# Implementing a Digital Assistant with Red Hat OpenShift AI on Dell APEX Cloud Platform for Red Hat OpenShift

## With a Language Model (LLM) and the Retrieval Augmented Generation (RAG) framework

November 2023

H19833

Design Guide

Abstract

This design guide describes the architecture and design of the Dell Technologies Validated Design for deploying a digital assistant on Dell APEX Cloud Platform for Red Hat OpenShift using Red Hat OpenShift AI. This solution leverages a LLM and the RAG technique in combination with a set of vectorized documents.

Dell Technologies Solutions

Dell
Validated Design

## Copyright

# Contents

# Chapter 1 Introduction

This chapter presents the following topics:

# Business challenge

Artificial Intelligence (AI) is evolving rapidly and is becoming critical for businesses to remain competitive. However, developing and managing complex algorithms and models to run AI systems can introduce challenges, as many organizations lack the AI expertise and time to design, deploy, and manage solution stacks.

AI applications have to scale to handle extremely large datasets and accommodate an increase in user loads. Efficiently maintaining a high-performance application can be a challenge for organizations that do not have the proper infrastructure. AI workloads can be resource-intensive and lead to high infrastructure and operational costs. It is essential to optimize resource usage to minimize costs while maintaining operational efficiency.

Organizations often have trouble searching through their internal documents, as they are limited to text-based search, which does not offer efficient querying. Large Language Model (LLM) based applications offer a solution to this challenge, but building, deploying, and managing a LLM can be complex and time consuming.

The cloud-native landscape is very complex. Red Hat® OpenShift® is a trusted and proven application platform that provides a turnkey solution enabling organizations to build, deploy, and run their applications.

# Solution introduction

## Overview

This design guide describes the process of deploying a digital assistant using Red Hat OpenShift AI on Dell APEX Cloud Platform for Red Hat OpenShift. This solution is designed to create a cloud native AI application, with the ease of deployment and manageability.

LLMs are highly sophisticated AI models that are designed to understand and generate human-like text, enabling a wide range of natural language processing applications. One limitation of LLMs is that once they are generated, they do not have access to information beyond the date that they were trained. Retrieval Augmented Generation (RAG) can extend the functionality of the LLMs by retrieving facts from an external knowledge base, in this case, from a Redis in-memory Vector database.

Digital assistants are designed to assist users by answering questions and processing simple tasks. Answers remain up to date and contain information unique to the organization by anchoring the model with relevant documentation.

In this solution, we have deployed a LLM based digital assistant that can answer user queries related to domain specific documents. As text-based searches are limited in obtaining the right data, a digital assistant can help retrieve more accurate and relevant results using semantic search and natural language processing.

The Red Hat OpenShift Container Platform provides a robust containerization platform and Kubernetes-based orchestration framework that enables organizations to build, deploy, and manage applications and databases efficiently across on-premises and multicloud environments.

Dell APEX Cloud Platform for Red Hat OpenShift is designed collaboratively with Dell Technologies and Red Hat to optimize and extend OpenShift deployments on-premises with an integrated operational experience. By combining Dell's expertise in delivering robust infrastructure solutions with Red Hat's industry-leading OpenShift Container Platform, this collaboration empowers organizations to start on a transformative journey towards modernization and innovation.

The APEX Cloud Platform for Red Hat uses separate storage nodes to provide persistent storage for the compute cluster. This separation enables the compute and storage nodes to scale independently.

Red Hat OpenShift AI offers organizations an efficient way to deploy an integrated set of common open-source and third-party tools to perform ML modeling. The ML models developed using Red Hat OpenShift AI are portable to deploy in production, on containers, on-premises, at the edge, or in the public cloud.

Utilizing object storage for LLMs and huge datasets is crucial for modern data-driven applications, due to their reliability and scalability. Object storage is leveraged in this solution to store Llama 2 model, relevant datasets, and artifacts.

### Document purpose

The purpose of this document is to provide design guidance for deploying a digital assistant using Red Hat OpenShift AI on Dell APEX Cloud Platform for Red Hat OpenShift.

**Note**: The contents of this document are valid for the described software and hardware versions. For information about updated configurations for newer software and hardware versions, contact your Dell Technologies sales representative.

### Audience

This design guide is intended for AI solution architects, data scientists, data engineers, IT infrastructure managers, and IT personnel who are interested in, or considering, implementing AI and ML deployments.

### Disclaimer

This document provides design guidance on deploying a digital assistant using open-source tools. It is the responsibility of the user to review and comply with the licensing terms and conditions for each tool mentioned in this document. Licensing information for each tool can be found on the respective tool's official website or in its documentation. See Appendix A to access the licensing page for some of the open-source tools used during validation of this solution.

# Introduction

Dell validated solutions offer customers faster time-to-market and reduced risks compared to creating their own solutions. For this solution, Dell systems engineers validated deploying a digital assistant using Red Hat OpenShift AI on Dell APEX Cloud Platform for Red Hat OpenShift.

This design guide outlines the concepts involved in developing the solution and presents the key elements of the solution, specifically the architecture and design. This guide also describes the Red Hat OpenShift AI deployment with Caikit and TGIS, followed by its

configuration. The guide also discusses digital assistant deployment, configuration, and execution.

# Terminology

The following table provides definitions for some of the terms that are used in this document.

**Table 1. Terminology**

| Term | Definition |
|---|---|
| API | Application Programming Interface |
| NLP | Natural Language Processing |
| LLM | Large Language Model |
| LLaMA | Large Language Model Meta AI |
| ACP | Dell APEX Cloud Platform |
| Red Hat OCP | Red Hat OpenShift Container Platform |
| RHCOS | Red Hat Enterprise Linux CoreOS |
| S3 | Simple Storage Service |
| Buckets | Buckets are object containers that are used to control access to objects. |
| DC | Domain controller |
| DNS | Domain Name System |
| gRPC | Google Remote Procedure Call |
| TGI | Text Generation Inference |
| AI | Artificial Intelligence |
| FIPS | Federal Information Processing Standards |
| ML | Machine Learning |
| NFD | Node Feature Discovery |
| PVC | Persistent Volume Claim |
| Git LFS | Git Large File Storage |
| isvc | Inference Service |
| SDS | software-defined storage |
| FIPS | Federal Information Protection Standard |

# Chapter 2 Solution Concepts

This chapter presents the following topics:

# Solution concepts

## Overview

This section discusses the concepts involved in building the solution. The following list includes the hardware and software layers in the solution stack.

- Dell APEX Cloud Platform for Red Hat OpenShift
- Red Hat OpenShift Container Platform
- Red Hat OpenShift AI
- Digital assistant related components (Llama 2, LangChain, Redis, Gradio, Caikit, and Text Generation Inference [TGI])

## Dell APEX Cloud Platform

Dell APEX is a portfolio of cloud services that is based on a cloud consumption model that delivers IT as-a-service, with no upfront costs and pay-as-you-go subscriptions. Dell APEX Cloud Platforms deliver innovation, automation, and integration across your choice of cloud ecosystems, empowering innovation in multicloud environments.

Dell APEX Cloud Platforms are a portfolio of fully integrated, turnkey systems integrating Dell infrastructure, software, and cloud operating stacks that deliver consistent multicloud operations by extending cloud operating models to on-premises and edge environments.

Dell APEX Cloud Platform provides an on-premises private cloud environment with consistent full-stack integration for the most widely deployed cloud ecosystem software including Microsoft Azure, Red Hat OpenShift, and VMware vSphere.

For more information, see the Dell APEX Cloud Platforms webpage.

## Dell APEX Cloud Platform for Red Hat OpenShift

Dell APEX Cloud Platform for Red Hat OpenShift is designed collaboratively with Red Hat to optimize and extend OpenShift deployments on-premises with a seamless operational experience.

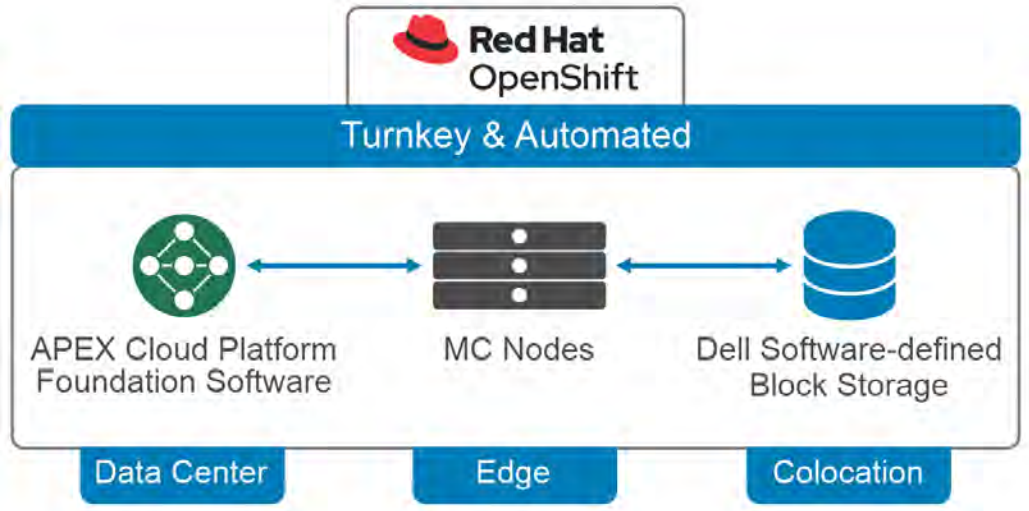


**Figure 1. Dell APEX Cloud Platform for Red Hat OpenShift high level physical architecture**

This turnkey platform provides:

- Deep integrations and intelligent automation between layers of Dell and OpenShift technology stacks, accelerating time-to-value and eliminating the complexity of management using different tools in disparate portals.
- Simplified, integrated management using the OpenShift Web Console.
- A bare metal architecture delivers the performance, predictability, and linear scalability needed to meet even the most stringent SLAs.

Additional benefits of Dell APEX Cloud Platform for Red Hat OpenShift include:

- Accelerating time to value with a purpose-built platform for OpenShift.
- Seamlessly extending applications and data across the IT landscape with a common SDS and advanced data services everywhere
- Enhanced control with multilayer security and compliance capabilities.

The Dell APEX Cloud Platform for Red Hat OpenShift uses separate storage nodes to provide persistent block storage for the compute cluster. This separation enables the compute and storage nodes to scale independently. The storage cluster is based on Dell PowerFlex software-defined storage architecture. This disaggregation allows customers with existing PowerFlex storage to deploy only the compute cluster (in a brownfield environment). Customers can opt for full integration in which the OpenShift Web Console is used to control both the compute and storage clusters.

