White paper
Cisco public

The bridge to possible

Run:ai on Cisco UCS Converged Infrastructure

Contents

Run:ai on Cisco UCS Converged Infrastructure optimizes AI workloads by providing a scalable and flexible platform that enhances resource utilization and accelerates time-to-insight with advanced orchestration capabilities.

Introduction

The integration of Run:ai with Cisco UCS® (Cisco Unified Computing System™) infrastructure offers an innovative approach to resource management for Artificial-Intelligence (AI) and Machine-Learning (ML) workloads. Run:ai provides a GPU-orchestration framework for AI and ML workloads that improves resource allocation and utilization by simplifying and automating the process. Cisco UCS provides an optimal environment for running data-intensive workloads by delivering high-performance computing power and simplifying the management of resources. Together, they deliver a solution that enhances the efficiency and productivity of data scientists and researchers.

This white paper explores the potential of combining Run:ai's advanced orchestration capabilities with the efficient and easy-to-automate Cisco UCS infrastructure. The paper provides an in-depth look into how this integration can revolutionize the way organizations handle AI and ML workloads. The combination ensures optimal hardware utilization, maximizes performance, and simplifies resource management – all crucial aspects in scaling AI and ML operations effectively and efficiently. This white paper further explores real-world use cases, illustrating the benefits and possibilities of using Run:ai with Cisco UCS.

Solution benefits

Run:ai is an AI orchestration platform that offers effective solutions for managing and streamlining AI workflows. When integrated with OpenShift on Cisco UCS X-Series, Run:ai can help optimize AI and machine-learning workloads. OpenShift, a Kubernetes-based platform, provides the perfect environment for deploying and managing Run:ai, enabling containerization and automation of AI workloads. Cisco UCS X-Series, a highly scalable and flexible modular computing platform, provides the necessary computing power and capacity to handle resource-intensive AI tasks.

The integration of Run:ai with OpenShift on Cisco UCS X-Series offers a holistic solution for AI workload management. It allows organizations to dynamically allocate resources, simplify workload management, and accelerate AI research. With Run:ai, enterprises can efficiently prioritize tasks, ensure optimal utilization of resources, and reduce operational costs.

Key features and benefits include:

- Fully utilized compute: **GPU scheduling**, **GPU Quota Management**, **Fractional GPU Sharing**, and **Dynamic MIG** (multi-instance GPU). Run:ai's platforms can help you better utilize resources in your infrastructure, on premises and in the cloud.
- Enterprise visibility: **real-time and historical metrics** by job, workload, and team in a single dashboard. Assign **compute guarantees** to critical workloads, **promote oversubscription**, and react to business needs easily.
- Central policy control: built-in **identity management** system integration, and a policies mechanism, allow you to control which team has access to which resources, create **node pools**, and **manage risk**.
- Zero-touch resources: promote practitioner productivity with the Run:ai GUI. Run:ai makes it simple for a practitioner to access compute and run workloads without being a technical expert. **Workspaces** and **templates** were built with end users in mind.

- Tool flexibility: provide flexibility to practitioners to **integrate** experiment-tracking tools and **development frameworks**. With Run:ai's rich integration options, you can work with your favorite ML stack right away.
- Cloud-like elasticity: Run:ai's Scheduler assures near on-demand access to GPUs from a finite resource pool. **Dynamic MIG** and **Fractional GPU Sharing** give you full flexibility when more GPU power is needed.

Furthermore, the solution can scale on demand, providing flexibility and agility to handle varying workload volumes. This combination of technologies provides a robust, scalable, and efficient solution for AI orchestration.

Solution design

The solution design for using Run:ai on OpenShift involves installing and integrating Run:ai into an OpenShift environment. The first step entails setting up and configuring OpenShift, a Kubernetes-based platform, to provide the infrastructure for the deployment of Run:ai. OpenShift offers a robust, scalable, and secure platform to run containerized applications, making it ideal for executing AI and ML workloads. OpenShift's built-in automation capabilities also make it easier to manage the lifecycle of Run:ai.

OpenShift installation

To install OpenShift on Cisco UCS infrastructure with GPUs, you can refer to one of the Cisco UCS validated designs such as FlashStack for Generative AI Inferencing or FlexPod Datacenter with Generative AI Inferencing. The FlashStack design features the Cisco UCS X-Series Modular System managed from Cisco Intersight® running Red Hat OpenShift with NVIDIA GPUs and Portworx Enterprise backed by Pure Storage FlashArray and FlashBlade. The FlexPod design also uses Cisco UCS X-Series managed from Cisco Intersight with NVIDIA GPUs and NetApp ONTAP on the NetApp AFF A800 with NetApp Astra Trident for persistent storage, and the latest release of the Red Hat OpenShift Container Platform (OCP).

After OpenShift is up and running, the next step is to install Run:ai. Run:ai's platform is designed to simplify the management of AI workloads, allowing for efficient scheduling, prioritizing, and execution of tasks. The installation process involves deploying the Run:ai operator on the OpenShift platform. Detailed instructions on how to install and use Run:ai can be found at https://docs.run.ai/latest/admin/runai-setup/cluster-setup/cluster-install/.

