red Hat", "Red Hat OpenShift Data Sc...", and "JupyterHub". The "JupyterHub" tab is active, displaying the URL jupyterhub-redhat-ods-applications.apps.rhods-internal.ju9j.p1.openshiftapps.com/hub/spawn-pending/adrezni. The page content indicates that the server is starting up and will be redirected automatically. Below this, an "Event log" section shows the following entries:

- 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"

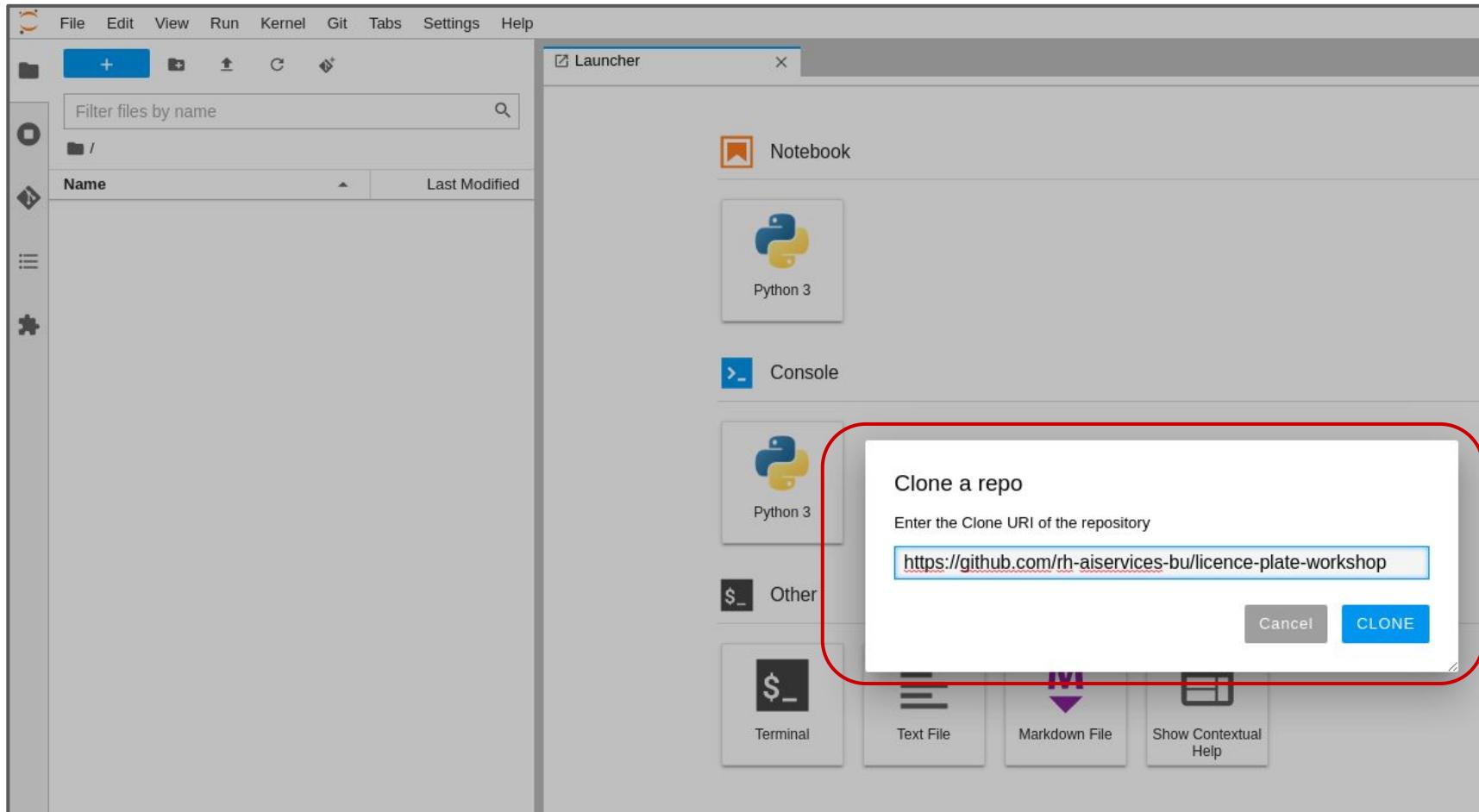
Welcome to JupyterLab!