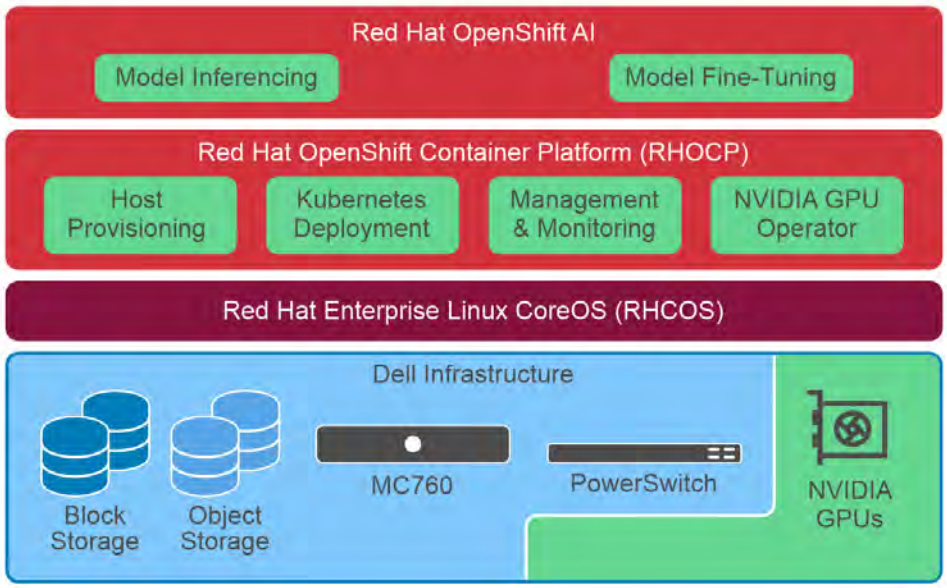


**Figure 2. High-level overview of OpenShift AI on Dell APEX Cloud Platform for Red Hat OpenShift logical architecture**

The Dell APEX Cloud Platform for Red Hat OpenShift introduces a new level of integration for running OpenShift on bare metal servers. Until now, all infrastructure management has been through separate OEM tooling and is managed separately from OpenShift. This dissociation requires IT staff with unique expertise to maintain the system, increasing operational costs.

The Dell APEX Cloud Platform Foundation Software mitigates this complexity by integrating the infrastructure management into the OpenShift Web Console. This integration enables administrators to update the hardware using the same workflow that updates the OpenShift software. It also enables OpenShift administrators to manage the infrastructure using the same management tools they use to control the cluster and the applications that run on it.

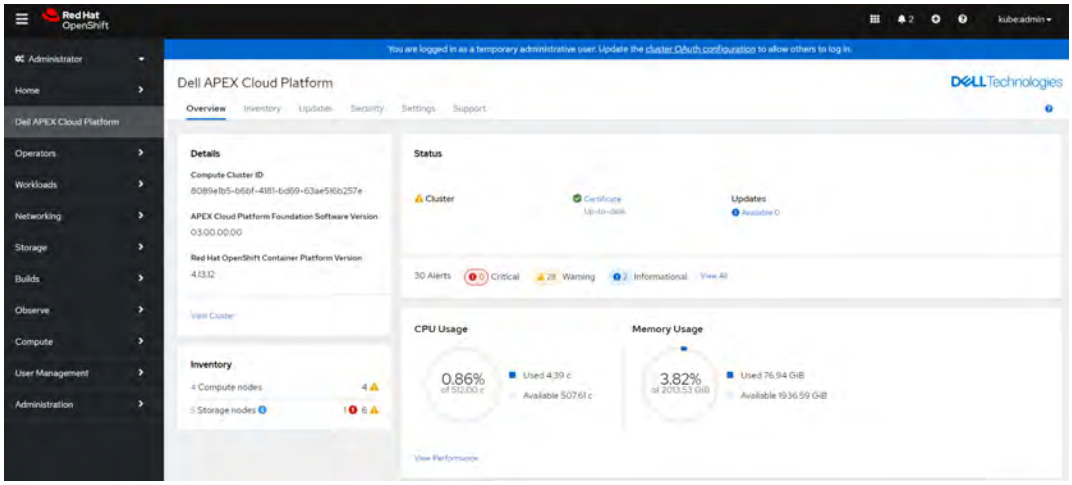


**Figure 3. Dell APEX Cloud Platform Foundation Software integration in OpenShift Web Console**

Ensuring that sensitive data, such as Personally Identifiable Information (PII), is securely stored in every stage of its life is an important task for any organization. To simplify this process, the National Institute of Science and Technology (NIST) developed standards, regulations, and best practices to protect data. These standards protect government data and ensure those working with the government comply with certain safety standards before they have access to data.

The Federal Information Protection Standard (FIPS) is one such standard that helps organizations simplify the process of protecting data.

Dell APEX Cloud Platform for Red Hat OpenShift is designed with FIPS-validated cryptographic based on RSA BSAFE Crypto Module. This provides support for various FIPS standards. For more information, see General FAQ for OpenShift and FIPS compliance.

**Note**: FIPS is enabled by default to provide a secure environment. This default setting is important to consider while developing your digital assistant applications, as some of the LangChain modules may not be FIPS-compliant.

## Red Hat OpenShift

Red Hat OpenShift Container Platform is a consistent hybrid cloud foundation for containerized applications, powered by Kubernetes. Developers and DevOps engineers using Red Hat OpenShift Container Platform can quickly build, modernize, deploy, run, and manage applications anywhere, securely, and at scale.

Red Hat OpenShift Container Platform is powered by open-source technologies and offers flexible deployment options ranging from physical, virtual, private cloud, public cloud, and Edge. Red Hat OpenShift Container Platform cluster consists of one or more control-plane nodes and a set of worker nodes.

For more information about Red Hat OpenShift Container Platform, see the Red Hat OpenShift web page.

### Red Hat OpenShift AI

AI/ML has quickly become crucial for businesses and organizations to remain competitive. However, deploying AI applications can be complicated due to a lack of integration among rapidly evolving tools. Popular cloud platforms offer attractive tools and scalability but often lock users in, limiting architectural and deployment options.

Red Hat OpenShift AI, formerly referenced as Red Hat OpenShift Data Science (RHODS), is a platform that unlocks the power of AI for developers, data engineers, and data scientists in Red Hat OpenShift. It is easily installed through a Kubernetes operator and provides a fully focused development environment called workbench that automatically manages the storage and integrates different tools. This provides users the ability to rapidly develop, train, test, and deploy machine learning models on-premises or in the public cloud environment.

Red Hat OpenShift AI allows data scientists and developers to focus on their data modeling and application development without waiting for infrastructure provisioning. The ML models developed using Red Hat OpenShift AI are portable to deploy in Production, on containers, on-premises, at the edge, or in the public cloud.

For more information about Red Hat OpenShift AI, see the Red Hat OpenShift AI web page.

### Llama 2

Llama 2 is an open-source pre-trained LLM that is freely available for research and commercial use. Llama 2 was trained on 40 percent more data than its predecessor, Llama 1, and has twice the context length (4096 compared to 2048 tokens). This means that Llama 2 can better understand context and generate more relevant and accurate information.

Llama 2 can be used in use cases to build digital assistant for consumers and enterprise usage, language translation, research, code generation, and various AI-powered tools.

For more information about Llama 2, see the Meta web page.

### LangChain

LangChain is an open-source framework for developing LLM-powered applications. It simplifies the process of building LLM powered applications by providing an abstracted standard interface that makes it easier to interact with different language models, including Llama 2.

LangChain's plug-and-play features allows users to use different data sources, LLMs, and UI tools without having to rewrite code and build powerful NLP applications with minimal effort.

LangChain provides a variety of tools and APIs to connect language models to other data sources, interact with their environment, and build complex applications. Developers are required to use language models such as Llama 2 to build applications using LangChain. LangChain can be used to build digital assistants to generate a question-answering system over domain specific information.

For more information about LangChain, see the LangChain web page.

### Redis

Redis is a vector store that offers an effective solution to efficiently query and retrieve relevant information from massive amounts of data. Vector stores are databases designed for storing and retrieving vector embeddings efficiently and to perform semantic searches. Vector stores can index and instantly search for similar vectors using similarity algorithms.

Redis is a popular in-memory data structure store. One feature of the Redis database is the ability to store embeddings with metadata to be used later by LLMs. Redis vector database is an excellent choice for applications that have to store and search vector data quickly and efficiently.

For more information about the Redis vector database, see the Redis webpage.

### Gradio

Gradio is an open-source Python library that enables incredibly fast development/ prototyping of the ML web applications with user interfaces. It provides a simple and intuitive API which is compatible with all Python programs and libraries. Gradio provides a variety of options to customize various elements of the user interface (UI).

Gradio is a fastest way to protype any ML model with a friendly web interface.

Gradio has a unique capability to visualize the intermediate steps or thought processes during a language model's decision-making process. This unique feature makes Gradio a useful tool for analyzing and debugging the decision processing abilities for language models.

For more information about Gradio, see the Gradio webpage.

### Caikit

Caikit provides an abstraction layer for developers where they can utilize AI models through APIs without the knowledge of the data form of the model. Caikit is an advanced AI toolkit that streamlines the process of working with AI models through developer-friendly APIs.

By implementing a model based on Caikit, an organization can:

- Perform training jobs to create models from dataset.
- Merge models from diverse AI communities into a common API
- Update applications to newer models for a specific task without client-side changes.

For more information about Caikit, see the Caikit webpage.

### Caikit-NLP

Caikit-NLP is a python library providing various Natural Language Processing (NLP) capabilities built on top of Caikit framework. Caikit-NLP is a powerful library that leverages prompt tuning and fine-tuning to add NLP domain capabilities to Caikit. This library also provides a ready to use python script to convert models from different format to Caikit format.

For more information, see the Caikit-NLP web page.

### Text Generation Inference

TGI is a toolkit designed for deploying and serving LLMs. TGI provides a high-performance text generation for the most popular open-source LLMs, such as Llama. It is powered by Python, Rust, and gRPC.

TGI implements many features to enhance the use of LLMs including:

- Serves the most popular LLMs with a simple launcher.
- Distributes the inference workload across multiple GPUs, which can significantly speed up generation for large models.
- Batches incoming requests together to improve throughput.
- Uses optimized versions of the transformers library to improve inference performance.

For more information about TGI, see the TGI GitHub webpage.

This validated solution uses many open-source technologies, frameworks, libraries, and methods.

# Chapter 3 Solution Architecture and Requirements

This chapter presents the following topics:

# Physical architecture

## Overview

The following figure demonstrates the physical architecture design of the solution used to run the LLM-based application on Dell APEX Cloud Platform for Red Hat OpenShift.

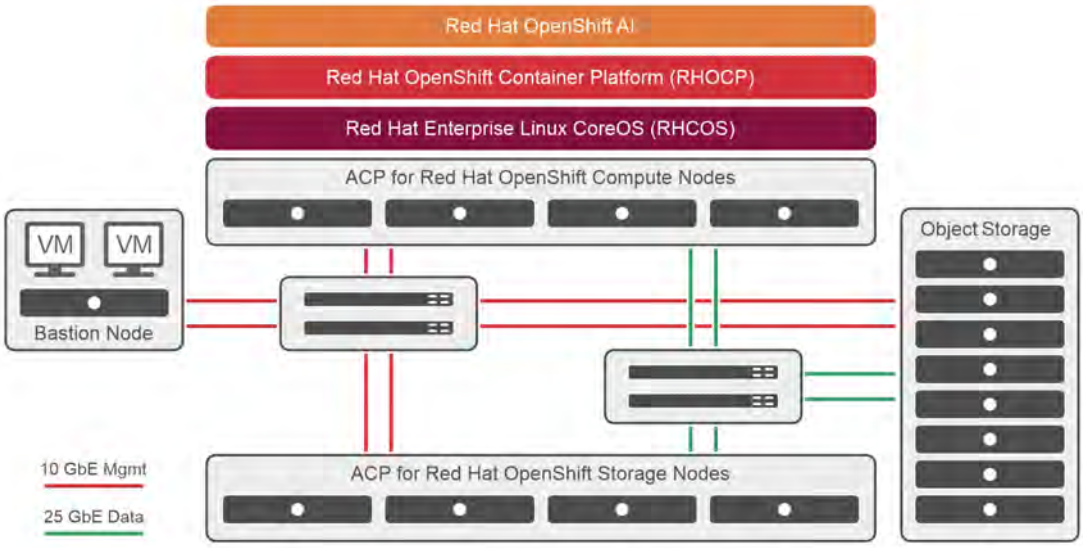


**Figure 4. Dell APEX Cloud Platform for Red Hat OpenShift physical architecture**

The following sections describe the various hardware stacks involved in designing this solution.

