

Red Hat OpenShift AI

RHOAI Getting Started



Agenda

- Data Science Projects
- Workbenches
- Data Connections
- Model Serving

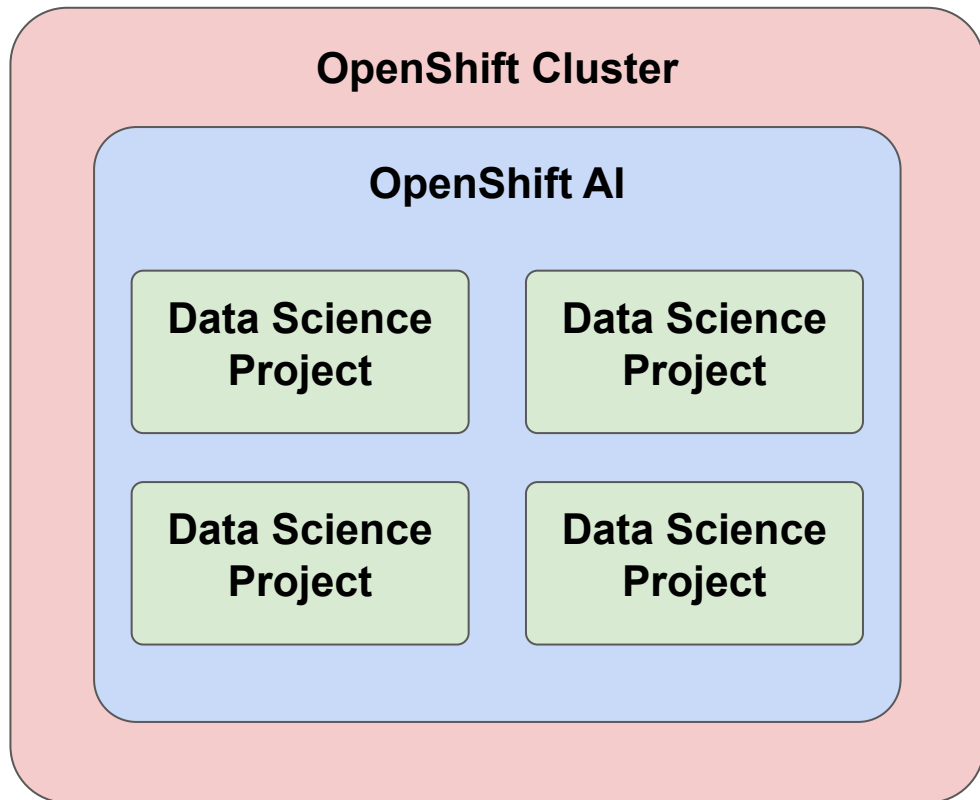
Data Science Projects

Data Science Projects

The screenshot displays the Red Hat OpenShift Data Science console interface. On the left is a dark sidebar with navigation options: Applications (with a dropdown arrow), Enabled, Explore, Data Science Projects (highlighted), Model Serving, Resources, and Settings (with a right arrow). The main content area is titled 'Data science projects' and includes the instruction 'View your existing projects or create new projects.' Below this, there is a search bar with a dropdown arrow and the text 'Find by name', a blue button labeled 'Create data science project', and a link labeled 'Launch Jupyter'. A table lists four projects: 'face detection' (admin), 'fraud detection' (admin), 'myproject' (admin), and 'object detection' (admin). Each project row shows its associated Workbench (JupyterLab, R Studio, VS Code, or development), its Status (all are 'Running' with a toggle switch), and its Created timestamp. The table has columns for Name, Workbench, Status, and Created.

Name ↑	Workbench	Status	Created ↓
face detection ⓘ admin	JupyterLab ↗ R Studio ↗ VS Code ↗	<input checked="" type="checkbox"/> Running <input checked="" type="checkbox"/> Running <input checked="" type="checkbox"/> Running	5/7/2023, 2:01:57 PM
fraud detection ⓘ admin	development ↗	<input checked="" type="checkbox"/> Running	5/7/2023, 2:05:15 PM
myproject ⓘ admin	myworkbench ↗	<input checked="" type="checkbox"/> Running	5/8/2023, 11:15:59 AM
object detection ⓘ admin	development ↗	<input checked="" type="checkbox"/> Running	5/7/2023, 2:03:58 PM

Data Science Projects



In OpenShift AI, you can have multiple **Data Science Projects (DSP)**.