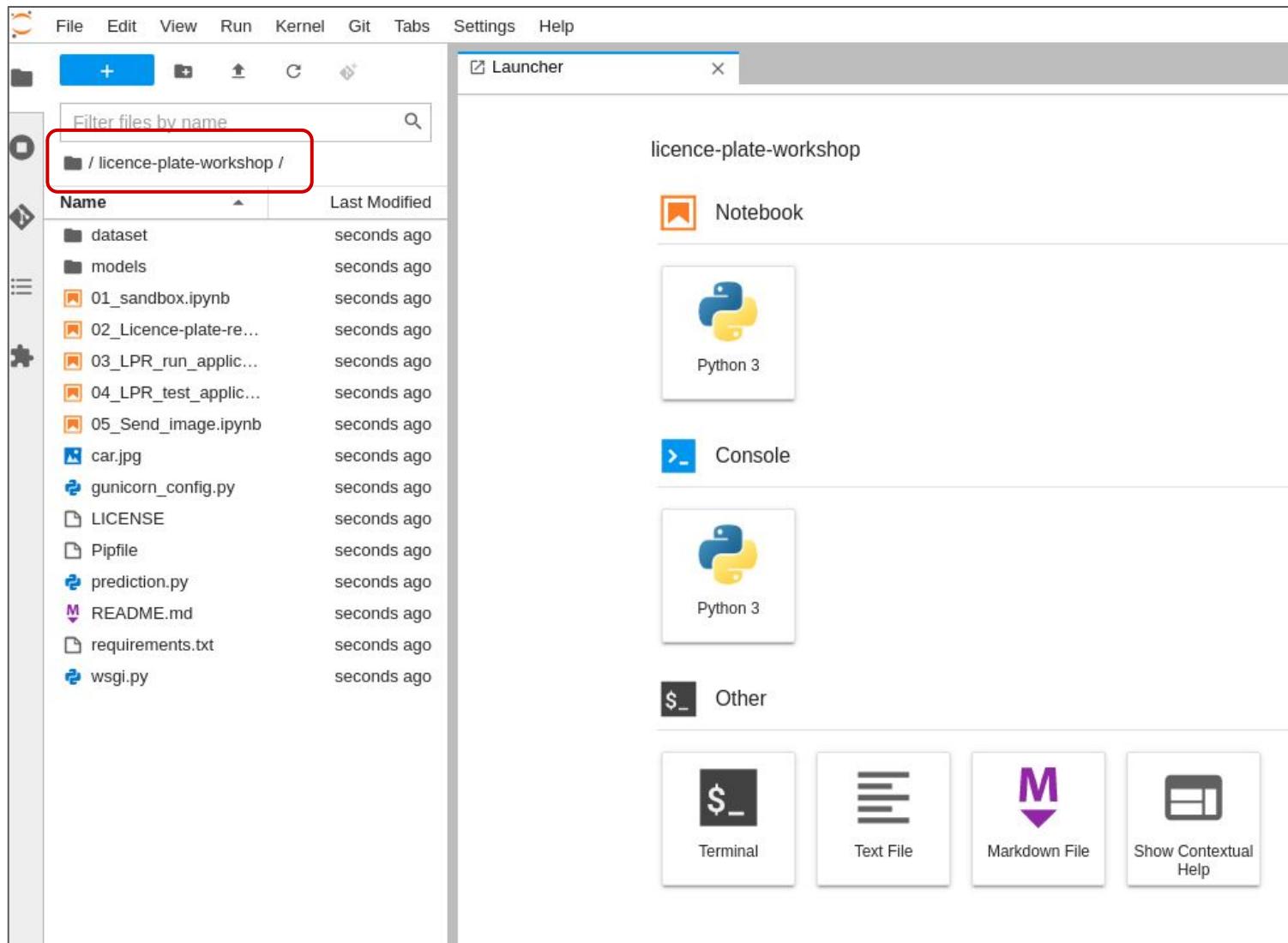
Choose “Clone a Repository” from the “Git” menu