Run:ai prerequisites

If you have performed the OpenShift installation following the instructions in the FlashStack or FlexPod Cisco Validated Design, most prerequisites for the Run:ai installation have already been met, including the following:

- Kubernetes (as part of the OpenShift Container Platform)
- NVIDIA GPU Operator
- Ingress Controller
- Prometheus

You will need to set up a self-hosted installation on OpenShift following the detailed instructions at https://docs.run.ai/v2.16/admin/runai-setup/self-hosted/ocp/prerequisites/ . One of the prerequisites for a self-hosted installation is that OpenShift must be configured with a trusted certificate. Run:ai installation relies on OpenShift to create certificates for subdomains. If you cannot use a trusted certificate authority, you can use a local certificate authority, and instructions are provide at https://docs.run.ai/latest/admin/runai-setup/config/org-cert/. If you're using a local certificate authority (CA), you will need to add the "--set global.customCA.enabled=true" to the helm commands as you perform the installation. When using a local CA, you need to create a secret for the runai-backend as documented at https://docs.run.ai/latest/admin/runai-setup/self-hosted/k8s/backend/#domain-certificate. Details on creating a TLS secret are at https://kubernetes.github.io/ingress-nginx/user-guide/tls/, and the domain name to use should be runai.apps.<OPENSHIFT-CLUSTER-DOMAIN> (for example, runai.apps.ai-inferencing-cluster.cisco.com). Once the self-signed certificate and private key are created, you will add these to Run:ai's backend namespace with the following command:

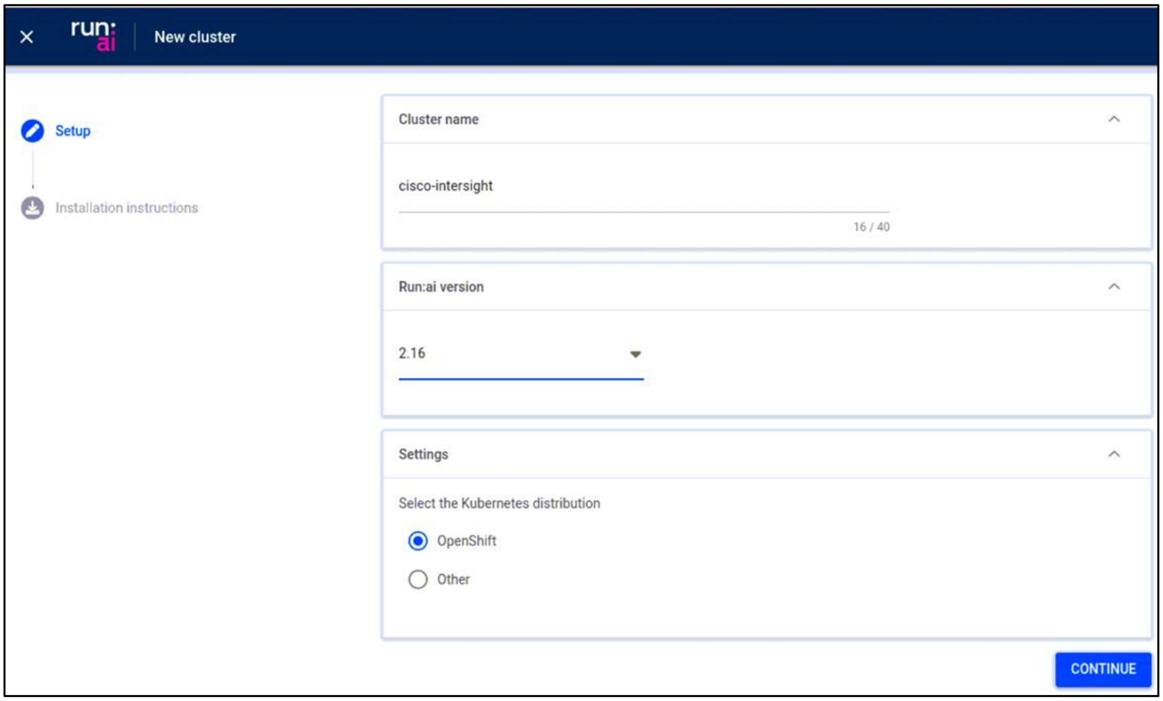
```
kubectl create ns runai-backend

kubectl create secret tls runai-backend-tls -n runai-backend \

   --cert /path/to/fullchain.pem \

   --key /path/to/private.pem
```

As part of a self-hosted deployment, you will install the Run:ai control plane as described at https://docs.run.ai/v2.16/admin/runai-setup/self-hosted/ocp/backend/. To install the control plane, a command similar to the following is needed (note that the flag "--set global.customCA.enabled=true" is only needed if you are using a local certificate authority):

```
helm repo add runai-backend https://runai.jfrog.io/artifactory/cp-charts-prod

helm repo update

helm upgrade -i runai-backend -n runai-backend runai-backend/control-plane \

      -- set global.domain=runai.apps.ai-inferencing-cluster.cisco.com\

      --set global.config.kubernetesDistribution=openshift \

      --set global.customCA.enabled=true
```

Run:ai cluster installation

Once the prerequisites are met, you can log in to your self-hosted Run:ai tenant (runai.apps.ai-inferencing-cluster.cisco.com) and create a “New cluster” in the Clusters menu. OpenShift and other versions of software used in this white paper are described in the FlashStack Cisco Validated Design or FlexPod Cisco Validated Design, and the Run:ai version used is 2.16.



