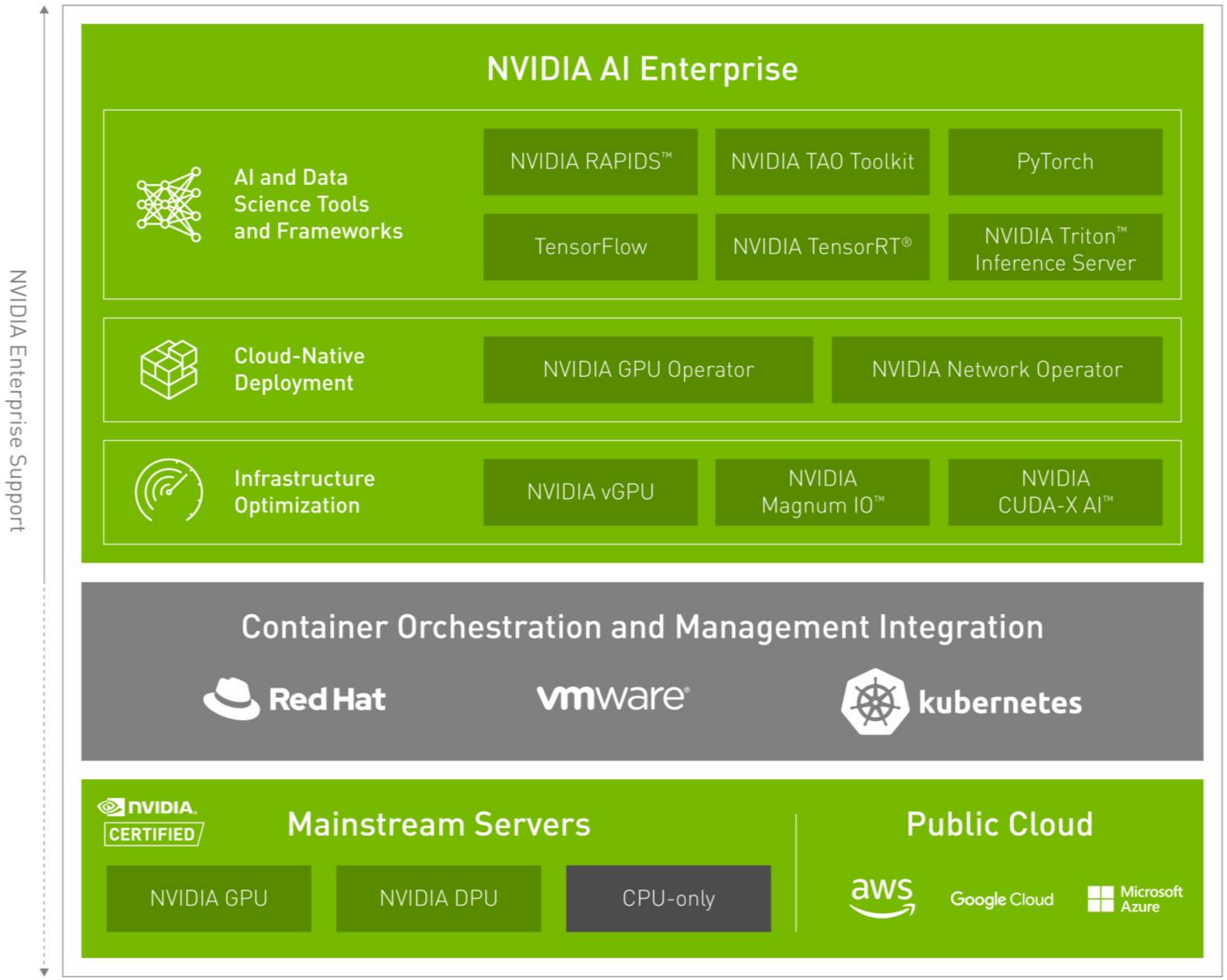
NVIDIA GTC

Running Cloud-native Apps in NVIDIA AI Enterprise

Joe Cullen | Technical Marketing Engineer | NVIDIA
Vinay Bagade | Technical Marketing Engineer | NVIDIA

- NVIDIA AI Enterprise Overview
  - Delivering AI to the Enterprise
  - NVIDIA Operators
- Orchestration Methods
- Red Hat OpenShift Deployment
- Orchestration with OCP and vSphere
- Machine Learning Pipeline on Kubernetes
  - Preprocessing
  - Training
  - Inference