## Dell servers

Dell APEX Cloud Platform for Red Hat OpenShift compute layer setup is configured with four Dell APEX MC-760 nodes for running AI workloads. Dell APEX MC-760 server is a 2U, two-socket fully featured enterprise rack server, designed to optimize even the most demanding workloads like Artificial Intelligence and Machine Learning. It provides a balanced CPU and storage for demanding workloads, powered by Intel Xeon processors and Nvidia GPUs.

Dell APEX MC-760 Servers offer:

- Up to two Next Generation Intel Xeon Scalable processors with up to 56 cores for faster and more accurate processing performance.
- Accelerate in-memory workloads with up to 32 DDR5 RDIMMS up to 4400 MT/sec (2DPC) or 4800 MT/sec for 1DPC (16 DDR5 RDIMMs max).
- Support for GPUs including 2 x double-wide or 6 x single-wide for workloads requiring acceleration.

For more information about Dell APEX MC-760 servers, see the Dell APEX Cloud Platform MC-760 Hardware Requirements and Specifications page. This solution includes Dell APEX MC-760 servers with Intel Xeon Platinum 8462Y+ processors which offer 32 cores per socket, operating at a speed of 2.80 GHz. Additionally, servers are equipped with NVIDIA A2 GPU which offer low power consumption, a small footprint, and high

performance. These specifications provide the necessary horsepower to handle the AI workloads.

### Storage

Dell APEX Cloud Platform for Red Hat OpenShift storage nodes are deployed as a separate cluster and used to provide persistent block storage for OpenShift containers. This separation enables the compute and storage nodes to scale independently. The storage cluster is based on Dell PowerFlex software-defined storage architecture.

Object storage systems, designed for scalable and cost-effective data management, provide an ideal solution for housing massive LLMs and large datasets. Object storage is used in this solution to store Llama 2 model, other datasets, and Red Hat OpenShift AI pipeline artifacts.

### Networking

Two 25 GbE Dell PowerSwitch network switches are being used for data plane connectivity, and two 10 GbE Dell PowerSwitch network switches are being used for management plane connectivity. Dell PowerSwitch enhances performance in data center fabrics of all sizes with efficient and flexible switching solutions.

### Bastion node

This node is used to run VMs, and Containers related to client application and tools required to manage the Red Hat OpenShift cluster, Object Storage, and LLM-based application. Also, infra VMs such as AD and DNS are running in this node.

## Logical architecture

The following figure demonstrates the logical architecture design of the digital assistant deployed on Dell APEX Cloud Platform for Red Hat OpenShift.

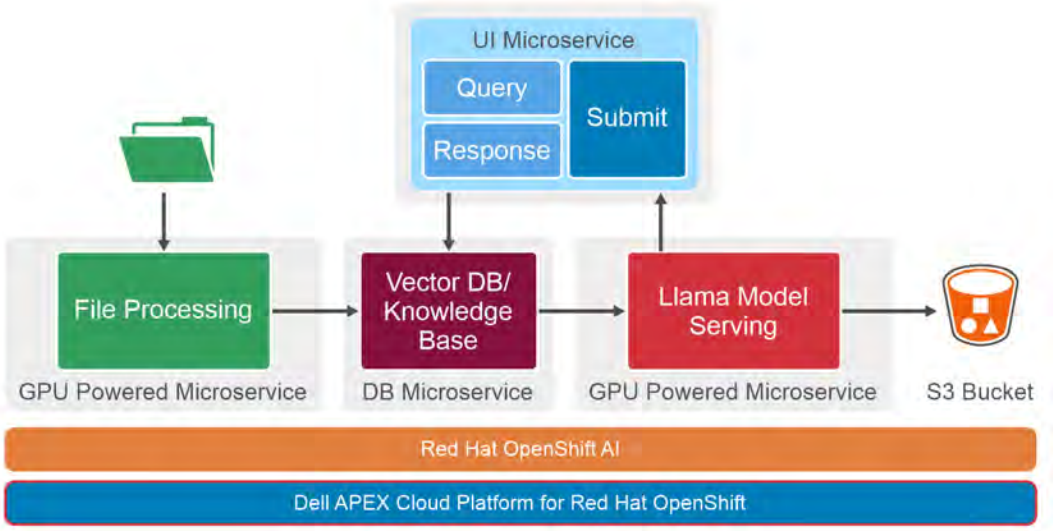


**Figure 5. Digital assistant logical architecture**

In our validated design, we use Llama 2 model for language processing, LangChain to integrate different tools of the LLM-based application together and to process the PDF files and web pages, Redis to store vectors, Caikit to serve the Llama 2 model, Gradio for user interface and object storage to store language model and other datasets. Solution components are deployed as microservices in the Red Hat OpenShift cluster.

The following list details the roles and responsibilities of each microservice.

- **Domain-specific document processing microservice:** This microservice provides a way to convert PDF files or web pages to vector embeddings using the LangChain library. Vector embeddings are a way of representing text as a vector of numbers, which can be used for semantic search to answer user queries.
- **Knowledge base microservice:** This microservice provides a scalable and efficient vector store for the digital assistant using Redis, an in-memory data structure store. It enables the digital assistant to perform real-time vector search and retrieval.
- **Llama Model serving microservice:** This microservice contains a Llama 2 model served using Caikit, an inference server for large language models. The model is also integrated with TGI. TGI enables high-performance text generation for the most popular LLMs.

- **UI Microservice:** This microservice provides a user-friendly interface for the digital assistant using Gradio, an open-source library for creating UIs for machine learning models. It enables users to interact with the digital assistant in a natural and intuitive way, making it more accessible and engaging. UI microservice also contains a LangChain library to chain together different components such as vector store and Llama model.

# Digital assistant design

RAG is an AI framework that allows LLMs to access additional domain specific data and generate better and more accurate answers, without having to be retrained. The following figure describes the RAG based digital assistant architecture design.

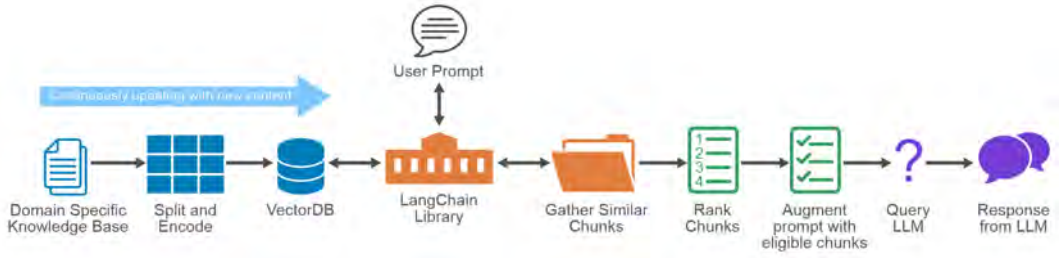


**Figure 6. Digital assistant design**

## Digital assistant workflow

The following list describes the steps involved in operating the digital assistant.

- **Ingest data:** PDF files or HTML web pages which contain domain specific information have to be identified and processed to help the Llama 2 model to answer user queries more accurately.
- **Extract and split data:** LangChain library is used to extract the data and split it into smaller and manageable chunks, as language models can only handle limited amount of text at a time, based on the context length on which it was trained.

Followed by converting text chunks to numerical values, known as embeddings. Embeddings are used to perform semantic searches.

- **Load embeddings into the vector store:** LangChain can store the embeddings to the vector store and retrieve it using different retrievers such as “similarity distance threshold.” Redis is used in this solution as the vector store to store vectors and to perform semantic search.
- **User Interface:** Gradio provides a simple and intuitive user interface to interact with the digital assistant. LangChain integrates the Gradio user interface with other components such as LLM and vector store to work together as a digital assistant.
- **Query processing:** When users submit a query, the query will be split into smaller chunks and then embeddings will be created for the same. Semantic search is performed against the vectors stored in Redis. Results from Redis are ranked and sent to the Llama 2 model, the Llama model answers the queries based on results from Redis and its pretrained capabilities.

# Solution requirements

## Server details

| Components | Details |
|---|---|
| Nodes | 4 x Dell APEX MC-760 |
| Server Model | Dell APEX MC-760 |
| CPU | 2 X Intel(R) Xeon(R) Platinum 8462Y+ |
| GPU | NVIDIA A2 |
| Memory | 512 GB (16 X 32 GB, DDR-5,4800 MT/s) |
| Storage Controller | BOSS-N1 Monolithic |
| | Dell HBA355i |
| Drives | 2 X 894 GB NVMe SSD |
| | 24 X 745 GB SAS SSD |
| NICs | Broadcom Adv. Dual 25 Gb Ethernet, 2 port |
| | Broadcom Adv. Dual 25 Gb Ethernet, 2 port |
| PSU | Dual, Redundant (1+1), Hot-Plug PSU,1800 W |

## Server firmware details

| Firmware/Software | Version |
|---|---|
| Server BIOS version | 1.5.6 |
| iDRAC | 6.10.85.00 |
| GPU | 94.07.5B.00.92 |
| System CPLD | 1.0.5 |
| Dell OS Driver Pack | 23.03.07 |
| Identity Module | 0.06 |

| Firmware/Software | Version |
|---|---|
| Power Supply | 00.42.B2 |
| TPM | 7.2.2.0 |
| HBA355i | 24.15.10.00 |
| BOSS-N1 | 2.1.13.2021 |
| Broadcom Adv. Dual 25 Gb Ethernet | 22.31.13.70 |
| Broadcom NetXtreme Gigabit Ethernet | 22.31.6 |

## Software details

| Software | Version |
|---|---|
| Server OS | RHCOS 4.13 |
| Red Hat OpenShift Container Platform | 4.13.12 |
| Kubernetes | 1.26.7 |
| APEX Cloud Platform Foundation Software | 03.00.00.00 |
| Red Hat OpenShift Data Science | 2.2.0 |
| Red Hat OpenShift serverless | 1.30.1 |
| Red Hat OpenShift service mesh | 2.4.4-0 |
| Red Hat OpenShift distributed tracing platform | 1.47.0-2 |
| Nvidia GPU operator | 23.6.1 |
| Node Feature Discovery operator | 4.13.0 |
| Redis enterprise operator | 7.2.4 |
| Kiali operator | 1.65.9 |
| Python | 3.11.4 |
| LangChain | 0.0.301 |
| Llama 2 | Llama-2-7b-Chat-hf |
| Gradio | 3.46.1 |
| Caikit | 0.22.0 |
| Caikit-nlp | 0.2.1 |
| TGI | 1.1.0 |
| Redis | 2.8.4 |

# Chapter 4 Solution Deployment

This chapter presents the following topics:

# Introduction

Deploying a LLM-based application is a complex multi-step process that requires the right combination of hardware and software for a robust AI/ML platform. Dell APEX Cloud Platform for Red Hat OpenShift reduces this complexity by providing an AI/ML solution for data scientists and data engineers to seamlessly deploy a LLM model.