Cluster name

cisco-intersight

16 / 40

Run:ai version

2.16

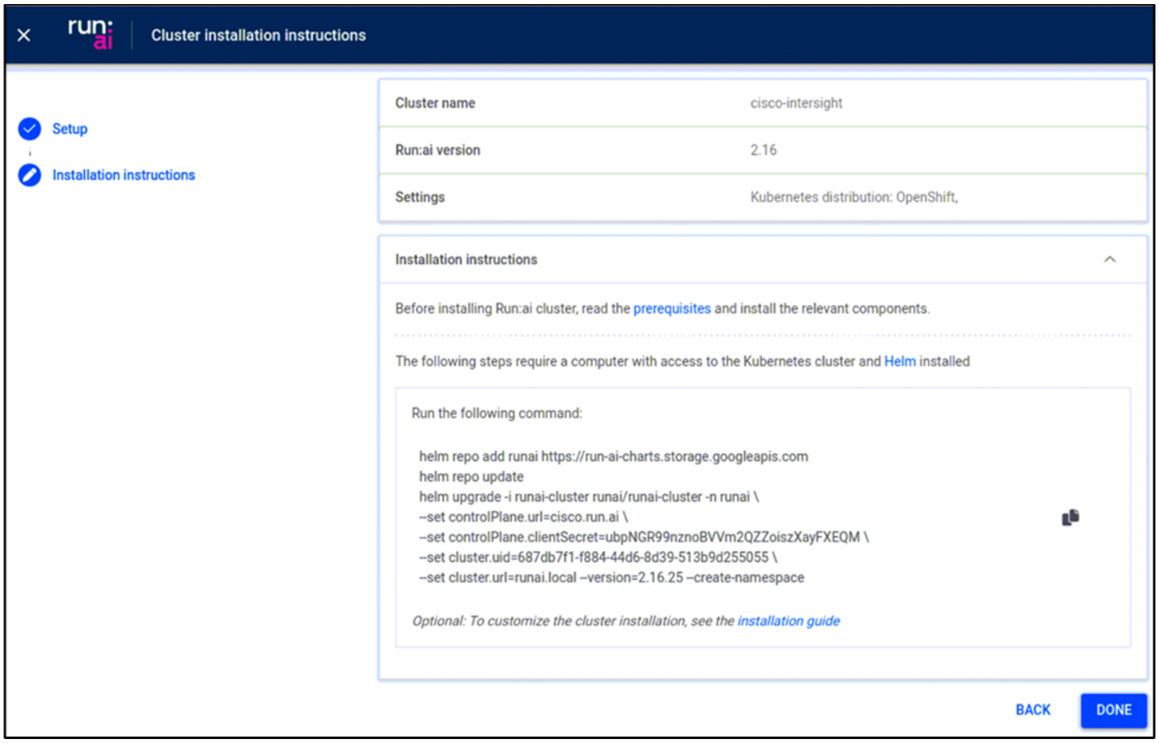
Settings

Select the Kubernetes distribution

● OpenShift

○ Other

Once a cluster name, version, and other settings are provided, you will be given installation instructions.



Cluster name	cisco-intersight
Run:ai version	2.16
Settings	Kubernetes distribution: OpenShift,

Installation instructions

Before installing Run:ai cluster, read the prerequisites and install the relevant components.

The following steps require a computer with access to the Kubernetes cluster and Helm installed

Run the following command:

```
helm repo add runai https://run-ai-charts.storage.googleapis.com
helm repo update
helm upgrade -i runai-cluster runai/runai-cluster -n runai \
--set controlPlane.url=cisco.run.ai \
--set controlPlane.clientSecret=ubpNGR99nznoBVVm2QZZoiszXayFXEQM \
--set cluster.uid=687db7f1-f884-44d6-8d39-513b9d255055 \
--set cluster.url=runai.local --version=2.16.25 --create-namespace
```

Optional: To customize the cluster installation, see the installation guide

Helm can be used for the cluster installation (controlPlane.url and cluster.url will need to be customized for your environment). Note that the flag "--set global.customCA.enabled=true" is only needed if you are using a local certificate authority.

```
helm repo add runai https://run-ai-charts.storage.googleapis.com

helm repo update

    helm upgrade -i runai-cluster runai/runai-cluster -n runai \

    --set controlPlane.url=runai.apps.ai-inferencing-cluster.cisco.com \

    --set controlPlane.clientSecret=pQCzH4kMXpNx7Wkei7Cet9r2BTwOIHox \

    --set cluster.uid=edcd477e-f183-4297-8446-5923c9843d12 \

    --set cluster.url=runai.apps.ai-inferencing-cluster.cisco.com \

    --version=2.16.11 --create-namespace \

    --set global.customCA.enabled=true
```