NVIDIA AI ENTERPRISE SOFTWARE SUITE



NVIDIA GTC

NVIDIA OPERATORS

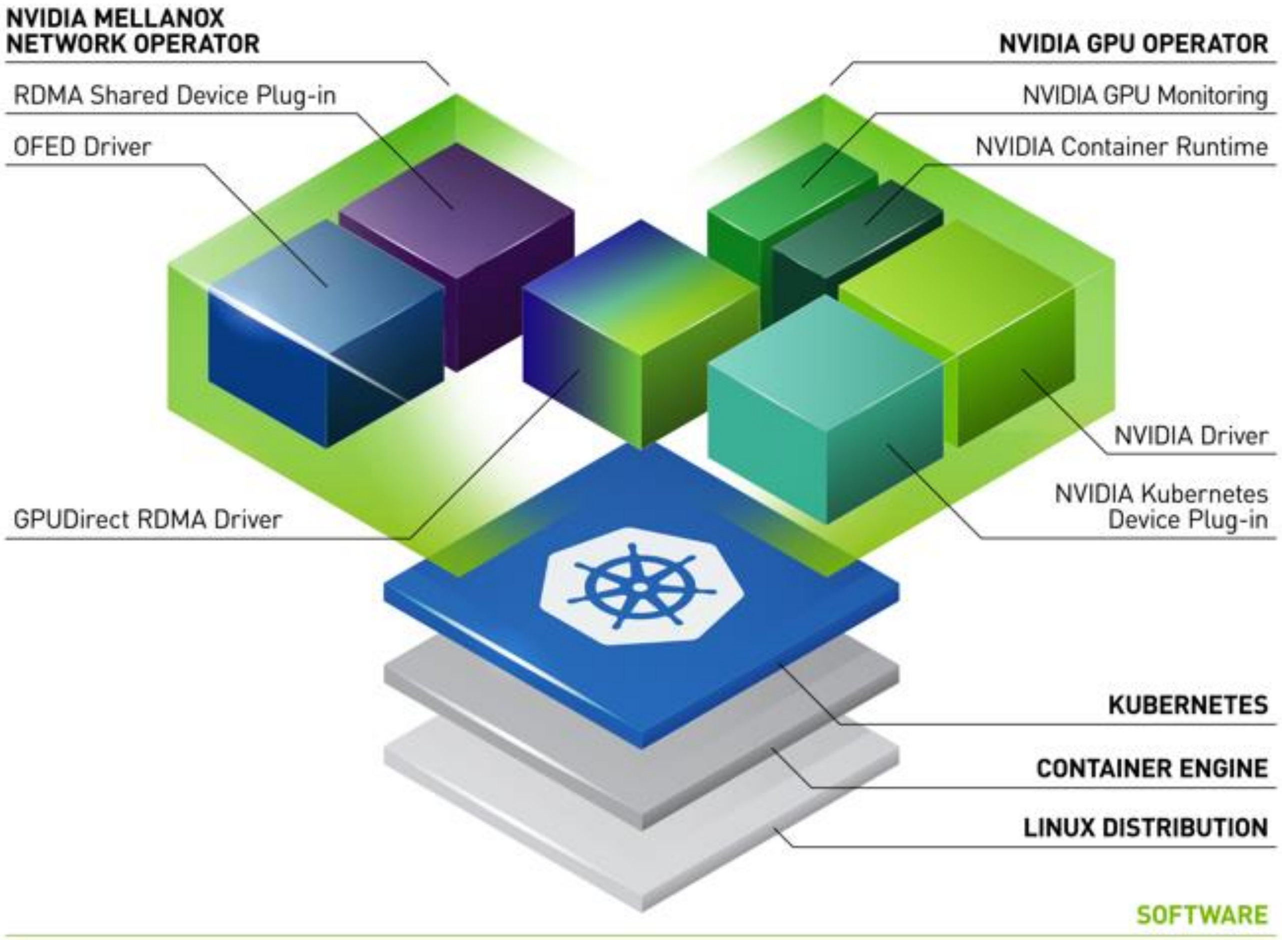
Only NVIDIA AI Enterprise customers have access to containerized vGPU drivers.

GPU Operator installs all software to make GPUs usable by applications running on VMware vSphere with Tanzu.

- Automates the installation of the vGPU Guest Driver, NVIDIA Container Toolkit, Device Plugin, DCGM, etc.
- Automatically scales to newly added GPU accelerated Tanzu nodes.

NVIDIA AI Enterprise customers have access to prebuild vGPU driver images.

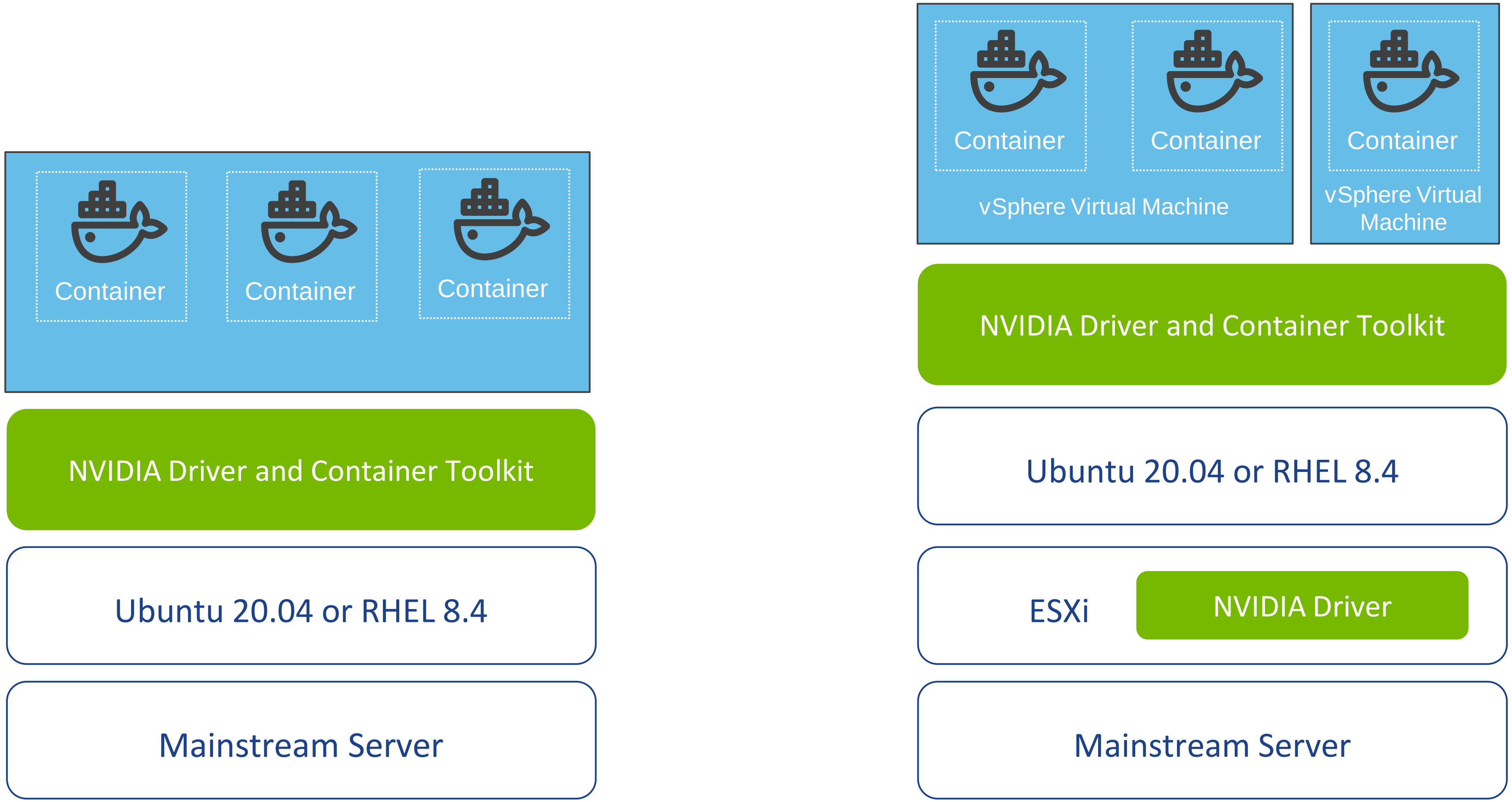
- GPU Operator installs a compatible vGPU Guest Driver.
- Only NVIDIA AI Enterprise customers have access to containerized vGPU drivers.