We deployed a LLM based digital assistant using Red Hat OpenShift AI on Dell APEX Cloud Platform for Red Hat OpenShift. The goal of this solution is to enable users to ask queries related to domain specific data and get accurate answers to the questions within the scope of the digital assistant. This chapter describes the necessary steps to deploy and configure the overall solution.

# Dell APEX Cloud Platform for Red Hat OpenShift

## Overview

Dell APEX Cloud Platform for Red Hat OpenShift is designed collaboratively with Red Hat to optimize and extend OpenShift deployments on-premises with an integrated operational experience. Dell APEX Cloud Platform for Red Hat OpenShift is a preconfigured setup based on Dell APEX MC-760 servers, Red Hat Enterprise Linux CoreOS (RHCOS) and Red Hat OpenShift Container Platform.

## Preparation

The following list describes the high-level tasks required to set up Dell APEX Cloud Platform for Red Hat OpenShift:

- Dell APEX Cloud Platform for Red Hat OpenShift comes pre deployed from Dell. Be sure to rack and stack the servers as per guidelines provided with server shipment.
- Log in and verify accessibility to Red Hat OpenShift Container Platform dashboard.
- Set up user accounts that are required to perform the solution deployment.

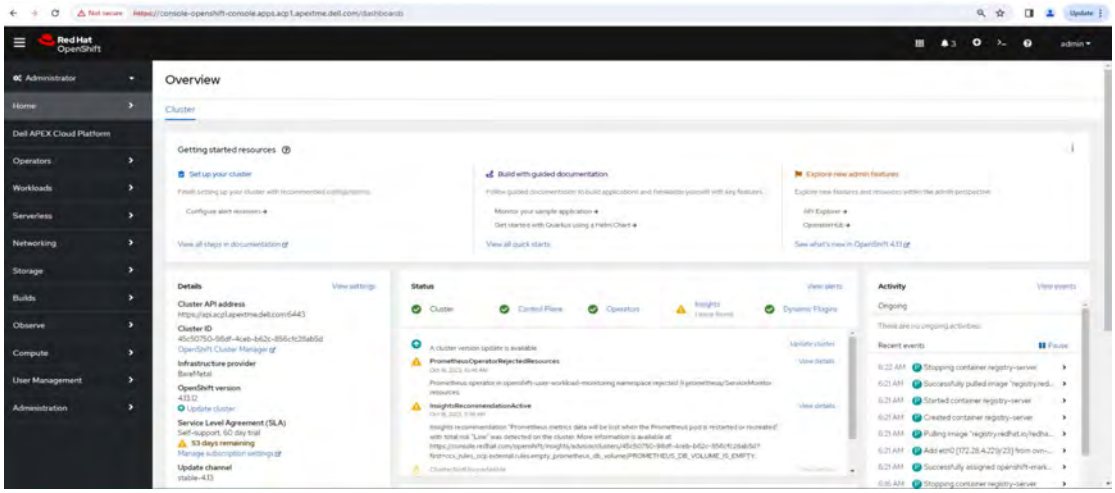


**Figure 7. Red Hat OpenShift dashboard**

# Object storage

**Overview**

S3-compatible object storage is used in this solution to store and access Llama 2 model and other datasets. Any S3 compatible storage such as Dell ObjectScale, AWS S3, Azure Blob, Dell ECS, Dell PowerScale and others can be leveraged as object storage.

**Preparation**

Create object storage buckets and configure the necessary permissions in the S3 bucket to store Llama model and datasets such as PDF files and other relevant artifacts.

# Digital assistant deployment and configuration

**Overview**

Deploying a digital assistant involves multiple steps, which includes preparing the environment and validating, deploying, and configuring the components. In this section we will briefly discuss the digital assistant deployment process.

**Prerequisites**

The following sections describe perquisites to the solution deployment.

### Configuring container registry

OpenShift Container Platform provides an internal, integrated container image registry that can be deployed to locally manage container images in the Red Hat OpenShift Container Platform cluster environment.

Follow the steps described in the GitHub repository to configure OpenShift Internal registry.

### Node Feature Discovery Operator deployment

The Node Feature Discovery Operator (NFD) manages the detection of hardware configuration and features in an OpenShift Container Platform cluster by labeling the nodes with node-specific attributes, such as PCI cards, kernel, operating system version, and so on.

Follow the steps described in GitHub repository to Install NFD Operator.

### Installing Nvidia GPU Operator

The NVIDIA GPU Operator uses the operator framework available within Kubernetes to automate the management of all NVIDIA software components needed to provision GPU. The Node Feature Discovery (NFD) Operator is a prerequisite for the NVIDIA GPU Operator.

Follow the steps described in the GitHub repository to Install Nvidia GPU Operator.

### Red Hat OpenShift AI

Red Hat OpenShift AI is an open-source machine learning platform. Red Hat OpenShift AI provides pre-configured tools and libraries that allow data scientists and developers to rapidly train, deploy, and monitor ML workloads and models on-premises and in the public cloud.

Installing Red Hat OpenShift AI involves the following high-level tasks:

- Confirm that your OpenShift cluster meets all requirements.
- Configure an identity provider and add administrative users.
- Subscribe to the OpenShift AI (RHODS) Add-on and install it.
- Access and verify the OpenShift Data Science dashboard.

For more information regarding installing Red Hat OpenShift AI, see the Red Hat web page.

## Model serving environment

This section describes the model serving environment preparation. This preparation includes step-by-step command installation of KServe and dependencies, along with Red Hat OpenShift AI (GitHub).

---

**Note**: You have the alternative option of installing the OpenShift AI KServe/Caikit/TGIS stack by using Automated Script installation.

---

### Prerequisites

The following list includes installation prerequisites:

- Your cluster needs a node with minimum 4 CPUs and 16 GB memory to support inferencing.
- You have cluster administrator permissions.
- You have installed the OpenShift CLI (oc).

### Procedure

The following procedure describes the steps to deploy RHODS Operator with the dependent services and operators such as Service Mesh, Istio, Knative Serving, Knative gateway, and Kserve.

1. Specify the data science operator by setting the TARGET_OPERATOR environment variable to RHODS.

For the RHODS Operator:

```
export TARGET_OPERATOR=rhods
```

2. Clone this repository and set up the environment.

```
git clone https://github.com/DellBizApps/dell-digital-assistent.git


cd dell-digital-assistent/01-rhods


source ./scripts/env.sh

export TARGET_OPERATOR_TYPE=$(getOpType $TARGET_OPERATOR)
```

```
export TARGET_OPERATOR_NS=$(getOpNS)

export KSERVE_OPERATOR_NS=$(getKserveNS)
```

3. Install the Service Mesh operators.

```
oc apply -f custom-manifests/service-mesh/operators.yaml

sleep 30

oc wait --for=condition=ready pod -l name=istio-operator -n
openshift-operators --timeout=300s

oc wait --for=condition=ready pod -l name=jaeger-operator -n
openshift-operators --timeout=300s

oc wait --for=condition=ready pod -l name=kiali-operator -n
openshift-operators --timeout=300s
```

4. Create an Istio instance.

```
oc create ns istio-system

oc apply -f custom-manifests/service-mesh/smcp.yaml

sleep 30

oc wait --for=condition=ready pod -l app=istiod -n istio-system --
timeout=300s

oc wait --for=condition=ready pod -l app=istio-ingressgateway -n
istio-system --timeout=300s

oc wait --for=condition=ready pod -l app=istio-egressgateway -n
istio-system --timeout=300s

oc wait --for=condition=ready pod -l app=jaeger -n istio-system --
timeout=300s
```

5. Install Knative Serving.

```
oc create ns ${KSERVE_OPERATOR_NS}

oc create ns knative-serving

oc -n istio-system apply -f custom-manifests/service-mesh/smmr-
${TARGET_OPERATOR_TYPE}.yaml

oc apply -f custom-manifests/service-mesh/peer-authentication.yaml

oc apply -f custom-manifests/service-mesh/peer-authentication-
${TARGET_OPERATOR_TYPE}.yaml
```

---

**Note**: These commands use PeerAuthentications to enable mutual TLS (mTLS) according to OpenShift Serverless Documentation.

---

```
oc apply -f custom-manifests/serverless/operators.yaml

sleep 30
```

```
oc wait --for=condition=ready pod -l name=knative-openshift -n
openshift-serverless --timeout=300s

oc wait --for=condition=ready pod -l name=knative-openshift-
ingress -n openshift-serverless --timeout=300s

oc wait --for=condition=ready pod -l name=knative-operator -n
openshift-serverless --timeout=300s
```

6. Create a KnativeServing Instance.

```
oc apply -f custom-manifests/serverless/knativeserving-istio.yaml

sleep 15

oc wait --for=condition=ready pod -l app=controller -n knative-
serving --timeout=300s

oc wait --for=condition=ready pod -l app=net-istio-controller -n
knative-serving --timeout=300s

oc wait --for=condition=ready pod -l app=net-istio-webhook -n
knative-serving --timeout=300s

oc wait --for=condition=ready pod -l app=autoscaler-hpa -n
knative-serving --timeout=300s

oc wait --for=condition=ready pod -l app=domain-mapping -n
knative-serving --timeout=300s

oc wait --for=condition=ready pod -l app=webhook -n knative-
serving --timeout=300s

oc delete pod -n knative-serving -l app=activator --force --grace-
period=0

oc delete pod -n knative-serving -l app=autoscaler --force --
grace-period=0

oc wait --for=condition=ready pod -l app=activator -n knative-
serving --timeout=300s

oc wait --for=condition=ready pod -l app=autoscaler -n knative-
serving --timeout=300s
```

7. Generate a wildcard certification for a gateway using OpenSSL.

```
export BASE_DIR=/tmp/kserve

export BASE_CERT_DIR=${BASE_DIR}/certs

export DOMAIN_NAME=$(oc get ingresses.config.openshift.io cluster
-o jsonpath='{.spec.domain}' | awk -F'.' '{print $(NF-1)"."$NF}')

export COMMON_NAME=$(oc get ingresses.config.openshift.io cluster
-o jsonpath='{.spec.domain}')



mkdir ${BASE_DIR}
```

```
mkdir ${BASE_CERT_DIR}


./scripts/generate-wildcard-certs.sh ${BASE_CERT_DIR}
${DOMAIN_NAME} ${COMMON_NAME}
```

8. Create the Knative gateway.

```
oc create secret tls wildcard-certs --
cert=${BASE_CERT_DIR}/wildcard.crt --
key=${BASE_CERT_DIR}/wildcard.key -n istio-system

oc apply -f custom-manifests/serverless/gateways.yaml
```

9. Apply the Istio monitoring resources.

```
oc apply -f ./custom-manifests/service-mesh/istiod-monitor.yaml

oc apply -f ./custom-manifests/service-mesh/istio-proxies-
monitor.yaml
```

10. Apply the cluster role to allow Prometheus access.

```
oc apply -f ./custom-manifests/metrics/kserve-prometheus-k8s.yaml
```

11. Deploy KServe with RHODS Operator 2.x.

```
oc create ns ${TARGET_OPERATOR_NS}

oc create -f custom-manifests/opendatahub/${TARGET_OPERATOR}-
operators-2.x.yaml


sleep 10

oc wait --for=condition=ready pod -l name=rhods-operator -n
${TARGET_OPERATOR_NS} --timeout=300soc create -f custom-
manifests/opendatahub/kserve-dsc.yaml
```

Once RHODS deployment completes along with dependent components, the RHODS dashboard option will become enabled in OpenShift console.