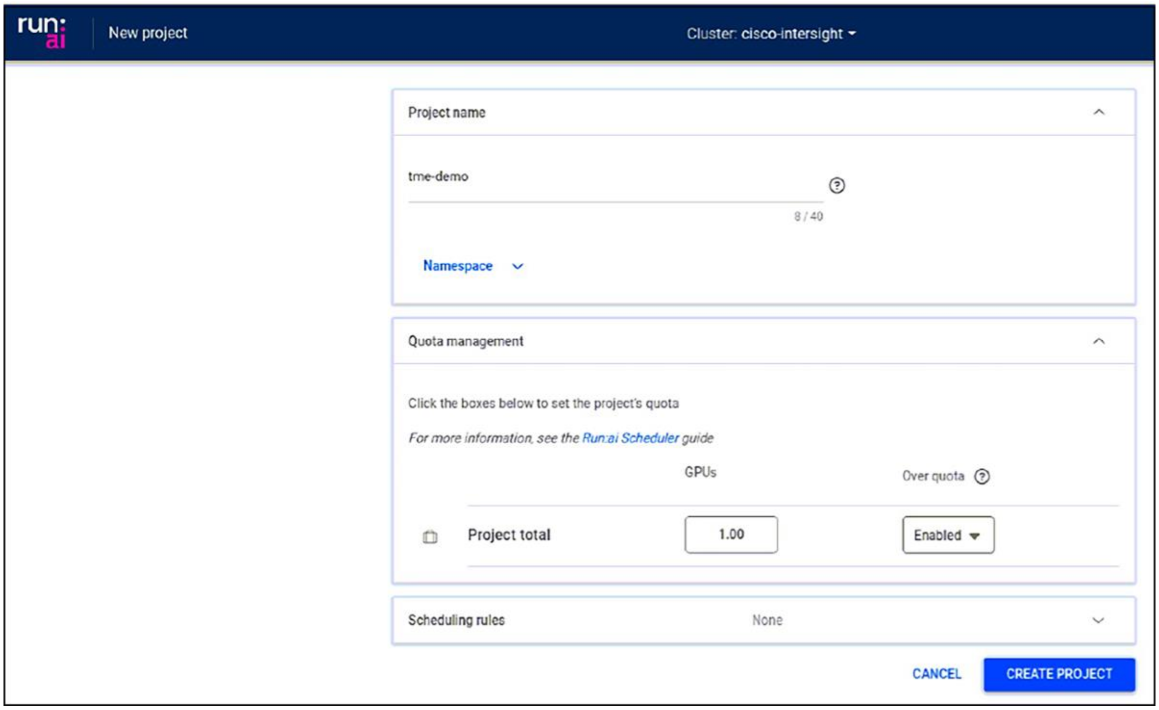
Please note that this is a high-level guide. You should refer to the official OpenShift, Cisco UCS, and Run:ai documentation for detailed steps and best practices.

Run:ai use cases

Once Run:ai is installed, the solution design involves configuring it to optimize the AI and ML workloads. This includes setting up resource-allocation policies to ensure that available compute resources are used effectively. Run:ai has a unique capability to create a pool of shared GPU resources that can be dynamically allocated based on the needs of different workloads. This ensures maximum utilization of resources and cost-effectiveness.

Run:ai project configuration

To manage resource allocation across Run:ai users and the jobs they submit, projects are used for GPU node pools and GPU/CPU resource quotas. An overview of project configuration is at https://docs.run.ai/latest/admin/admin-ui-setup/project-setup/. As shown below, projects can set GPU quotas in specific namespaces and enable "Over quota," if needed, to allow use of unallocated resources.



Project name

tme-demo

8 / 40

Quota management

Click the boxes below to set the project's quota

For more information, see the Run:ai Scheduler guide

	GPUs	Over quota
Project total	1.00	Enabled

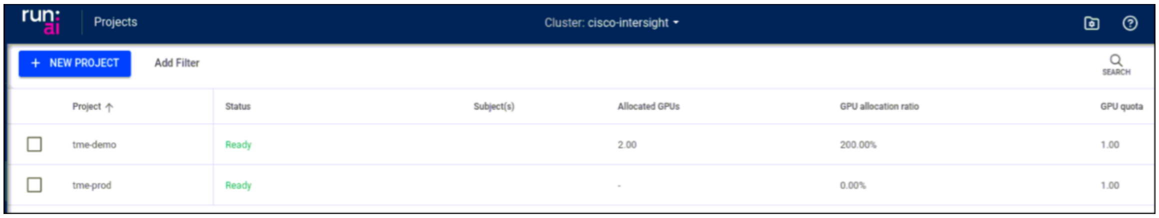
Scheduling rules None

If needed, you can configure Run:ai users and assign access roles to projects, as described at https://docs.run.ai/latest/admin/admin-ui-setup/admin-ui-users/.