Orchestration Methods

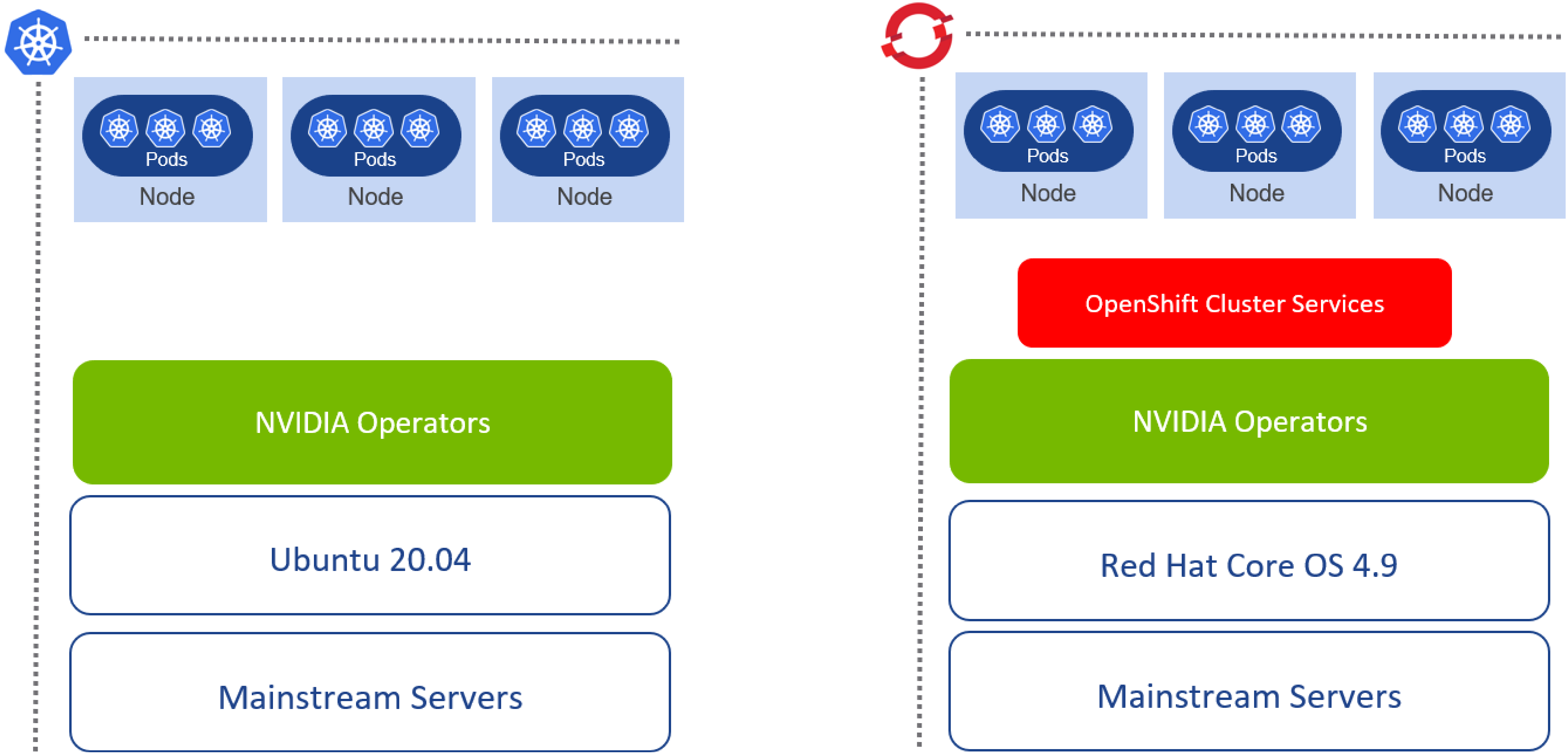
DELIVERING AI WORKLOADS WITH NVIDIA AI ENTERPRISE

Orchestration with Containers



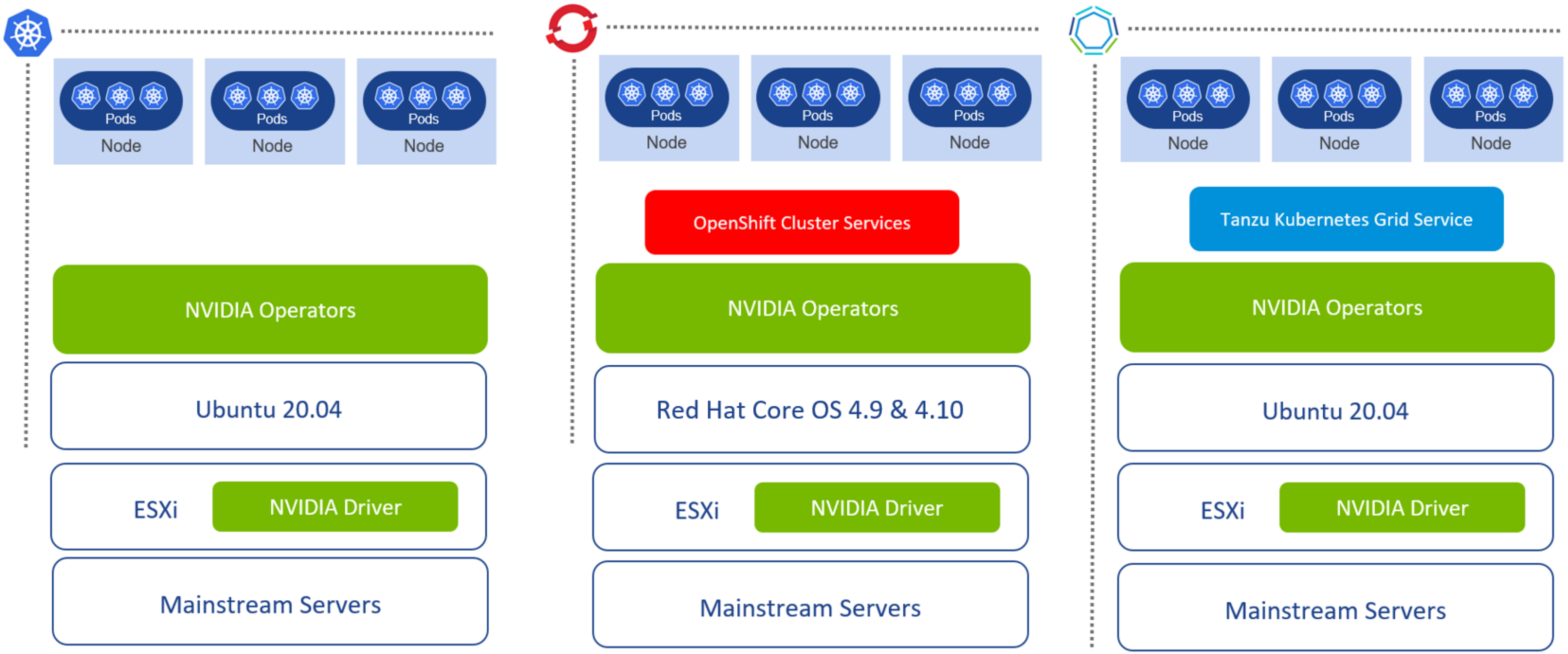
DELIVERING AI WORKLOADS WITH NVIDIA AI ENTERPRISE