A **DSP** can be created by administrators, or directly by users if they are allowed to.

Each DSP is **isolated** from the others. Only user with **access privileges** can see what is inside a DSP (applications, data,...).

You can **grant access** to other **users** and **groups** to your DSP.

Collaborate in a project

Components

Permissions

Add users and groups that can access the project.

Users

Name ↑	Permission ↕	Date added ↕
<input type="text" value="another-user"/>	<div>Edit ▼</div>	<div>✓ ✕</div>

+

 Add user

Workbenches

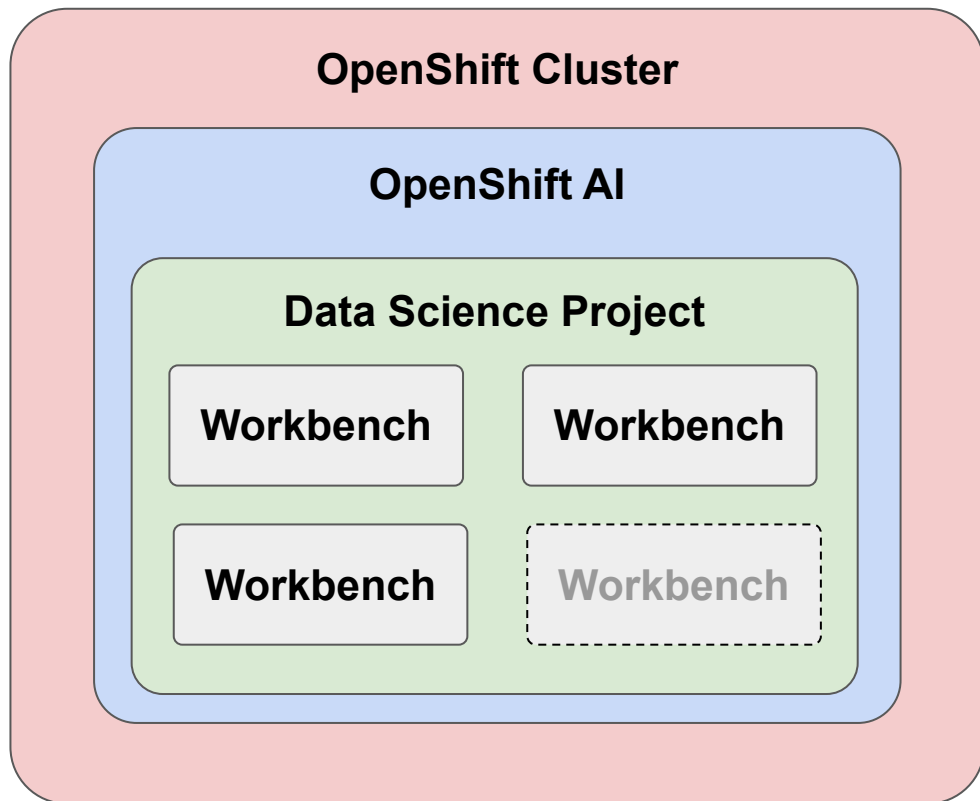
Workbenches

Workbenches

Create workbench

Name ↑	Notebook image	Container size	Status ↑
▼ Failure prediction ?	Standard Data Science Python v3.9	Small	<input type="checkbox"/> Stopped Open ↗ ⋮
Workbench storages Failure prediction Max 20Gi Mount path: / + Add storage	Packages Boto3 v1.26 Kafka-Python v2.0 Kfp-tekton v1.5 Matplotlib v3.6 Numpy v1.24 Pandas v1.5 Scikit-learn v1.2 Scipy v1.10 Elyra v3.15	Limits 2 CPU, 2Gi Memory Requests 1 CPU, 2Gi Memory	