Run:ai jobs and workspaces

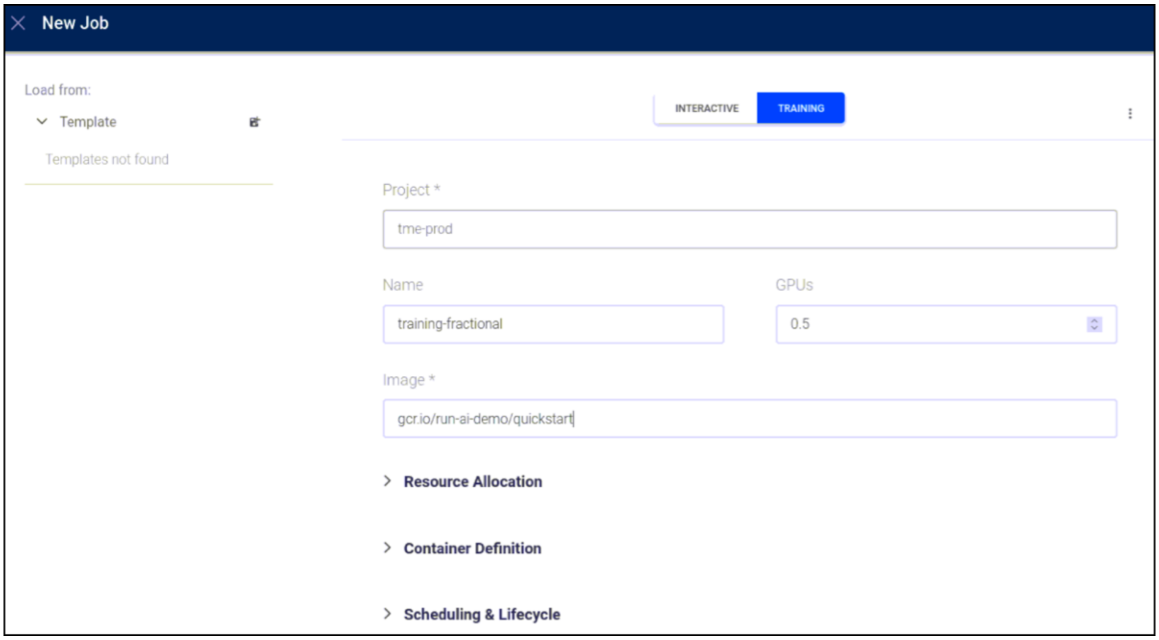
Once you have projects configured, you can submit jobs from the Run:ai web interface or Command Line Interface (CLI). For information on downloading and using the Run:ai CLI, see https://docs.run.ai/v2.16/admin/researcher-setup/cli-install/. Below, we will look at jobs and workspaces using the web interface.

As jobs are submitted, Run:ai will allocate resources based on project settings. In the example below, the tme-demo project has a GPU quota of 1 but currently has 2 GPUs allocated because the project allows jobs to go over quota.



	Project ↑	Status	Subject(s)	Allocated GPUs	GPU allocation ratio	GPU quota
□	tme-demo	Ready		2.00	200.00%	1.00
□	tme-prod	Ready		-	0.00%	1.00

Below a new job is submitted in the tme-prod project with a request for a fractional GPU.



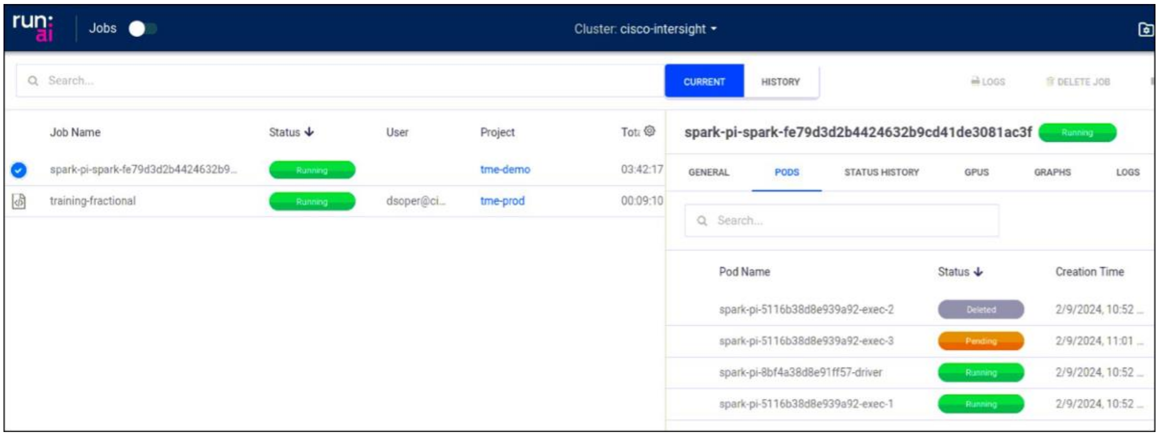
Project *: tme-prod

Name: training-fractional

GPUs: 0.5

Image *: gcr.io/run-ai-demo/quickstart

You can see Run:ai's GPU resource management scale back GPU usage in over-quota projects. Below, the Apache Spark RAPIDS training job has automatically had over-quota pods deleted to bring the job back to its quota of 1 GPU.

