Reference Architecture

Generative AI with Intel AI and Red Hat OpenShift AI

Table of Contents

1 Introduction 2
1.1 Scope of this document 2
1.2 Key objectives 2
1.3 Intended audience 2
2 Solution overview 3
2.1 The challenge of generative AI in production 3
2.1.1 Design time 3
2.1.2 Integration time 3
2.1.3 Runtime 3
2.2 What a production generative AI system looks like 3
2.3 Benefits of choosing Intel AI for generative AI workloads 4
2.3.1 Intel Xeon 6 processor family 4
2.3.2 Intel Gaudi 3 accelerators 4
2.3.3 Intel Xeon 6 processor as a head node for another accelerator 5
2.4 Benefits of using Red Hat OpenShift AI along with Intel AI 6
2.4.1 What is OpenShift AI? 6
2.4.2 Design-time benefits 8
2.4.3 Integration-time benefits 8
2.4.4 Runtime benefits 8
2.5 Primary use cases 8
2.5.1 Inference 8
2.5.2 Fine-tuning 9
3 Architecture design 9
3.1 High-level architecture 9
3.2 Hardware overview 10
3.2.1 Compute (CPU/HPU) 10
3.2.2 Network (HPU fabric) 10
3.2.3 Storage 10
3.3 Software overview 11
3.4 Stack comparison against competition 12
4 Deployment guide 13
4.1 Components and versions used in this guide 13
4.2 Key steps 13
4.3 Prepare the cluster 13
4.3.1 OpenShift SNO 13
4.3.2 Pod PID limit 13
4.4 Install operators 14
4.4.1 LVM Storage Operator 14
4.4.2 Intel Gaudi Base Operator 14
4.4.3 Red Hat OpenShift Serverless Operator 17
4.4.4 Red Hat OpenShift Service Mesh 2 Operator 17
4.4.5 Red Hat OpenShift AI Operator 17
4.5 OpenShift image registry 20
4.5.1 Pushing an image to the internal registry 20
5 Deploy workloads using OpenShift AI 21
5.1 Running an inference server 21
5.2 Running a fine-tuning job 28
6 Monitoring and observability 31
6.1 Red Hat OpenShift Observability 31
6.2 Intel Gaudi Prometheus Exporter (in-band) 31
6.2.1 Prometheus metrics using curl 31
6.2.2 Visualizing Prometheus metrics (OpenShift web console) 32
6.2.3 Visualizing Prometheus metrics (Grafana) 33
6.3 Intel Gaudi BMC exporter (out-of-band) 36
7 Performance optimization 36
7.1 Performance benchmarking 36
7.1.1 MLPerf 36
7.2 Model optimization 36
7.3 PyTorch optimization 36
7.4 Debugging 36
8 References and additional resources 37
8.1 Product links 37
8.2 Documentation links 37
8.3 Official support 37
8.4 Community support 37
9 References 38
10 Notices and Disclaimers 38

Reference Architecture

Generative AI with Intel AI and Red Hat OpenShift AI

1 Introduction

1.1 Scope of this document

This document is designed to help organizations see the value proposition of using OpenShift and OpenShift AI with the Intel AI portfolio. It provides guidance on hardware and software requirements and offers step-by-step deployment instructions to ensure successful implementation of the most common generative AI (GenAI) solutions.

This document is a comprehensive guide for understanding, deploying, managing and operating AI workloads using single-node Intel® Xeon® processors and Intel® Gaudi® 3 AI accelerators in conjunction with Red Hat OpenShift AI. It is a technical reference entry point for organizations leveraging OpenShift AI to support generative AI workloads on Intel Xeon processors or Intel Gaudi 3 hardware.

1.2 Key objectives

The objectives of this document are to:

1. **Define:** Recommend a reference architecture and integration for OpenShift AI on Intel Xeon processors and Intel Gaudi accelerators.
2. **Deploy:** Provide deployment guidance for setting up a solution using Intel Xeon processors and Intel Gaudi accelerators on OpenShift AI.
3. **Run:** Explain best practices for running the most common generative AI workloads on OpenShift and Intel AI.
4. **Showcase Value:** Highlight how Intel Xeon processors and Intel Gaudi accelerators, combined with OpenShift AI, enhance generative AI development and deployment.

1.3 Intended audience

This reference design is intended to be helpful to a wide range of people involved in deploying an AI system:

- **Enterprise Architects:** Evaluating generative AI infrastructure solutions.
- **AI Engineers and Data Scientists:** Deploying models on OpenShift AI.
- **DevOps and MLOps Professionals:** Optimizing generative AI pipelines.
- **Infrastructure Engineers:** Managing generative AI infrastructure on OpenShift.
- **System Integrators:** Deciding what configuration to prepare.
- **Original Device Manufacturers (ODMs):** Needing a simple reference setup to consume.
- **AI Researchers:** Desiring to use Intel's unique feature set.

2 Solution overview

2.1 The challenge of generative AI in production

A production-quality system for generative AI needs to provide a comprehensive, balanced solution to many challenges. There are obvious factors like cost, performance and enterprise-quality hardware. But there are many other factors like total cost of ownership, open-source support and vendor lock-in that are important for developing and running a production-quality inference system. When all the factors are considered, Intel has some of the best solutions for production-quality inference.

2.1.1 Design time

Developers ask for day-zero support for large language models (LLMs) like Llama. A good developer experience is essential, so platforms must offer optimized tools, comprehensive documentation, seamless integration with widely used frameworks such as PyTorch, blueprints for popular AI models and pre-validated solutions that accelerate time to deployment. System architects look for rapid return on investment (ROI), open-source ecosystems and adherence to open standards that help avoid vendor lock-in, which can result from proprietary technologies.

2.1.2 Integration time

During integration, security and scalability become top priorities. Enterprises need platforms supporting new and legacy models with easy migration paths and robust connectivity. Solutions that eliminate the need for specialized hardware are particularly valuable, allowing teams to leverage standard data center infrastructure to reduce costs and simplify deployment and maintenance. Pre-optimized solutions help integrate quickly and efficiently without sacrificing performance or flexibility.

2.1.3 Runtime

At runtime, the focus shifts to operational efficiency and reducing long-term total cost of ownership (TCO) while maintaining performance and support for a wide range of models and frameworks to ensure flexibility and future proofing. Long-term TCO requires validated, enterprise-grade technologies that are easy to use and maintain. Successfully running generative AI in production hinges on a platform's ability to deliver a reliable trusted partnership and cost-effective performance across the entire lifecycle.

2.2 What a production generative AI system looks like

An AI solution is a collection of components that deliver some business value. Those components are usually arranged in layers.

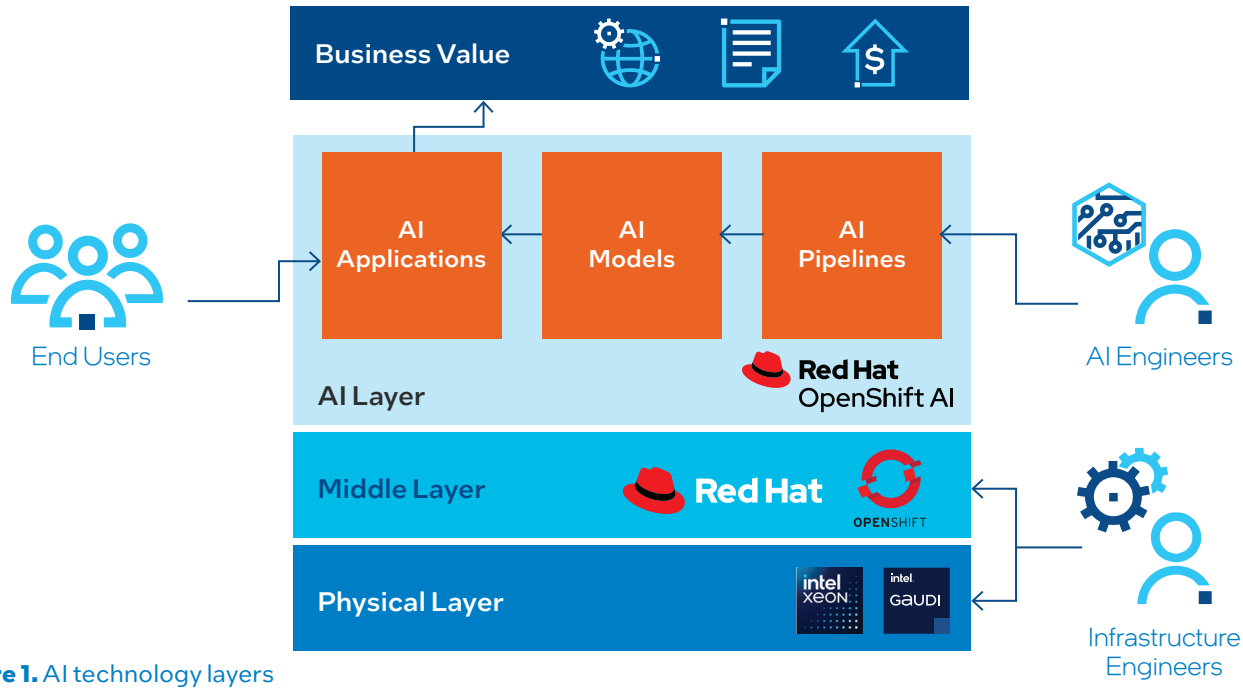


Figure 1. AI technology layers

Figure 1 illustrates how the technology layers interact with end users, AI engineers and infrastructure engineers to deliver AI-driven business value. In the "AI layer," end users engage with AI applications that leverage trained AI models and pipelines, developed and maintained by AI engineers using Red Hat OpenShift AI, to perform tasks such as data analysis and content generation. In the "middle layer," these applications are orchestrated by Red Hat OpenShift, ensuring efficient resource allocation and scalability. The "physical layer," comprising Intel Xeon processors and/or Intel Gaudi accelerators, provides the necessary computational power for AI workloads, managed by infrastructure engineers. These components integrate seamlessly to produce business value.

2.3 Benefits of choosing Intel AI for generative AI workloads

Intel offers two accelerated AI data center solutions, allowing enterprises to flexibly choose whichever best suits their workload:

1. **Intel Xeon 6 Processor Family with AI Acceleration:** AI Acceleration using Intel® Advanced Matrix Extensions (Intel® AMX) is suitable for small inference GenAI workloads and retrieval-augmented generation (RAG) use cases due to its well-optimized vector search capabilities.

2. **Intel Gaudi 3 Accelerators:** Ideal for inference and fine-tuning generative AI workloads.

2.3.1 Intel Xeon 6 processor family

The Intel Xeon 6 processor has been upgraded with AI accelerators that make it suitable for AI inferencing if you have small models or a mixture of workloads. Features include:

- **Cost-Effectiveness:** For small models, the CPU can be fast enough and lower cost than purchasing additional AI hardware and infrastructure in your data center.
- **Eliminate Data Bottlenecks Inherent When Using Discrete Accelerators:** The CPU has direct access to system memory. Intel Xeon 6 processors have more CPU cores, more memory bandwidth and more cache to improve AI performance. Multiplexed Rank DIMMs (MRDIMMs) deliver improved memory bandwidth and up to 504 MB low-latency last-level cache (LLC) to boost performance for memory-bound AI workloads.
- **Integrated AI Acceleration:** Intel Xeon 6 processors already have integrated AI accelerators. Intel Xeon 6 processors with P-cores include Intel AMX and Intel® Advanced Vector Extensions 512 (Intel® AVX-512) acceleration in every core to boost AI workloads. Intel AMX includes support for INT8, BF16 and now FP16 data types. Optimizations are integrated into the mainstream distributions of popular frameworks like TensorFlow, PyTorch, llama.cpp, vLLM and others.
- **Easy Scaling:** Easily scale power efficiency and server consolidation. Address growing power usage and space constraints by refreshing aging infrastructure. Intel Xeon 6 processors with P-cores bring improved energy efficiency that scales with utilization. Consolidating servers powered by Intel Xeon 6 processors reduces server space requirements and energy consumption for a lower TCO.
- **Flexibility:** If your workload is diverse (e.g., AI+HPC or AI+RAG) and most of your workload isn't AI, then a CPU solution that improves all your workload might be more sensible than an accelerator that accelerates just the AI part, especially if you don't know what your future workload mix will be. A faster CPU makes everything faster and can run a wide range of AI and HPC workloads.
- **Open Standards:** Intel Xeon processors use standard data center hardware and tools and have an open AI stack. No need to retrain your engineering team or employ specialists to support specialist AI hardware that causes vendor lock-in.
- **Security:** Intel Xeon processors have decades of hardening against threats. Specific security features include Intel® Crypto Acceleration, Intel® QuickAssist Technology (Intel® QAT), Intel® Trust Domain Extensions (Intel® TDX), Intel® Control-flow Enforcement Technology (Intel® CET) and Intel® Platform Firmware Resilience (Intel® PFR).[1]
- **Stability:** Intel Xeon processors provide the stability of knowing that your legacy workloads will continue to work. You won't be forced to migrate everything that runs on the head node to ARM CPUs to follow someone's roadmap.
- **ML Framework Support:** The leading frameworks and data analytics have been optimized for Intel Xeon processors (e.g., PyTorch, TensorFlow, PaddlePaddle, MXNet, Pandas, NumPy and SciPy).[2]
- **Intel Extensions for PyTorch (IPEX):** IPEX is a Python package that extends the official PyTorch with feature optimizations that make it easy to use the performance features on Intel Xeon processors. Optimizations take advantage of Intel AVX-512, Vector Neural Network Instructions (VNNI) and Intel AMX on Intel CPUs. IPEX supports the most popular ML models.[3]
- **Precision Support:** Intel Xeon 6 processors have Intel AMX BF16 support for both real-time inference and training workloads.

2.3.2 Intel Gaudi 3 accelerators

An Intel Gaudi accelerator is dedicated AI hardware that is designed to accelerate inference and fine tuning generative AI workloads. Intel Gaudi accelerators offer several advantages:

- **Fundamentally Designed for Accelerating AI, Not Graphics:** AI applications increasingly demand faster and more energy-efficient hardware solutions, and the Intel Gaudi 3 AI accelerator was designed to answer that demand.
- **Rapid ROI:** Intel Gaudi 3 accelerators are engineered to be quickly ready for production:
  - **Optimized for Developers:** Utilize software tools and developer resources to get up to speed.
  - **Integrated With PyTorch:** Keep working with a library the team already knows.
  - **Supports New and Existing Models:** Customize reference models, start fresh or migrate existing models using open-source tools, including resources from Hugging Face.

  - **Easy Migration of GPU-Based Models:** Quickly port existing solutions using purpose-built software tools.