Workbenches



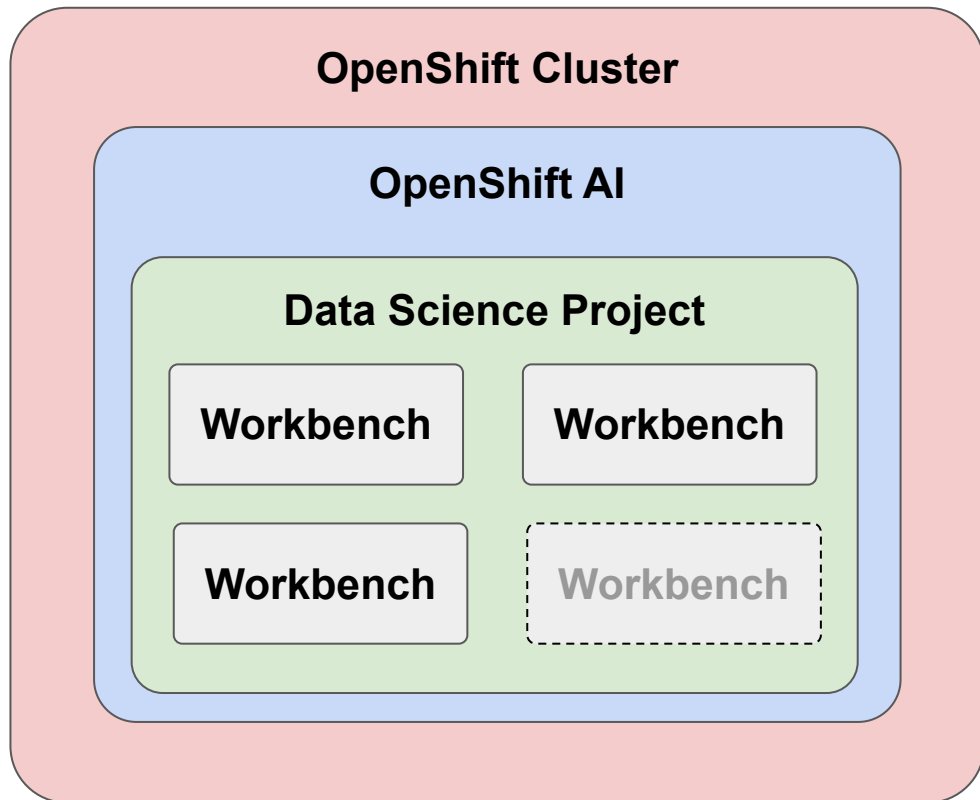
A **Workbench** is a **working space** that you create in a Data Science Project.

A Workbench can be based on different development environments: **Jupyter**, **VSCode**, **RStudio**,...

Different **flavors** or Workbenches are provided with OpenShift AI, including different built-in libraries and tools: **Tensorflow**, **PyTorch**,...

You can **extend** the capabilities by **importing** your own **custom Workbench**.

Workbenches - continued



You can have as **many Workbenches** as you want in a Data Science Project.

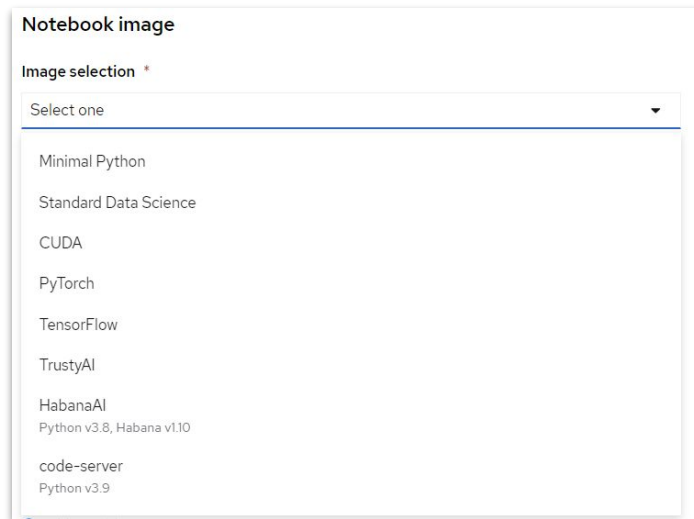
Each **Workbench** can have a **different configuration**: CPU, RAM, GPU, environment,...

Workbenches can be **started and stopped** at any time, their configuration is preserved.