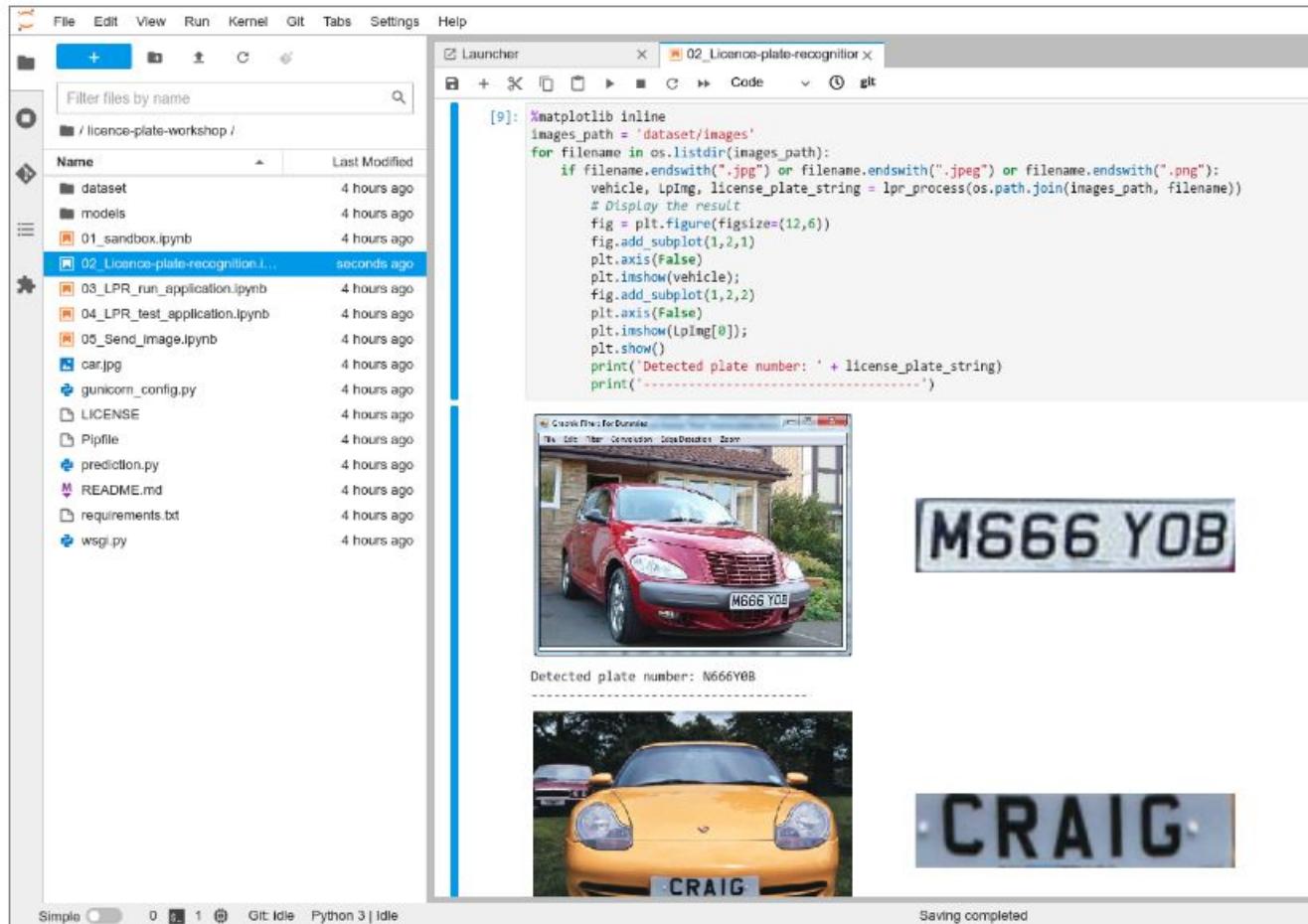
Clone license plate GIT repo



License plate repo



Ready to Experiment using a Jupyter Notebook



The screenshot shows a Jupyter Notebook interface. On the left is a file browser with a list of files in the directory `/licence-plate-workshop/`. On the right is a code editor and an output area.