Orchestration with Kubernetes on **Bare Metal**



DELIVERING AI WORKLOADS WITH NVIDIA AI ENTERPRISE

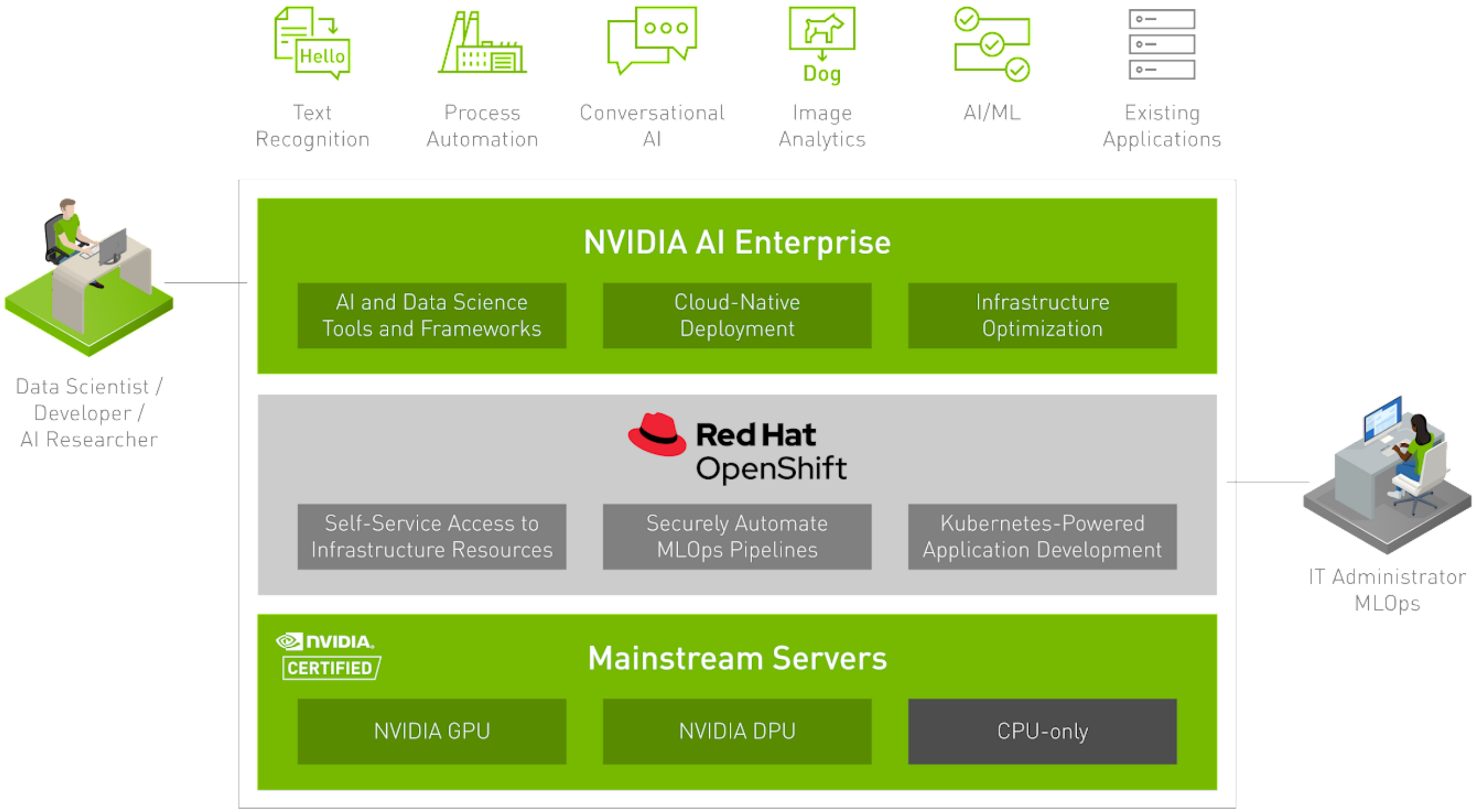
Orchestration with Kubernetes and **Virtualization**



