ClearML & NVIDIA AI Enterprise Deployment Guide on VMware vSphere with Tanzu

Version 1.0.2

Table of Contents

Introduction

ClearML delivers a machine learning solution that maximizes resource utilization and accessibility while minimizing the DevOps workload

Easier Infrastructure Management

ClearML offers a little- to no-overhead solution for DevOps managing GPU machines. Installed within minutes, ClearML is the easiest solution to get an NVIDIA DGX™ system or NVIDIA-certified server up and running. As part of ClearML Orchestrate, the ClearML Agent can be installed on bare metal or as a Kubernetes client.

Bare Metal

ClearML eliminates DevOps overhead when installed on bare metal, making machines instantly and fully available to machine learning teams. The solution supports containerized applications and virtual environment applications and is truly plug-and-play – it's easy to install, with no need to configure any firewall rules or customize network setup.

A bare metal installation is a turnkey solution that offers fully accessible GPU machines without the need for Kubernetes installation and maintenance. It is available as a fully managed SaaS service or with on-prem dedicated support.

Kubernetes

ClearML minimizes DevOps overhead when installed on top of a Kubernetes cluster. As a Kubernetes-native solution, ClearML provides the user management layer and adds scheduling capabilities that build and launch job queues for the machine, enabling dynamic GPU slicing for multiple users. ClearML supports Vanilla Kubernetes, Rancher, OpenShift, and Tizen (VMWare).

The ClearML Kubernetes installation makes it easy for machine learning teams to utilize their GPU resources without exposing them to Kubernetes or needing to support them.

Slurm

ClearML can also be installed on top of a Slurm cluster, providing the user management layer that makes orchestration accessible and automated with minimal DevOps overhead.

More Accessibility for Teams

ClearML supports SSO, LDAP integration, and role-based access control. The platform ensures fully secured access to NVIDIA DGX systems, and users can interact directly with the hardware from their development environment / IDE.

Platform users can launch jobs directly from the UI programmatically as well as build full automation pipelines with no additional infrastructure needed. With the ClearML SaaS solution, Data Scientists and ML Engineers can even launch and manage jobs from the convenience of wherever they happen to be working.

ClearML is certified on NVIDIA AI Enterprise, ISO 27001-certified and provides enterprises with a fully secure end-to-end platform for continuous machine learning.

Optimal Cluster Utilization

ClearML's scheduling and automation functionality allows teams to fully utilize their resources. Users can build pipelines of sequential tasks, and the jobs of all users can be put into queues and prioritized based on resource availability, ensuring machines are utilized to capacity.

By allowing the user to quickly automate the creation and scheduling of multiple jobs, data scientists can work more optimally and achieve their research goals faster with no additional overhead.

Centralized Control and Visibility

ClearML provides ML and DevOps teams with detailed information for dashboards as well as analytics that monitor all resources and workloads. With the ability to set policies based on business goals, the platform allows for controlled allocation and oversight of resources across departments, projects, and users.

Truly Open and Extensible

Maximize efficiency in your ML development process by utilizing the optimized built-in pipelines, workflows, and automations in ClearML. ClearML's modular, open source architecture allows ML teams and DevOps to seamlessly integrate their own infrastructure and tools for complete management of their MLOps.

One End-to-End Platform

ClearML's unified, end-to-end platform includes a best-of-breed experiment management module with scheduling and orchestration to maximize the efficiency of the development process.

ClearML was purpose-built by data scientists and ML practitioners to bring CI/CD methodology to the ML space, fully automated by NVIDIA hardware. ClearML empowers ML teams to develop, manage, deploy, and monitor the complete ML lifecycle process from a single fully integrated platform – all with just two lines of code. With ClearML, customers significantly shorten their time-to-value and time-to-revenue, ensuring ML projects are executed successfully and make it to production efficiently. The open source platform can seamlessly integrate with any organization's existing infrastructure and tools, allowing companies and teams a fast, frictionless customer onboarding experience and workflow. Organizations can run ClearML on any infrastructure, whether it is cloud, virtual private cloud (VPC), or on premises.