Code Editor:

```
[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:

Two images of cars are shown, each with its detected license plate displayed below it.

- 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**.

Below the images, the text "Saving completed" is visible.

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).

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

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

Package the Model as an API - Launch the Server

The screenshot shows a Jupyter Notebook interface with a single code cell. The cell contains Python code to import sys and install requirements from requirements.txt if the executable is not sys. It also includes a note about launching the Flask application and ignoring CUDA errors. A message at the bottom indicates the server is running on port 5000.

```
[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).

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 - Test the Flask App

The screenshot shows a Jupyter Notebook interface with a single open cell. The cell contains code and explanatory text for testing a Flask application.

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

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 `msg1.py` file.

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

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 'Project Details' view for the 'audreytest' project in the 'Red Hat OpenShift Dedicated' namespace. The URL in the browser is `console-openshift-console.apps.rhods-internal.ju9j.p1.openshiftapps.com/project-details/ns/audreytest`. The left sidebar includes links for Developer, +Add, Topology, Monitoring, Search, Builds, Pipelines, Helm, Project, ConfigMaps, and Secrets. The main content area displays the project's name, status (Active), utilization (CPU usage over the last hour), and activity (ongoing and recent events). A sidebar on the right lists Red Hat applications and OpenShift Managed Services.

Project: audreytest

Overview Details Project access

Details

Name: audreytest

Requester: adrezni

Labels: control-plane=kubeflow, katic-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
CPU	4.02m	6m	4m	2m	1000 MIR

Activity

Ongoing: There are no ongoing activities.

Recent events: There are no recent events.

Red Hat applications

- OpenShift Cluster Manager

OpenShift Managed Services

- Red Hat OpenShift Data Science

Build the application inside OpenShift

The screenshot shows a web browser window with the URL `console-openshift-console.apps.rhods-internal.ju9j.p1.openshiftapps.com/project-details/ns/audreytest`. The page is titled "Project Details · Red Hat". The main content area displays a project named "audreytest" (status: Active). Below the project name are tabs for "Overview", "Details", and "Project access". A large modal dialog is centered on the screen, prompting the user to "Select Project...". The dialog includes a dropdown menu set to "Project: user1-project" and a dropdown menu set to "Application: all applications". It also features a "Create Project" button and a list of existing projects, with "user1-project" highlighted in blue and a cursor pointing at it. To the right of the modal, there is a sidebar titled "Red Hat applications" listing "OpenShift Cluster Manager" and "Red Hat OpenShift Data Science". The bottom right corner of the page has the "Red Hat OpenShift" logo.

Build the application inside OpenShift

The screenshot shows the Red Hat OpenShift Data Science 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 left sidebar has sections for 'Developer' (selected), 'Topology', 'Monitoring', 'Search', 'Builds', 'Pipelines', 'Helm', 'Project' (selected), 'ConfigMaps', and 'Secrets'. The main content area displays a large card with a GitHub icon and the text 'From Git'. Below it, it says 'Import code from your Git repository to be built and deployed' with a 'Select' button. At the bottom of the card, there is a 'Create' button. A hand cursor is shown pointing at the 'Create' button. To the right of the main card, there is a sidebar titled 'Red Hat applications' containing 'OpenShift Cluster Manager' and 'Red Hat OpenShift Data Science'. The bottom right corner features the 'Red Hat OpenShift' logo.

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.