Deploying NVIDIA AI Enterprise with Red Hat OpenShift

NVIDIA AI ENTERPRISE WITH RED HAT OPENSHIFT

Supported on Bare Metal or with VMware vSphere



AI ENTERPRISE WITH OPENSHIFT

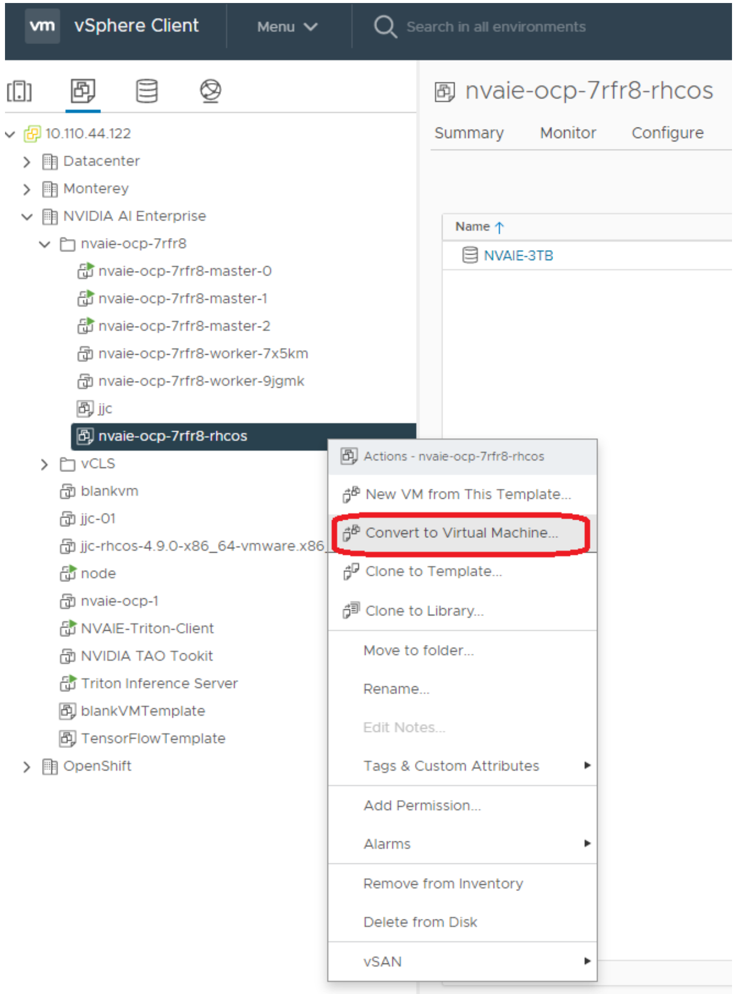
Requirements and Prerequisites for a **Virtualized** Deployment

- OpenShift CLI
  - Interact with cluster using oc
- NGC CLI
  - Pull/Push drivers, operators, containers, resources
- NVIDIA License
  - CLS or DLS instance
- NVIDIA AI Enterprise VIB on Hosts
  - Needed for virtualization
- Deployed OpenShift Cluster with **EFI Boot**
  - User Provisioned Infrastructure (UPI)
  - Installer Provisioned Infrastructure (IPI)