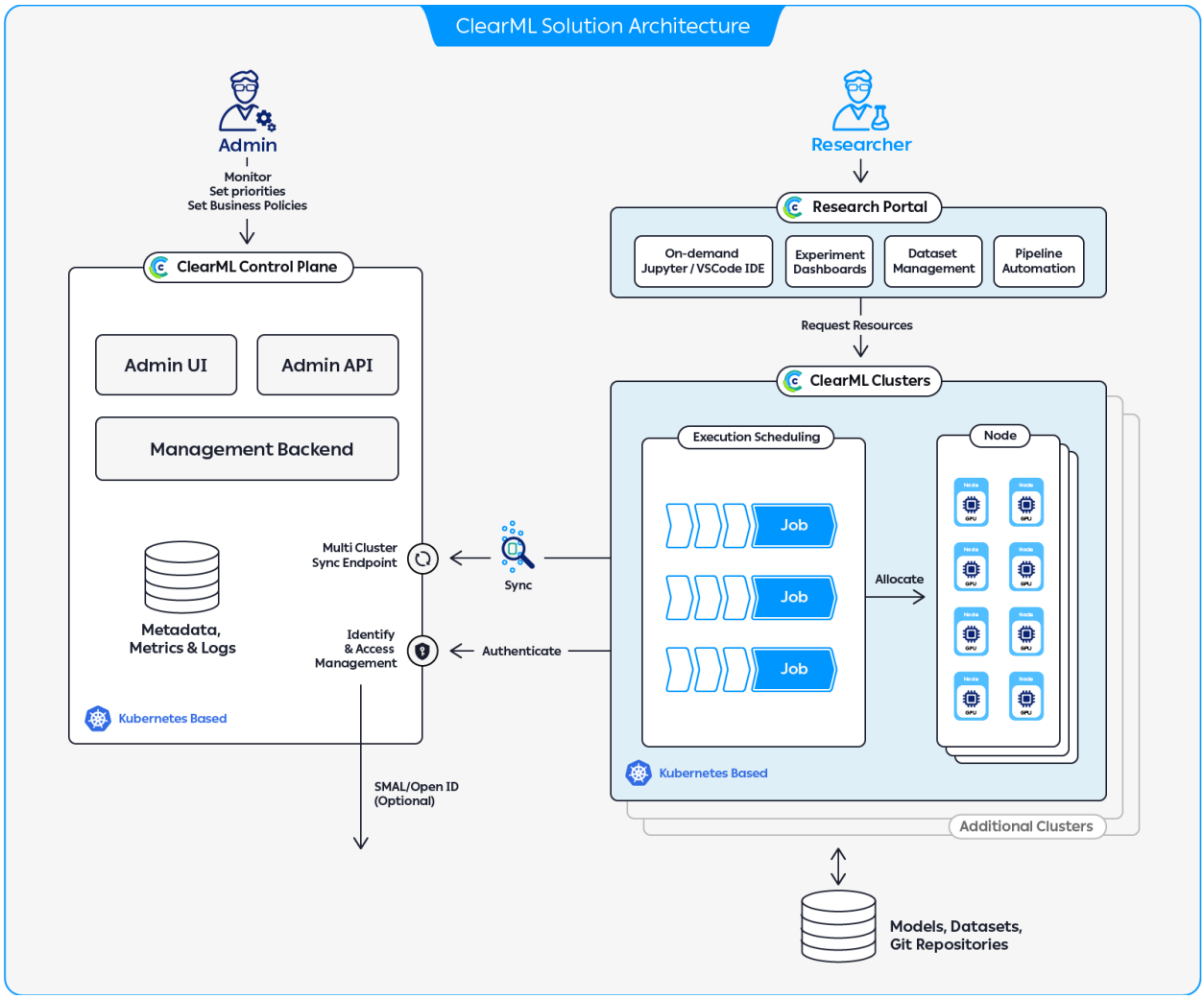
Accelerate Hybrid Cloud

ClearML provides centralized management and visibility of on-premises and cloud-based resources, making hybrid cloud ML infrastructure a viable option for organizations. ClearML also provides the ability for controlled spillover onto cloud infrastructure when needed. Please contact us for more details if you are interested in an auto-scaling solution for AWS, GCP, or Azure.

Architecture

Solution Overview

ClearML is installed over a Kubernetes Cluster which serves as the infrastructure to log ML experiments' configuration, data and results as well as orchestrate their execution over a distributed cluster of lightweight execution agents. ClearML can be deployed on various container orchestrators from Tanzu, OpenShift, Rancher, and Vanilla Kubernetes to cloud service based orchestrators like Google Kubernetes Engine (GKE), Azure Kubernetes Service (AKS), and Amazon's Elastic Kubernetes Service (EKS). ClearML provides researchers and data scientists frictionless access to compute capacity as provisioned by DevOps on any mix of compute clusters.



The ClearML installation consists of two components. The ClearML cluster (pictured above, right) and the ClearML Control Plane or Backend (left). Each of these components include options for how they can be deployed with the options being explained below.

ClearML Cluster

The ClearML cluster consists of lightweight agents that are installed in the Kubernetes cluster and orchestrate the workloads local to the deployed infrastructure. ClearML makes use of the default Kubernetes scheduler to schedule workloads sent by researchers and data scientists based on business rules and quotas defined in the ClearML Control Plane. The ClearML agents are responsible for sending monitoring data to the ClearML Control Plane.

The cluster components allow researchers and data scientists to directly submit and interact with workloads that utilize the compute infrastructure. The cluster includes components that provide the scheduling and orchestration capabilities to ensure fair sharing of GPU resources while also allowing bursting of capacity to eliminate idle resources. The cluster components also ensure that any policies set at an administrative level are enforced to ensure the sharing of GPU resources based on business priorities while also maintaining secure operation of the cluster.

Researchers and data scientists have various methods for submitting and interacting with their workloads. They have the ability to submit machine learning workloads via the ClearML Command-Line Interface (CLI), or via the ClearML GUI. The ClearML GUI allows for the simple submission of workloads by assigning them to administrator defined queues that implement the desired resource allocation policy.

Each workload's custom execution environment can be created by the ClearML agents, be they elaborate python repositories or standalone Jupyter notebooks. This allows administrators to provision available resources and streamline onboarding new users to the platform.

In addition to all of the tooling that ClearML provides out of the box, it is an open platform in terms of the tools that your data scientists and researchers will be able to utilize: ClearML allows researchers to orchestrate workflows with built-in pipelines, or to easily plug in ClearML jobs into workflow tools they are familiar with like Kubeflow, MLflow, Airflow, JupyterHub and Argo Workflows.
ClearML also provides on-demand availability for Visual Studio Code, Jupyter notebooks and PyCharm.

ClearML Control Plane (Backend)

The ClearML Control Plane stores ML experiments configuration and results, as well as aggregates monitoring and performance information related to compute infrastructure and running workloads. The control plane allows for the joining of multiple clusters to a single backend so you can manage and monitor multiple clusters from a single pane of glass. The control plane component for Saas customers is fully maintained by ClearML.

Researchers and data scientists use the ClearML web user interface (GUI) to examine their ML experiments' configuration, logs and results: compare experiments, configure tracking tables and create live reports. Additionally, the ClearML GUI allows data scientists to easily create new variants of previously executed experiments facilitating speedy model optimization and accelerating time-to-deployment.

The control plane is also the location where administrative changes can be made to the platform. Some of these administrative changes include integration with SSO, managing of projects and resource quotas, toggling of features and managing end users of the platform.

The cluster sends information to the control plane for the purpose of control as well as monitoring and experiment exploration through dashboards. This includes source control references, execution environment definitions (such as Docker container references, environment variables and Python packages), execution logs, and ML metrics.

For artifact storage (such as any debug samples, training data, models, checkpoints and the like) users can make use of the backend incorporated file server, or use their own storage solution (Local, corporate network, or cloud based) to guarantee the data does not leave their corporate premises.
The directions below include installation for the SaaS version ClearML.

If you are interested in installing the self hosted version of ClearML where the control plane is hosted in your Kubernetes cluster, please follow the instructions here:
https://clear.ml/docs/latest/docs/deploying_clearml/clearml_server_kubernetes_helm

Prerequisites

General ClearML prerequisites

The latest ClearML version supports Kubernetes versions 1.21 through 1.26.
NVIDIA's GPU Operator is also a prerequisite for using ClearML. Installation instructions for the GPU Operator are included below.

ClearML leverages Elasticsearch and Mongodb for experiment results and task information databases. If you are installing your own ClearML control plane, the installation will, by default, install these, but it can also connect to existing instances installed by an organization.