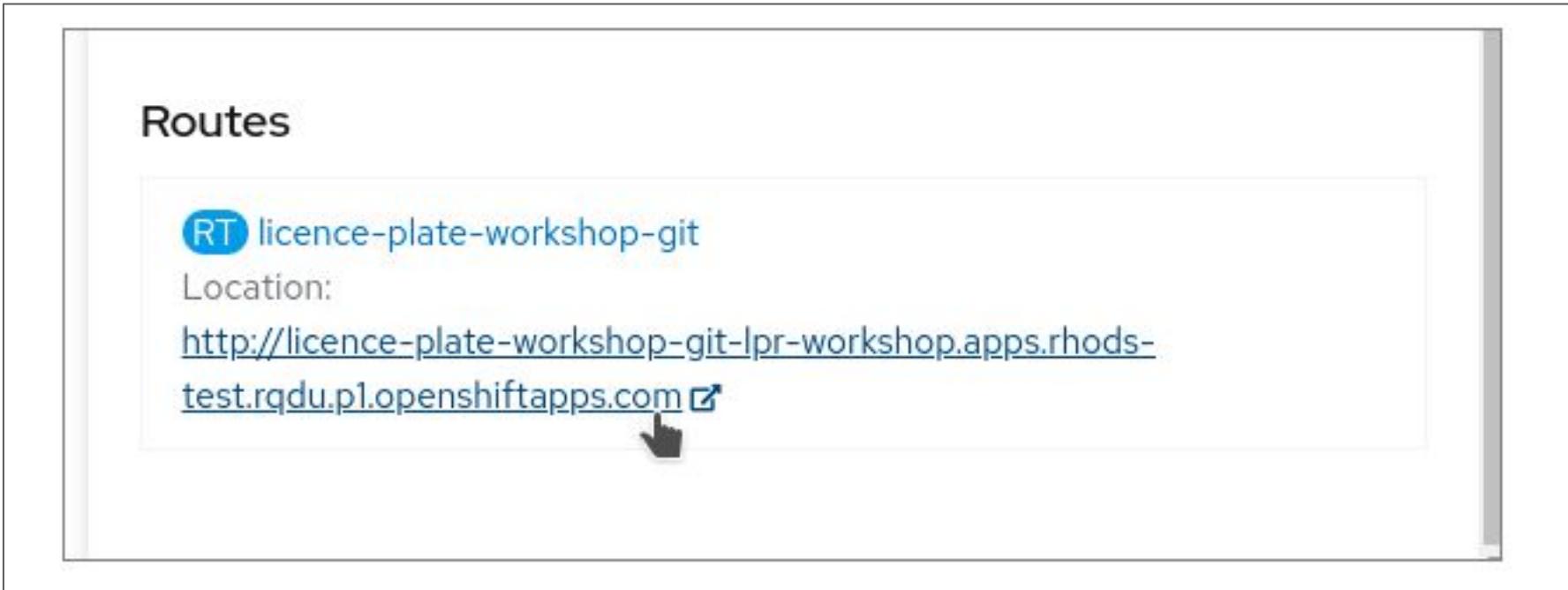
Advanced options

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](#)

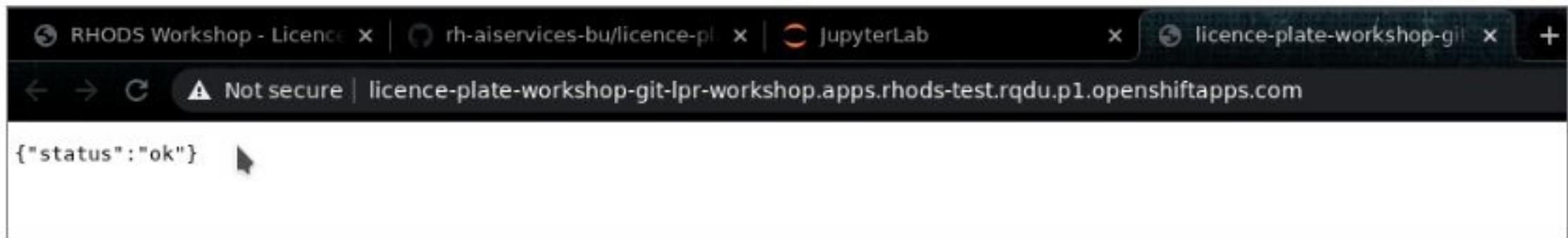
Build the application inside OpenShift



The screenshot shows the 'Routes' section of the OpenShift web interface. A single route is listed:

- 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):