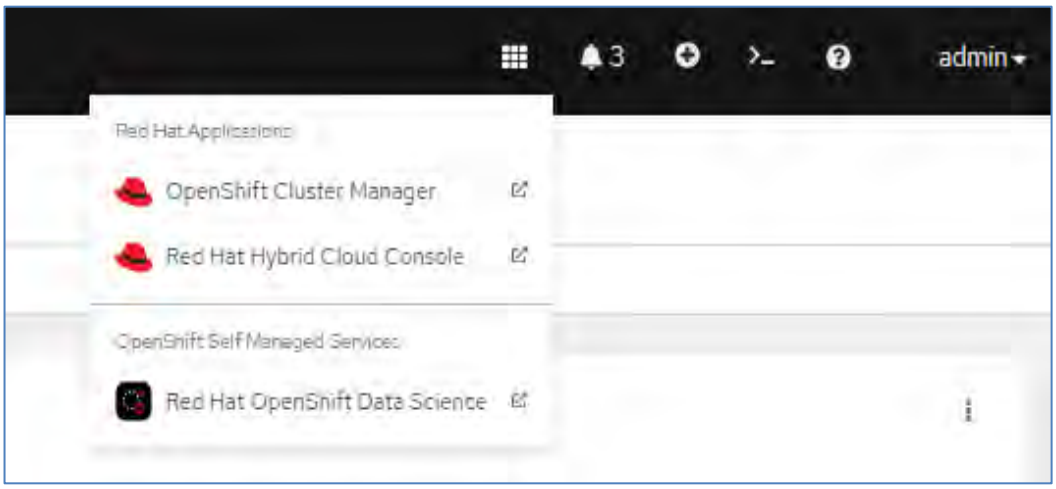


**Figure 8. RHODS navigation link in OpenShift console**

We can navigate to the RHODS Dashboard by clicking the following link. This grants access to the RHODS dashboard shown in Figure 9.

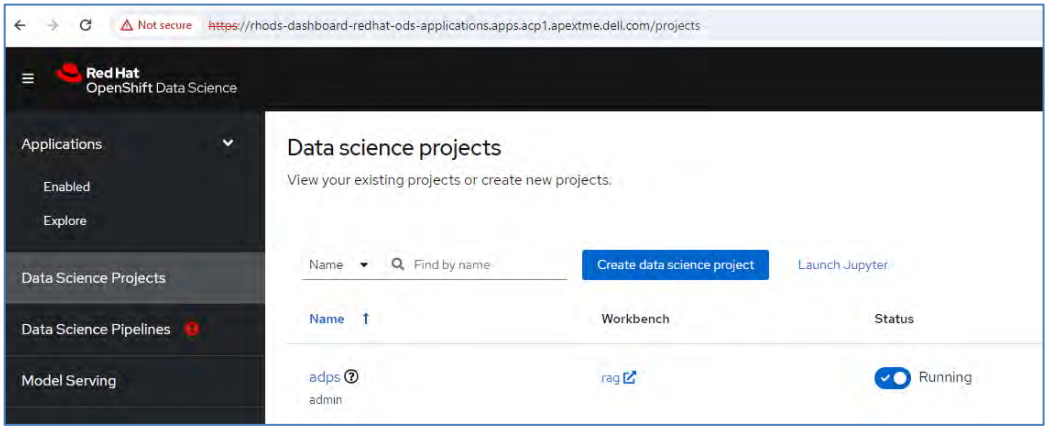


**Figure 9. RHODS dashboard in OpenShift console**

## Data science project in Red Hat OpenShift AI

A data science project in Red Hat OpenShift AI is a logical grouping that uses the Red Hat OpenShift AI platform to develop, train, and deploy machine learning models.

### Create and configure data science project

To create and configure a data science project, go to the **Create data science project** page on the Red Hat OpenShift Data Science dashboard (Red Hat OpenShift Data Science dashboard > Data Science Projects > Create data science project > enter the required fields [such as project name and description] > **Create**).

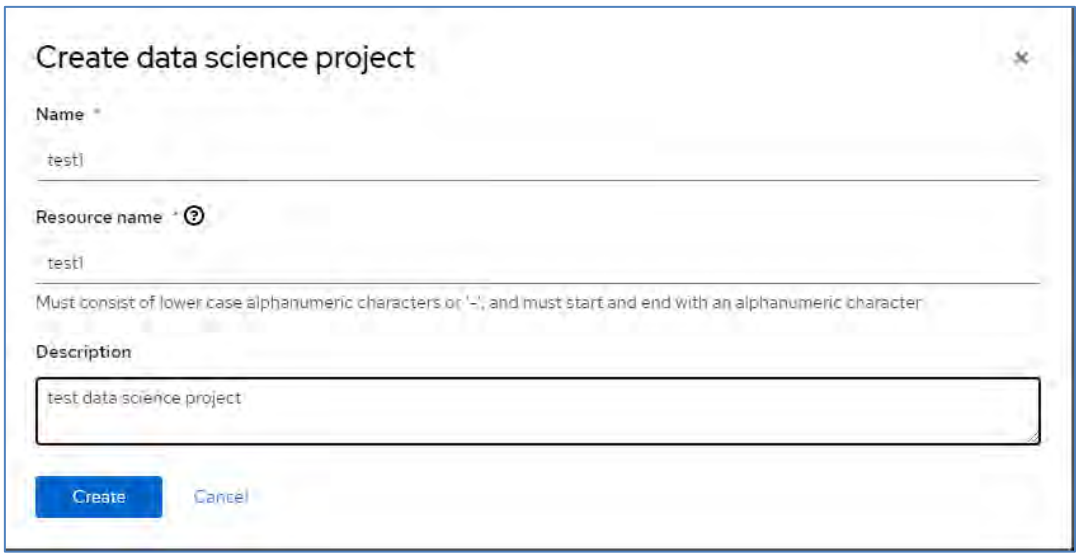


**Figure 10. Data science project creation**

For more information, see the Red Hat webpage.

Within the data science project, you can add the following configuration options:

- **Workbenches**: Development environments within your project where you can access notebooks and generate models.
- **Cluster storage**: Storage for your project in your OpenShift cluster.
- **Data connections**: A list of data sources that your project uses.
- **Models and model servers**: A list of models and model servers that your project uses.

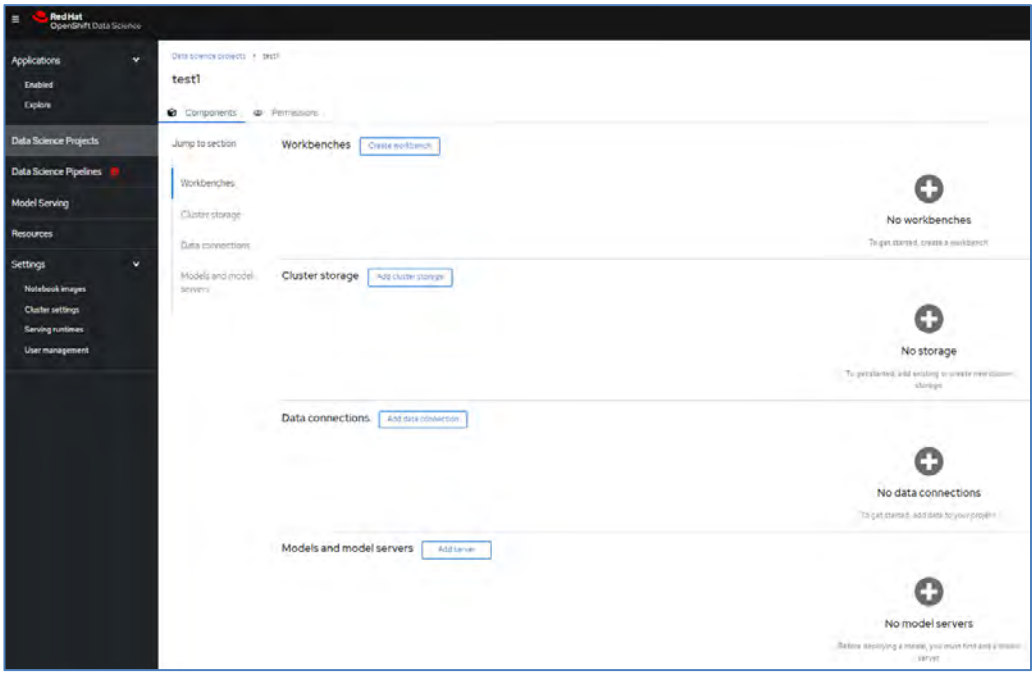


**Figure 11. Different components inside data science project**

## Project Workbench

Workbench provides you with an isolated area to examine and work with data models. Workbench allows you to create a new Jupyter Notebook from an existing notebook container image to access its resources and properties. For data science projects that have to retain data, you can add container storage to the workbench you are creating.

### Create and configure Workbench

To create Workbench, navigate to the Red Hat OpenShift Data Science dashboard (Red Hat OpenShift Data Science dashboard > Data Science Projects > Select the name of the project that you want to add the workbench > Create workbench > Configure the properties of the workbench [such as notebook image to use, container size, and storage] > **Create workbench**).