ClearML also provides unique model serving capabilities as well that allows you to serve and scale Triton and other inference processes in a flexible and efficient deployment supporting auto-batching and canary A/B strategies. More information on the requirements for enabling ClearML's model serving capabilities can be found at the following link:
https://clear.ml/docs/latest/docs/clearml_serving/clearml_serving_setup

For production installs, it is recommended to leverage ClearML system nodes to reduce downtime and save CPU cycles on GPU Machines. ClearML recommends that clusters contain two or more CPU worker nodes, designated for ClearML server. These nodes do not have to be dedicated to ClearML however from a resource perspective; the ClearML server components require: 4 CPUs, 8GB of RAM and 50GB of Disk space. ClearML also requires few network settings for pulling container images as well as pushing data to the control plane. These network requirements are defined in the ClearML cluster setup section.

Helm is another prerequisite for installing the ClearML cluster as well as the NVIDIA GPU Operator. Helm is a package manager that helps you manage Kubernetes applications using charts that define, install, and upgrade even the most complex Kubernetes applications.
Installation instructions for Helm are provided below in addition to full documentation at the link provided:
https://helm.sh/docs/intro/install/

- **`curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3`**
- **`chmod 700 get_helm.sh`**
- **`./get_helm.sh`**

For the most up to date information please refer to https://github.com/allegroai/clearml-helm-charts/blob/main/INSTALL.md

Running kubectl and CLI commands

The commands in the deployment guide are expected to be executed from one of the control nodes of the cluster or from a node that has the kubeconfig file of the Tanzu cluster exported locally to it.

For more information on kubeconfig files and how to use them, please review the official Kubernetes documentation:
https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/

Deploy a Tanzu Kubernetes cluster

VMware vSphere with Tanzu overview

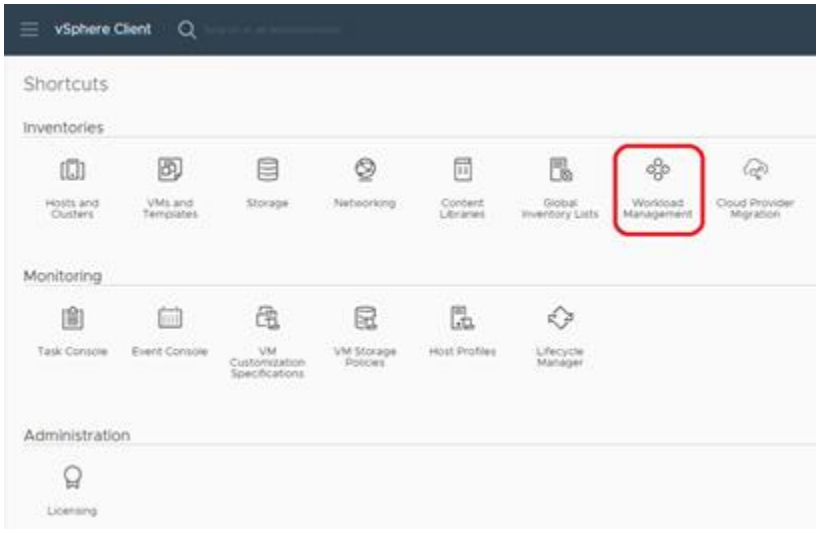
The goal of this portion of the deployment guide is to provide a blueprint for deploying a Tanzu Kubernetes cluster with GPU nodes. VMware vSphere with Tanzu directly integrates with vSphere, which provides a simplified orchestration solution. Tanzu is declarative, so creating and interacting with a GPU-enabled cluster often requires fewer steps than upstream Kubernetes, and it can be done on-demand. Tanzu lets you create and operate Tanzu Kubernetes clusters natively in vSphere. These instructions are not designed to be leveraged for a production environment, but rather to provide a bare minimum setup to demonstrate the abilities of ClearML in conjunction with VMware vSphere with Tanzu.

The instructions assume you have vSphere installed and you have administrative access to the infrastructure. The first step will be to ensure you can connect to vCenter to begin managing your infrastructure.

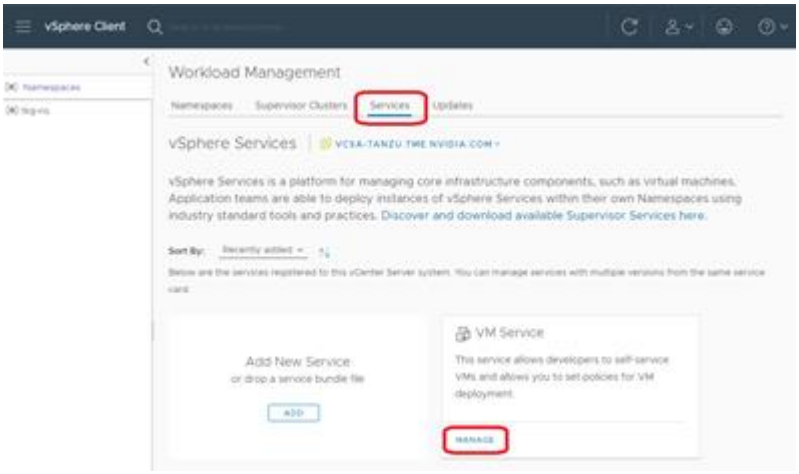
Create a VM Class with GPUs

To leverage a Tanzu Kubernetes cluster with GPUs, you will need to create a virtual machine class. This virtual machine class specifies the underlying compute resources both CPU and GPU that will be dedicated to your Kubernetes worker. VMware vSphere with Tanzu provides default classes for compute or you can create your own VM classes as described below.

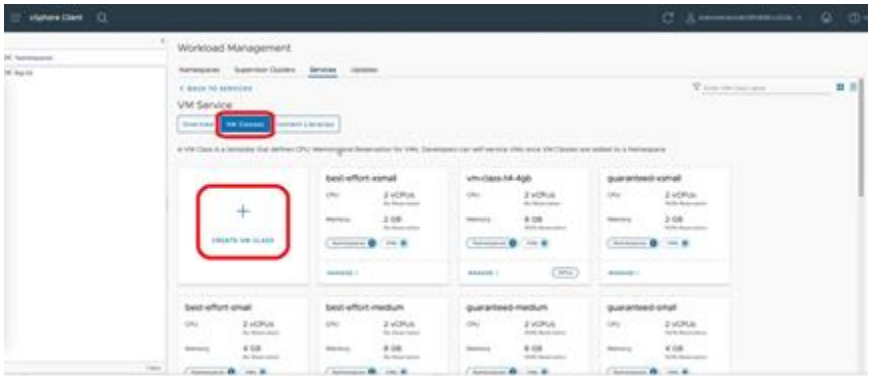
To create your own VM classes for your GPU nodes, first login to vCenter. Navigate to Shortcuts and then to Workload Management.