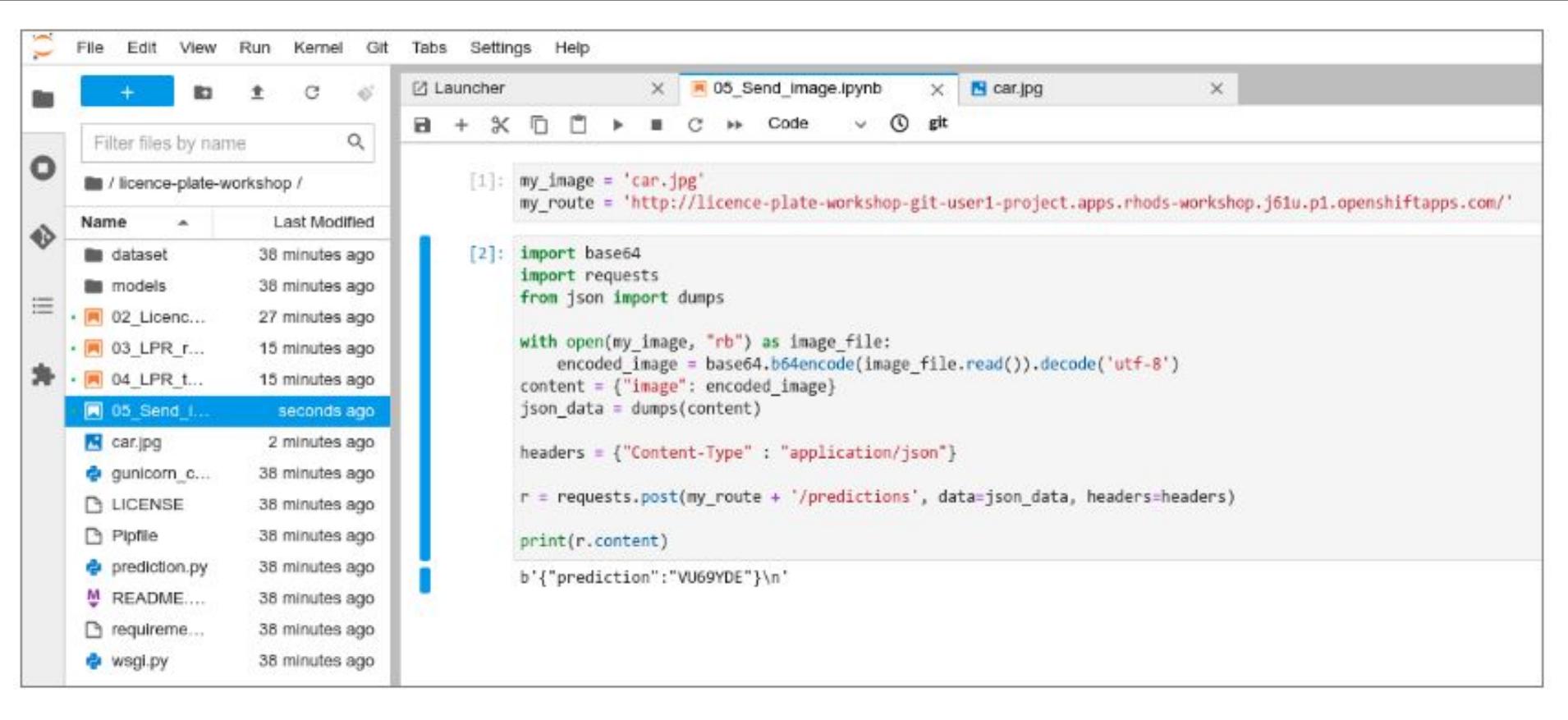
```
(echo -n '{"image": ""}; base64 car.jpg; echo "")' | curl -H "Content-Type: application/json" -
```

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. On the left, there's a file browser with a list of files in the 'licence-plate-workshop' directory. The file '05_Send_Image.ipynb' is selected and highlighted with a blue border. 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