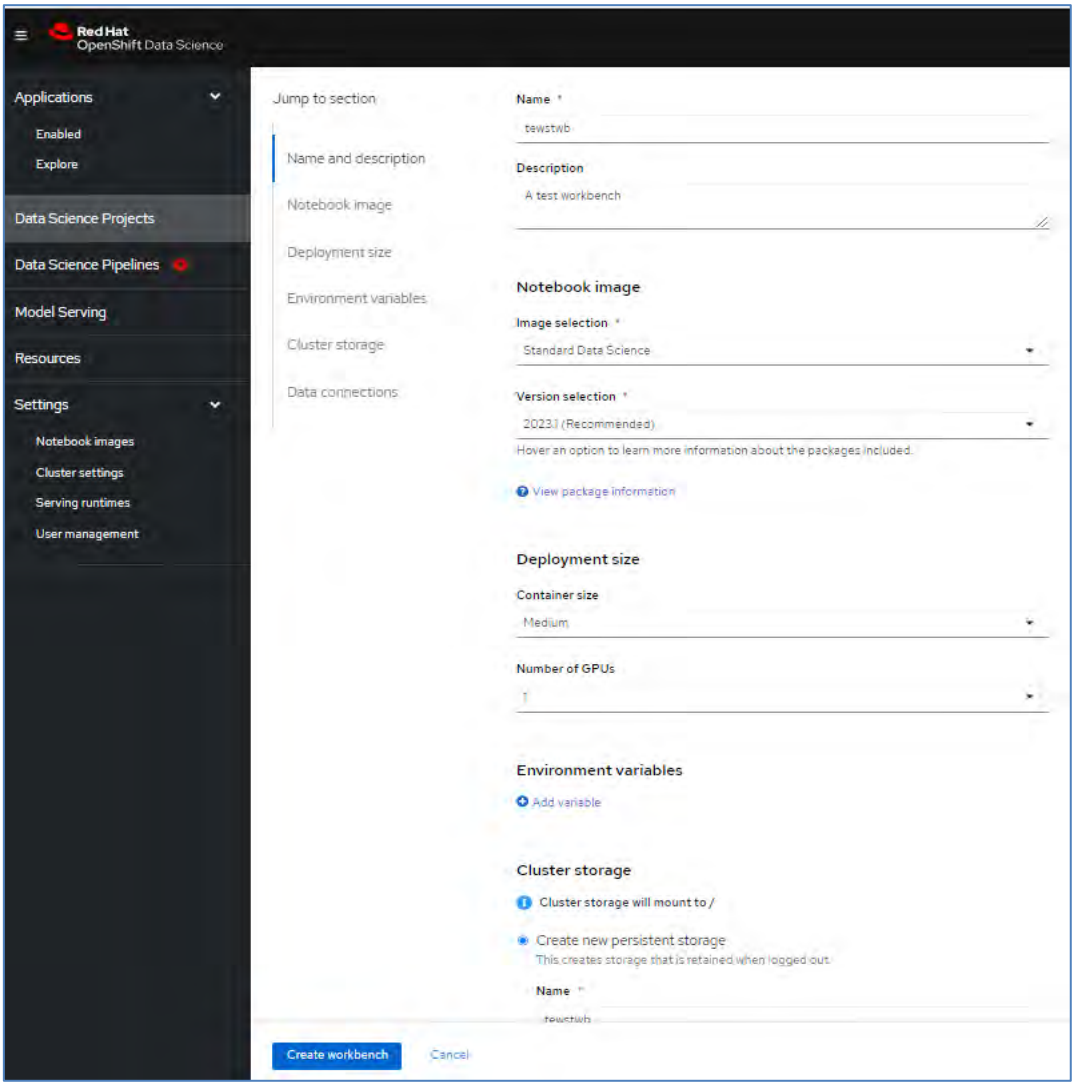


**Figure 12. Creating Workbench in Data Science project**

For more information, see the Red Hat webpage.

## LLM model preparation

This section covers the LLM model preparation required before deploying it in the Caikit model serving environment. For the model to be served using Caikit, it must be converted to Caikit format before deployment.

**Note**: Obtain the required access for Llama 2 from the Meta webpage before proceeding with the following steps.

### Convert Llama 2 in Caikit format

The following steps describe the process to convert Llama 2 in Caikit format. For more information, see GitHub.

1. Download and install git-lfs.

```
!wget https://github.com/git-lfs/git-lfs/releases/download/v3.4.0/git-lfs-linux-amd64-v3.4.0.tar.gz
!tar -xvzf git-lfs-linux-amd64-v3.4.0.tar.gz
!PREFIX=/opt/app-root/src/.local ./git-lfs-3.4.0/install.sh
```

2. Clone Caikit nlp repository.

```
!git clone https://github.com/caikit/caikit-nlp.git
```

3. Install caikit-nlp.

```
!pip install ./caikit-nlp
```

4. Clone caikit tgis serving repository.

```
!git clone https://github.com/opendatahub-io/caikit-tgis-serving.git
```

5. Copy model conversion python script "convert.py" to the current location.

```
!cp caikit-tgis-serving/utils/convert.py .
```

6. Install and upgrade huggingface_hub module and log in to your huggingface account.

**Note**: Make sure you have your HUGGINGFACE_USER and HUGGINGFACE_TOKEN added as environment variable of this notebook

```
!pip install --upgrade huggingface_hub
!huggingface-cli login --token $HUGGINGFACE_TOKEN
```

7. Download Llama-2-7b-hf model from Hugging Face.

```
!git clone https://${HUGGINGFACE_USER}:${HUGGINGFACE_TOKEN}@huggingface.co/meta-llama/Llama-2-7b-hf
```

8. Convert Llama 2 model to Caikit format. Your converted model will be saved to Llama-2-7b-hf-caikit directory.

```
!./convert.py --model-path ./Llama-2-7b-hf/ --model-save-
path ./Llama-2-7b-hf-caikit
```

### Upload Model to Object Storage

Use the following steps to upload the model to Object Storage (GitHub).

1. Install boto module and libraries if it does not exist in your current session.

```
#! pip install --upgrade pip
#! pip install boto3 botocore
```

2. Create helper functions for easier uploading and listing model.

**Note**: Make sure you have the following environment variables set for connection to your object storage.

- AWS_S3_ENDPOINT
- AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY
- AWS_DEFAULT_REGION
- AWS_S3_BUCKET

```
import os
import boto3
import botocore

aws_access_key_id = os.environ.get('AWS_ACCESS_KEY_ID')
aws_secret_access_key = os.environ.get('AWS_SECRET_ACCESS_KEY')
endpoint_url = os.environ.get('AWS_S3_ENDPOINT')
region_name = os.environ.get('AWS_DEFAULT_REGION')
bucket_name = os.environ.get('AWS_S3_BUCKET')

session =
boto3.session.Session(aws_access_key_id=aws_access_key_id,

aws_secret_access_key=aws_secret_access_key)

s3_resource = session.resource(
    's3',
    config=botocore.client.Config(signature_version='s3v4'),
    endpoint_url=endpoint_url,
    region_name=region_name)

bucket = s3_resource.Bucket(bucket_name)


def upload_directory_to_s3(local_directory, s3_prefix):
    for root, dirs, files in os.walk(local_directory):
        for filename in files:
            file_path = os.path.join(root, filename)
            relative_path = os.path.relpath(file_path,
local_directory)
            s3_key = os.path.join(s3_prefix, relative_path)
```

```
            print(f"{file_path} -> {s3_key}")
            bucket.upload_file(file_path, s3_key)


def list_objects(prefix):
    filter = bucket.objects.filter(Prefix=prefix)
    for obj in filter.all():
        print(obj.key)
```

3. Upload your model directory to object storage.

**Note**: Do not upload your model to root of the bucket

```
upload_directory_to_s3("Llama-2-7b-chat-caikit", f"example-
models/llm/models/Llama-2-7b-chat")
```

4. List uploaded models.

```
list_objects(f"example-models/llm/models/Llama-2-7b-chat")
```

## Model deployment

This section describes deploying the LLM model to the Caikit model serving environment.

### Deploying a LLM model with the Caikit + TGIS Serving runtime (GitHub)

There are two options for deploying an LLM model:

1. Following the step-by-step commands that are described in this procedure.
2. Running short scripts as described in Using scripts to deploy an LLM model with the Caikit + TGIS Serving runtime.

This procedure includes deploying an example LLM model Llama-2-7b-chat-hf with the Caikit + TGIS Serving runtime.

**Note**: The Llama-2-7b-chat-hf LLM model has been uploaded to the object storage bucket.

#### *Prerequisites*

- You have installed the Caikit-TGIS-Serving stack as described in the Caikit-TGIS-Serving README file.
- Your current working directory is the /01-rhods/ directory.
- If your LLM model is in an S3-like object storage (for example, Dell ObjectScale, AWS S3, MinIO), change the connection data in storage-config-secret.yaml and serviceaccount.yaml
- Change the storageUri to the location of the model stored in the object storage bucket, within the Caikit-isvc.yaml.

#### *Procedure*

1. Deploy the LLM model with Caikit+TGIS Serving runtime.
    a. Create a new namespace and patch ServiceMesh related object.

```
export TEST_NS=kserve-demo
oc new-project ${TEST_NS}
```

```
oc patch smmr/default -n istio-system --type='json' -p="[{'op':
'add', 'path': '/spec/members/-', 'value': \"$TEST_NS\"}]"
```

b. Create a caikit ServingRuntime. By default, it requests 4CPU and 8Gi of memory. You can adjust these values as required.

```
oc apply -f ./custom-manifests/caikit/caikit-servingruntime.yaml -
n ${TEST_NS}
```

c. Deploy the MinIO data connection and service account.

```
oc apply -f ./custom-manifests/caikit/storage-config-secret.yaml -
n ${TEST_NS}
```

```
oc create -f ./custom-manifests/Caikit/serviceaccount.yaml -n
${TEST_NS}
```

d. Deploy the inference service. It will point to the model located in the modelmesh-example-models/llm/models directory.

```
oc apply -f ./custom-manifests/caikit/caikit-isvc.yaml -n
${TEST_NS}
```

e. Verify that the inference service's READY state is True.

```
oc get isvc/caikit-example-isvc -n ${TEST_NS}
```

2. Perform inference with Remote Procedure Call (gPRC) commands.

a. Determine whether the HTTP2 protocol is enabled in the cluster.

```
oc get ingresses.config/cluster -ojson | grep
ingress.operator.openshift.io/default-enable-http2
```

If the annotation is set to true, skip to Step 2c.

b. If the annotation is set to either false or not present, enable it.

```
oc annotate ingresses.config/cluster
ingress.operator.openshift.io/default-enable-http2=true
```

c. Run the following grpcurl command for all tokens in a single call:

```
export KSVC_HOSTNAME=$(oc get ksvc caikit-example-isvc-predictor -
n ${TEST_NS} -o jsonpath='{.status.url}' | cut -d'/' -f3)
```

```
grpcurl -insecure -d '{"text": "At what temperature does liquid
Nitrogen boil?"}' -H "mm-model-id: flan-t5-small-caikit"
${KSVC_HOSTNAME}:443
caikit.runtime.Nlp.NlpService/TextGenerationTaskPredict
```

The response should be similar to:

```
{
  "generated_token_count": "5",
  "text": "74 degrees F",
  "stop_reason": "EOS_TOKEN",
  "producer_id": {
```

```
    "name": "Text Generation",
    "version": "0.1.0"
  }
}
```

d. Run the grpcurl command to generate a token stream.

```
grpcurl -insecure -d '{"text": "At what temperature does liquid
Nitrogen boil?"}' -H "mm-model-id: flan-t5-small-caikit"
${KSVC_HOSTNAME}:443
caikit.runtime.Nlp.NlpService/ServerStreamingTextGenerationTaskPre
dict
```

The response should be similar to:

```
{
  "details": {
  }
}
{
  "tokens": [
    {
      "text": "_",
      "logprob": -1.599083423614502
    }
  ],
  "details": {
    "generated_tokens": 1
  }
}
{
  "generated_text": "74",
  "tokens": [
  {
      "text": "74",
      "logprob": -3.3622500896453857
    }
```

```
  ],
  "details": {
    "generated_tokens": 2
  }
}
....
```

## Redis deployment

This section describes the Redis vector store deployment. Redis is an open source, in-memory data store that can be used as a Vector Store for your embeddings in a RAG scenario.