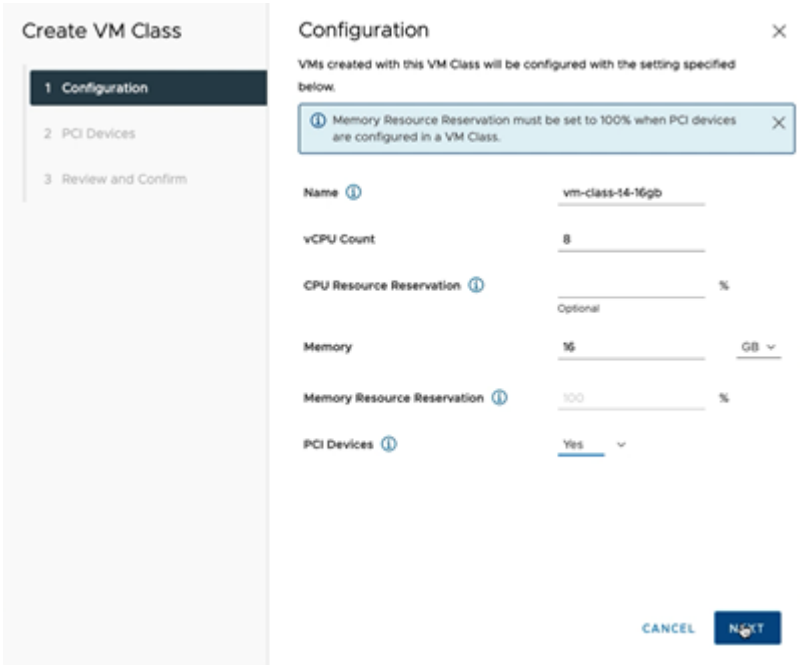
On the Services tab of Workload Management, select Manage under the VM Service tile.



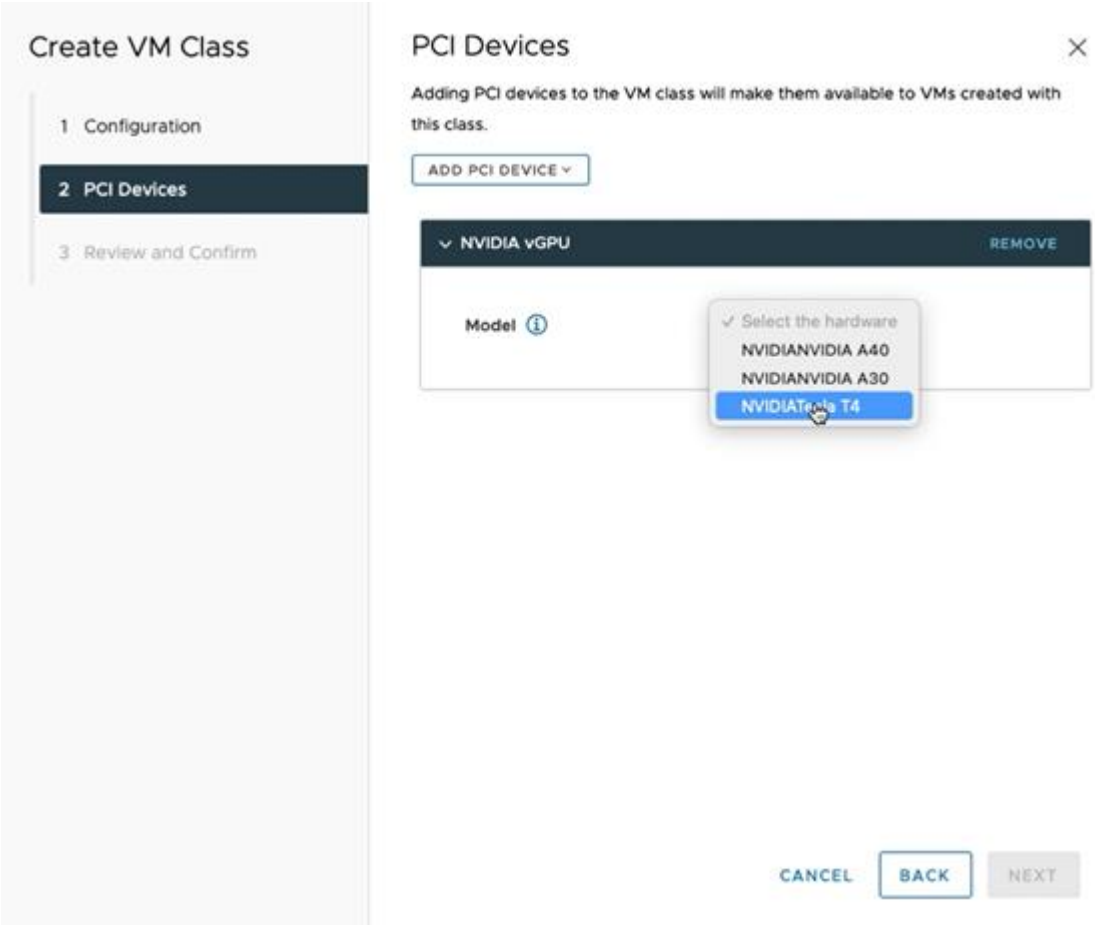
Select VM Classes and then select the tile to Create VM Class.