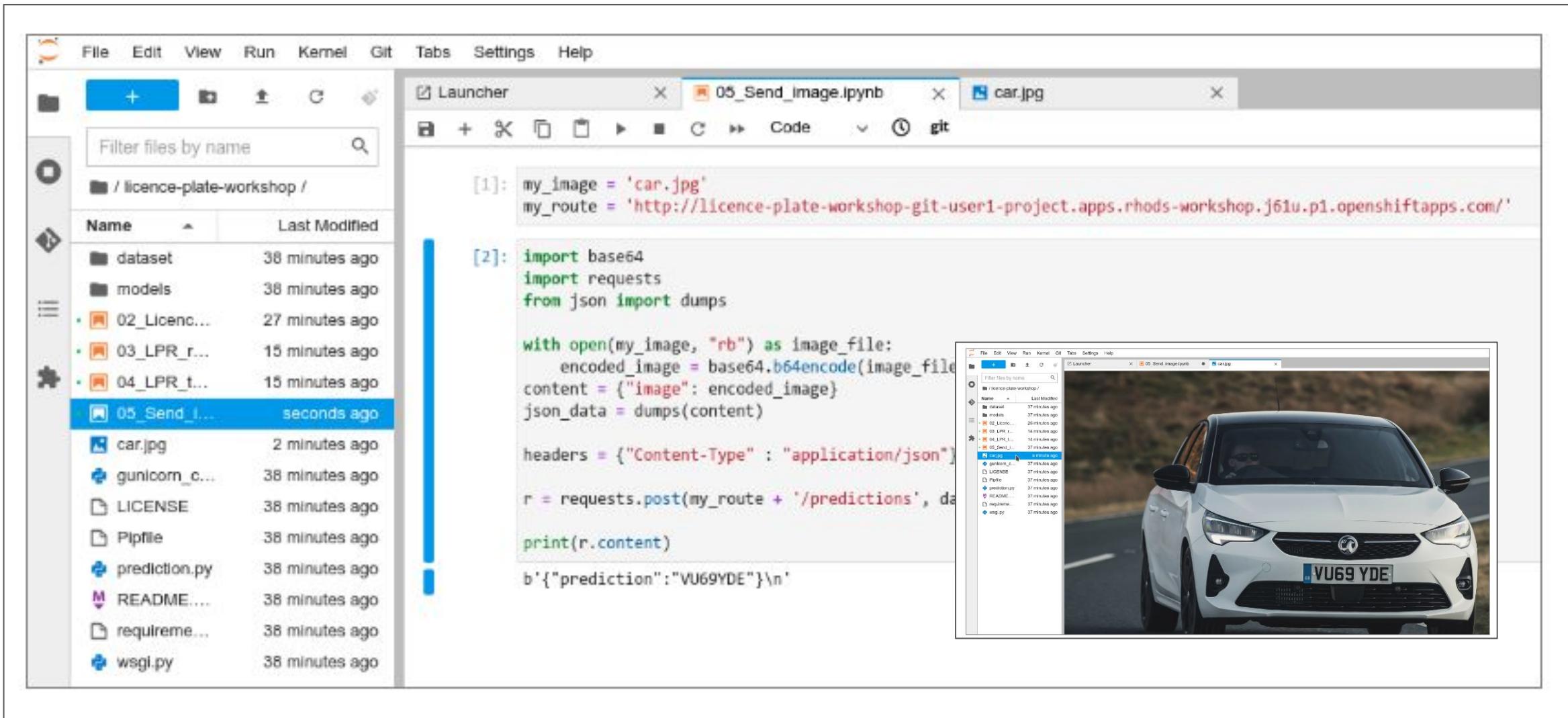
with open(my_image, "rb") as image_file:
    encoded_image = base64.b64encode(image_file.read()).decode('utf-8')
content = {"image": encoded_image}
json_data = dumps(content)

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 Menu:** File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help.
- Toolbar:** Includes icons for file operations like new, open, save, and a search bar labeled "Filter files by name".
- File Explorer:** Shows the directory structure of the workspace:
 - / licence-plate-workshop /
 - Name Last Modified
 - dataset 38 minutes ago
 - models 38 minutes ago
 - 02_Licens... 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
 - wsgl.py 38 minutes ago
- Code Cell:** Contains 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