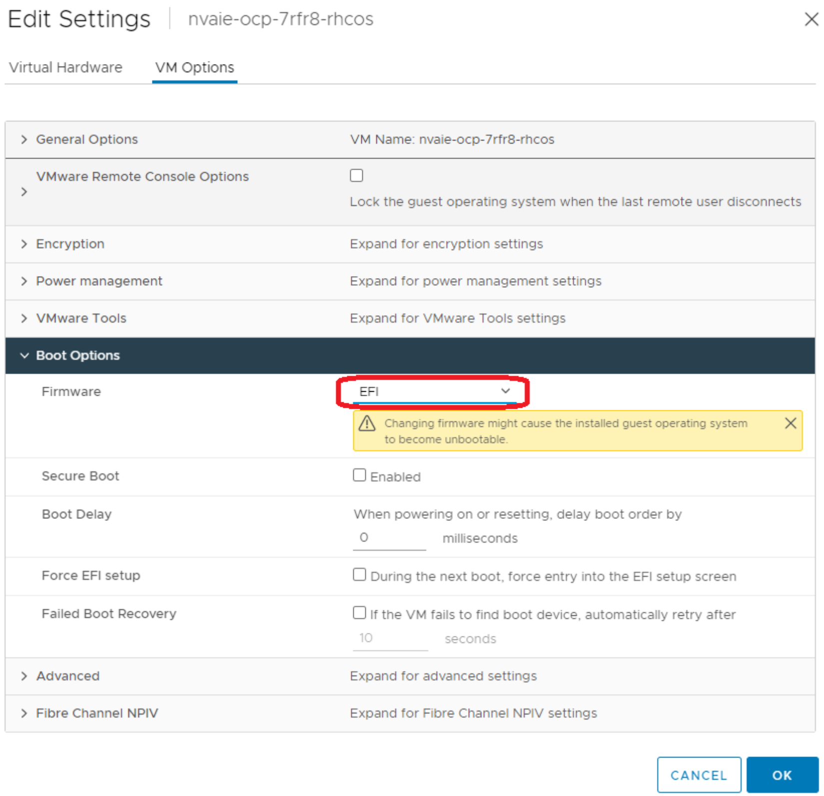
AI ENTERPRISE WITH OPENSHIFT

EFI Boot with IPI (Installer Provisioned Infrastrucutre)