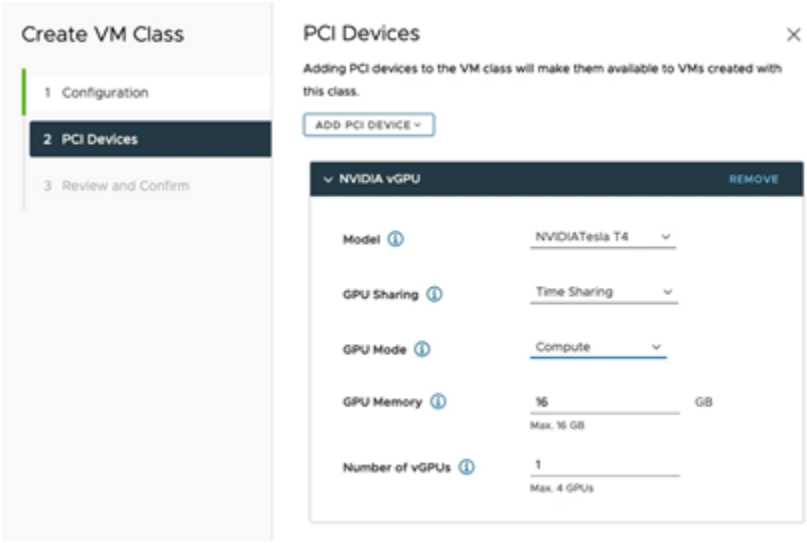
Provide the configuration for the VM class. It is recommended to name the VM class relative to the resources on the node itself such as GPU type or you can leverage your organizations naming convention to name and identify the nodes. Provide a vCPU count, Memory allotment and ensure that you select Yes for the PCI devices from the drop down.



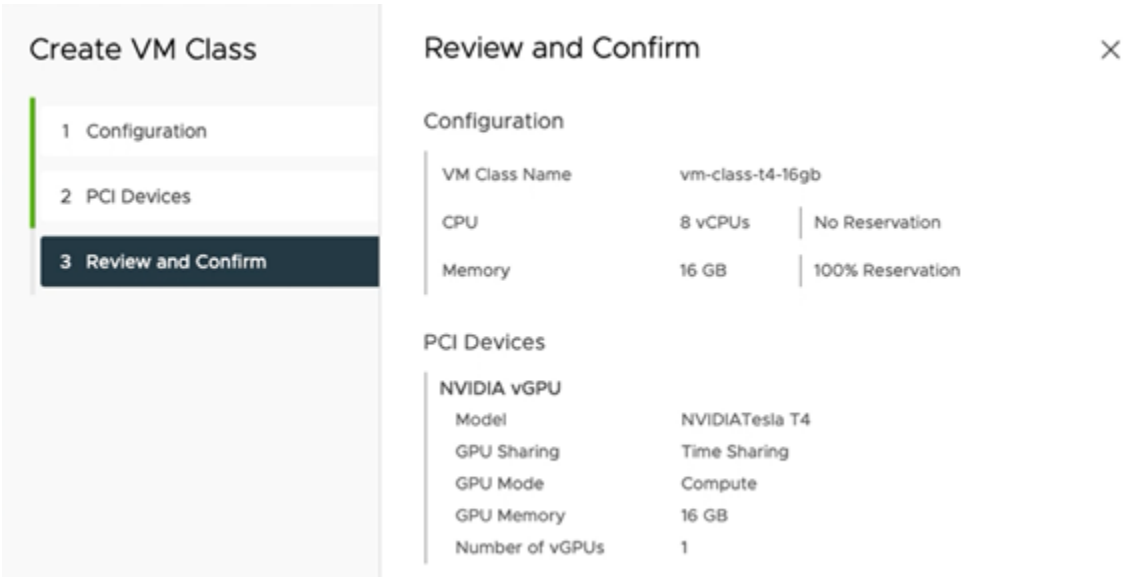
Next on the PCI devices tab, select NVIDIA vGPU from the dropdown and then select the GPU model that you want to add to your VM class.



After selecting the GPU model, additional details will need to be provided such as the GPU sharing mode (Time Sharing), GPU Mode (Compute), GPU Memory (typically the maximum memory is allotted) in addition to the total number of vGPUs per node.



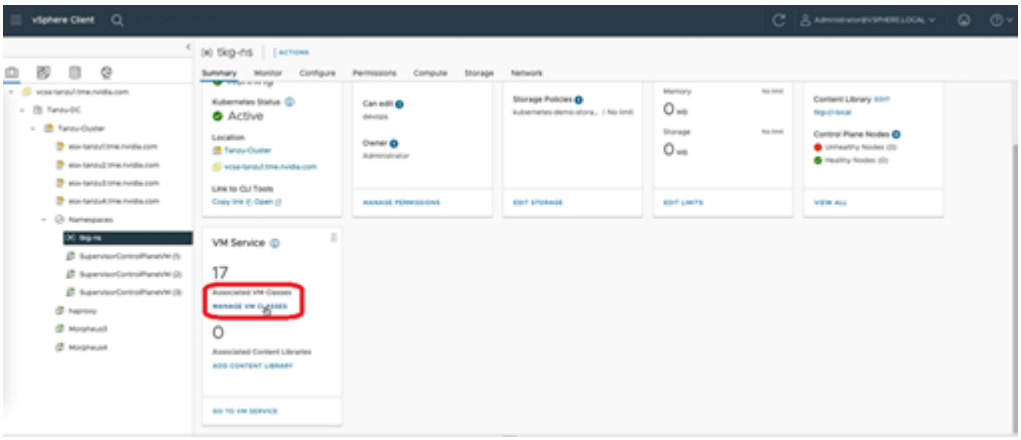
On the final tab, review the details of the VM class and click on Finish to finalize the creation of the VM class.



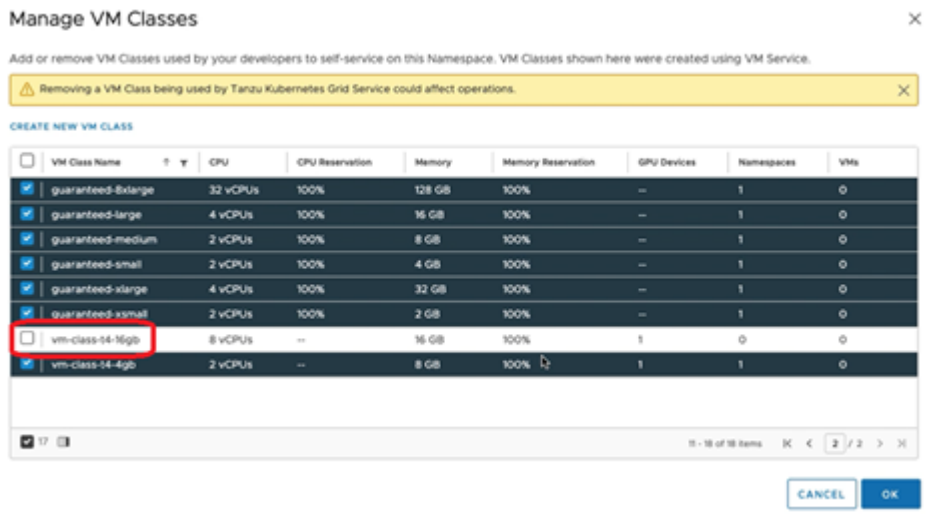
Assign the GPU accelerated VM Class to the Supervisor Namespace

After creating a GPU accelerated VM class it will need to be associated with the Supervisor Namespace. VM classes can reside in one or more namespaces within a Supervisor Cluster. Additionally Supervisor clusters can include multiple different types of VM classes.