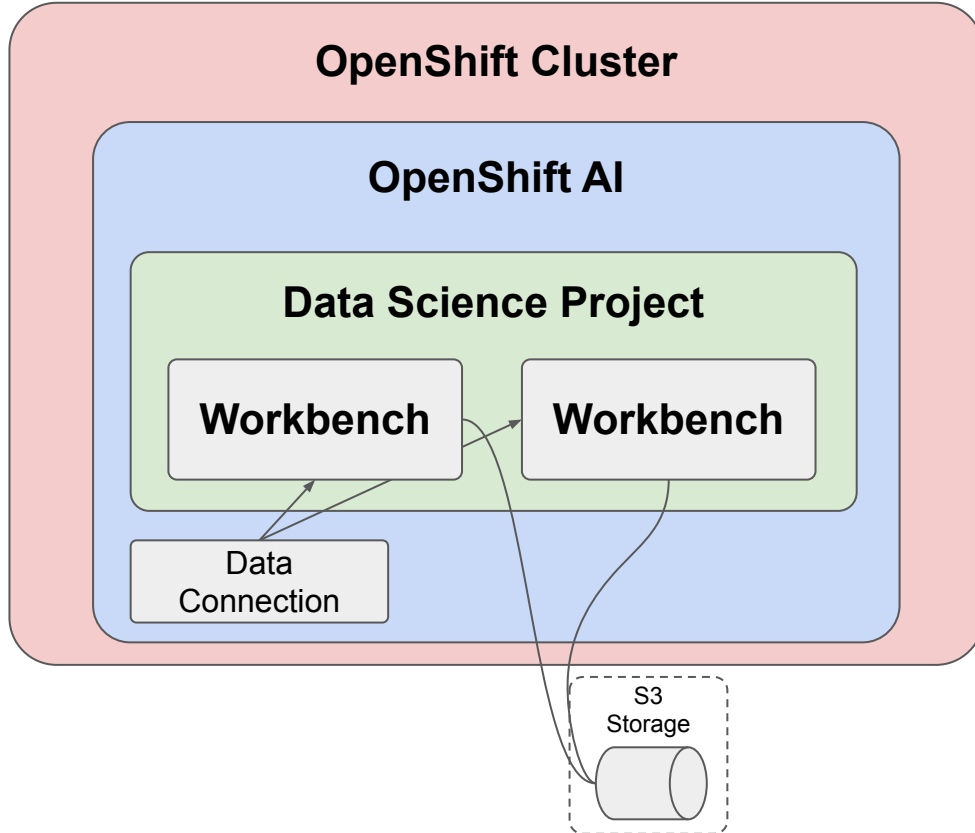
Each **Workbench** has its own dedicated **persistent storage** that gets **reconnected** to it every time you start the Workbench.

Supported Workbenches

- Minimal Python
- Standard Data Science
- CUDA
- PyTorch
- TensorFlow
- TrustyAI
- HabanaAI
- code-server



Workbenches - Data

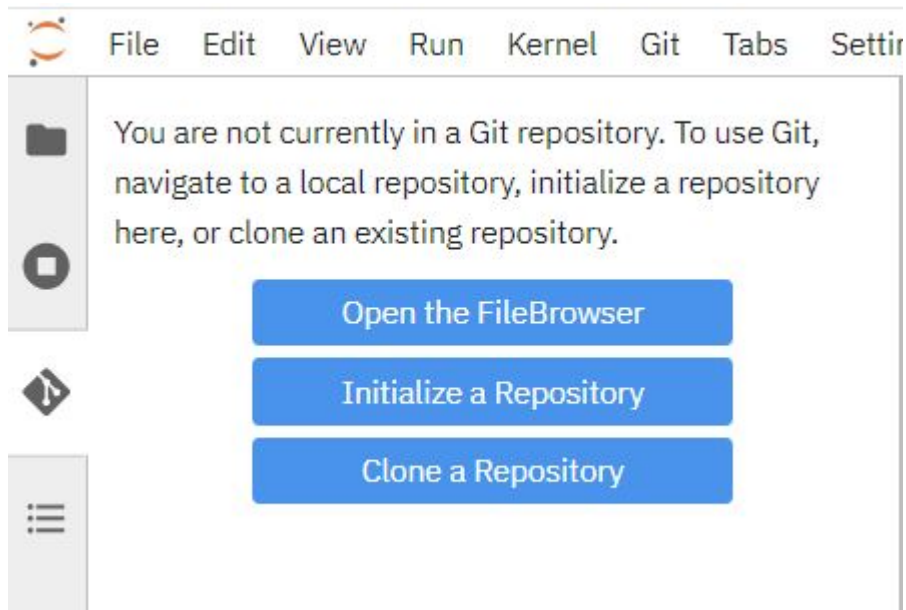


Generally, Workbenches **cannot share** standard **storage**.