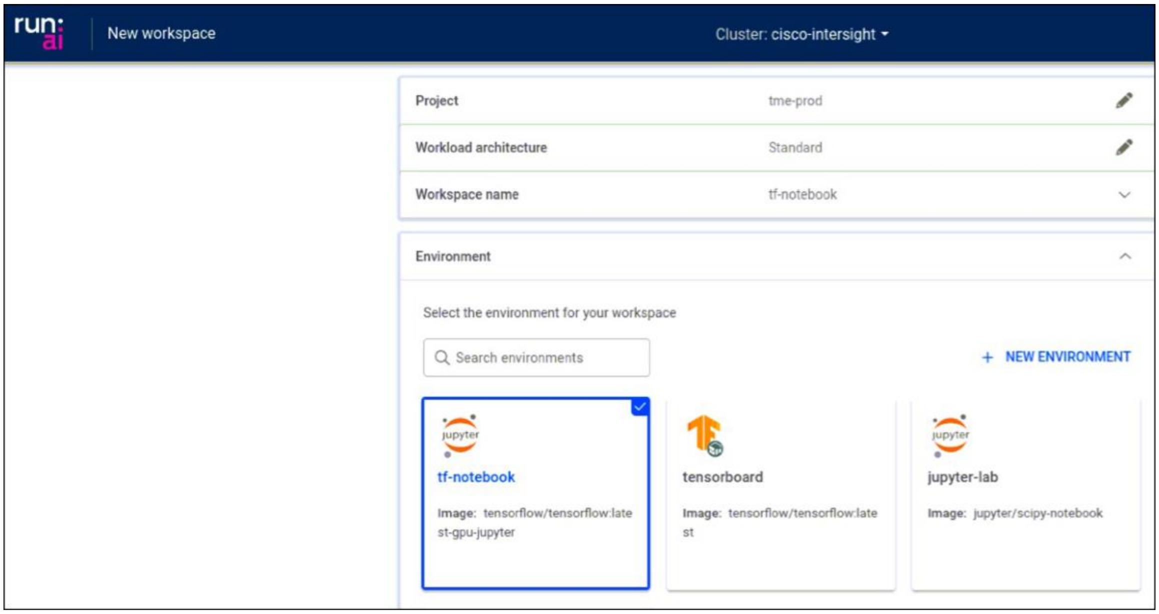


Job Name	Status	User	Project	Tot:
spark-pi-spark-fe79d3d2b4424632b9...	Running		tme-demo	03:42:17
training-fractional	Running	dsoper@ci...	tme-prod	00:09:10

spark-pi-spark-fe79d3d2b4424632b9cd41de3081ac3f Running

Pod Name	Status	Creation Time
spark-pi-5116b38d8e939a92-exec-2	Deleted	2/9/2024, 10:52 ...
spark-pi-5116b38d8e939a92-exec-3	Pending	2/9/2024, 11:01 ...
spark-pi-8bf4a38d8e91ff57-driver	Running	2/9/2024, 10:52 ...
spark-pi-5116b38d8e939a92-exec-1	Running	2/9/2024, 10:52 ...

Workspaces provide another way for you to define and submit workloads using predefined templates and other required settings. Interactive jobs such as Jupyter Notebooks can be created easily.



Project: tme-prod

Workload architecture: Standard

Workspace name: tf-notebook

Environment

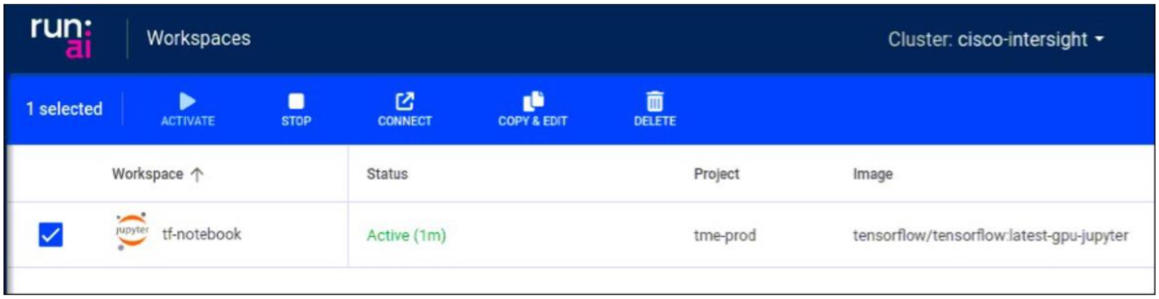
Select the environment for your workspace