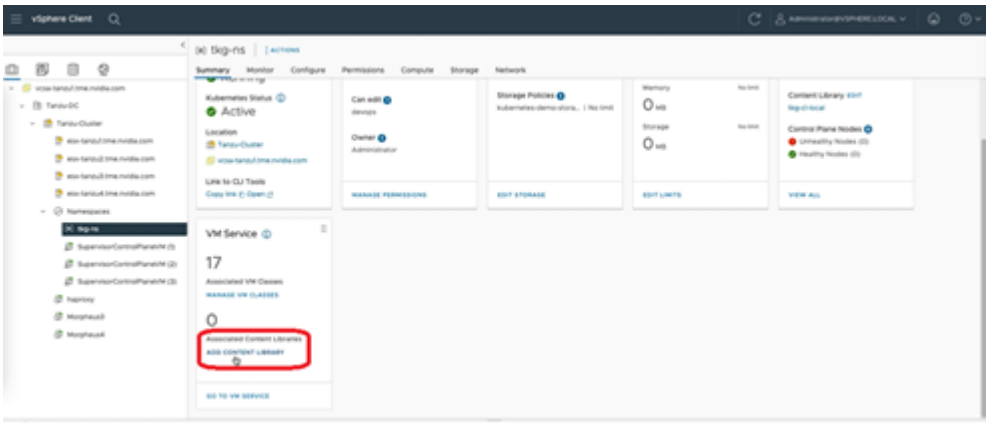
Within vCenter, navigate to inventory and expand your Tanzu cluster and associated namespace. Select the appropriate namespace and click on Add VM Class or Manage VM Classes.



Upon entering Add VM Class or Manage CM Classes, you can select the check box next to the VM class that was created in the prior steps as well as any other relevant VM classes to make it available within the namespace.

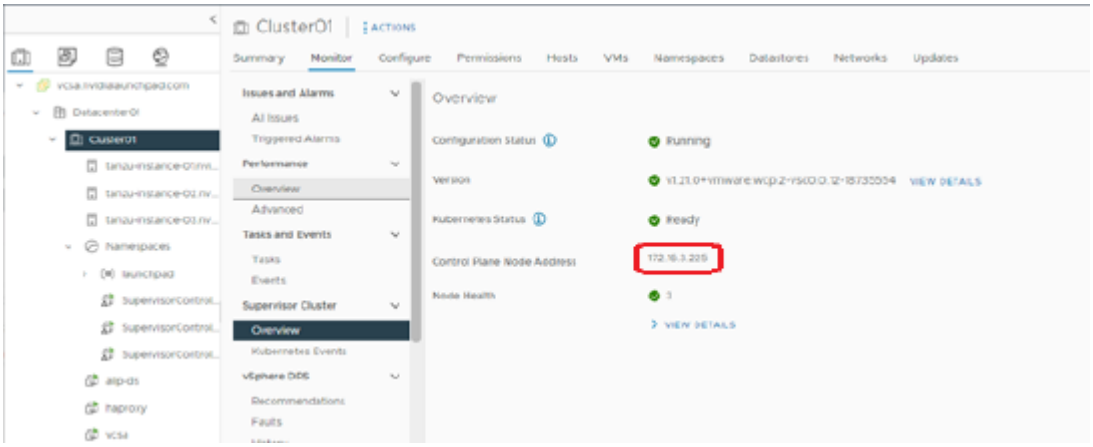


Validate that the Content Library is associated with the Supervisor Namespace by clicking on the Add Content Library button in the VM Service card. Ensure that the VM template is present in the Subscribed Content Library so it can be used by NVIDIA AI Enterprise.



Connect to the Supervisor Cluster

Identify the IP address of the control plane node (<KUBERNETES-CONTROL-PLANE-IP-ADDRESS>) and leverage the IP address to connect to the Supervisor Cluster VM in vCenter.



Use the following command to connect to the Supervisor Cluster VM:

- **`kubectl vsphere login --server=<KUBERNETES-CONTROL-PLANE-IP-ADDRESS> --vsphere-username administrator@vsphere.local --insecure-skip-tls-verify --tanzu-kubernetes-cluster-namespace <namespace>`**

Run the following commands on the CLI to ensure you are connected and submitting commands to the correct cluster:

- **`kubectl config get-contexts`**
- **`kubectl config use-context <context name>`**

Validate your connection to the cluster by running the following commands to view the outputs of the nodes that are in the cluster in addition to all of the running pods.

- **`kubectl get nodes`**
- **`kubectl get pods -A`**

Create a GPU Accelerated Tanzu Cluster

The next step to enabling the groundwork for ClearML is to provision the GPU accelerated TKG cluster. To ensure that your VM class created in the previous steps is available, run the following kubectl command:

- **`kubectl get virtualmachineclasses`**

To validate that the virtual machine class is configured with GPUs, you can describe the VM class to gather more information.