- **Connectivity:** Intel Gaudi 3 accelerators have extensive I/O connectivity per accelerator so you can enable massive scale-up and scale-out.[4]
- **Open Standards**
  - **No Vendor Lock-in:** Other AI hardware vendors have proprietary stacks that lock customers into their ecosystem and limit their ability to customize. Avoid risky investments in locked, proprietary technologies such as NVLink, NVSwitch, and InfiniBand.
  - **Open Source:** Intel offers an open ecosystem, with a stack comprising open-source components. Intel has teamed with industry partners and the open-source community to provide a rich ecosystem of validated technologies and seamless integration with common operating systems, compilers, libraries and frameworks.
  - **Uses Ethernet:** Use the networking infrastructure you already own, and support future needs with standard Ethernet hardware.
- **Ease of Use**
  - **Blueprints:** Intel provides blueprint solutions[5] and performance figures[6] for the most popular AI models.
  - **Pre-Optimized:** Get new products and AI services into production fast using existing platforms and optimized AI libraries and frameworks.[7]
  - **Scalable:** Enterprise-ready CPU and GPU solutions. Add Intel® AI accelerators for more performance as needed.[8]
- **LLM Support**
  - **Engine Support:** Intel Gaudi accelerators have support for the TGI and vLLM inference engines.[9]
  - **Model Support:** Fast support for the most popular open-source LLMs, like DeepSeek, Llama 4, Mistral, Qwen and more thanks to support for TGI and vLLM.[10,11,12,13]
- **Security**
  - **Cultivate a Second Supplier**: Reliance on a single vendor can be risky. It reduces your pricing leverage and makes you vulnerable to your supplier's changes of pricing model, focus, licensing terms or bundling of hardware and services.
  - **Diversify Your Supply Chain:** Reliance on a single vendor's supply chain is a particular concern in this time of fast-changing tariffs and globally distributed supply chains.
  - **Future-Proof Your Stack:** Developing for one specific proprietary hardware platform makes it harder to adopt other solutions in the future. Many upcoming accelerator products from many vendors can be addressed by adopting an open shared software stack. By choosing a stack that is open to a global ecosystem of hardware and software vendors, solutions can be matched to future business needs.

2.3.3 Intel Xeon 6 processor as a head node for another accelerator

Intel Xeon 6 processors are great head nodes for OpenShift AI clusters, even if you choose an AI accelerator from AMD or NVIDIA.

Intel Xeon processors continue to be the host CPU of choice on the world's most powerful AI accelerator platforms for data preprocessing support.

Intel Xeon CPUs have been consistently chosen as head nodes or host CPUs by major AI implementors[14,15] because, with their high core count and improved memory bandwidth, they are well suited to efficiently feed data to the GPU for AI training and inference.

Intel Xeon 6 processors provide the performance needed for a wide range of workloads, including AI and high-performance computing (HPC).[16]

Intel Xeon CPUs enable efficient data transfer:

- **Faster PCIe Data Transfer Between the CPU and GPU:** Intel Xeon 6 processors deliver up to 20% more PCIe lanes than the previous generation,[17,16] accelerating data offloads. Intel Xeon 6 processors have up to 96 lanes of PCIe 5.0 per socket compared to the 80 PCIe lanes in Intel Xeon 5 processor, giving more bandwidth for data transfer and more flexibility in connecting peripherals and storage devices.[18]
- **Faster Memory:** Intel Xeon 6 processors (both P-cores and E-cores) support DDR5 6400 high-speed memory, giving 6,400 mega transfers per second (MT/s), up from Intel Xeon processor Gen 5's 5,600 MT/s. The Intel Xeon 6 P-core version is the first processor family to incorporate the fast type of DDR5 memory called MRDIMMs (Multiplexed Rank DIMMs) that can deliver 37% greater memory bandwidth than RDIMMs, with an expected data transfer rate of up to 8,800 MT/s.[19]

- **Optimized Performance per Core:** Intel Xeon 6 processors with P-cores are optimized for performance, with up to 504 MB L3 cache and exceptionally low latency at large L3 access sizes and Intel AVX512, which is only supported on Intel Xeon 6 processors with P-cores. Intel Xeon 6 processors with P-cores are great for public cloud workloads with improved performance per vCPU for floating point operations, transactional databases and HPC workloads.[16]
- **More Cores:** P-cores (Performance-cores) have up to 128 cores per socket, double the previous Intel Xeon generation. E-cores (Efficient-cores) offer up to 288 cores per socket.[16]
- **AI Data Preparation:** Intel Xeon CPUs can be used to prepare data, boosting the speed of vector math common to HPC and classical AI workloads.

2.4 Benefits of using Red Hat OpenShift AI along with Intel AI

Red Hat OpenShift AI offers a comprehensive suite of tools and features designed to streamline the development process for cloud-native AI applications.

2.4.1 What is OpenShift AI?

Red Hat OpenShift AI is a flexible, scalable MLOps platform with tools to build, deploy and manage AI-enabled applications. Built using open-source technologies, it provides trusted, operationally consistent capabilities for teams to experiment, serve models and deliver innovative apps.[20] Red Hat OpenShift AI provides an environment to develop, train, serve, test and monitor AI/ML models and applications on-premises or in the cloud.[21] OpenShift AI is a platform built on top of Red Hat OpenShift to provide a secure, supported and proven enterprise-grade environment for efficiently developing and deploying AI applications. It provides a suite of tools to streamline the development process and management of cloud-native AI applications by offering foundational elements from open source. OpenShift AI flexibly supports hybrid cloud environments, hardware acceleration and many AI/ML tools.

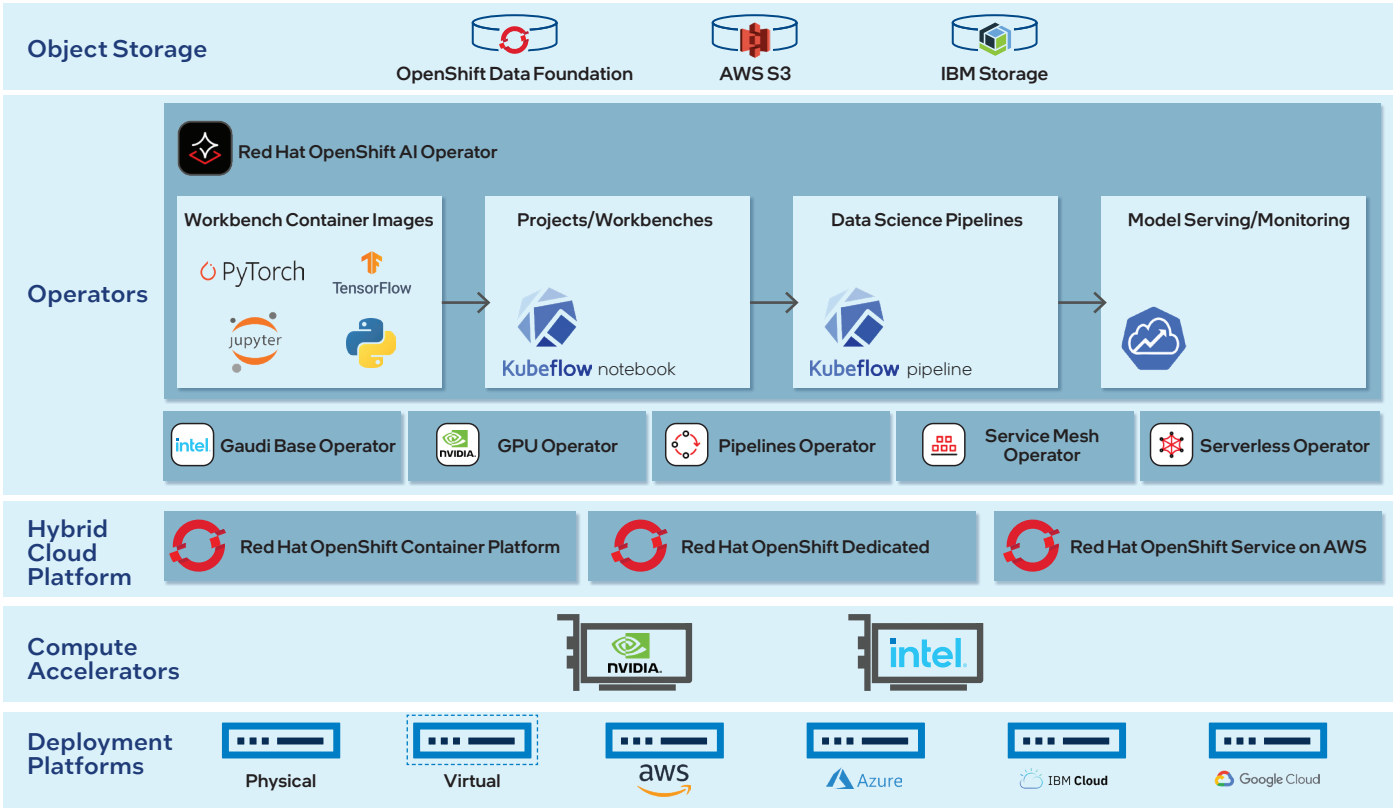


Figure 2. Key components of OpenShift AI

The key components of OpenShift AI, shown in Figure 2, are:

- **Object Storage:** Stores models and data.
  - **Red Hat OpenShift Data Foundation:** Software-defined storage for containers and persistent storage for cloud-native applications, ensuring features like encryption, replication and availability across hybrid clouds.
  - **AWS S3:** The S3 API is the de facto standard for HTTP-based access to object storage services.
  - **IBM Storage:** Scalable and resilient object storage service on IBM Cloud.

- **Operators:** Operators automate the creation, configuration and management of instances of Kubernetes-native applications and provide automation throughout the stack. With Red Hat OpenShift Certified Operators found in the embedded OperatorHub, developers and cluster admins have access to a library of workloads "as-a-service."
    - **OpenShift AI Operator:** Includes everything needed for data scientists and IT operations administrators.
        - **Workbench Container Images:** Creates a "workbench" containerized environment for data scientists. OpenShift AI includes a number of workbench images optimized with tools and libraries such as PyTorch that are ready to use for multiple common data science stacks and for model development.[22]
        - **Projects/Workbenches:** Workbenches run as OpenShift pods and are designed for machine learning and data science. Workbenches include the Jupyter notebook execution environment and data science libraries.
        - **Data Science Pipelines:** A workflow that executes scripts or Jupyter notebooks in an automated way. Using pipelines, you can automate the execution of different steps and store the results (e.g., gathering data, cleaning data, training a model, evaluating the model and finally saving the model to S3 storage).
        - **Model Serving/Monitoring:** Model Server uses a data connection to download the model file from S3-compatible storage. After the download, the model server exposes the model via REST or gRPC APIs. OpenShift AI uses KServe as the model-serving platform and supports model runtimes such as OpenVINO. [23]
    - **Intel Gaudi Base Operator:** Automates the management of all necessary Intel Gaudi software components, including drivers, Kubernetes device plugin, container runtime, feature discovery and monitoring tools.[24]
    - **Pipelines Operator:** Installs Red Hat OpenShift Pipelines, including the custom resources for the Pipelines configuration.[25] OpenShift Pipelines assists developers by allowing them to create advanced continuous integration (CI) workflows for their applications. It is a cloud-native continuous integration (CI) solution that integrates with OpenShift's console, allowing for automated builds and deployments alongside application development. It helps automate the process of building, testing and deploying machine learning models.[26]
    - **Service Mesh Operator:** Installs OpenShift Service Mesh, which provides a way to connect, manage, observe and secure microservices-based applications. It lets developers integrate communications policies without changing application code.[27]
    - **Serverless Operator:** Installs OpenShift Serverless, which simplifies the development of hybrid cloud applications by abstracting infrastructure complexities and dynamically adjusting application resources to match demand.[28]
- **Hybrid Cloud Platform:** Offers tools to quickly deliver applications, with security and compliance across operating environments, clouds and developer environments.
    - **Red Hat OpenShift Container Platform:** Supports confidential AI through in-use encryption, leveraging Intel® Tiber™ Trust Authority Services to ensure data security during AI operations.
    - **Red Hat OpenShift Dedicated:** A managed Red Hat OpenShift offering available on Google Cloud or Amazon Web Services that lets you take advantage of native Google Cloud or AWS services.[29,30]
    - **Red Hat OpenShift Service on AWS:** A fully managed turnkey application platform with native AWS service and integration, including integrated support and billing.[31]
- **Compute Accelerators**: OpenShift AI supports AI accelerator integration. OpenShift AI supports native hardware AI accelerators for training and inference, including Intel Gaudi accelerators and Intel Data Center GPUs.
- **Deployment Platforms:** Manage your environment on any supported infrastructure or cloud — or let OpenShift manage it for you on one of our partner cloud platforms. Choose how and where you build. One platform. Any workload. Any infrastructure.[32]
    - **Physical:** A suite of solutions ranging from simple container orchestration up to advanced multi-cluster security and management to build and scale applications across environments.
    - **Virtual:** A cost-effective solution to migrate, deploy, manage and scale virtual machines.
    - **Cloud Provider:** OpenShift AI includes a managed cloud service available as a managed application platform on the cloud provider of your choice. It can be AWS, Azure, IBM Cloud or Google Cloud.

Additional supporting resources are available for OpenShift projects:

- **Open Data Hub:** integrates open-source projects to provide a blueprint for building AI-as-a-Service on OpenShift. It simplifies the development of portable and cloud-native AI applications.[33]

2.4.2 Design-time benefits

By engaging with a single supplier, developers can avoid the complexities of coordinating multiple resources, ensuring seamless integration of components. OpenShift AI provides integrated development tools that enhance the developer experience, including multilanguage support, CI/CD pipelines, service mesh, serverless capabilities, model serving and monitoring and logging features.[34] The platform is enterprise-grade, offering certified components that remain patched throughout deployment, and is validated with IBM's cloud offerings and AI products, accelerating the impact of generative AI in workflows. OpenShift adheres to open-source standards, incorporating OCI containers and Kubernetes for container orchestration, and boasts a robust ecosystem with a wide variety of third-party integrations.

2.4.3 Integration-time benefits

During integration time, OpenShift AI components work out of the box, saving time and speeding up time-to-market. The platform ensures portability, allowing workloads to be deployed and configured consistently across different environments, thanks to OCI industry-standard container images. OpenShift is already integrated with Intel Xeon processors and Intel Gaudi worker nodes, ensuring compliance with industry standards and regulations.

Automated installation and upgrades are supported across various infrastructures, including on-premises and cloud environments, with services from the OperatorHub being deployable and upgradable with a single click. The platform offers streamlined automation for container and app builds, deployments, scaling and health management, along with multi-cluster management capabilities through Red Hat Advanced Cluster Management for Kubernetes.

2.4.4 Runtime benefits

At runtime, Red Hat OpenShift provides enterprise-grade support, simplified management and patching for all components, ensuring a consistent user experience across hybrid cloud environments. The platform includes integrated monitoring and automated maintenance operations, giving IT teams control and visibility over deployments and code pipelines. OpenShift optimizes resource utilization by integrating Intel Infrastructure Processing Units, enhancing efficiency and reducing power consumption. Security is bolstered through automated policies for container deployment and runtime protection, with advanced capabilities like runtime threat detection and vulnerability management. OpenShift improves performance for AI applications by integrating with AI acceleration features in Intel Xeon processors and offers scalability to thousands of instances across hundreds of nodes. The platform provides flexibility for hybrid infrastructure deployment, allowing organizations to choose between self-managed or fully managed services across on-premises, cloud and hybrid environments.

2.5 Primary use cases

Intel Xeon processors and Intel Gaudi accelerators are particularly well-suited for several key inference and fine-tuning use cases:

2.5.1 Inference

Inference use cases for Intel Xeon processors and Intel Gaudi accelerators:

- **Text Generation:** Generative AI models like GPT (generative pre-trained transformer) require substantial computational resources to generate coherent and contextually relevant text.
- **Image Synthesis:** Applications such as creating realistic images from textual descriptions or generating new images based on existing datasets benefit from the accelerator's ability to process large models and datasets, improving the quality and speed of image synthesis.
- **Music and Audio Generation:** Generative AI models that compose music or generate audio effects can leverage the parallel processing capabilities of Intel Xeon processors and Intel Gaudi accelerators to produce high-quality audio outputs in real time.
- **Style Transfer:** In tasks where the style of one image is applied to another, such as transforming photographs into paintings, the accelerators can efficiently manage the complex neural networks involved, enhancing the speed and quality of style transfer.
- **3D Model Generation:** Creating 3D models from 2D images or textual descriptions requires significant computational power, which Intel Xeon processors and Intel Gaudi accelerators can provide, enabling more detailed and accurate 3D model generation.
- **Video Generation and Enhancement:** Generative AI models that produce or enhance video content, such as generating realistic animations or improving video resolution, benefit from the accelerator's ability to handle large volumes of data and complex computations.
- **Chatbots and Conversational Agents:** These applications require real-time text generation and understanding, tasks that are well-supported by the accelerator's ability to process large language models quickly and efficiently.