### Redis Deployment and Configuration (GitHub)

1. From the OperatorHub, install the Redis Enterprise Operator. You can install the operator with the default value in the namespace that you want to create your Redis cluster (the operator is namespace based).
2. Create new Project in OpenShift cluster:

```
oc new-project redisdb
```

3. Create Security Context Constraints using the following yaml manifests:

```
apiVersion: security.openshift.io/v1

kind: SecurityContextConstraints

metadata:

 name: redis-enterprise-scc-v2

 annotations:

  kubernetes.io/description: redis-enterprise-scc is the minimal
SCC needed to run Redis Enterprise nodes on Kubernetes.

   It provides the same features as restricted-v2 SCC, but allows
pods to enable the SYS_RESOURCE capability,

   which is required by Redis Enterprise nodes to manage file
descriptor limits and OOM scores for database shards.

   Additionally, it requires pods to run as UID/GID 1001, which
are the UID/GID used within the Redis Enterprise node containers.

allowedCapabilities:

- SYS_RESOURCE

allowHostDirVolumePlugin: false

allowHostIPC : false
```

```
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegeEscalation: false
allowPrivilegedContainer: false
readOnlyRootFilesystem: false
runAsUser:
 type: MustRunAs
 uid: 1001
fsGroup:
 type: MustRunAs
 ranges:
 - min: 1001
   max: 1001
seLinuxContext:
 type: MustRunAs
seccompProfiles:
- runtime/default
supplementalGroups:
 type: RunAsAny
```

4. Provide the operator permissions for Redis Enterprise Operator and Cluster pods.

```
oc adm policy add-scc-to-user redis-enterprise-scc
system:serviceaccount:redisdb:redis-enterprise-operator

 oc adm policy add-scc-to-user redis-enterprise-scc
system:serviceaccount:redisdb:rec
```

5. Redis Enterprise Operator deployment is now available using OpenShift dashboard. For a visual walkthrough of Redis Enterprise Operator installation in Openshift cluster, see the Developer Redis page.

6. Once the operator is deployed, create a Redis cluster. You can use the following YAML definition as an example (adapt to your needs regarding size, persistency, storageClass, and more).

```
apiVersion: app.redislabs.com/v1
kind: RedisEnterpriseCluster
metadata:
  name: rec
spec:
  serviceAccountName: rec
  redisEnterpriseNodeResources:
    limits:
      cpu: '4'
      ephemeral-storage: 10Gi
      memory: 4Gi
    requests:
      cpu: '4'
      ephemeral-storage: 1Gi
      memory: 4Gi
  bootstrapperImageSpec:
    repository: registry.connect.redhat.com/redislabs/redis-
enterprise-operator
  clusterCredentialSecretName: rec
  nodes: 3
  persistentSpec:
    enabled: true
    storageClassName: gp3-csi
    volumeSize: 20Gi
  createServiceAccount: true
  username: your_admin_username
  clusterCredentialSecretRole: ''
  podStartingPolicy:
    enabled: false
    startingThresholdSeconds: 540
  redisEnterpriseServicesRiggerImageSpec:
```

```
    repository: registry.connect.redhat.com/redislabs/services-
manager
  redisEnterpriseImageSpec:
    imagePullPolicy: IfNotPresent
    repository: registry.connect.redhat.com/redislabs/redis-
enterprise
  uiServiceType: ClusterIP
  clusterCredentialSecretType: kubernetes
  servicesRiggerSpec:
    databaseServiceType: 'cluster_ip,headless'
    serviceNaming: bdb_name
  services:
    apiService:
      type: ClusterIP
```

7. Once the Redis cluster is ready, you can deploy a database to host the vector store. It is important to enable the search module and set enough memory to hold the initial index capacity. See the following steps for the example.
8. Create the secret for the Redis database using following YAML definition as an example.

```
apiVersion: v1
kind: Secret
metadata:
  name: redb-my-doc
  namespace: redisdb
type: Opaque
stringData:
  username: default
  password: 1a2b3c4d
```

9. Create the Redis database using following YAML definition as an example. apiVersion: app.redislabs.com/v1alpha1

```
kind: RedisEnterpriseDatabase
```

```
metadata:
  name: my-doc
  namespace: redisdb
spec:
  databaseSecretName: redb-my-doc
  memorySize: 4GB
  modulesList:
    - name: search
      version: 2.8.4
  persistence: snapshotEvery12Hour
  replication: true
  tlsMode: disabled
  type: redis
```

10. Once the database is deployed you will have:

- A Secret named redb-my-doc (or the name you put in the YAML). It holds the password to the default user account for this database.
- A Service named my-doc-headless (or the name you put in the YAML). From this Service you will get: The full URL to the service from within the cluster (such as my-doc-headless.my-namespace.svc.cluster.local) and the port that Redis is listening to (like 14155).

11. Your Redis URL is in the different notebooks or applications within this repository, the full URI you can construct will be similar to:

    **redis://default:password@server:port.**

With our example, it would be:

redis://default:1a2b3c4d@my-doc-headless.my-namespace.svc.cluster.local:14155.

## Document embeddings

This section describes the procedure to store domain specific documents embeddings in the Redis vector store.

### Creating an index and populating it with web page content using Redis (GitHub)

This section provides an example on how to ingest web document content into a Redis vector store.

---

**Note**: To ingest PDF documents content into Redis vector store, follow this procedure.

---

Requirements to create an index include a Redis cluster and a Redis database with at least 2GB of memory (to match with the initial index cap).

# Base parameters, the Redis information:

```
redis_url = "redis://server:port"
index_name = "dellwebdocs"
```

# Imports:

```
from langchain.document_loaders import WebBaseLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.embeddings.huggingface import HuggingFaceEmbeddings
from langchain.vectorstores.redis import Redis
```

# Ingesting new documents:

```
loader =
WebBaseLoader(["https://infohub.delltechnologies.com/l/design-
guide-sql-server-2022-database-solution-with-object-storage-on-
dell-hardware-stack/business-challenge-193/",

"https://infohub.delltechnologies.com/l/design-guide-sql-server-
2022-database-solution-with-object-storage-on-dell-hardware-
stack/solution-introduction-81/",

"https://infohub.delltechnologies.com/l/design-guide-sql-server-
2022-database-solution-with-object-storage-on-dell-hardware-
stack/design-guide-introduction-28/"
                        ])
data = loader.load()
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1024,
                                               chunk_overlap=40)
all_splits = text_splitter.split_documents(data)
embeddings = HuggingFaceEmbeddings()
rds = Redis.from_existing_index(embeddings,
                                redis_url=redis_url,
                                index_name=index_name
schema="dellwebdocs_redis_schema.yaml")
rds.add_documents(all_splits)
```

# Write the schema to a yaml file to be able to open the index later:

```
rds.write_schema("redis_schema.yaml")
```

## Gradio deployment

This section describes the procedure to deploy Gradio to provide a user interface for a digital assistant.

### Gradio UI for RAG using Caikit + TGIS and Redis (GitHub)

This is a simple UI example for a RAG-based Chatbot using Gradio, Caikit + TGIS for LLM inference, and Redis as a vector database.

Deployment requirements include:

- A Caikit + TGIS deployment with a deployed LLM. This example is based on Llama2 but depending on your LLM you may need to adapt the prompt.
- A Redis installation with a database and an index already populated with documents. See GitHub for deployment instructions.
- An index in the Redis database that you have populated with documents. See GitHub for an example.

### Deployment on OpenShift

A pre-built container image of the application is available at: `quay.io/rh-aiservices-bu/gradio-caikit-rag-redis:latest`

The deployment folder includes the necessary files necessary to deploy the application:

- **cm_redis_schema.yaml**: A ConfigMap containing the schema for the database you have created in Redis (see the ingestion Notebook).
- **cm_certificate.yaml**: You must put your Caikit endpoint certificate and create this ConfigMap prior to the deployment.
- **deployment.yaml**: You must provide the URL of your inference server in the placeholder on L54 and Redis information for L56 and L58. Please feel free to modify other parameters as you see fit.
- **service.yaml**
- **route.yaml**

The different parameters you must pass as environment variables in the deployment are:

```
INFERENCE_SERVER_URL - mandatory
REDIS_URL - mandatory
REDIS_INDEX - mandatory
MAX_NEW_TOKENS - optional, default: 512
MIN_NEW_TOKENS - optional, default: 100
```

Create a new project in OpenShift cluster using the following command:

```
oc new-project dell-digital-assistant
```

Clone dell digital assistant GitHub repository and go to:
`dell-digital-assistant/05-UI/Gradio/caikit-rag-redis/`

```
git clone https://github.com/DellBizApps/dell-digital-assistent.git

cd dell-digital-assistent/05-UI/Gradio/caikit-rag-redis/

oc apply -f deployment/
```

---

**Note**: You can also deploy the application in OpenShift dashboard by using the source to image feature of the OpenShift cluster by following the instructions available on the Red Hat Customer Portal.

---

The deployment replicas are initially set to 0 to let you properly fill in the environment variables based on your setup. Change the values accordingly and scale the deployment replicas from 0 to 1.

### Deployment summary

Caikit was deployed to serve the Llama 2 model, TGI provided text generation, and Gradio and LangChain were used for the user interface and tool integration. The Llama 2 model was served using Caikit. Redis database was also deployed and the documents were embedded into its vector store.

# Chapter 5 Solution execution

This chapter presents the following topics:

# Digital assistant execution

## Overview

This chapter describes the function of digital assistant using an example. This chapter also describes how to access and validate the digital assistant.

## Accessing the digital assistant

Once the digital assistant is deployed and configured successfully, go to the project where the digital assistant has been deployed in the OpenShift dashboard (Network > Routes). On the Routes page, you will find the URL to launch the digital assistant.

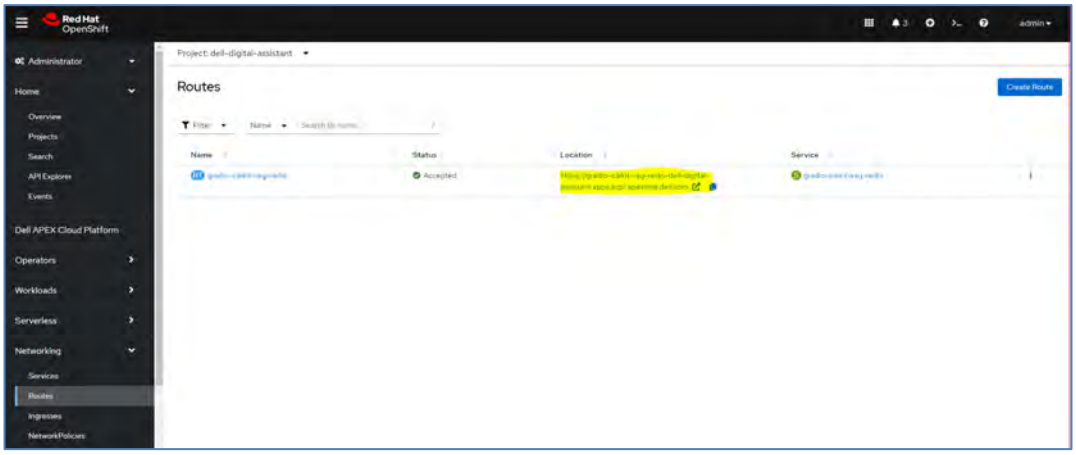


**Figure 13. Link to access the digital assistant.**

The following figure showcases the digital assistant user interface.