- **`kubectl describe virtualmachineclass <vmclass-name>`**

The final step for creating the cluster will be to use a text editor to create a yaml file that defines the cluster and includes the VM classes with the GPUs we intend to leverage. An example yaml file below (tanzucluster.yaml) is provided below with some default parameters.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
 name: tkg-cluster
 namespace: tkg-clearml
spec:
  distribution:
    fullVersion: 1.21.2+vmware.1-tkg.1
  settings:
    network:
      cni:
        name: antrea
      pods:
       cidrBlocks:
         - 192.0.2.0/16
     serviceDomain: local
      services:
        cidrBlocks:
          - 198.51.100.0/12
    storage:
      defaultClass: clearml-kubernetes
  topology:
    controlPlane:
      replicas: 1
      storageClass: clearml-kubernetes
      vmClass: guaranteed-medium
    nodePools:
      -
        name: nodepool-t4
        replicas: 2
        storageClass: clearml-kubernetes
        vmClass: vm-class-t4-16gb
        volumes:
          -
            capacity:
              storage: 100Gi
            mountPath: /var/lib/containerd
            name: containerd
```

Once the yaml file is edited, it can be applied to deploy the cluster.

- **kubectl apply -f tanzucluster.yaml**

Check the status of the cluster and wait until the cluster shows ready.

- **kubectl get tkc**

Once the cluster shows as Ready, you can use the following command to connect to the GPU accelerated cluster to deploy NVIDIA's GPU Operator as well as proceed with the installation of ClearML.

- **kubectl vsphere login --server=<KUBERNETES-CONTROL-PLANE-IP-ADDRESS> --vsphere-username administrator@vsphere.local --insecure-skip-tls-verify --tanzu-kubernetes-cluster-name tkg-cluster --tanzu-kubernetes-cluster-namespace tkg-clearml**

NVIDIA GPU Operator

Install the NVIDIA GPU Operator

ClearML requires the NVIDIA GPU Operator to be installed on the Kubernetes cluster in order to make the GPUs available to the ClearML cluster. The NVIDIA GPU Operator uses the operator framework within Kubernetes to automate the management of all NVIDIA software components needed to provision GPUs. These components include the NVIDIA drivers (to enable CUDA), Kubernetes device plugin for GPUs, the NVIDIA Container Runtime, Node Feature discovery for automatic node labeling and DCGM based monitoring.
The recommended installation steps for installing the NVIDIA GPU Operator are included below and the full installation instructions including potential customization options are provided by NVIDIA at the following link:
https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/getting-started.html

Add the NVIDIA helm repository:

- **helm repo add nvidia https://helm.ngc.nvidia.com/nvidia && helm repo update**

Install the GPU Operator

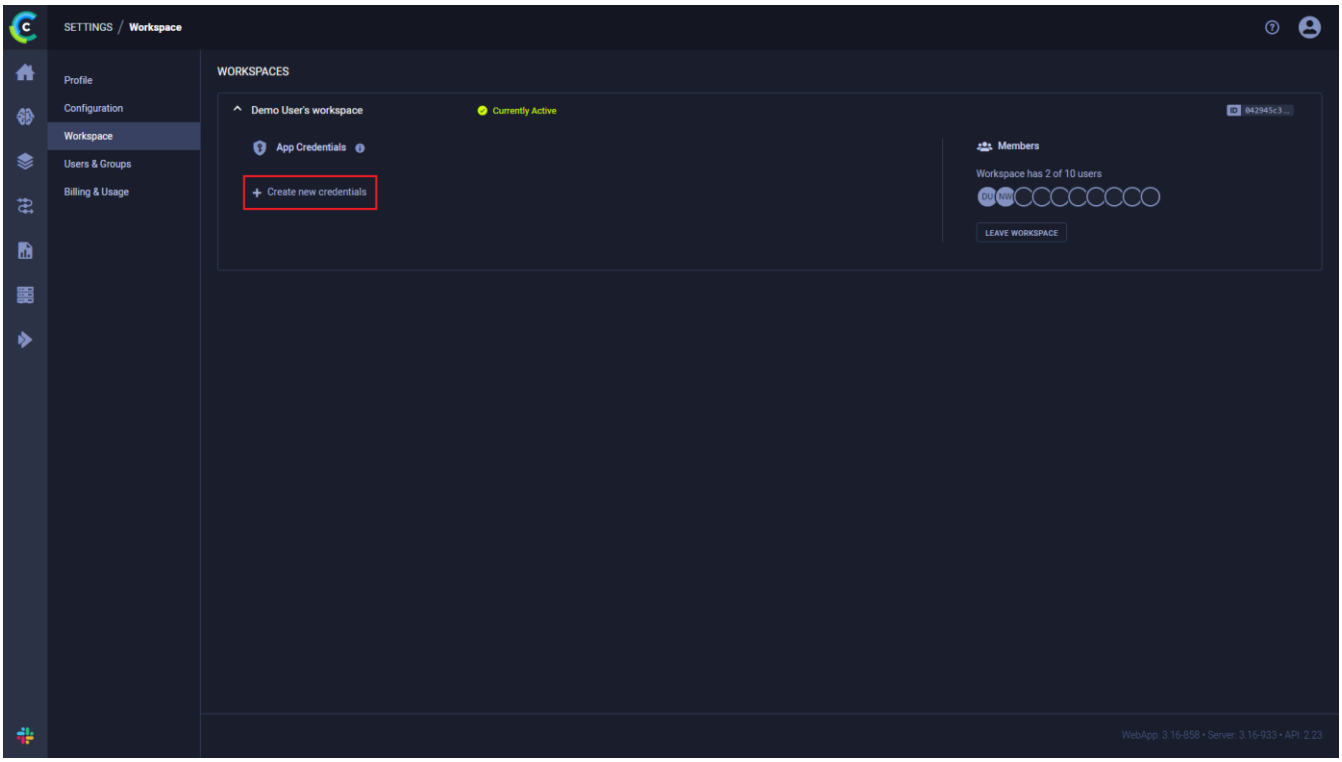
- **helm install --wait --generate-name -n gpu-operator --create-namespace nvidia/gpu-operator**

Deploy the ClearML stack

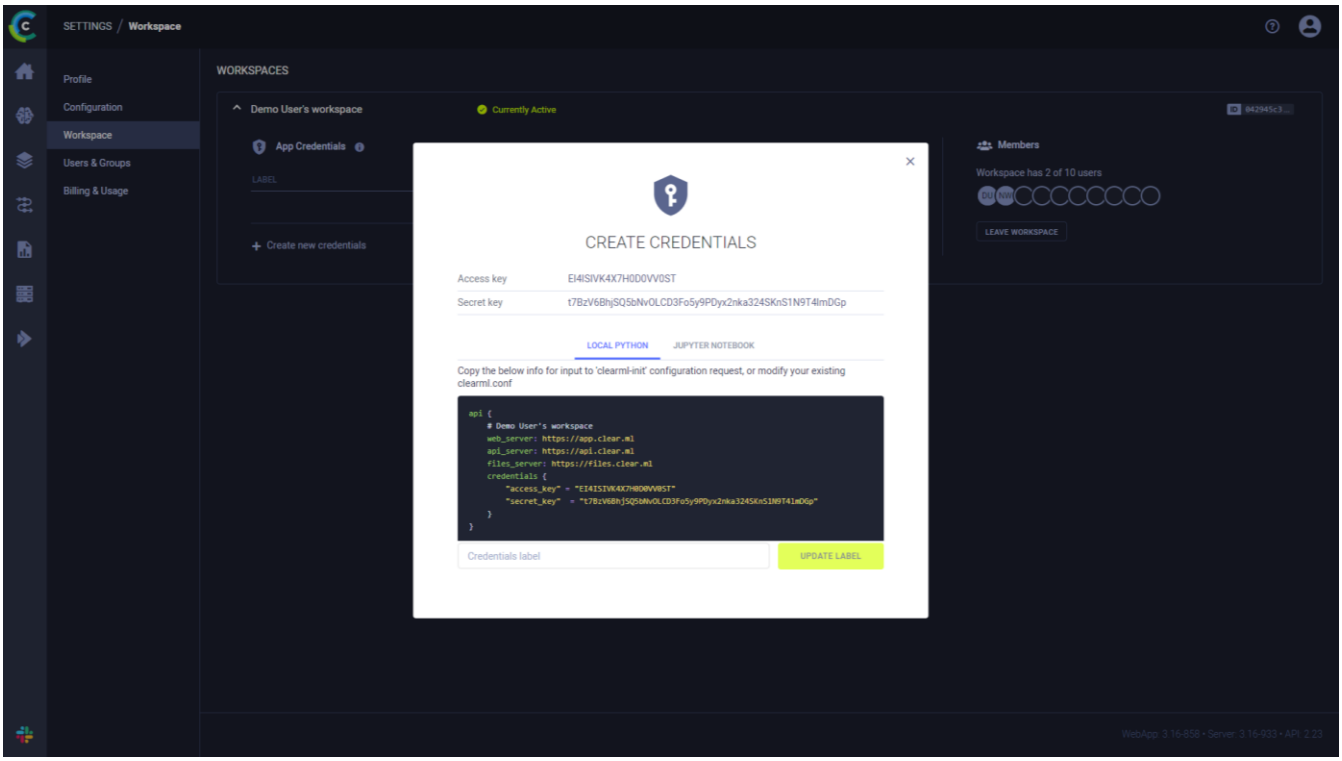
Set Up Access Credentials

Begin setting up ClearML by navigating to your SaaS tenant URL. The credentials to authenticate to this page will be provided by ClearML customer support. For convenience the examples in this guide make use of the public multi-tenant ClearML service: https://app.clear.ml.
If you are working with a ClearML managed dedicated tenant, the format for your tenant will typically be **https://app.<organization>.hosted.allegro.ai/**.

The first step is to create access credentials that will enable the ClearML cluster to authenticate and communicate with the ClearML backend.
Go to the Settings/Workspace page and click "+ Create new credentials"



A new set of credentials is created. Copy the ClearML access key and secret key for use further down the installation process.



Install the ClearML cluster

Once all of the prerequisites have been completed including the provisioning of a vSphere Tanzu cluster and installation of the NVIDIA GPU Operator, proceed to the installation of ClearML.

Configure Role Binding

Get the latest role binding definitions file: https://github.com/allegroai/clearml-helm-charts/blob/main/platform-specific-configs/tanzu/rolebinding.yaml

Fill in your configured Kubernetes namespace, which should result with a rolebinding.yaml file similar to:

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: clearml-tanzu-rolebinding
 namespace: tkg-clearml
roleRef:
  kind: ClusterRole
  name: psp:vmware-system-privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
```