2.5.2 Fine-tuning

Fine-tuning use cases for Intel Xeon processors and Intel Gaudi accelerators:

- **Domain-Specific Language Models:** Fine-tuning large language models like GPT for specific industries or domains (e.g., legal, medical or financial) allows these models to generate more relevant and accurate text. Intel Xeon processors and Intel Gaudi accelerators can efficiently handle the computational demands of adapting models to specialized vocabularies and contexts.
- **Custom Image Generation:** Fine-tuning generative models to produce images that align with specific artistic styles or brand aesthetics can be computationally intensive. The accelerators provide the necessary power to adjust models for unique visual characteristics and requirements.
- **Personalized Recommendation Systems:** By fine-tuning models on user-specific data, businesses can enhance recommendation systems to deliver more personalized content or product suggestions. Intel Xeon processors and Intel Gaudi accelerators facilitate the rapid processing of large datasets to refine model outputs.
- **Voice and Speech Adaptation:** Fine-tuning speech synthesis models to mimic specific voices or accents requires significant computational resources. The accelerators can efficiently manage the complex adjustments needed to achieve high-quality, personalized audio outputs.
- **Enhanced Style Transfer:** Fine-tuning models to apply specific artistic styles or effects to images or videos can improve the quality and uniqueness of style transfer applications. Intel Xeon processors and Intel Gaudi accelerators support the detailed computations required for these customizations.
- **Interactive Chatbots and Virtual Assistants:** Fine-tuning conversational models to understand and respond to user-specific queries or preferences enhances the interactivity and relevance of chatbots. The accelerators enable efficient processing of large language models to refine conversational capabilities.
- **3D Content Personalization:** Fine-tuning models to generate 3D content tailored to specific user preferences or project requirements can be resource intensive. Intel Xeon processors and Intel Gaudi accelerators provide the computational power needed to adapt models for detailed and personalized 3D outputs.

Overall, Intel Xeon processors and Intel Gaudi accelerators excel in generative AI use cases that demand high computational throughput, low latency and the ability to manage large and complex models, making them ideal for advancing the capabilities of generative technologies across various domains.

3 Architecture design

3.1 High-level architecture

Figure 3 presents a high-level architecture for deploying AI applications and models using a combination of Red Hat and Intel AI technologies.

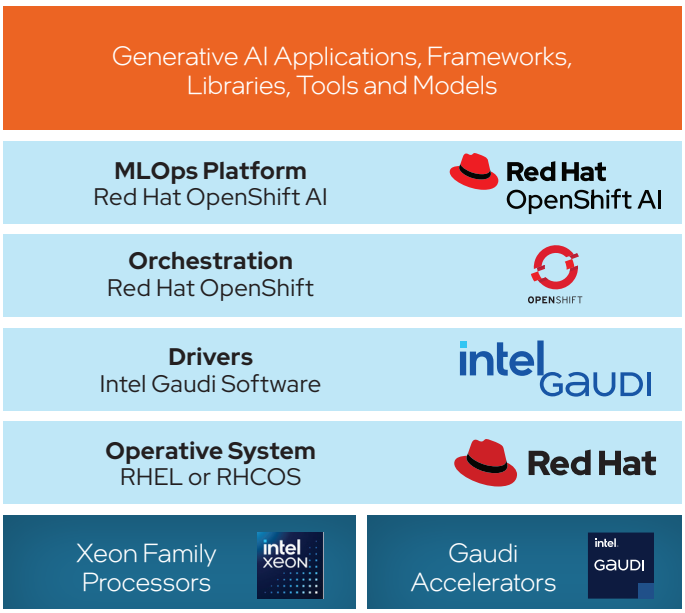


Figure 3. High-level architecture

At the foundation of this architecture are the hardware components: the Intel Xeon processor family and Intel Gaudi accelerators, which provide the computational power necessary for generative AI workloads.

The software stack begins with the operating system layer, comprising Red Hat Enterprise Linux (RHEL) or Red Hat CoreOS (RHCOS), which offer a stable and secure environment for running applications. Above this layer, Intel Gaudi Software drivers are integrated to optimize the performance of Intel Gaudi accelerators, ensuring efficient processing of AI tasks.

The orchestration layer is managed by Red Hat OpenShift, a Kubernetes-based platform that facilitates the deployment, scaling and management of containerized applications. This layer ensures seamless integration and operation across both on-premises and cloud environments.

At the top of the stack is the MLOps platform, Red Hat OpenShift AI, which provides tools and frameworks for developing, deploying and managing machine learning models. This platform is designed to streamline the lifecycle of AI applications, enabling businesses to leverage generative AI technologies effectively.

Overall, this architecture offers a comprehensive solution for building and scaling AI applications, combining robust hardware with a sophisticated software stack to meet diverse AI business needs.

3.2 Hardware overview

3.2.1 Compute (CPU/HPU)

Leading OEMs now offer AI computing systems equipped with Intel Xeon 6 processors and Intel Gaudi 3 accelerators. This reference design, shown in Figure 4, uses a compute element, referred to as a node, that consists of an air-cooled server chassis housing two Intel Xeon 6 processors and eight Intel Gaudi 3 accelerators.

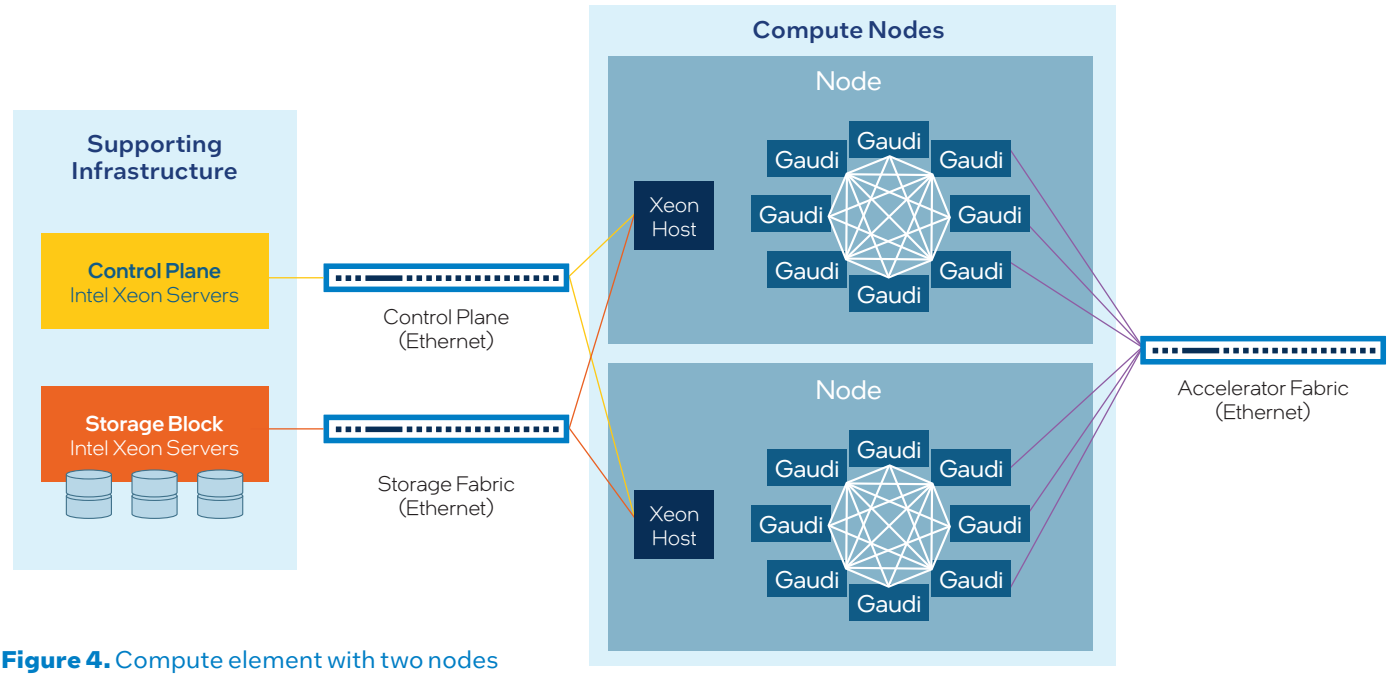


Figure 4. Compute element with two nodes

This node is specifically engineered to facilitate the rapid development, training and deployment of large generative AI models. It can be configured into scalable clusters comprising multiple nodes to efficiently process vast datasets for AI applications, making it ideal for large language models (LLMs), multi-modal models, recommendation engines, neural network, RAG and agent-based applications.

Intel Xeon 6 processors and Intel Gaudi 3 accelerators have established themselves as a competitive alternative because of their generative AI compute capability, pricing, energy efficiency and market availability.

3.2.2 Network (HPU fabric)

Intel Gaudi accelerators are designed for exceptional scalability, with each accelerator interconnected to every other within a node in an all-to-all configuration. Each accelerator includes 24 RDMA over Converged Ethernet (RoCE) ports — 21 dedicated to scale-up connectivity within an eight-card universal baseboard, and three reserved for scale-out connectivity.

To extend scalability beyond a single node, 800-Gbps OSFP Ethernet switches are employed for the accelerator fabric. These switches directly interconnect all AI accelerators within the cluster, delivering up to 25.6 Tbps of throughput.[35]

3.2.3 Storage

Storage is the component of the solution most shaped by customer needs and workload demands. Whether deployed in a single-node or multi-node cluster, various storage solutions are available to accommodate these differing requirements.

3.2.3.1 Single node

In single-node deployments, local disks configured with logical volumes are preferred. The OpenShift Data Foundation Logical Volume Manager Operator provides the ideal implementation for this purpose. For further details, please consult the following documentation: Red Hat OpenShift Data Foundation Single Node Documentation.

3.2.3.2 Multi node

OpenShift AI and Intel AI can seamlessly scale to multi-node configurations utilizing networked storage. This initial document does not cover these configurations due to the diverse array of distributed storage solutions available for customers to choose from. For more comprehensive information, please refer to the Red Hat OpenShift Data Foundation Architecture Documentation.

3.3 Software overview

Figure 5 illustrates the various software layers in the Intel AI and Red Hat AI software stack, utilized in different GenAI solutions. The lowest layer of the stack pertains to Intel hardware and is included solely for context and consistency purposes.

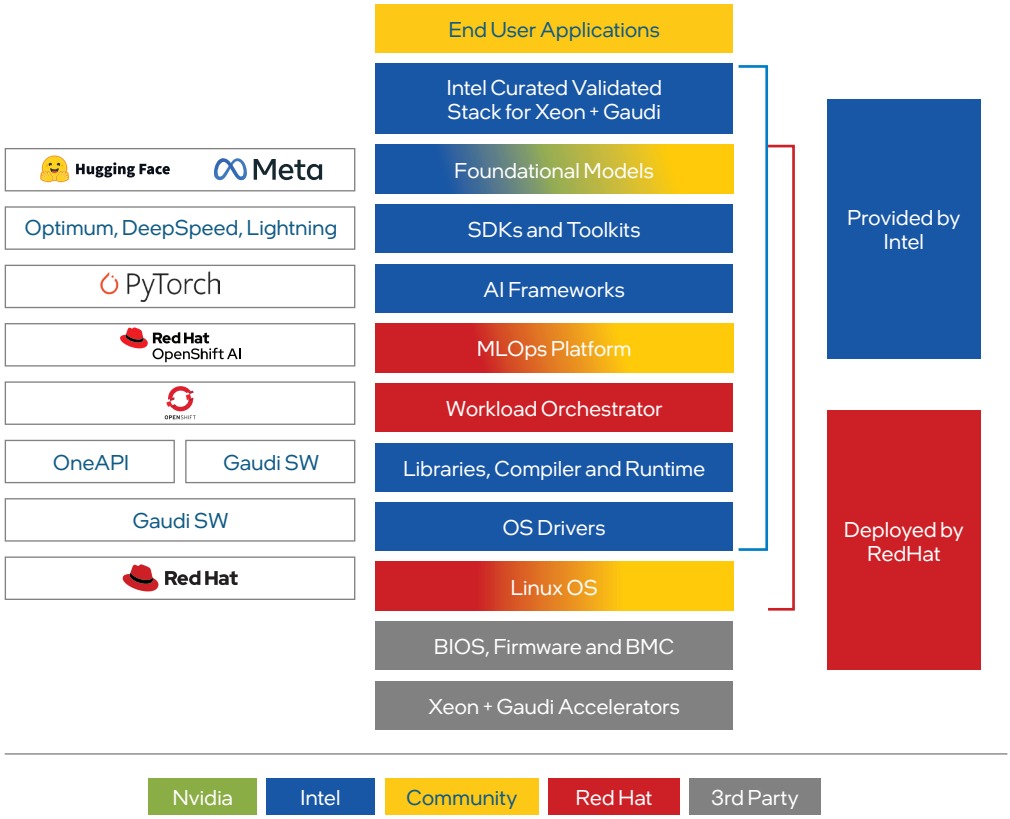


Figure 5. Software layers in the Intel AI and Red Hat AI software stack

The layers in the software stack include the following:

- **End-User Applications:** The top layer represents applications that users interact with, leveraging the capabilities of the underlying stack to deliver AI-driven functionalities and solutions.
- **Intel Curated Validated Stack for Intel Xeon Processors With Intel Gaudi Accelerators:** A comprehensive, validated software stack optimized for Intel Xeon processors and Intel Gaudi accelerators, ensuring compatibility and performance for AI workloads.
- **Foundational Models:** Pre-trained models from various sources, such as Meta, serving as the basis for developing AI applications.
- **SDKs and Toolkits:** Software development kits and toolkits, such as Optimum, DeepSpeed and Lightning, offering essential tools for building and optimizing AI applications.
- **AI Frameworks:** Popular AI frameworks like PyTorch, providing the infrastructure for developing machine learning models and applications.
- **MLOps Platform:** Red Hat OpenShift AI, providing a platform for managing the lifecycle of generative AI models, from development to deployment and monitoring.
- **Workload Orchestrator:** Red Hat OpenShift, offering orchestration capabilities for efficient management and scaling of containerized workloads across various environments.
- **Libraries, Compiler and Runtime:** Intel's OneAPI and Intel Gaudi software, providing libraries, compilers and runtime environments necessary for efficient execution of AI applications.

- **OS Drivers:** Operating system drivers, including Intel Gaudi software, that facilitate communication between hardware and software components.
- **Linux OS:** The operating system layer, featuring Red Hat Enterprise Linux (RHEL) or Red Hat CoreOS (RHCOS), which offers a stable and secure environment for running applications and managing resources.
- **BIOS, Firmware and BMC:** Essential components for hardware initialization and management, including the basic input/output system, firmware and baseboard management controller.
- **Intel Xeon Processors With Intel Gaudi Accelerators:** The foundational hardware layer comprising the Intel Xeon processors and Intel Gaudi accelerators, providing the necessary computational power for AI workloads.

3.4 Stack comparison against competition

Figure 6 presents a comparison between the components from one of our competitors, NVIDIA, with their corresponding counterparts offered by Intel, Red Hat and the open-source community.