tf-notebook — Image: tensorflow/tensorflow:latest-gpu-jupyter

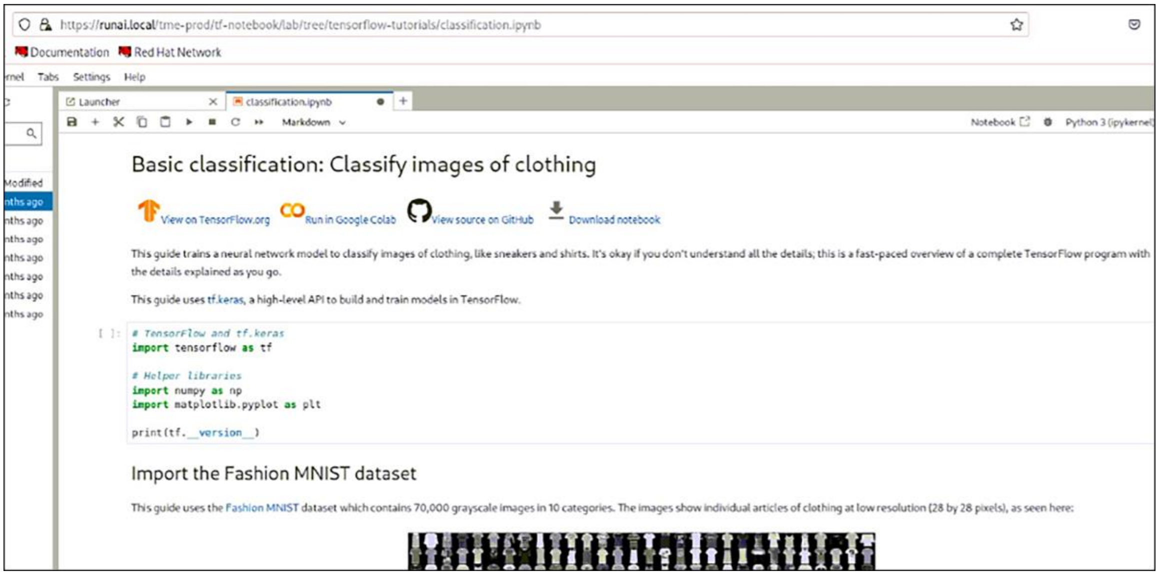
tensorboard — Image: tensorflow/tensorflow:latest

jupyter-lab — Image: jupyter/scipy-notebook

Run:ai also provides web browser connections to the running workspace so that you do not have to configure any additional container networking to interact with the workload.



	Workspace	Status	Project	Image
☑	tf-notebook	Active (1m)	tme-prod	tensorflow/tensorflow:latest-gpu-jupyter



Basic classification: Classify images of clothing

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details; this is a fast-paced overview of a complete TensorFlow program with the details explained as you go.

This guide uses tf.keras, a high-level API to build and train models in TensorFlow.

```
[ ]: # TensorFlow and tf.keras
     import tensorflow as tf

     # Helper libraries
     import numpy as np
     import matplotlib.pyplot as plt

     print(tf.__version__)
```

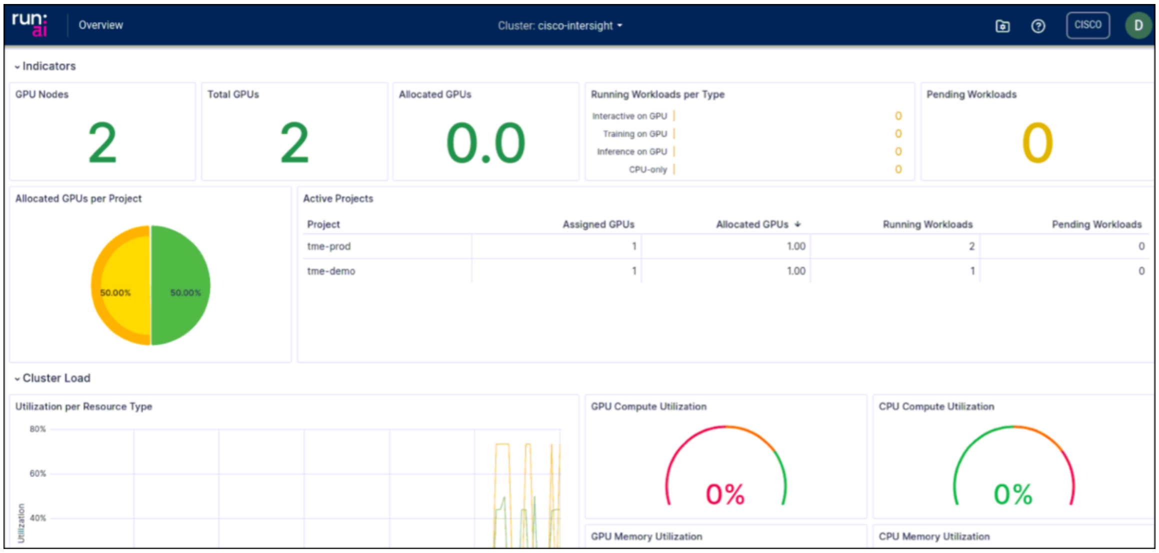
Import the Fashion MNIST dataset

This guide uses the Fashion MNIST dataset which contains 70,000 grayscale images in 10 categories. The images show individual articles of clothing at low resolution (28 by 28 pixels), as seen here:

See https://docs.run.ai/v2.16/Researcher/user-interface/workspaces/overview/ for more information on workspaces and other ways to manage workloads.