However different workbenches can be **connected** simultaneously to the same **Object Storage** resource (S3) to exchange data and resources.

Object Storage resources can be easily **defined** and used in Workbenches using **Data Connections**.

Collaborate using Git



Data Connections

Simple secrets

Data connections			Add data connection
Name	Type	Connected workbenches	
My Storage ?	Object storage	VS Code Workshop	
Pipeline Artifacts ?	Object storage	No connections	

Edit data connection

Name *
my-first-data-connection

Access key *
[REDACTED]

Secret key *
[REDACTED]

Endpoint *
https://s3.eu-de.cloud-object-storage.appdomain.cloud

Region
eu-de

Bucket
my-bucket

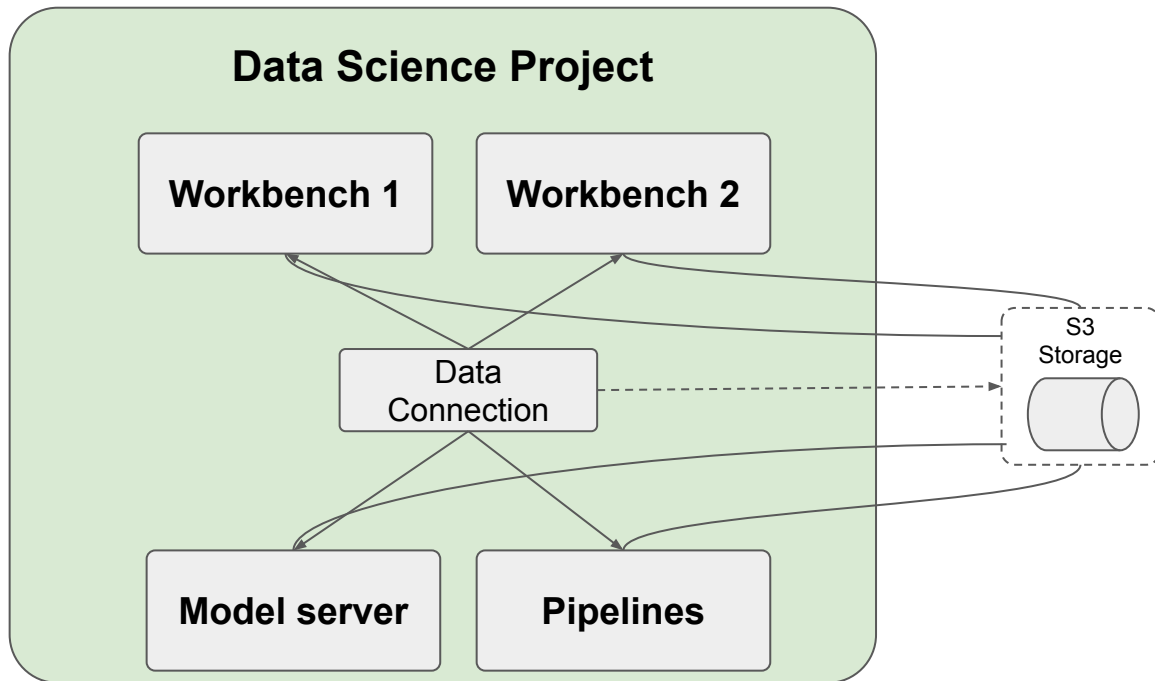
Connected workbench
my-first-workbench x my-second-workbench x Select a workbench to connect

Connect to workbenches that do not already have a data connection

Unsaved work will be lost
Running workbenches will restart upon updating. To avoid losing your work, save any recent data in the following running workbenches: my-second-workbench.