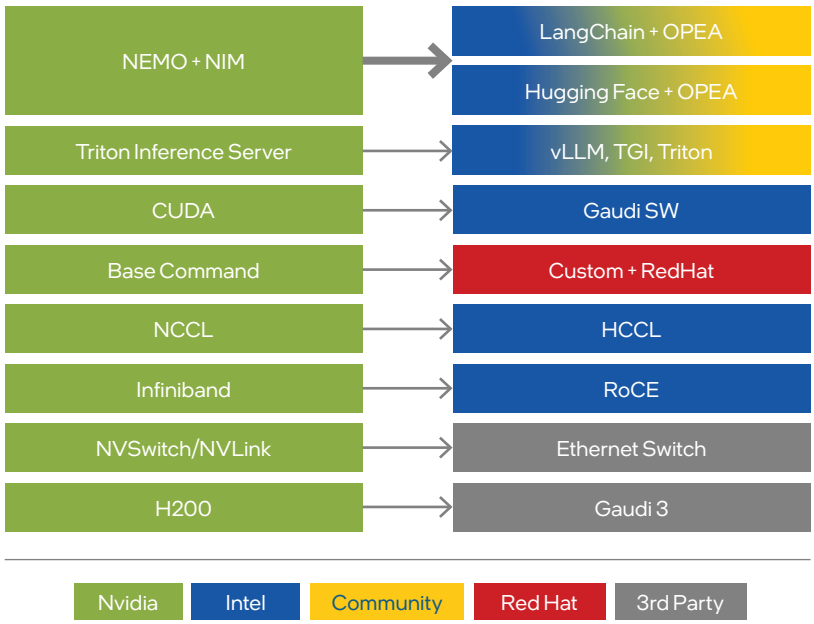


Figure 6. Nvidia H200 and Intel Gaudi 3 stack comparison

4 Deployment guide

This guide provides step-by-step instructions for deploying AI workloads on one specific hardware configuration: a node that contains 2x Intel Xeon 6 processors and 8x Intel Gaudi 3 accelerators on Red Hat OpenShift AI.

4.1 Components and versions used in this guide

This guide uses the following components:

- 1x Intel AI node with
  - 2x Intel Xeon 4 processors
  - 8x Intel Gaudi 3 accelerators
  - 2 TB RAM
  - 1 TB NVMe boot drive
  - 1x 7 TB NVMe for data storage
- Red Hat OpenShift OCP v4.20.0
- Intel Gaudi Base Operator v1.22.1-6-2
- LVM Storage v4.20.0
- Red Hat OpenShift Serverless v1.36.1
- Red Hat OpenShift Service Mesh 2 v2.6.11-0
- Red Hat OpenShift AI v2.25.0
- Intel Gaudi PyTorch container base images v1.22.1 with PyTorch v2.7.1

4.2 Key steps

The key steps presented here include the following:

- **Prepare the Cluster:** Set up your cluster to ensure it's ready for AI deployment.
- **Install Operators:** Add the necessary components to support AI operations.
- **Deploy Workloads:** Implement your AI workloads to utilize Intel's AI capabilities fully.

Follow this guide to efficiently deploy AI workloads using Intel technology on Red Hat OpenShift AI.

4.3 Prepare the cluster

This guide uses a single-node OpenShift (SNO) installation as the deployment environment for AI Intel Gaudi accelerators. While this setup simplifies initial testing and deployment, the same approach can be extended to multi-node clusters for production-scale deployment.

4.3.1 OpenShift SNO

This guide uses the Assisted Installer to deploy a single-node OpenShift cluster for simplicity and rapid setup. For alternative approaches, refer to the OpenShift Installation guide (https://docs.redhat.com/en/documentation/openshift_container_platform/4.20/html-single/installation_overview/index)

4.3.2 Pod PID limit

OpenShift's default Process ID (PID) limit per pod is 4096. We need to increase it to be able to run workloads using multiple Intel Gaudi 3 devices.

```
oc label machineconfigpool worker custom-kubelet=set-pod-pid-limit-kubelet
```

Create a Kubelet config file custom-kubelet-pidslimit.yaml:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
metadata:
  name: custom-kubelet-pidslimit
spec:
  kubeletConfig:
    PodPidsLimit: 32768
  machineConfigPoolSelector:
    matchLabels:
      custom-kubelet: set-pod-pid-limit-kubelet
```

Then apply it:

```
oc apply -f custom-kubelet-pidslimit.yaml
```

NOTE: This will cause the node to reboot.

4.4 Install operators

4.4.1 LVM Storage Operator

In a production environment, it is common to utilize the OpenShift Data Foundation Operator along with distributed storage solutions to manage persistent storage needs efficiently. However, for this single-node use case, we can opt for the LVM Storage Operator to provision persistent storage using local volumes. This approach is suitable for scenarios where the complexity and overhead of distributed storage are unnecessary, allowing for a simpler and more direct method of managing storage on a single node.

Make sure to select the right device/volume: in this case we are using /dev/nvme7n1

This will delete all data on the disk

```
apiVersion: lvm.topolvm.io/v1alpha1
kind: LVMCluster
metadata:
  name: lvmcluster
  namespace: openshift-lvm-storage
spec:
  tolerations:
  - effect: NoSchedule
    key: xyz
    operator: Equal
    value: "true"
  storage:
    deviceClasses:
    - name: vg2
      fstype: ext4
      default: true
      deviceSelector:
        paths:
        - /dev/nvme7n1
        forceWipeDevicesAndDestroyAllData: true
      thinPoolConfig:
        name: thin-pool-1
        sizePercent: 90
        overprovisionRatio: 10
        chunkSize: 128Ki
        chunkSizeCalculationPolicy: Static
        metadataSizeCalculationPolicy: Host
```

To verify the status:

```
oc get lvmclusters.lvm.topolvm.io -o jsonpath='{.items[*].status}' -n openshift-lvm-storage
```

If no error is reported, check the creation of the default storage class:

```
oc get sc
NAME                 PROVISIONER   RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
lvms-vg1
(default)   topolvm.io    Delete          WaitForFirstConsumer   true                   4m56s
```

4.4.2 Intel Gaudi Base Operator

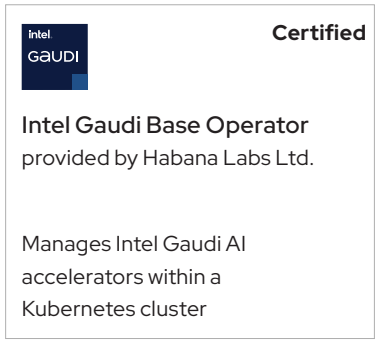
This section is for enabling Intel Gaudi accelerators using the related operator. The CPU does not require an operator.

4.4.2.1 Intel Gaudi firmware

Make sure you have the latest firmwares Installed on the Intel Gaudi 3 accelerators server. See this link for more details: https://docs.habana.ai/en/latest/Installation_Guide/Firmware_and_Platform_Components/index.html#firmware-and-platform-components

4.4.2.2 Deploying Intel Gaudi Base Operator

Make sure to use the latest version of Intel Gaudi Operator. Once the operator is installed, create a ClusterPolicy. As an example, for version 1.22.1, see the YAML file below:



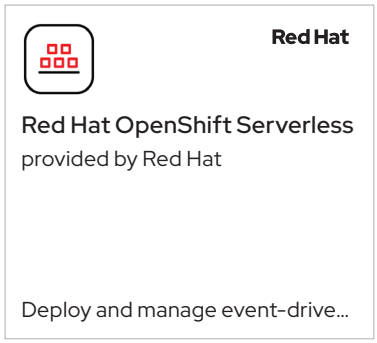

```yaml
apiVersion: habanalabs.habana.ai/v2
kind: ClusterPolicy
metadata:
  name: habana-ai
spec:
  feature_discovery:
    runner:
      image:
        repository: vault.habana.ai/habana-ai-operator/habanalabs-feature-discovery
        tag: 1.22.1-6
      resources:
        limits:
          cpu: 20m
          memory: 64Mi
        requests:
          cpu: 10m
          memory: 32Mi
    nfd_plugin: false
  driver:
    driver_loader:
      resources:
        limits:
          cpu: 4000m
          memory: 16Gi
        requests:
          cpu: 2000m
          memory: 8Gi
      images:
        rhel_8.6:
          repository: vault.habana.ai/habana-ai-operator/driver/rhel8.6/driver-installer
          tag: 1.22.1-6
        rhel_9.2:
          repository: vault.habana.ai/habana-ai-operator/driver/rhel9.2/driver-installer
          tag: 1.22.1-6
        rhel_9.4:
          repository: vault.habana.ai/habana-ai-operator/driver/rhel9.4/driver-installer
          tag: 1.22.1-6
        rhel_9.6:
          repository: vault.habana.ai/habana-ai-operator/driver/rhel9.6/driver-installer
          tag: 1.22.1-6
        tencentos_3.1:
          repository: vault.habana.ai/habana-ai-operator/driver/tencentos3.1/driver-installer
          tag: 1.22.1-6
        ubuntu_22.04:
          repository: vault.habana.ai/habana-ai-operator/driver/ubuntu22.04/driver-installer
          tag: 1.22.1-6
        ubuntu_24.04:
          repository: vault.habana.ai/habana-ai-operator/driver/ubuntu24.04/driver-installer
          tag: 1.22.1-6
      mlnx_ofed_repo_path: artifactory/gaudi-installer/deps
      mlnx_ofed_version: mlnx-ofed-5.8-2.0.3.0-rhel8.4-x86_64.tar.gz
```

```
    repo_path: artifactory/gaudi-installer/repos
    repo_server: vault.habana.ai
  driver_runner:
    image:
      repository: vault.habana.ai/habana-ai-operator/driver/ubuntu22.04/driver-installer
      tag: 1.22.1-6
    resources:
      limits:
        cpu: 20m
        memory: 64Mi
      requests:
        cpu: 10m
        memory: 32Mi
bmc_monitoring:
  image:
    repository: vault.habana.ai/habana-bmc-exporter/bmc-exporter
    tag: 1.22.1-6
  resources:
    limits:
      cpu: 250m
      memory: 250Mi
    requests:
      cpu: 150m
      memory: 100Mi
metric_exporter:
  runner:
    image:
      repository: vault.habana.ai/gaudi-metric-exporter/metric-exporter
      tag: 1.22.1-6
    resources:
      limits:
        cpu: 150m
        memory: 120Mi
      requests:
        cpu: 100m
        memory: 100Mi
  interval: 20
  port: 41611
runtime:
  configuration:
    container_engine: crio
  runner:
    image:
      repository: vault.habana.ai/habana-ai-operator/habana-container-runtime
      tag: 1.22.1-6
    resources:
      limits:
        cpu: 20m
        memory: 64Mi
      requests:
        cpu: 10m
        memory: 32Mi
device_plugin:
  image:
    repository: vault.habana.ai/docker-k8s-device-plugin/docker-k8s-device-plugin
    tag: 1.22.1-6
  resources:
    limits:
      cpu: 20m
      memory: 64Mi
    requests:
      cpu: 10m
      memory: 32Mi
image_registry: vault.habana.ai
```

4.4.3 Red Hat OpenShift Serverless Operator

This operator enables the installation and use of Knative Serving, Knative Eventing, and Knative Kafka on an OpenShift cluster.

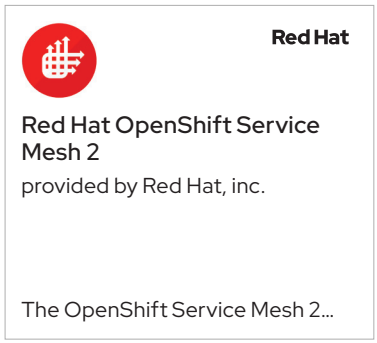
Use the web UI to install OpenShift Serverless Operator from Operator Hub:[36]



4.4.4 Red Hat OpenShift Service Mesh 2 Operator

This operator is based on the open-source Istio,[37] Envoy and Kiali projects. It enables a uniform way to connect, manage, observe and provide security for microservices-based applications on top of an OpenShift cluster.

Use the web UI to install OpenShift Service Mesh 2 Operator from Operator Hub:



4.4.5 Red Hat OpenShift AI Operator

OpenShift AI Operator can be installed using the web console or the CLI.

Using the web console search of OpenShift AI Operator:

Once the operator is installed, create a DataScienceCluster, as shown in Figure 7. We will not be using all the features so you can use default values for now. Make sure the status is Ready.

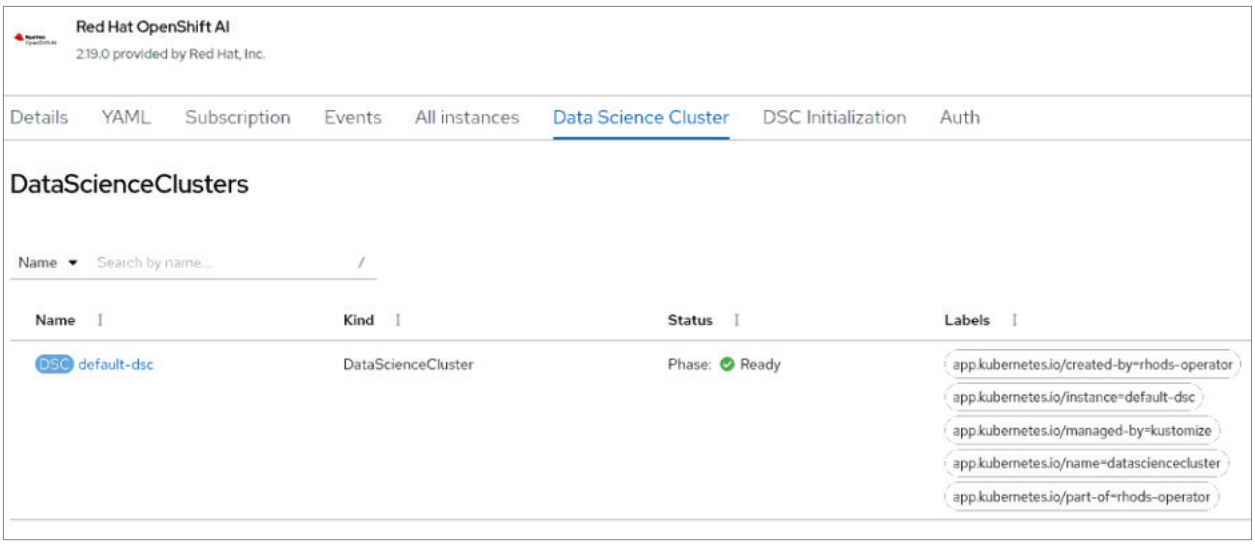


Figure 7. Creating a DataScienceCluster

4.4.5.1 Configure Intel Gaudi hardware profile

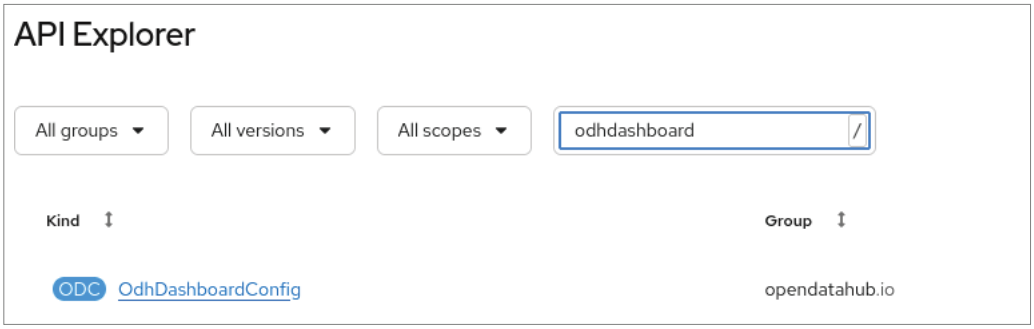
To enable Intel Gaudi hardware profile in Openshift AI create the yaml file below and then apply it:

```yaml
# Copyright (c) 2025 Intel Corporation
# SPDX-License-Identifier: Apache-2.0
apiVersion: infrastructure.opendatahub.io/v1alpha1
kind: HardwareProfile
metadata:
  annotations:
    opendatahub.io/disabled: 'false'
    opendatahub.io/display-name: gaudi
  name: gaudi
  namespace: redhat-ods-applications
spec:
  identifiers:
    - defaultCount: 2
      displayName: CPU
      identifier: cpu
      maxCount: '20'
      minCount: 1
      resourceType: CPU
    - defaultCount: 100Gi
      displayName: Memory
      identifier: memory
      maxCount: 200Gi
      minCount: 2Gi
      resourceType: Memory
    - defaultCount: 1
      displayName: gaudi
      identifier: habana.ai/gaudi
      maxCount: 8
      minCount: 1
      resourceType: Accelerator
```

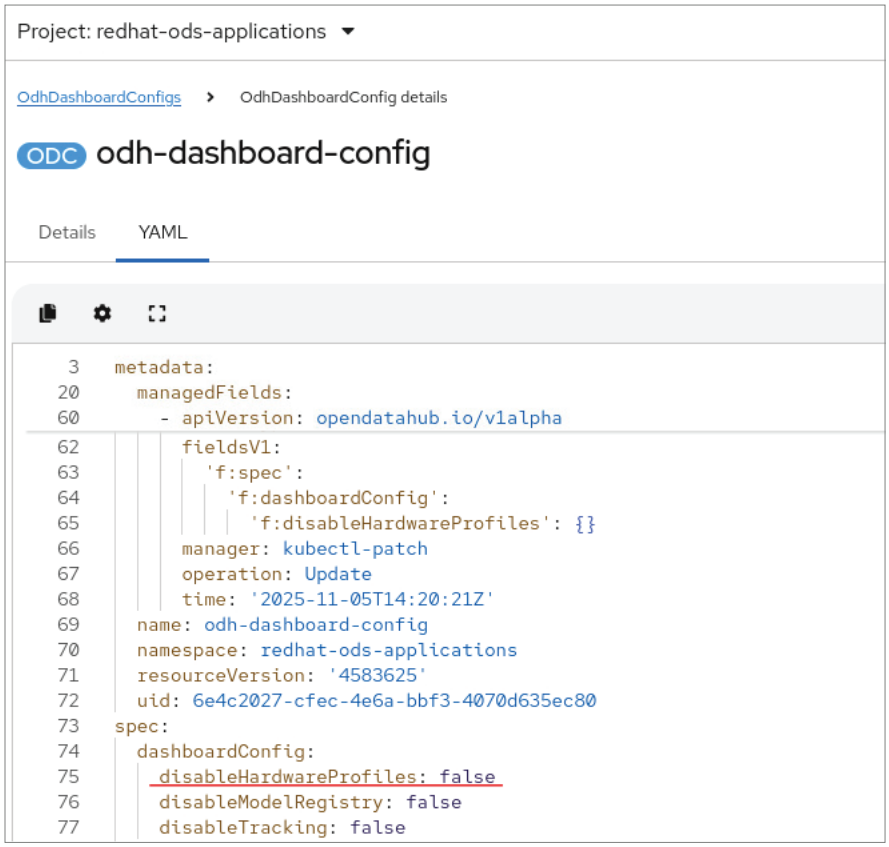
If you prefer to use the UI you first need to make sure that Hardware profile setting is available in OpenShiftAI dashboard::

Using OCP console navigate to API Explorer:

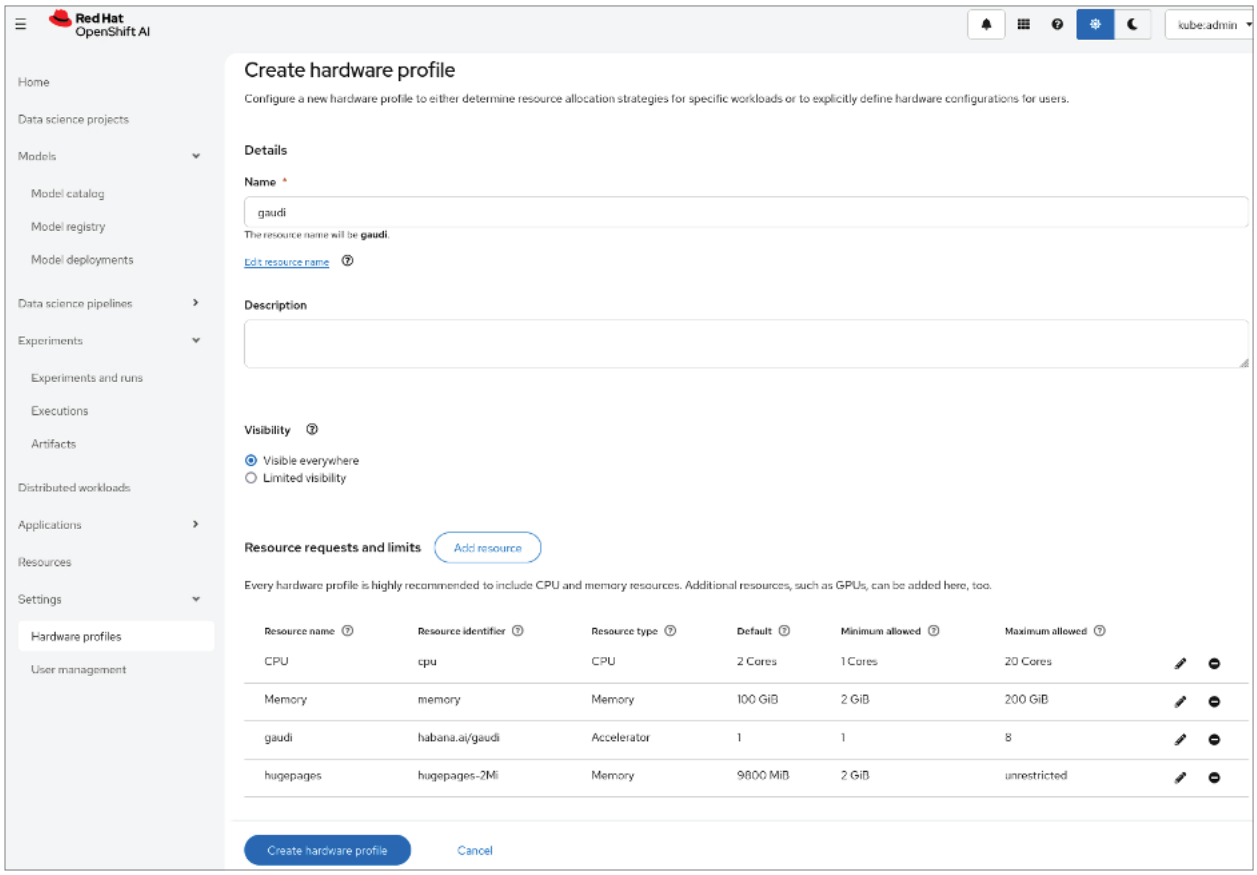
Then search for odhDashboardConfig



Make sure to select the project: redhat-ods-applications and then update spec.dashboardCofig disableHardwareProfiles: false



Then a Hardware profile can be created like below:



4.5 OpenShift image registry

This section covers the setup of the OpenShift image registry, which is essential for managing local container images in your cluster. Note that the configuration provided is suitable for non-production environments, such as development or testing. For production use, a high-availability solution is recommended; refer to the following link for more information: https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html/registry/setting-up-and-configuring-the-registry

The storage size for the registry can be configured in a PVC spec like the one below:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage
  namespace: openshift-image-registry
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
```

Then update the image registry config using the commands below:

```
oc patch configs.imageregistry.operator.openshift.io/cluster --type 'json' -p='[{"op": "add", "path": "/spec/storage/pvc", "value": {"claim": "image-registry-storage"}}]'
```

```
oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":{"rolloutStrategy":"Recreate","replicas":1}}'
```

```
oc patch configs.imageregistry.operator.openshift.io/cluster --type merge -p '{"spec":{"defaultRoute":true}}'
```

Make sure to add the registry route to /etc/hosts or your DNS server:

You can get the route using the following command:

```
oc get route default-route -n openshift-image-registry --template='{{ .spec.host }}'
```

4.5.1 Pushing an image to the internal registry

First make sure you're logged in:

```
oc login -u kubeadmin -p <password>
```

Log in using podman to the registry:

```
export REGISTRY=`oc get route default-route -n openshift-image-registry --template='{{ .spec.host }}'`
podman login -u kubeadmin -p `oc whoami --show-token` ${REGISTRY}
```

Tag and then push the image:

For example, we are here pushing an image test to the default namespace/project:

```
podman tag <SRC_IMAGE> ${REGISTRY}/default/test:latest
podman push  ${REGISTRY}/default/test:latest
```

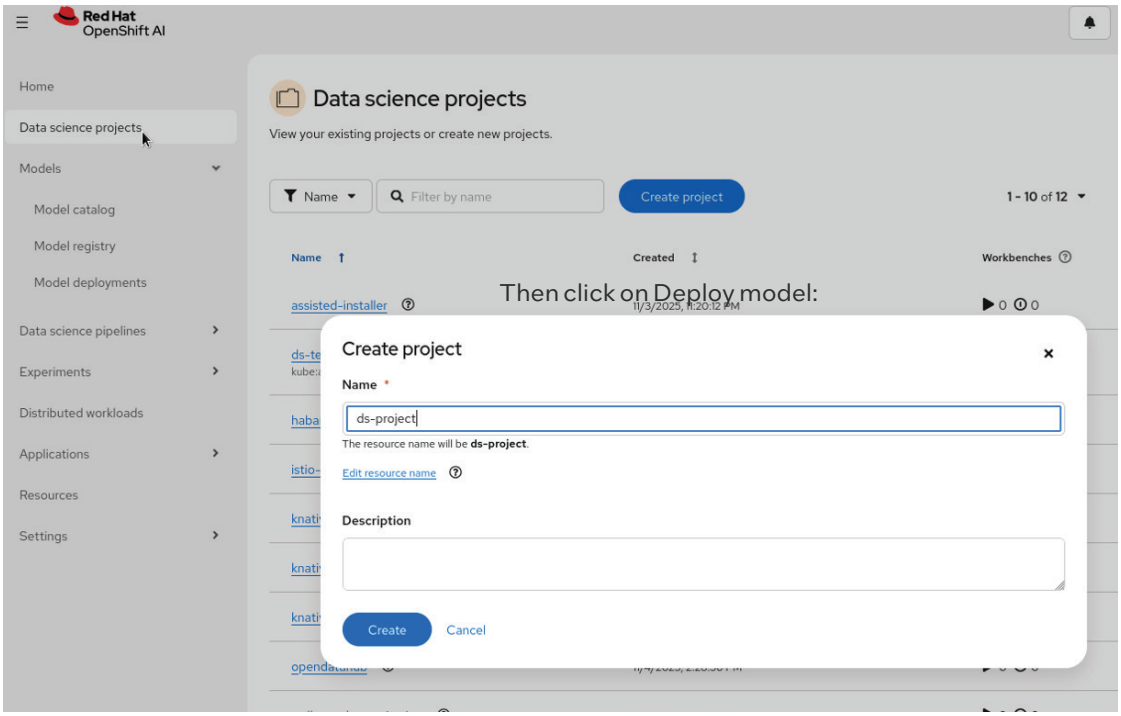
5 Deploy workloads using OpenShift AI

In this section, we demonstrate how to deploy workloads. We start with a basic set of steps to deploy an inference server using vLLM and OpenShift and run a Llama 3.1 fine-tuning job. Then we demonstrate a real-world chatQnA example that exercises the whole system.

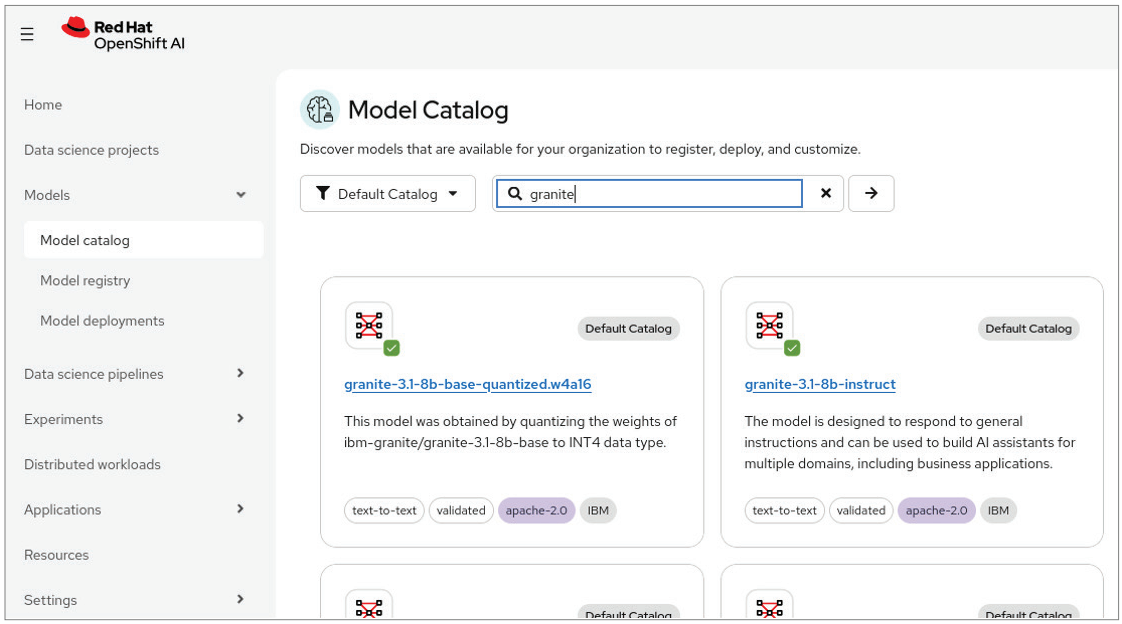
5.1 Running an inference server

This example provides a basic and straightforward method to set up a scalable inference server using OpenShift AI and inference models, leveraging the computational power of Intel Gaudi 3 processors.