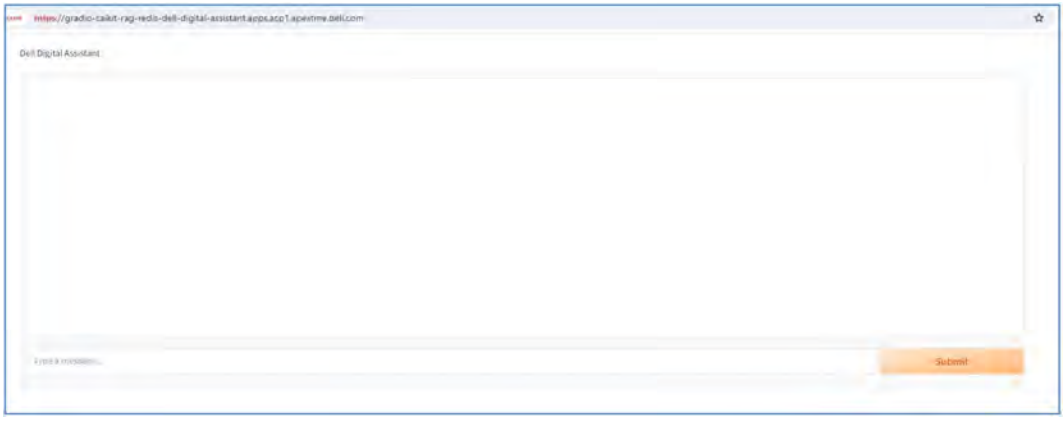


**Figure 14. Digital assistant UI**

## Digital assistant demonstration

Figure 15 shows an example of the user querying the digital assistant before embedding the domain specific information into the vector store. The digital assistant response shows that it does not have relavant information about Dell ObjectScale.

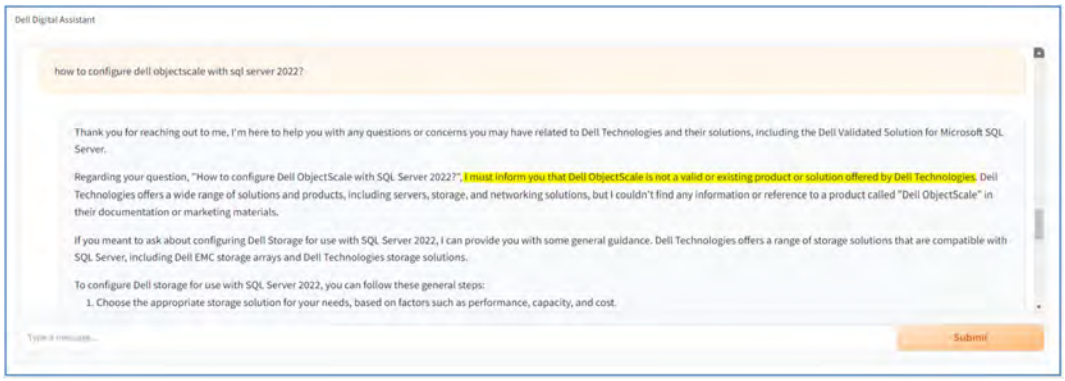


**Figure 15. Response before domain specific information.**

The following figure shows an example of the response after adding the domain specific embeddings into the vector store.

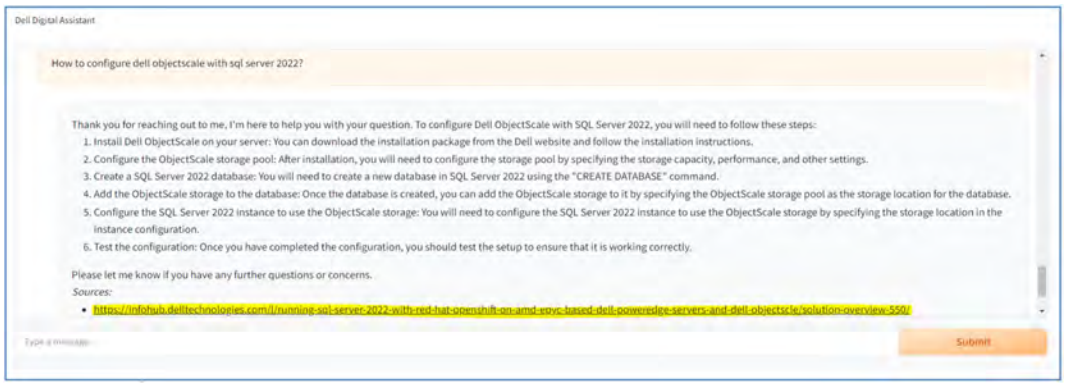


**Figure 16. Response after embedding domain specific information.**

Figure 17 demonstrates that the digital assistant is able to provide more relevant information using the knowledge base from Redis vector database. The digital assistant can also provide the accurate links to source documents. Users can open the provided links to access the source document for further reference.

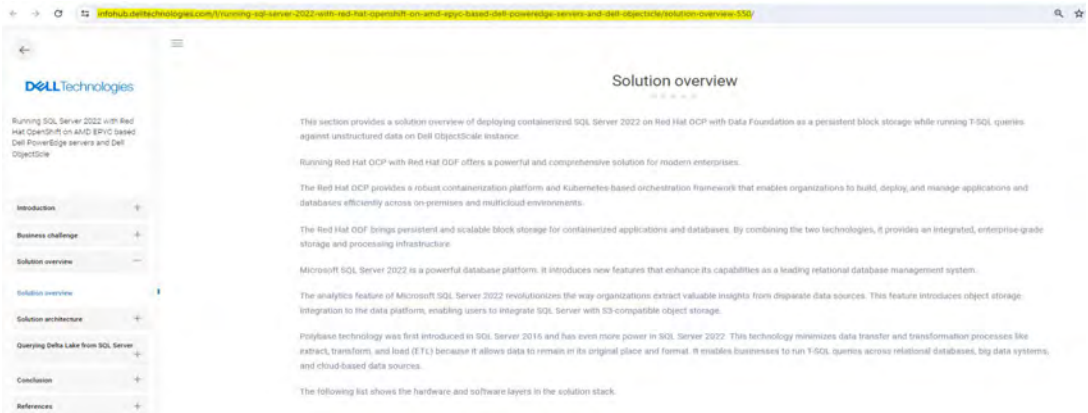


**Figure 17. Accessing the source document.**

# Chapter 6 Summary

This chapter presents the following topics:

# Summary

## Overview

Implementing a digital assistant with Red Hat OpenShift AI on Dell APEX Cloud Platform for Red Hat OpenShift has been created to address the needs of organizations that need to develop and run a custom LLM-based application using domain-specific information that is relevant to their own organization.

The technology behind AI is evolving at a rapid phase, and companies might lack AI experts or the time to design, deploy, and manage solution stacks at the necessary pace. Dell Technologies provides validated design guidance to enable customers to build their own AI solutions on Dell APEX Cloud Platform for Red Hat OpenShift with speed and confidence. Validated workload-optimized architectures can reduce customer proof-of-concept time and eliminate months of design and testing time that is required to plan correct configurations prior to the deployment.

Dell Technologies and Red Hat have teamed up to deliver customers a full-stack AI solution built on the Dell APEX Cloud Platform for Red Hat OpenShift utilizing Red Hat OpenShift AI. The partnership between Dell and Red Hat provides a powerful solution that brings reliability, performance, and scalability to AI workloads. Red Hat OpenShift AI offers organizations an efficient way to deploy an integrated set of common open-source and third-party tools to perform AI/ML modeling.

Building, deploying, and managing a LLM-based application can be complex. In this validated design, we have provided design guidance on deploying real-world LLM-based digital assistant on Dell APEX Cloud Platform for Red Hat OpenShift using Red Hat OpenShift AI and have successfully demonstrated the RAG capabilities to provide domain specific context to the LLM.

# We value your feedback

Dell Technologies and the authors of this document welcome your feedback on the solution and the solution documentation. Contact the Dell Technologies Solutions team by email.

**Authors:** Sanjeev Ranjan, Balakrishnan R – *Dell Technologies*

**Contributors**:

James A Martin, Tom Dau, Ava English – *Dell Technologies*

Christopher Chase, Guillaume Moutier – *Red Hat*

**Note**: For links to additional documentation for this solution, the Dell Technologies Solutions Info Hub for Artificial Intelligence.

# Chapter 7 References

This chapter presents the following topics:

## Dell Technologies documentation

The following Dell Technologies documentation provides additional and relevant information. Access to these documents depends on your login credentials. If you do not have access to a document, contact your Dell Technologies representative.

- *Dell ObjectScale 1.2.x Administration Guide*

## Red Hat documentation

The following Red Hat documentation provides additional and relevant information:

- *Product Documentation for OpenShift Container Platform 4.13*
- *Product Documentation for Red Hat OpenShift Data Science*

## Llama 2 documentation

The following Llama 2 webpage provides additional and relevant information:

- *Meta Llama 2*

## LangChain documentation

The following LangChain documentation provides additional and relevant information:

- *LangChain Python Docs*

## Redis documentation

The following Redis documentation provides additional and relevant information:

- *Redis docs*

## Gradio documentation

The following Gradio documentation provides additional and relevant information:

- *Gradio docs*

## Caikit documentation

The following Caikit GitHub documentation provides additional and relevant information:

- *Caikit getting started*
- *Caikit-NLP Github*

# Text Generation Inference documentation

The following TGI GitHub documentation provides additional and relevant information:

- *Text Generation Inference GitHub*

# Appendix A Open-Source License

This appendix presents the following topics:

# Open-Source Licensing Information

This document provides design guidance on configuring a digital assistant using open-source tools. Open-source tools are distributed under a variety of licenses, each with its own terms and conditions. It is important to review and understand the license terms of each open-source tool you use before using it in your own solution. This will help you to ensure that you are complying with the license requirements and that you are not infringing on the intellectual property rights of the copyright holders.

It is the responsibility of the user to review and comply with the licensing terms and conditions of each tool mentioned in this document. Licensing information for each tool can be found on the respective tool's official website or in its documentation. We have also listed links to some of the open-source tools license page below.

## Llama 2

Llama 2 is an open-source LLM. The licensing information for Llama 2 can be found on Meta web page.

## LangChain

LangChain is an open-source tool that can be used to work with LLMs. The licensing information for LangChain can be found on LangChain GitHub web page.

## Redis

Redis is an open-source, in-memory data structure store used as a database, cache, message broker, and streaming. The licensing information for Redis can be found on Redis web page.

## Gradio

Gradio is an open-source tool that helps to generate an easy-to-use UI machine learning web apps with few lines of code. The licensing information for Gradio is available on the Gradio GitHub web page.

## Caikit

Caikit is an open-source AI toolkit that enables users to manage models. For licensing information for Caikit, see the Caikit GitHub web page.

## TGI

TGI is an open-source tool that enables high-performance text generation for open-source LLMs. The licensing information for TGI is available on the TGI GitHub web page.