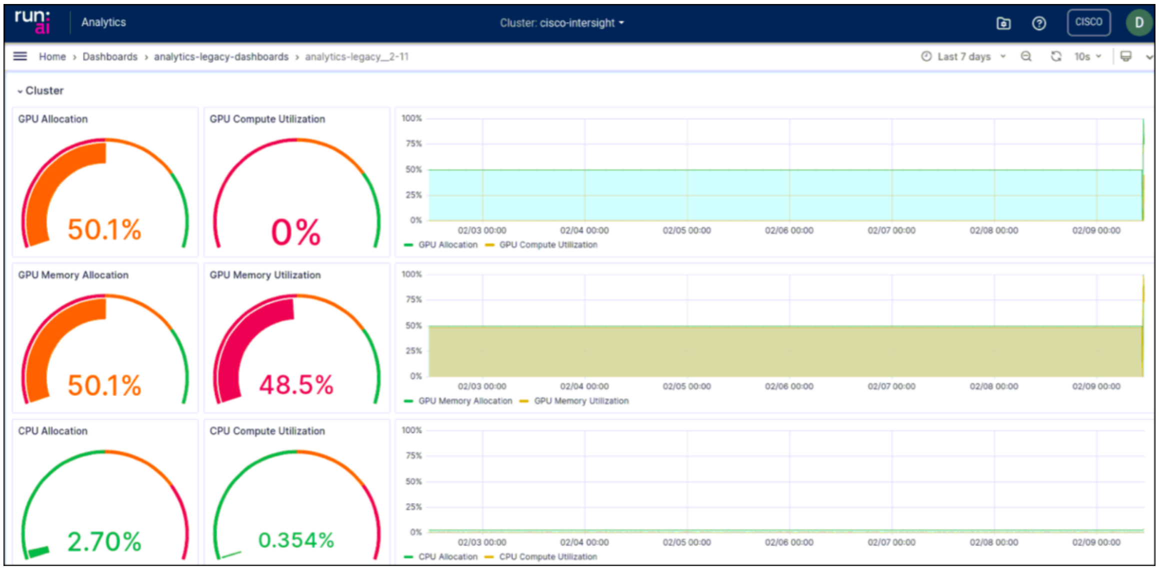
Run:ai dashboard and analytics

Finally, the solution design includes monitoring and managing the Run:ai platform using its built-in tools. These tools provide visibility into the performance and utilization of resources, helping to identify any bottlenecks or inefficiencies. Run:ai's dashboards provide overviews of workloads deployed across projects. In the following screenshot you can see GPU usage and workloads in both the tme-prod and tme-demo projects.



Project	Assigned GPUs	Allocated GPUs ↓	Running Workloads	Pending Workloads
tme-prod	1	1.00	2	0
tme-demo	1	1.00	1	0

Run:ai's analytics provide additional detail and allow you to customize views of resource usage and allocation.



The monitoring capabilities also help to ensure that the solution is running optimally, allowing for any necessary adjustments to be made swiftly. This results in a seamless, efficient, and optimized AI and ML workflow.

Conclusion

In conclusion, the integration of Run:ai with OpenShift on Cisco UCS X-Series provides a robust, scalable, and effective solution for managing and optimizing AI and machine-learning workloads. Run:ai's AI orchestration capabilities, coupled with the power of OpenShift's containerization and automation features, and the high-performance computing offered by Cisco UCS X-Series, together create a unified platform that can streamline AI workflows, maximize resource utilization, and accelerate AI research. This unique combination enables businesses to drive innovation, maintain a competitive edge, and significantly reduce operational costs associated with AI workloads.

The capability of Run:ai to dynamically allocate resources and efficiently manage workloads provides a level of agility and flexibility that is essential in today's fast-paced, data-driven environment. Meanwhile, the scalability of OpenShift and Cisco UCS X-Series ensures that the solution can grow with the evolving needs of a business, handling increased workloads as required. Together, Run:ai, OpenShift, and Cisco UCS X-Series provide a complete solution for AI orchestration, offering a path to faster results, greater efficiencies, and a higher return on AI investments.

For more information

Ready to harness the power of AI orchestration? Discover how Run:ai, integrated with OpenShift on Cisco UCS X-Series, can revolutionize your AI and ML workloads. Visit Run:ai’s installation and usage guide for step-by-step instructions on how to deploy Run:ai on OpenShift. Also, check out Cisco’s UCS Validated Designs for comprehensive information on the power and performance of the Cisco UCS X-Series. Recent additions to the Cisco UCS validated design zone that use Red Hat OpenShift and NVIDIA GPUs include FlashStack for Generative AI Inferencing with Pure Storage and FlexPod Datacenter with Generative AI Inferencing with NetApp storage. Don't miss this opportunity to streamline your AI workflows, maximize resource utilization, and drive innovation in your organization.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.

Printed in USA C11-4257761-00 03/24