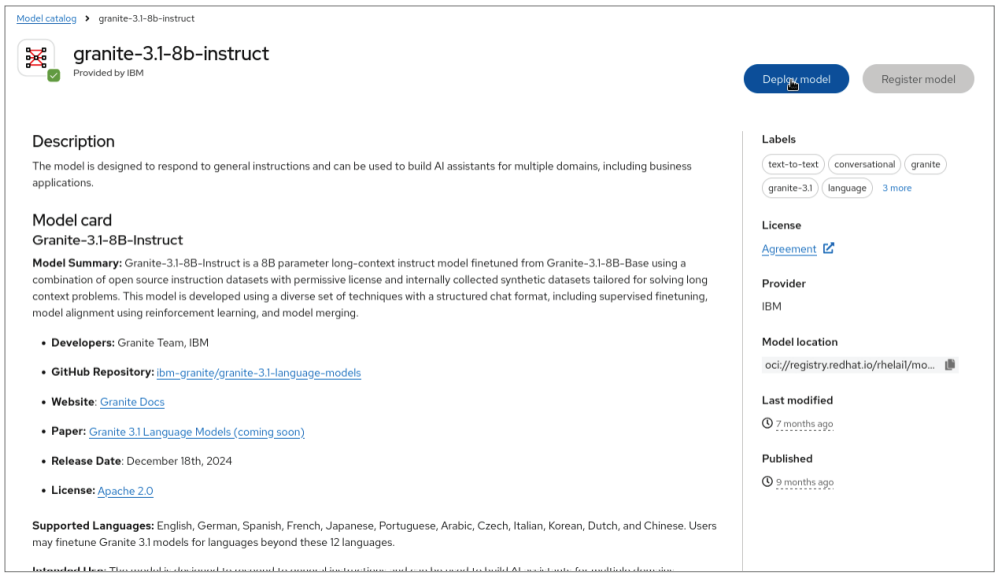
Open Red Hat Openshift AI dashboard then create a Data science project:



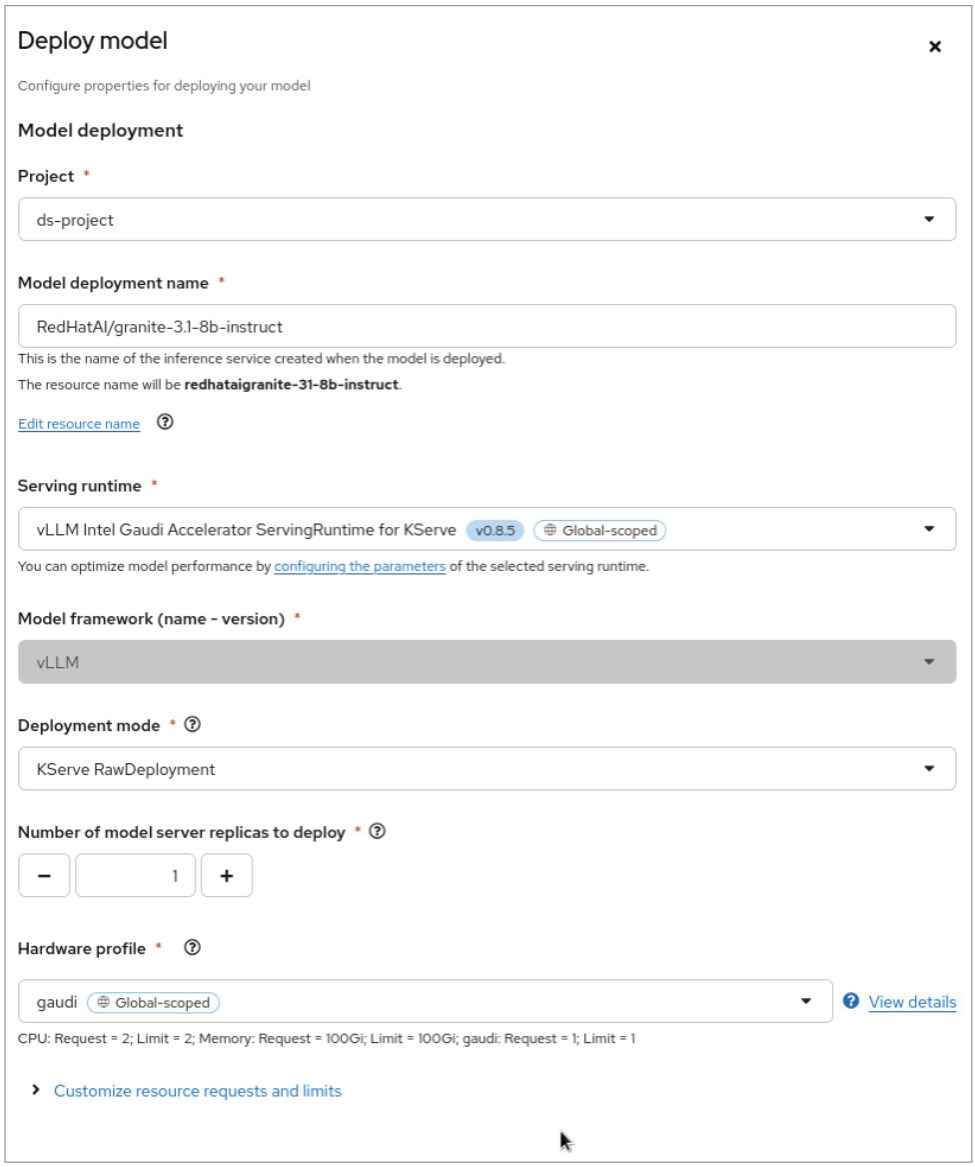
Navigate to Model catalog and pick granite-3.1-8b-instruct in this case:



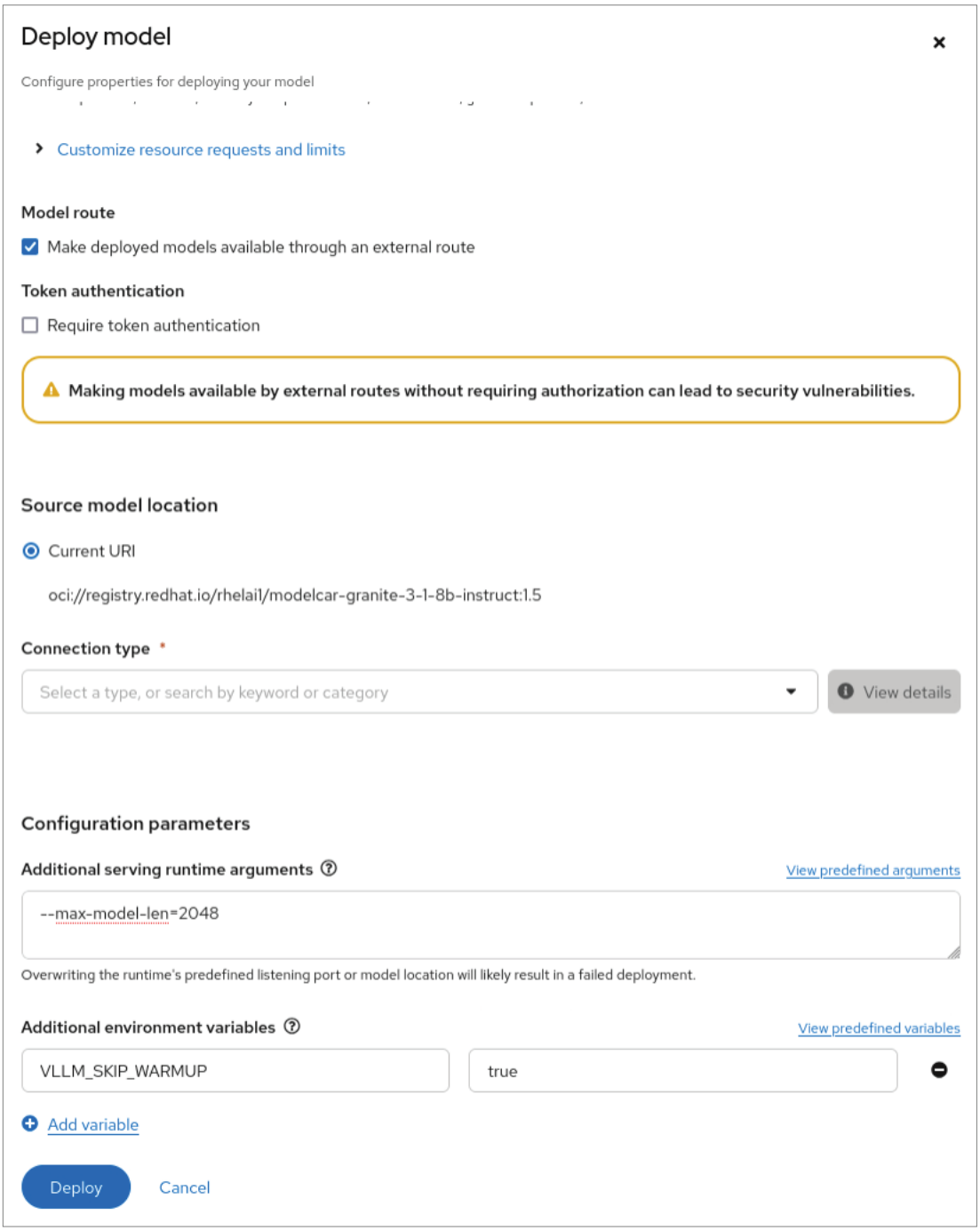
Then click on Deploy model:



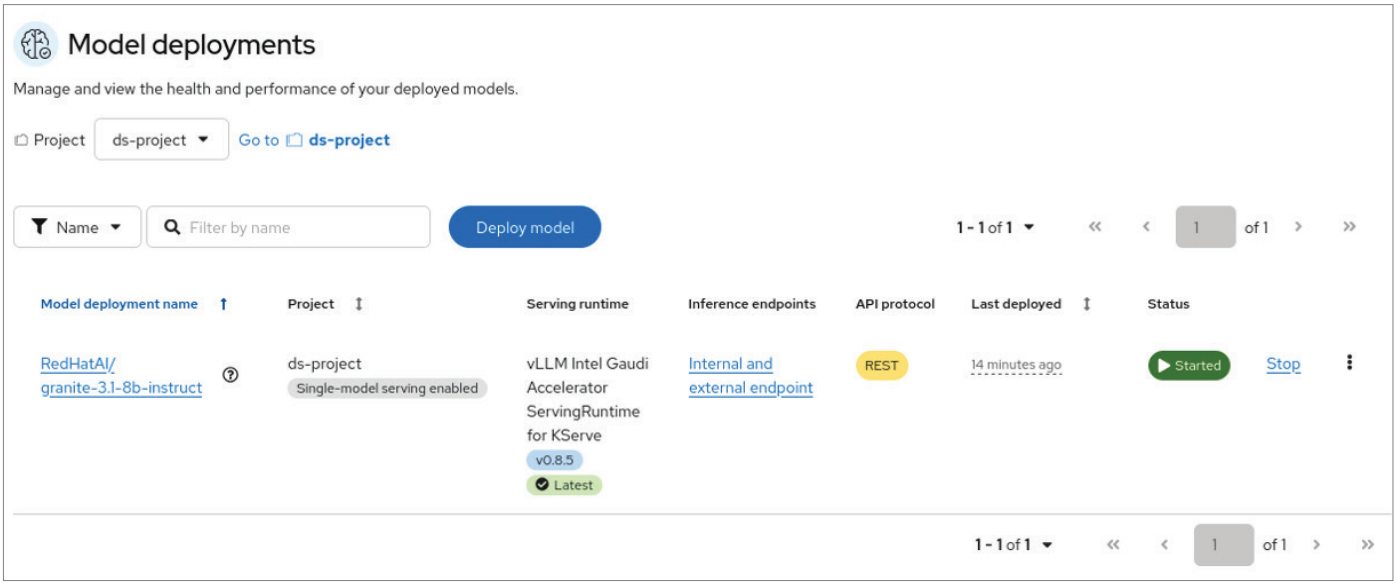
A new context window will pop up. Select ds-project we created before, then update Serving runtime to use vLLM Gaudi and make sure Hardware profile is matching the gaudi hardware profile we created before.



To expose the inference endpoint, select Model route and update any additional arguments or environment variables as shown below, and finally click on Deploy:



Once the inference model is instatiated, the web console will add this new instantiation to the list of Model deployments:



If you prefer to use the CLI, here is an example YAML file in order to deploy the inference model:

Create the project if it's not already there:

```
oc create ns ds-project
```

Apply the YAML file:

```
oc apply -f inference-service.yaml
servingruntime.serving.kserve.io/redhataigranite-31-8b-instruct created
inferenceservice.serving.kserve.io/redhataigranite-31-8b-instruct created
```

YAML file content:

```
---
apiVersion: serving.kserve.io/v1alpha1
kind: ServingRuntime
metadata:
  annotations:
    opendatahub.io/accelerator-name: ''
    opendatahub.io/apiProtocol: REST
    opendatahub.io/recommended-accelerators: '["habana.ai/gaudi"]'
    opendatahub.io/runtime-version: v0.8.5
    opendatahub.io/serving-runtime-scope: global
    opendatahub.io/template-display-name: vLLM Intel Gaudi Accelerator ServingRuntime for KServe
    opendatahub.io/template-name: vllm-gaudi-runtime
    openshift.io/display-name: redhataigranite-31-8b-instruct
  name: redhataigranite-31-8b-instruct
  namespace: ds-project
  labels:
    opendatahub.io/dashboard: 'true'
spec:
  annotations:
    prometheus.io/path: /metrics
    prometheus.io/port: '8080'
  builtInAdapter:
    modelLoadingTimeoutMillis: 90000
  containers:
    - args:
        - '--port=8080'
        - '--model=/mnt/models'
        - '--served-model-name={{.Name}}'
      command:
        - python
        - '-m'
        - vllm.entrypoints.openai.api_server
      env:
        - name: HF_HOME
          value: /tmp/hf_home
      image: 'registry.redhat.io/rhoai/odh-vllm-gaudi-rhel9@sha256:1041bb23d3078ad4767e77cf1996959e4
ab9214730bbac657741891b4fb0bba9'
      name: kserve-container
      ports:
        - containerPort: 8080
          protocol: TCP
      volumeMounts:
        - mountPath: /dev/shm
          name: shm
  multiModel: false
  supportedModelFormats:
    - autoSelect: false
      name: vLLM
```

```
  volumes:
    - emptyDir:
        medium: Memory
        sizeLimit: 2Gi
      name: shm
---
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    opendatahub.io/hardware-profile-name: gaudi
    opendatahub.io/hardware-profile-namespace: redhat-ods-applications
    openshift.io/display-name: RedHatAI/granite-3.1-8b-instruct
    serving.kserve.io/deploymentMode: RawDeployment
  name: redhataigranite-31-8b-instruct
  namespace: ds-project
  labels:
    networking.kserve.io/visibility: exposed
    opendatahub.io/dashboard: 'true'
spec:
  predictor:
    automountServiceAccountToken: false
    maxReplicas: 1
    minReplicas: 1
    model:
      args:
        - '--max-model-len=2048'
      env:
        - name: VLLM_SKIP_WARMUP
          value: 'true'
      modelFormat:
        name: vLLM
      name: ''
      resources:
        limits:
          cpu: '20'
          habana.ai/gaudi: '1'
          memory: 100Gi
        requests:
          cpu: '20'
          habana.ai/gaudi: '1'
          memory: 100Gi
      runtime: redhataigranite-31-8b-instruct
      storageUri: 'oci://registry.redhat.io/rhelai1/modelcar-granite-3-1-8b-instruct:1.5'
```