Update data connection Cancel

S3 as Central Artifact Store



Boto3: Interacting with S3

```
import os
import boto3

key_id = os.getenv("AWS_ACCESS_KEY_ID")
secret_key = os.getenv("AWS_SECRET_ACCESS_KEY")
endpoint = os.getenv("AWS_S3_ENDPOINT")
bucket_name = os.getenv("AWS_S3_BUCKET")

s3_data_path = "dataset.csv"

s3 = boto3.client(
    "s3",
    aws_access_key_id=key_id,
    aws_secret_access_key=secret_key,
    endpoint_url=endpoint,
    use_ssl=True)

s3.download_file(bucket_name, s3_data_path,
    "my/local/path/dataset.csv")
```

```
import os
import boto3

source_path = "model.onnx"
s3_destination_path = "models/model.onnx"

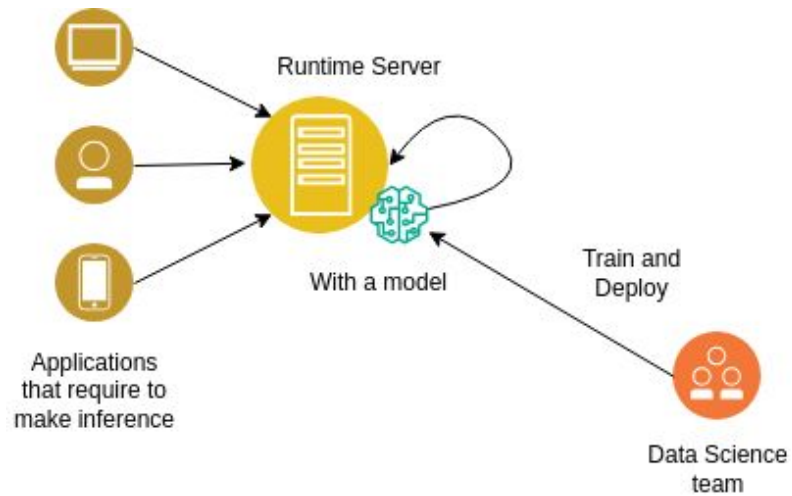
key_id = os.getenv("AWS_ACCESS_KEY_ID")
secret_key = os.getenv("AWS_SECRET_ACCESS_KEY")
endpoint = os.getenv("AWS_S3_ENDPOINT")
bucket_name = os.getenv("AWS_S3_BUCKET")

s3 = boto3.client(
    "s3",
    aws_access_key_id=key_id,
    aws_secret_access_key=secret_key,
    endpoint_url=endpoint,
    use_ssl=True)

s3.upload_file(source_path, bucket_name, Key=s3_destination_path)
```

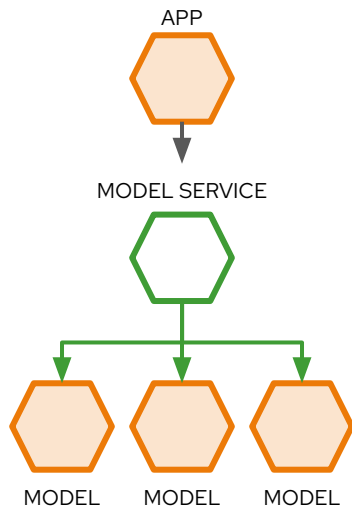
Model Serving

What is Model Serving?