Apply the role binding definitions:

- **kubectl apply -f rolebinding.yaml**

Update Helm Repo Definitions

Add the ClearML helm repo:

- **helm repo add allegroai https://allegroai.github.io/clearml-helm-charts**

Update helm repos:

- **helm repo update**

Install the Helm Chart

- **helm install clearml-agent allegroai/clearml-agent --set clearml.agentk8sglueKey=ACCESSKEY --set clearml.agentk8sglueSecret=SECRETKEY --set agentk8sglue.apiServerUrlReference="https://api.clear.ml" --set agentk8sglue.fileServerUrlReference="https://files.clear.ml" --set agentk8sglue.webServerUrlReference="https://app.clear.ml"**

This deploys the full ClearML stack on the cluster in your configured namespace.

For the most up to date installation instructions please refer to the following link:
https://github.com/allegroai/clearml-helm-charts/blob/main/INSTALL.md

Validate the installation

To validate that the install was successful, run the following command to validate that all of the pods in the tkg-clearml namespace are running.

- **`kubectl get pods -n tkg-clearml`**

Once the installation is complete, you can go to the Workers and Queues page of the Saas tenant where your cluster will appear as an available worker.

Note that your cluster needs to be configured for network access to your SaaS tenant.

Getting Started

Install the ClearML SDK

To enable researchers to log their work onto ClearML, they should install the ClearML python package

- **`pip install clearml`**

Experiments can be logged automatically by instrumenting their code with just 2 lines and executing them. See https://clear.ml/docs/latest/docs/getting_started/ds/ds_first_steps.

Existing code can be launched on your cluster:

- **`clearml-task --project my_proj --name my_experiment --script experiment.py --args epochs=1 --queue default`**

See https://clear.ml/docs/latest/docs/apps/clearml_task

Configure Queues

ClearML queues are the mechanism through which experiments are scheduled for execution on the ClearML cluster.
You can implement flexible resource allocation policies by defining specific queues to match each specific resource requirement (e.g. pods utilizing a specific number of GPUs), and configure the ClearML cluster to allocate the desired resource budget to that queue.
For more information on using ClearML queues for orchestration, see the following link:
https://clear.ml/docs/latest/docs/webapp/webapp_workers_queues

Additional Resources

ClearML provides a diverse set of educational resources for enabling data scientists and researchers to supercharge their ML workflows. These resources are available at the following links:

- Quickstart Guides
- Video Tutorials
- Integrations