To test the inference model, a request can be made to the inference endpoint:

```
curl -s -k https://redhataigranite-31-8b-instruct-ds-project.apps.frankie.ocp.lab/v1/completions   -H
"Content-Type: application/json"   -d '{
    "model": "redhataigranite-31-8b-instruct",
    "prompt": "Create a plan for staying healthy",
    "max_tokens": "1000"
  }' | jq
{
  "id": "cmpl-e4affe1b063d4797aa417cc88d215832",
  "object": "text_completion",
  "created": 1762361490,
  "model": "redhataigranite-31-8b-instruct",
  "choices": [
    {
      "index": 0,
      "text": ", even after lockdown ends.\" \n\n\n\n1. **Maintain a Balanced Diet:** Continue
eating a variety of fruits, vegetables, lean proteins, and whole grains. Limit processed foods,
sugars, and unhealthy fats.\n\n2. **Stay Active:** Regular physical activity is crucial. Aim for
at least 150 minutes of moderate aerobic activity or 75 minutes of vigorous activity each week,
alongside strength training exercises on two or more days. Consider outdoor activities, such as
walking, running, or cycling, to enjoy fresh air and sunlight.\n\n3. **Practice Good Hygiene:**
Even after lockdown, maintain good hand hygiene by washing hands frequently with soap and water
for at least 20 seconds. Carry hand sanitizer with at least 60% alcohol for situations where
soap and water are not available.\n\n4. **Get Adequate Sleep:** Prioritize getting 7-9 hours of
quality sleep each night. Establish a regular sleep schedule and create a restful environment by
reducing noise and light exposure.\n\n5. **Manage Stress:** Engage in stress-reducing activities,
such as meditation, deep breathing exercises, or progressive muscle relaxation. Seek professional
help if stress becomes overwhelming.\n\n6. **Socialize Safely:** Gradually reintroduce social
interactions while following public health guidelines. Prioritize outdoor gatherings, maintain
physical distance, and wear masks when necessary.\n\n7. **Regular Health Check-ups:** Schedule
routine medical appointments to monitor overall health and address any concerns. This includes
preventive screenings and vaccinations.\n\n8. **Stay Informed:** Keep up-to-date with the latest
public health guidelines and recommendations to ensure safety and adjust your plan as needed.\n\
n9. **Connect with Nature:** Spend time outdoors to boost mental well-being and vitamin D levels.
Participate in activities like gardening, park visits, or outdoor sports.\n\n10. **Maintain Mental
Health:** Stay connected with friends, family, and support networks through virtual means if in-
person interactions are not feasible. Seek professional mental health services if needed.\n\n11.
**Limit Alcohol and Avoid Tobacco:** Reduce alcohol consumption and avoid tobacco products to
promote overall health and well-being.\n\n12. **Continuous Learning:** Stay curious and open to
learning new skills or hobbies that promote mental and physical well-being, such as yoga, tai chi,
or cooking.",
      "logprobs": null,
      "finish_reason": "stop",
      "stop_reason": null,
      "prompt_logprobs": null
    }
  ],
  "usage": {
    "prompt_tokens": 7,
    "total_tokens": 601,
    "completion_tokens": 594,
    "prompt_tokens_details": null
  }
}
```

5.2 Running a fine-tuning job

OpenShift AI is a versatile platform for managing and scaling machine learning workloads across different frameworks and hardware configurations. Below is an MPIJob configuration for fine-tuning the Llama 3.1 8B model using OpenShift AI. This setup uses DeepSpeed and Habana Intel Gaudi 3 processors for efficient distributed training within the node.

This example can be easily modified to scale the fine-tuning job on multiple nodes. All that you need to do is update the replicas.

```
apiVersion: kubeflow.org/v1
kind: MPIJob
metadata:
  name: gaudi-llm-ds-ft-multi-8b
  namespace: default
spec:
  slotsPerWorker: 8
  runPolicy:
    cleanPodPolicy: Running
  launcherCreationPolicy: WaitForWorkersReady
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
        spec:
          containers:
            - image: vault.habana.ai/gaudi-docker/1.22.1/ubuntu24.04/habanalabs/pytorch-installer-
2.7.1:latest
              name: gaudi-llm-ds-ft-multi-8b-no-ssh-launcher
              env:
                - name: HF_HOME
                  value: "/root/.cache/huggingface"
                - name: no_proxy
                  value: "localhost,127.0.0.1"
                - name: LLM_MODEL
                  value: meta-llama/Meta-Llama-3.1-8B-Instruct
                - name: NUM_HPU
                  value: "8"
                - name: NUM_EPOCHS
                  value: "3"
                - name: HUGGING_FACE_HUB_TOKEN
                  value: ""
                - name: http_proxy
                  value: ""
                - name: https_proxy
                  value: ""
                - name: "MPI_ROOT"
                  value: "/opt/amazon/openmpi"
                - name: "OPAL_PREFIX"
                  value: "/opt/amazon/openmpi"
                - name: "MPICC"
                  value: "/opt/amazon/openmpi/bin/mpicc"
              command: ["/bin/bash", "-c"]
              args:
                - >-
```

```
                  /usr/bin/ssh-keygen -A;
                  echo "    KexAlgorithms curve25519-sha256,sntrup761x25519-sha512@openssh.com" |
tee -a /etc/ssh/ssh_config;
                  /usr/sbin/sshd;

                  PATH=$MPI_ROOT/bin:$PATH;
                  HOSTSFILE=$OMPI_MCA_orte_default_hostfile;
                  MASTER_ADDR="$(head -n 1 $HOSTSFILE | sed -n s/[[:space:]]slots.*//p)";
                  export no_proxy=$no_proxy,$KUBERNETES_SERVICE_HOST;
                  NUM_NODES=$(wc -l < $HOSTSFILE);
                  N_CARDS=$((NUM_NODES*NUM_HPU));

                  mpirun --npernode 1 \
                    --tag-output \
                    --allow-run-as-root \
                    --prefix $MPI_ROOT \
                    -x http_proxy=$http_proxy \
                    -x https_proxy=$https_proxy \
                    -x no_proxy=$no_proxy \
                    -x HF_HOME=$HF_HOME \
                    -x HUGGING_FACE_HUB_TOKEN=$HUGGING_FACE_HUB_TOKEN \
                    -x PT_HPU_MAX_COMPOUND_OP_SIZE=10 \
                    -x DEEPSPEED_HPU_ZERO3_SYNC_MARK_STEP_REQUIRED=1 \
                    bash -i -c '
                      git clone https://github.com/huggingface/optimum-habana /optimum-habana
                      cd /optimum-habana && git checkout v1.19.1
                      pip install .
                      pip install -r examples/language-modeling/requirements.txt
                      pip install git+https://github.com/HabanaAI/DeepSpeed.git@1.19.0
                    ';

                  MODEL_PATH=/optimum-habana/examples;
                  FINE_TUNE_CMD="python3 $MODEL_PATH/language-modeling/run_lora_clm.py \
                        --model_name_or_path $LLM_MODEL \
                        --dataset_name tatsu-lab/alpaca \
                        --bf16 True \
                        --output_dir ./model_lora_llama_ddp \
                        --num_train_epochs $NUM_EPOCHS \
                        --per_device_train_batch_size 8 \
                        --gradient_accumulation_steps 2 \
                        --evaluation_strategy "no" \
                        --save_strategy "no" \
                        --learning_rate 3e-4 \
                        --warmup_ratio  0.03 \
                        --lr_scheduler_type "constant" \
                        --max_grad_norm  0.3 \
                        --logging_steps 1 \
                        --do_train \
                        --do_eval \
                        --use_habana \
                        --use_lazy_mode \
                        --throughput_warmup_steps 3 \
```

```
                    --lora_rank=8 \
                    --lora_alpha=16 \
                    --lora_dropout=0.05 \
                    --lora_target_modules q_proj v_proj \
                    --dataset_concatenation \
                    --max_seq_length 512 \
                    --ddp_bucket_cap_mb 50 \
                    --adam_epsilon 1e-08 \
                    --validation_split_percentage 4 \
                    --low_cpu_mem_usage True";

                cd $MODEL_PATH;
                mpirun -np ${N_CARDS} \
                  --allow-run-as-root \
                  --bind-to core \
                  --map-by ppr:4:socket:PE=6 \
                  -rank-by core --report-bindings \
                  --tag-output \
                  --merge-stderr-to-stdout --prefix $MPI_ROOT \
                  -x RDMAV_FORK_SAFE=1 \
                  -x FI_EFA_USE_DEVICE_RDMA=1 \
                  -x MASTER_ADDR=$MASTER_ADDR \
                  -x http_proxy=$http_proxy \
                  -x https_proxy=$https_proxy \
                  -x no_proxy=$no_proxy \
                  -x LLM_MODEL=$LLM_MODEL \
                  -x HF_HOME=$HF_HOME \
                  -x HUGGING_FACE_HUB_TOKEN=$HUGGING_FACE_HUB_TOKEN \
                  -x NUM_EPOCHS=$NUM_EPOCHS \
                  $FINE_TUNE_CMD;

            volumeMounts:
              - name: datasets
                mountPath: /root/.cache
        volumes:
          - name: datasets
            persistentVolumeClaim:
              claimName: shared-model
              readOnly: false
    Worker:
      replicas: 1
      template:
        spec:
          hostIPC: true
          containers:
            - image: vault.habana.ai/gaudi-docker/1.22.1/ubuntu24.04/habanalabs/pytorch-installer-
2.7.1:latest
              name: gaudi-llm-ds-ft-multi-8b-no-ssh-worker
              command: ["/bin/bash", "-c"]
              args:
                - >-
                  /usr/bin/ssh-keygen -A;
```

```
            /usr/sbin/sshd;
            sleep 365d;
        resources:
          limits:
            habana.ai/gaudi: 8
            memory: 250Gi
            hugepages-2Mi: 9800Mi
          requests:
            habana.ai/gaudi: 8
            memory: 250Gi
            hugepages-2Mi: 9800Mi
        volumeMounts:
          - name: datasets
            mountPath: /root/.cache
      volumes:
        - name: datasets
          persistentVolumeClaim:
            claimName: shared-model
            readOnly: false
```

6 Monitoring and observability

Intel Gaudi 3 accelerators support both in-band metrics monitoring using a Prometheus exporter and out-of-band monitoring through a server's BMC Redfish endpoints.

6.1 Red Hat OpenShift Observability

Observability is a comprehensive set of capabilities that provides deep insights into the performance and health of OpenShift-based applications and infrastructure across any footprint: the public cloud, on-premises and edge. Red Hat OpenShift Observability provides real-time visibility, monitoring and analysis of various system metrics, logs, traces and events to quickly diagnose and troubleshoot issues before they impact applications or end users. It streamlines metrics, traces and logs, while aggregating and transporting data. Red Hat OpenShift Observability allows you to gain visibility into your clusters through efficient user interfaces and empowers your teams to make data-driven decisions.

6.2 Intel Gaudi Prometheus Exporter (in-band)

These metrics are available to OpenShift cluster end users.[38] When the Intel Gaudi Operator is installed, it automatically installs a metrics exporter in the habana-ai-operator namespace. The endpoint and Prometheus scrapers are automatically created.

6.2.1 Prometheus metrics using curl

The easiest way to verify metrics is to use a pod that already has curl in it. The habana-ai-device-plugin pod in the same namespace as the metric exporter can be used for this. Below is a single command that will list out all of the metrics by curling the metric-exporter endpoint.

```
$ metrics_url=$(oc get ep habana-ai-metric-exporter-svc -n habana-ai-operator --no-headers | awk
'{print $2}') &&
oc exec -it $(oc get pods -n habana-ai-operator --no-headers | grep '^habana-ai-device-plugin-ds-'
| awk '{print $1}') -n habana-ai-operator -- curl http://${metrics_url}/metrics
```

6.2.2 Visualizing Prometheus metrics (OpenShift web console)

Red Hat OpenShift comes with observability capabilities that can be accessed through the Dashboard "Observe" view. This view provides a set of built-in dashboards and available cluster metrics. You can check Intel Gaudi metrics using the Metrics tab, as shown in Figure 8.

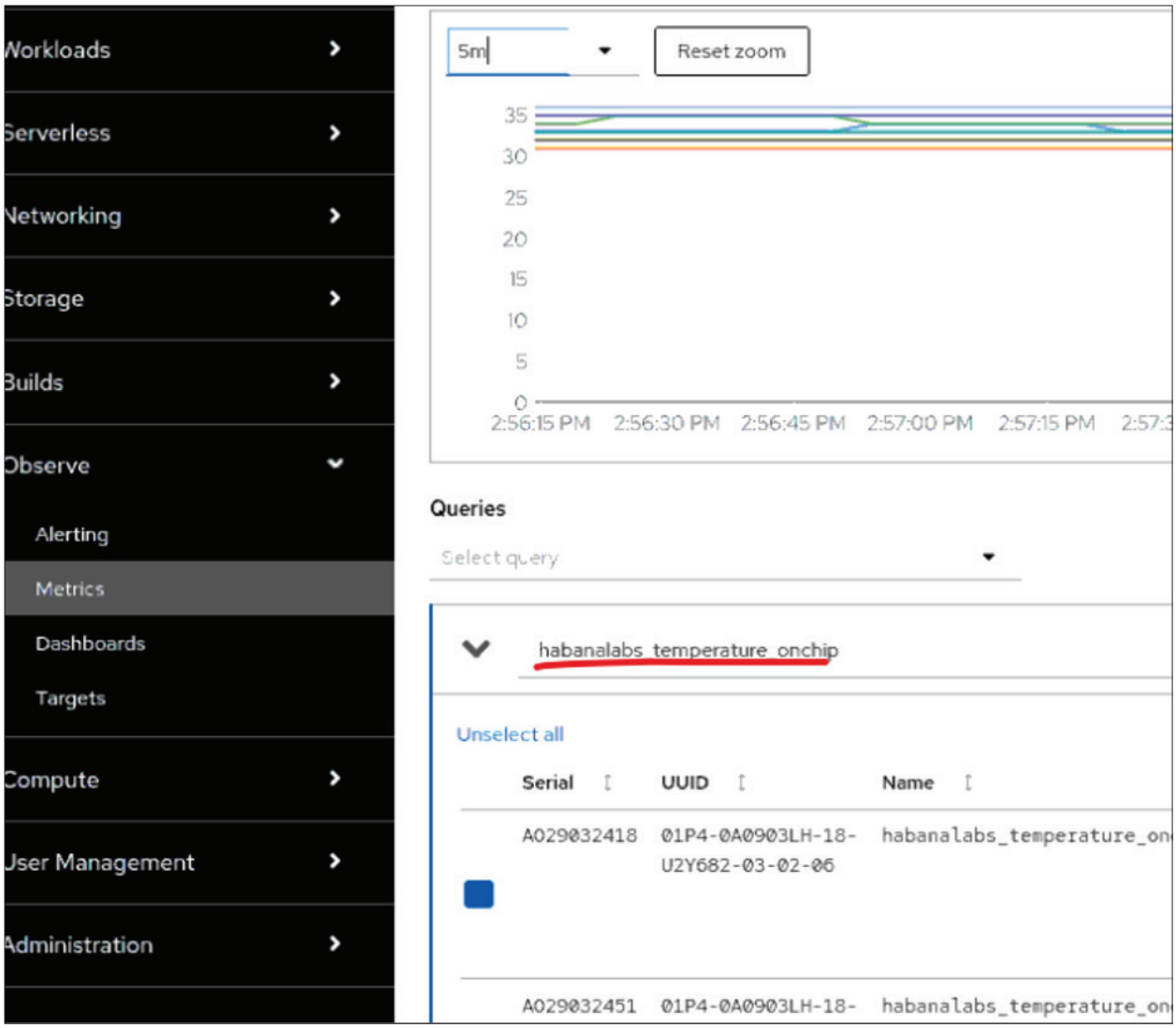


Figure 8. Check Intel Gaudi metrics using the Metrics tab

Red Hat and Intel are working towards an integrated dashboard that will be available in the near future. Today, there is not a way to create your own dashboards, so there is an alternative "Grafana" (see below) where custom dashboards can be created and Intel Gaudi 3 accelerator metrics integrated.

6.2.3 Visualizing Prometheus metrics (Grafana)

Grafana is an open-source dashboard visualization tool that displays metrics from various data sources, such as Prometheus. It can be integrated with Alertmanager, which manages alerts and notifies users of potential issues.

Grafana is a community-supported operator on OpenShift. Install the Grafana Operator and create a Grafana Instance. If you don't want to lose the Grafana configuration when the pod restarts or node is rebooted, then make sure to configure the Grafana instance with a persistentVolumeClaim. Make sure to note the admin_password and admin_user to use later when logging in.