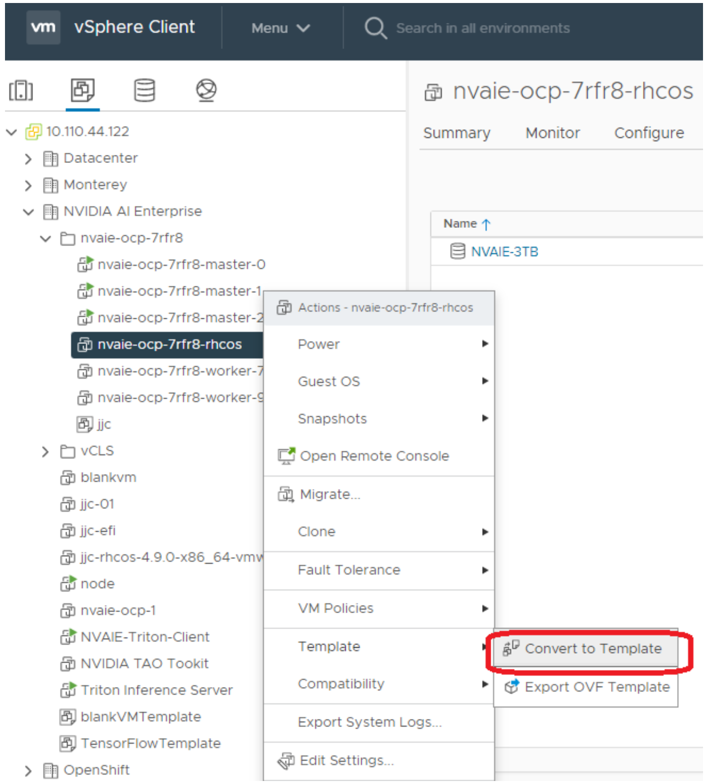
Convert to Virtual Machine



Change boot mode



Convert back to template



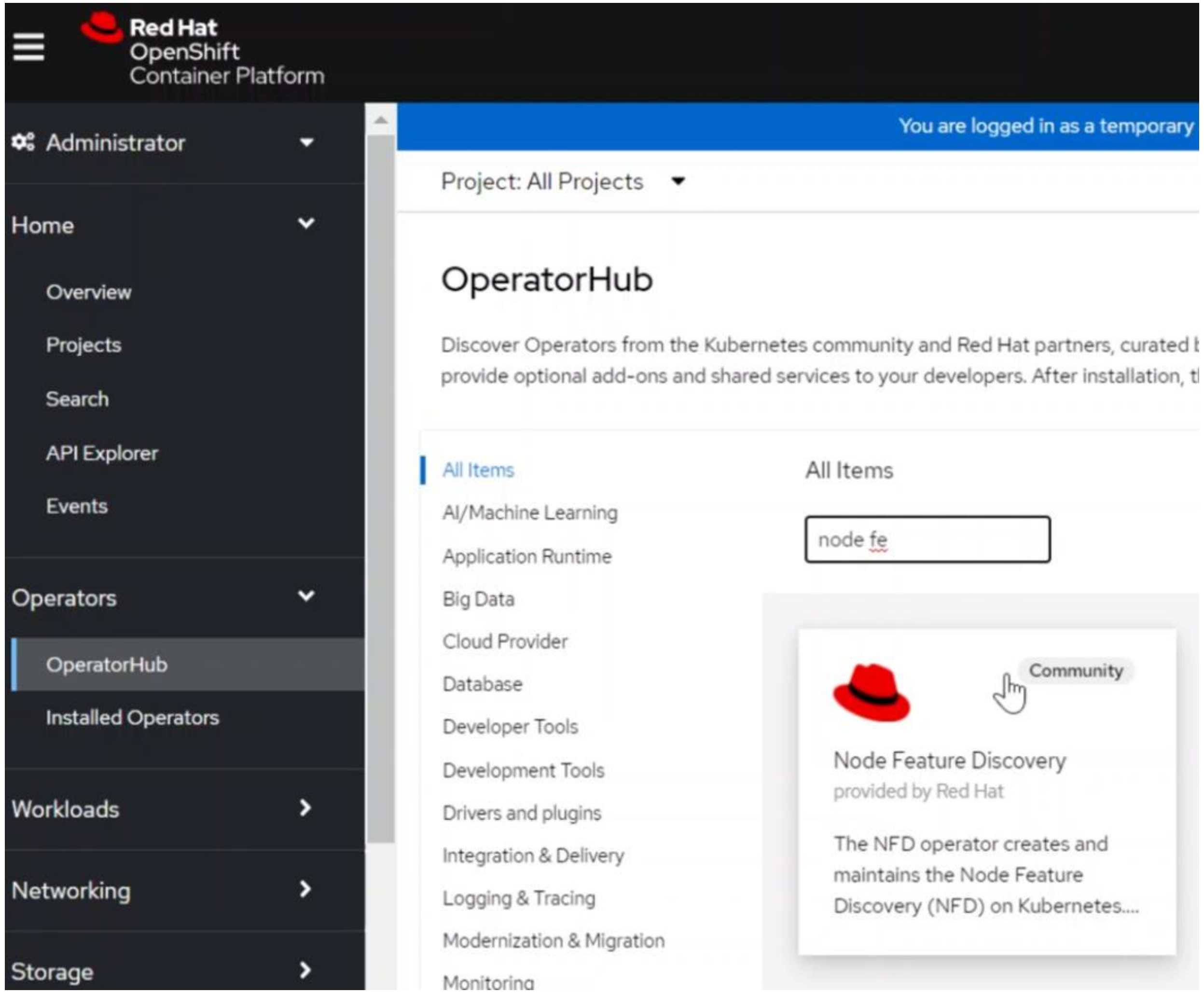
AI ENTERPRISE WITH OPENSHIFT CONTAINER PLATFORM

Orchestration on OCP

- Step 1: Install the Node Feature Discovery (NFD) Operator
- Step 2: Create NLS License Config Map
- Step 3: Import NGC Secret
- Step 4: Install the NVIDIA Network Operator (optional)
- Step 5: Install the NVIDIA GPU Operator
- Step 6: Create the Cluster Policy Instance

AI ENTERPRISE WITH OPENSHIFT

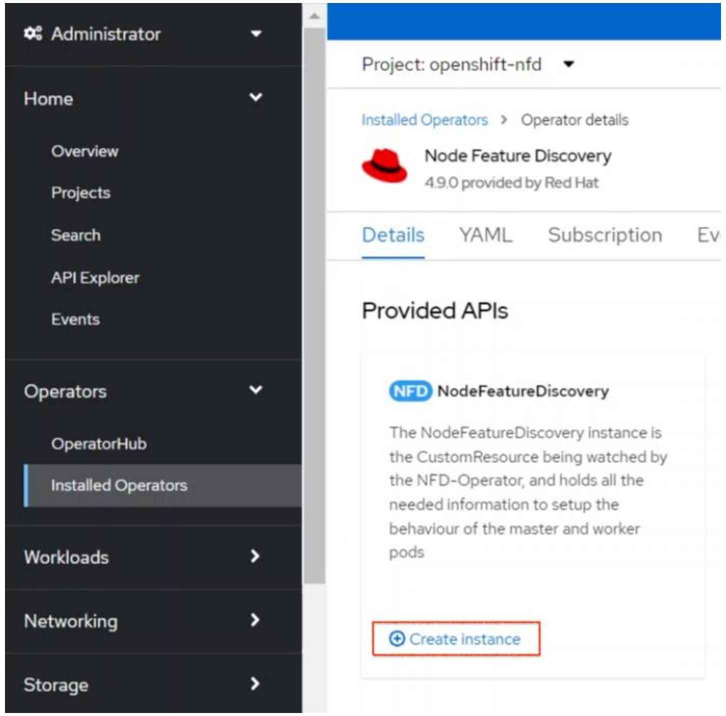
Step 1: Install the Node Feature Discovery (NFD) Operator

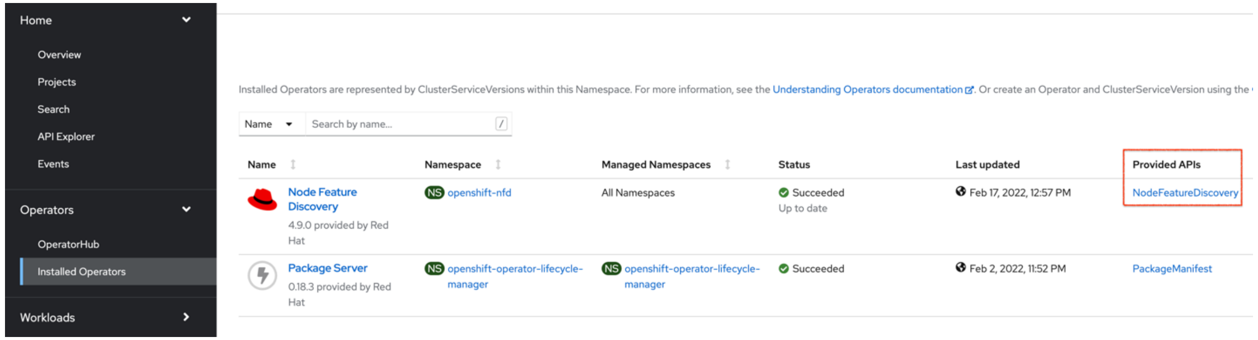


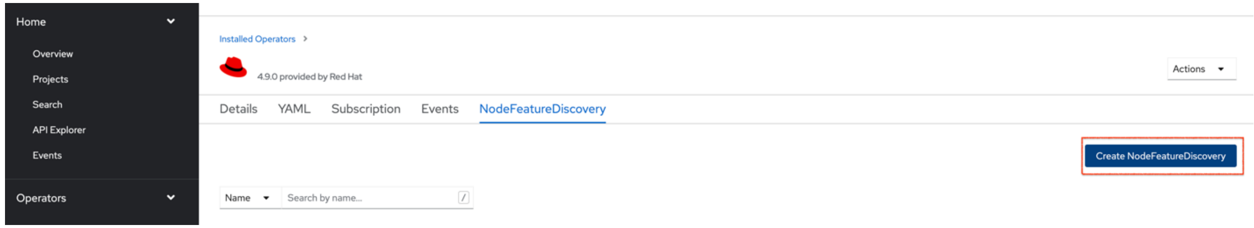

```
$ oc get pods -n openshift-nfd
NAME                                 READY   STATUS    RESTARTS   AGE
nfd-controller-manager-7f86ccfb58-nqgxm   2/2     Running   0          11m
```