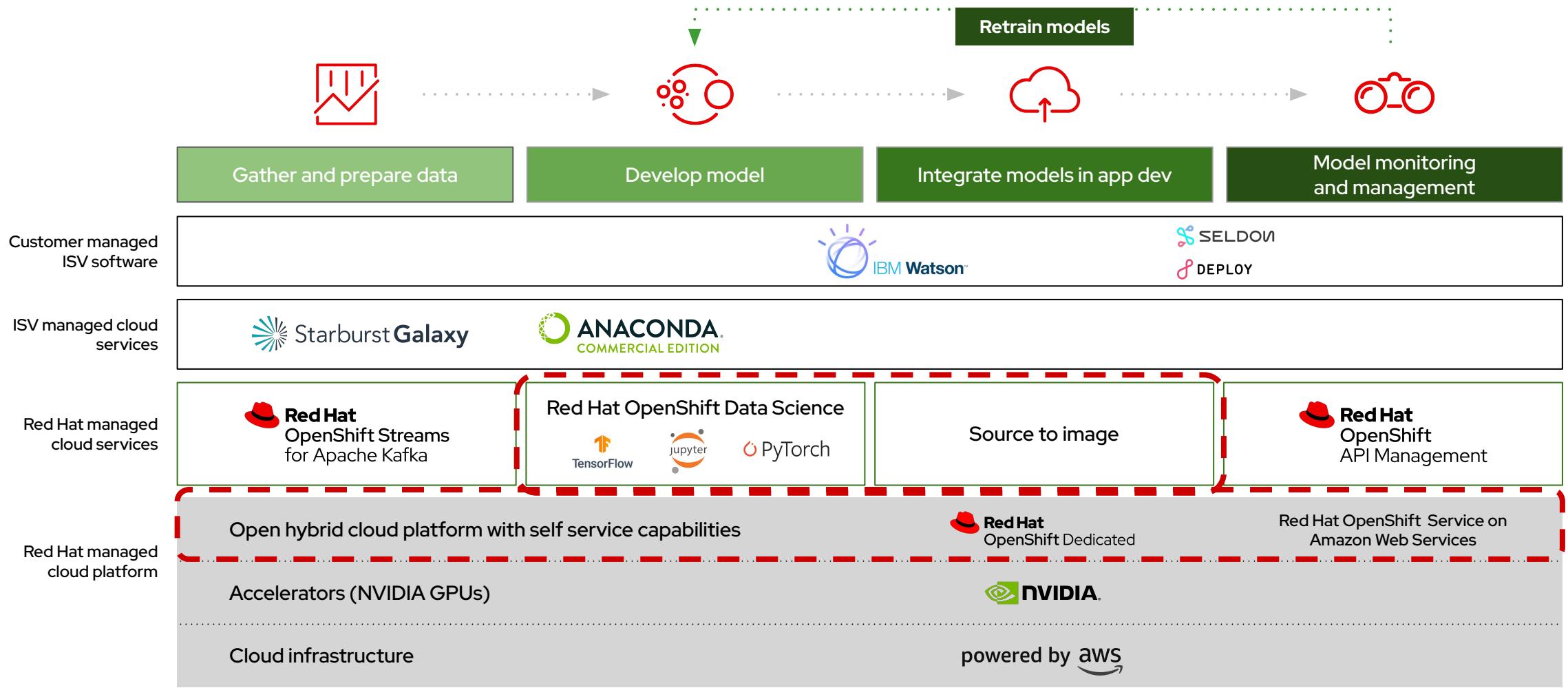
with open(my_image, "rb") as image_file:
    encoded_image = base64.b64encode(image_file.read())
content = {"image": encoded_image}
json_data = dumps(content)

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

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

print(r.content)
b'{"prediction":"VU69 YDE"}\n'
```
- Output Cell:** Displays the prediction result from the deployed application, showing a white car with the license plate "VU69 YDE".

Red Hat Managed Cloud Services



What did we learn today?

- ▶ A Model's role in an Intelligent Application
- ▶ What are Managed Services?
 - Who uses Managed Services? Hint... not only Data Scientists
- ▶ What do Managed Services have to do with Model Delivery?
- ▶ Where do I find Managed Services?
- ▶ Demo of using Red Hat OpenShift Data Science platform and it's Managed Services.

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



youtube.com/openshift



facebook.com/OpenShift



twitter.com/openshift