To create a Grafana instance, you can check the example below (make sure you update username and password):

Community

Grafana Operator
provided by Grafana Labs

Deploys and manages Grafana instances, dashboards and data sources

```yaml
apiVersion: grafana.integreatly.org/v1beta1
kind: Grafana
metadata:
  labels:
    dashboards: grafana-a
    folders: grafana-a
  name: grafana-a
  namespace: openshift-operators
spec:
  config:
    auth:
      disable_login_form: 'false'
    log:
      mode: console
    security:
      admin_password: <pass>
      admin_user: <user>
  persistentVolumeClaim:
    metadata: {}
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 10Gi
      storageClassName: lvms-vg1
  route:
    metadata: {}
    spec:
      to:
        kind: Service
        name: grafana-a-service
        weight: 100
```

Once Grafana is created, make sure to apply this policy:

```
oc adm policy add-cluster-role-to-user cluster-monitoring-view -z grafana-a-sa
```

Use the following command to get a Prometheus Bearer token:

```
oc create token prometheus-k8s --duration=8760h -n openshift-monitoring
```

A Prometheus data source needs to be connected. You will need to first get a token to automate the login to Prometheus.

You can use the Grafana UI to add Prometheus as a Datasource: Connections --> Data sources --> Add new data source. Select Prometheus and configure it.

Under Connection use the following url:

https://thanos-querier.openshift-monitoring.svc.cluster.local:9091

in TLS settings use Skip TLS certificate validation

Alternatively, you can use this YAML file to do that.

```yaml
apiVersion: grafana.integreatly.org/v1beta1
kind: GrafanaDatasource
metadata:
  name: grafanadatasource
  namespace: openshift-operators
spec:
  allowCrossNamespaceImport: true
  datasource:
    access: proxy
    isDefault: true
    jsonData:
      httpHeaderName1: Authorization
      timeInterval: 5s
      tlsSkipVerify: true
    name: prometheus
    secureJsonData:
      httpHeaderValue1: Bearer <insert token here>
    type: prometheus
    url: 'https://thanos-querier.openshift-monitoring.svc.cluster.local:9091'
  instanceSelector:
    matchLabels:
      dashboards: grafana-a
  plugins:
    - name: grafana-clock-panel
      version: 1.3.0
  resyncPeriod: 5m
```

When creating a dashboard, all the Intel Gaudi metrics are prefaced with "habanalabs_*." Figure 9 shows an example of an Intel Gaudi dashboard showing eight different Intel Gaudi accelerators, identified by their unique serial numbers. These serial numbers are what shows up in the "curl" output from above.

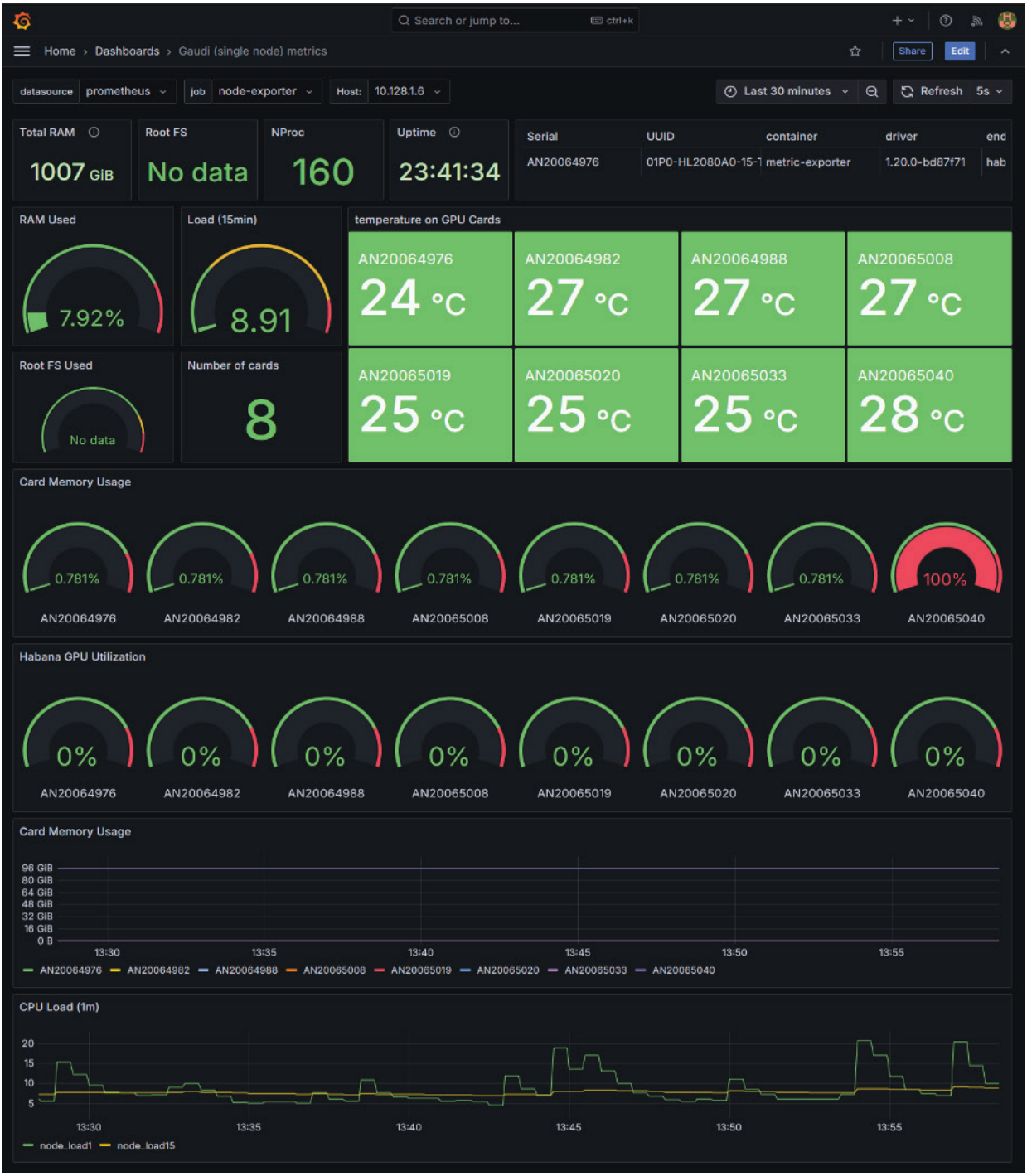


Figure 9. Intel Gaudi dashboard

If you need to create a custom dashboard, the following guide explains how to make your own dashboard using Grafana: https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-prometheus/.

6.3 Intel Gaudi BMC exporter (out-of-band)

Baseboard Management Controllers (BMC) are embedded processors on server motherboards that provide remote monitoring and management capabilities. The BMC exporter extracts and exposes data from a BMC and translates the BMC data into a format that Prometheus can understand, allowing for monitoring of hardware health. The BMC exporter uses Redfish protocol to transmit commands during each data scrape. With Redfish, users can interact with systems through standard web services, like HTTP and REST APIs. A table of endpoint categories and instructions for using the BMC exporter can be found in: https://docs.habana.ai/en/latest/Orchestration/BMC_Exporter/BMC_Exporter.html.

7 Performance optimization

Achieving peak performance for AI workloads is crucial for maximizing the capabilities of your hardware. This section delves into the essentials of performance optimization, focusing on Intel Xeon processors and Intel Gaudi accelerators. While detailed instructions are reserved for separate documentation, here you will find links to best practices and performance benchmarks, along with resources for troubleshooting bottlenecks and optimizing models.

7.1 Performance benchmarking

Intel publishes updated performance figures for popular AI models running on Intel hardware:

- **Intel Xeon Processors:**
  https://edc.intel.com/content/www/us/en/products/performance/benchmarks/intel-xeon-6/
- **Intel Gaudi Accelerators:**
  https://www.intel.com/content/www/us/en/developer/platform/gaudi/model-performance.html

7.1.1 MLPerf

MLCommons publishes MLPerf numbers for Intel hardware:

- **Intel Xeon Processors:** MLPerf results for Intel Xeon hardware and can be found at:
  https://mlcommons.org/benchmarks/inference-datacenter/
- **Intel Gaudi Accelerators:** MLPerf numbers for Intel Gaudi 2 accelerators were published in MLPerf V4 at:
  https://mlcommons.org/benchmarks/inference-datacenter/. (Intel Gaudi3 accelerator results will be published soon).

7.2 Model optimization

- **Intel Xeon Processors:**
  - https://www.intel.com/content/www/us/en/developer/articles/technical/tuning-guide-for-ai-on-the-4th-generation.html
- **Intel Gaudi Accelerators:**
  https://docs.habana.ai/en/latest/PyTorch/Model_Optimization_PyTorch/index.html

7.3 PyTorch optimization

- **Intel Xeon Processors:**
  - https://docs.pytorch.org/tutorials/recipes/xeon_run_cpu.html
  - https://www.intel.com/content/www/us/en/developer/articles/technical/tuning-guide-for-ai-on-the-4th-generation.html#inpage-nav-5
- **Intel Gaudi Accelerators:**
  - https://docs.habana.ai/en/latest/PyTorch/Model_Optimization_PyTorch/Optimization_in_PyTorch_Models.html
  - https://docs.habana.ai/en/latest/PyTorch/Model_Optimization_PyTorch/Dynamic_Shapes.html

7.4 Debugging

- **Intel Xeon Processors:**
  https://www.intel.com/content/www/us/en/developer/articles/guide/xeon-performance-tuning-and-solution-guides.html
- **Intel Gaudi Accelerators:**
  https://docs.habana.ai/en/latest/PyTorch/Reference/Debugging_Guide/index.html

8 References and additional resources

8.1 Product links

- **Intel Xeon Processors:**
  https://www.intel.com/content/www/us/en/products/details/processors/xeon.html
- **Intel Gaudi Accelerators:**
  https://www.intel.com/content/www/us/en/products/details/processors/ai-accelerators/gaudi.html
- **Red Hat OpenShift AI:**
  https://www.intel.com/content/www/us/en/products/details/processors/ai-accelerators/gaudi.html

8.2 Documentation links

- **Intel Xeon Processors:**
  https://www.intel.com/content/www/us/en/products/details/processors/xeon.html
- **Intel Gaudi Accelerators:**
  https://docs.habana.ai/en/latest/
- **Red Hat OpenShift AI:**
  https://docs.redhat.com/en/documentation/red_hat_openshift_ai_self-managed/3.0

8.3 Official support

- **Intel Xeon Processors:**
  https://www.intel.com/content/www/us/en/support/products/873/processors.html
- **Intel Gaudi Accelerators:**
  https://www.intel.com/content/www/us/en/support.html
- **Red Hat OpenShift AI:**
  https://access.redhat.com/support

8.4 Community support

- **Intel Xeon Processors:**
  https://community.intel.com/t5/Intel-Xeon-Processor-and-Server/bd-p/server-products
- **Intel Gaudi Accelerators:**
  https://community.intel.com/t5/Intel-Gaudi-AI-Accelerator/bd-p/IntelGaudiAIAccelerator
- **Red Hat OpenShift AI:**
  https://commons.openshift.org/

9 References

[1] [Online]. Available: https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/security-engines.html.
[2] [Online]. Available: https://community.intel.com/t5/Blogs/Tech-Innovation/Cloud/Part-1-Unlocking-AI-ML-Potential-with-Intel-Xeon-Based-CPUs/post/1533754.
[3] [Online]. Available: https://github.com/intel/intel-extension-for-pytorch.
[4] [Online]. Available: https://www.intel.com/content/www/us/en/products/details/processors/ai-accelerators/gaudi.html.
[5] [Online]. Available: https://www.intel.com/content/www/us/en/developer/platform/gaudi/develop/overview.html.
[6] [Online]. Available: https://www.intel.com/content/www/us/en/developer/platform/gaudi/model-performance.html.
[7] [Online]. Available: https://docs.habana.ai/en/latest/PyTorch/PyTorch_Main.html.
[8] [Online]. Available: https://www.intel.com/content/www/us/en/products/details/processors/ai-accelerators/gaudi.html.
[9] [Online]. Available: https://huggingface.co/blog/intel-gaudi-backend-for-tgi and https://vllm-gaudi.readthedocs.io/en/latest/index.html.
[10] [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-ai-solutions-support-llama-4-release.html.
[11] [Online]. Available: https://www.intel.com/content/www/us/en/developer/platform/gaudi/models/catalog.html?f:curated=Large%20Language%20Models.
[12] [Online]. Available: https://huggingface.co/docs/optimum/habana/index.
[13] [Online]. Available: https://docs.habana.ai/en/latest/PyTorch/vLLM_Inference/vLLM_FAQs.html.
[14] [Online]. Available: https://docs.nvidia.com/dgx/dgxh100-user-guide/introduction-to-dgxh100.html.
[15] [Online]. Available: https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/gpu-accelerated/nd-h200-v5-series.
[16] [Online]. Available: https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2024-05/intel-xeon-6-product-brief.pdf.
[17] [Online]. Available: https://newsroom.intel.com/artificial-intelligence/new-intel-xeon-6-cpus-maximize-gpu-ai-performance.
[18] [Online]. Available: https://www.intel.com/content/www/us/en/products/details/processors/xeon/xeon6-p-cores.html.
[19] [Online]. Available: https://newsroom.intel.com/data-center/new-ultrafast-memory-boosts-intel-data-center-chips.
[20] [Online]. Available: https://developers.redhat.com/products/red-hat-openshift-ai/overview.
[21] [Online]. Available: https://www.solutionpatterns.io/solution-pattern-predict-stock-price-trend-using-rhoai/solution-pattern-predict-stock-price-trend-using-rhoai/02-architecture.html.
[22] [Online]. Available: https://docs.redhat.com/en/documentation/red_hat_openshift_ai_cloud_service/1/html-single/getting_started_with_red_hat_openshift_ai_cloud_service/index.
[23] [Online]. Available: https://developers.redhat.com/articles/2024/08/06/red-hat-openshift-ai-and-machine-learning-operations#architecture_of_red_hat_openshift_ai.
[24] [Online]. Available: https://catalog.redhat.com/en/software/container-stacks/detail/6683b2cce45daa25e36bddcb
[25] [Online]. Available: https://docs.redhat.com/en/documentation/red_hat_openshift_pipelines/1.18/html/installing_and_configuring/installing-pipelines.
[26] [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift/pipelines.
[27] [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift/what-is-openshift-service-mesh.
[28] [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift/serverless.
[29] [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift/dedicated.
[30] [Online]. Available: https://docs.redhat.com/en/documentation/openshift_dedicated/4/html/introduction_to_openshift_dedicated/osd-understanding.
[31] [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift/aws.
[32] [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift.
[33] [Online]. Available: https://opendatahub.io/.
[34] [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift/features.
[35] [Online]. Available: https://cdrdv2-public.intel.com/833842/gaudi-3-ai-accelerator-cluster-ref-design-white-paper.pdf
[36] [Online]. Available: https://docs.redhat.com/en/documentation/red_hat_openshift_ai_self-managed/2.19/html/installing_and_uninstalling_openshift_ai_self-managed/installing-the-single-model-serving-platform_component-install#configuring-automated-installation-of-kserve_componen.
[37] [Online]. Available: https://www.redhat.com/en/topics/microservices/what-is-istio.
[38] [Online]. Available: https://docs.habana.ai/en/latest/Orchestration/Prometheus_Metric_Exporter.html.

10 Notices & Disclaimers

1126/RM/MESH/PDF 360859-001US