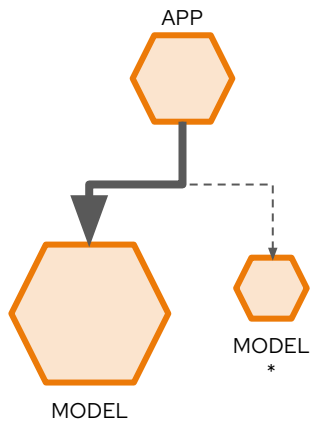


Models as stateless microservices

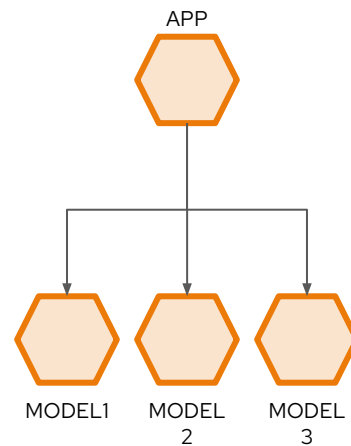
SCALE HORIZONTALLY



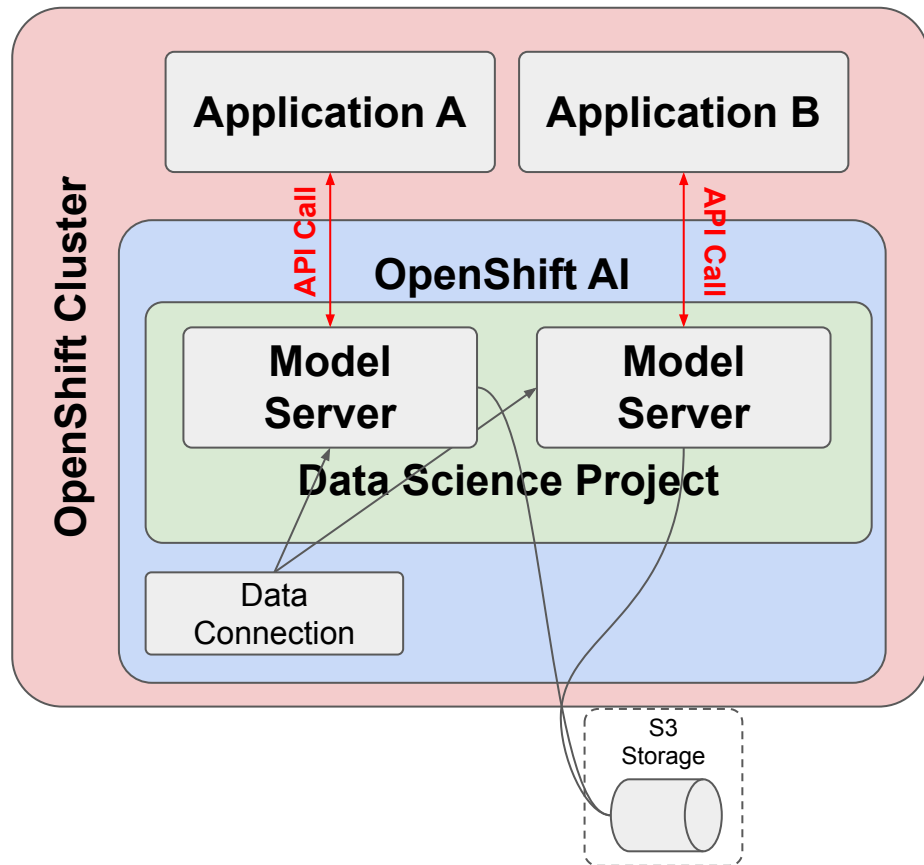
PHASED ROLLOUTS



MULTIPLE TRIALS



Model Serving



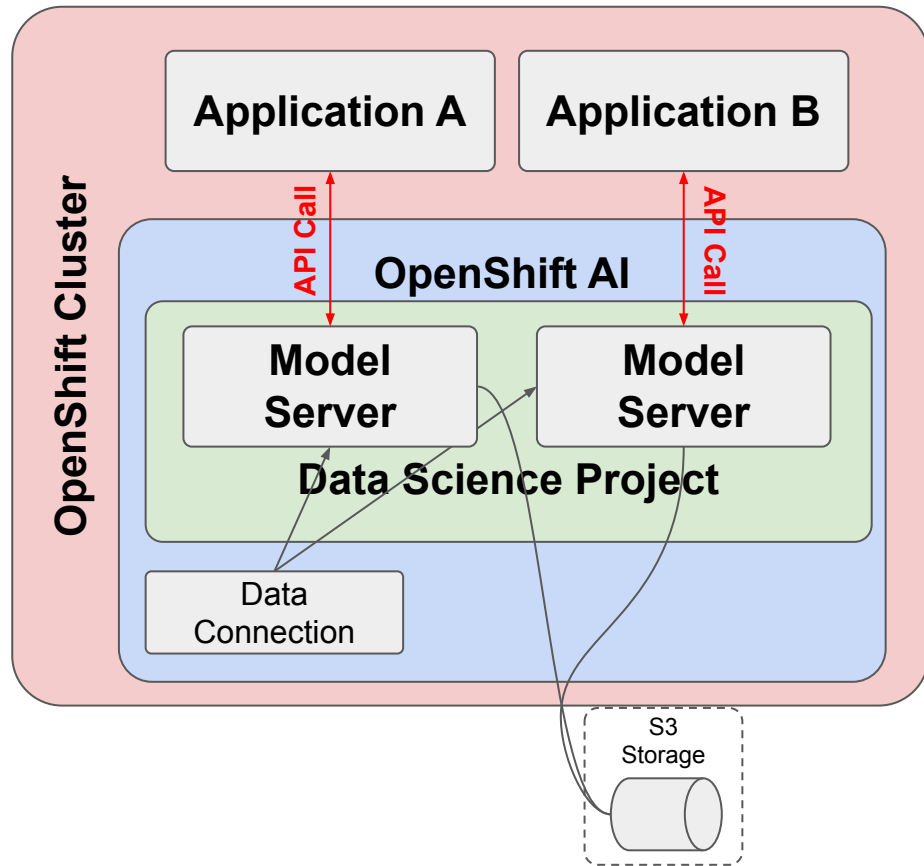
In a Data Science Project, you can deploy one or several **Model Servers**.

Models must be stored on **Object Storage (S3)**.

When a **Model Server** starts, it will read a model file and service it through an **API**.

Your application can then directly query this **API** to make an **Inference** using the served model.

Model Serving - continued

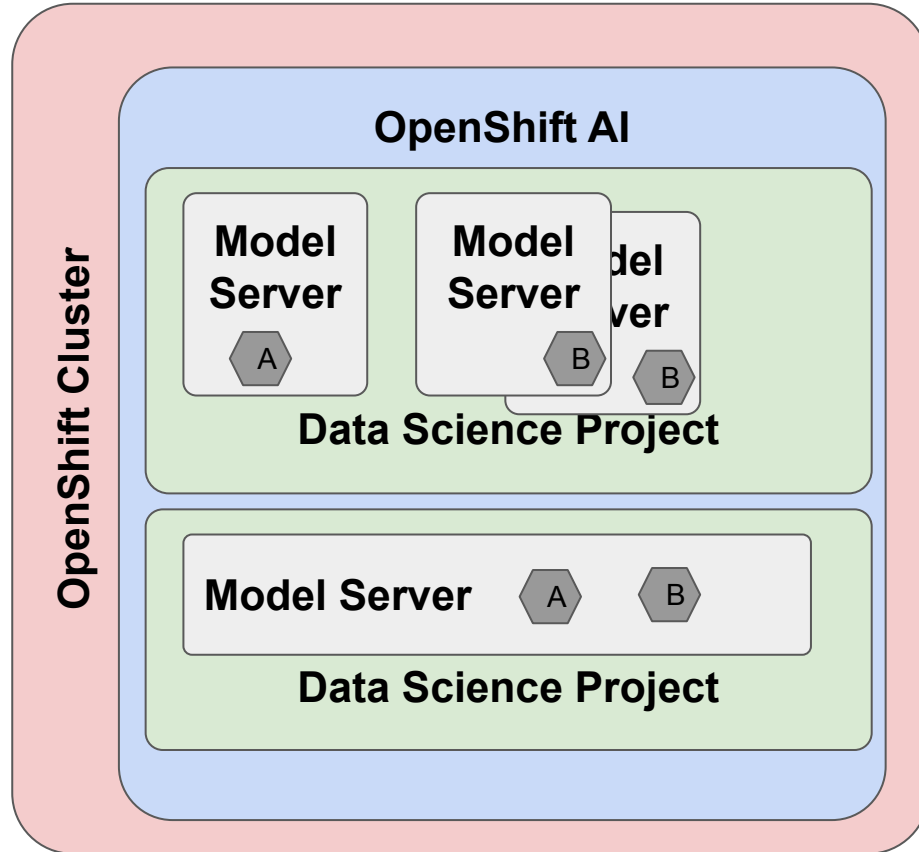


Different **types of Model Servers** are available for different needs: lots of small models (**Multi-model serving**), single larger models like Large Languages Models (**Single stack model serving**).

Different **Runtimes** are available, adapted to different types of models (LLMs, Predictive AI,...).

You can **extend** the capabilities by **importing** your own **runtime**.

Model Serving - single vs multiple



Top: Single-Model Model Serving
Each model is in its own server
You can have multiple "replicas"

Bottom: Multi-Model model Serving
Multiple Models are served by the
same server.
You can also have multiple
"replicas" (not depicted)

Supported Runtimes and their Frameworks

Single-model serving

- Caikit TGIS ServingRuntime for Kserve
 - Caikit
- Text Generation Inference Server
 - PyTorch
- OpenVino Model Server
 - ONNX
 - OpenVino IR
 - TensorFlow

Multi-model serving

- OpenVino Model Server
 - ONNX
 - OpenVino IR
 - TensorFlow

Use the one that fits your use case and is compatible with your format.