AI ENTERPRISE WITH OPENSHIFT

Step 1: Install the Node Feature Discovery (NFD) Operator

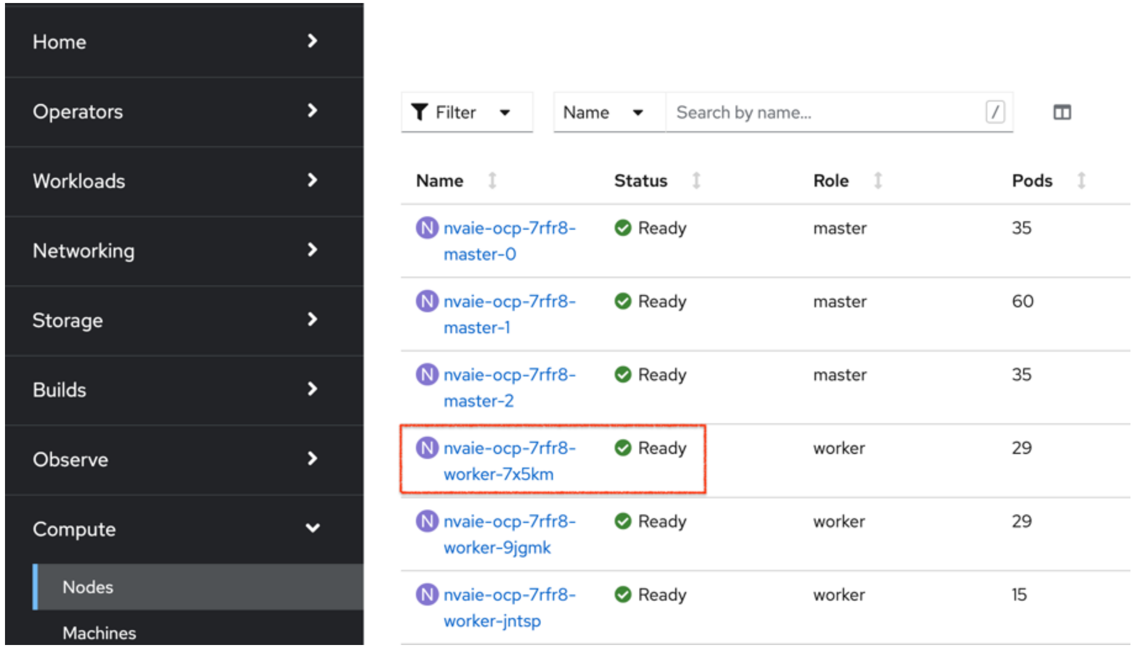


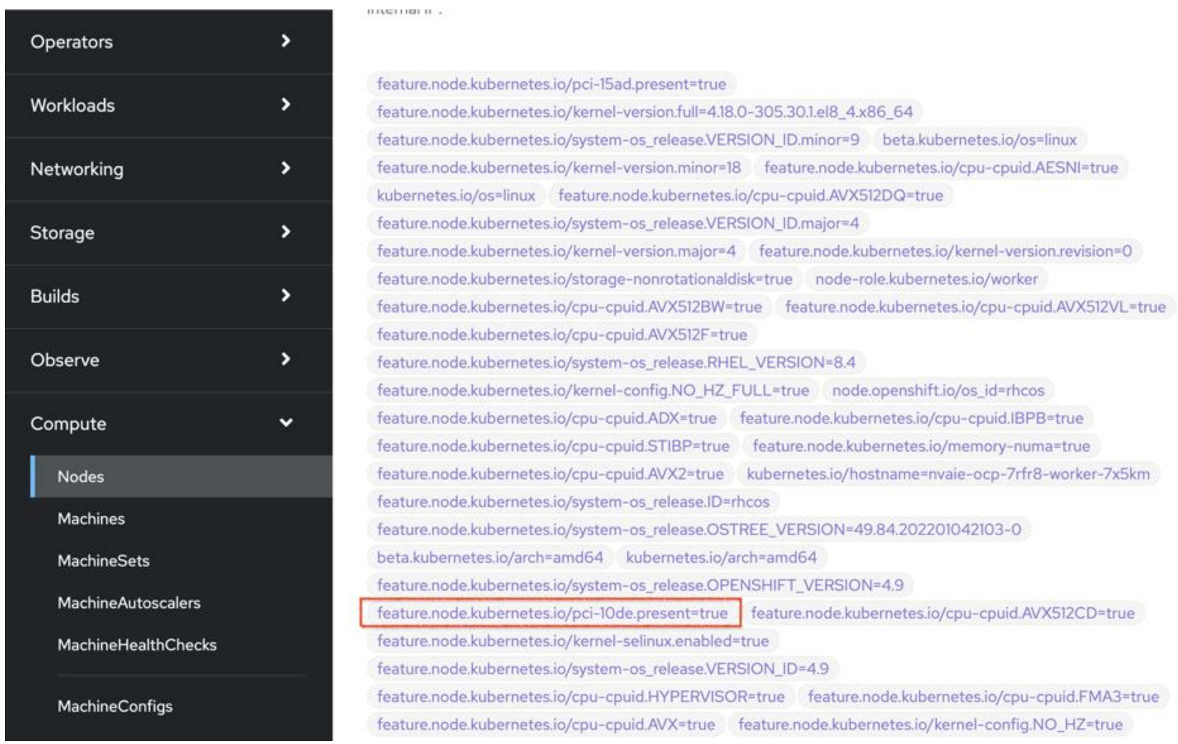




AI ENTERPRISE WITH OPENSHIFT

Step 1: Install the Node Feature Discovery (NFD) Operator





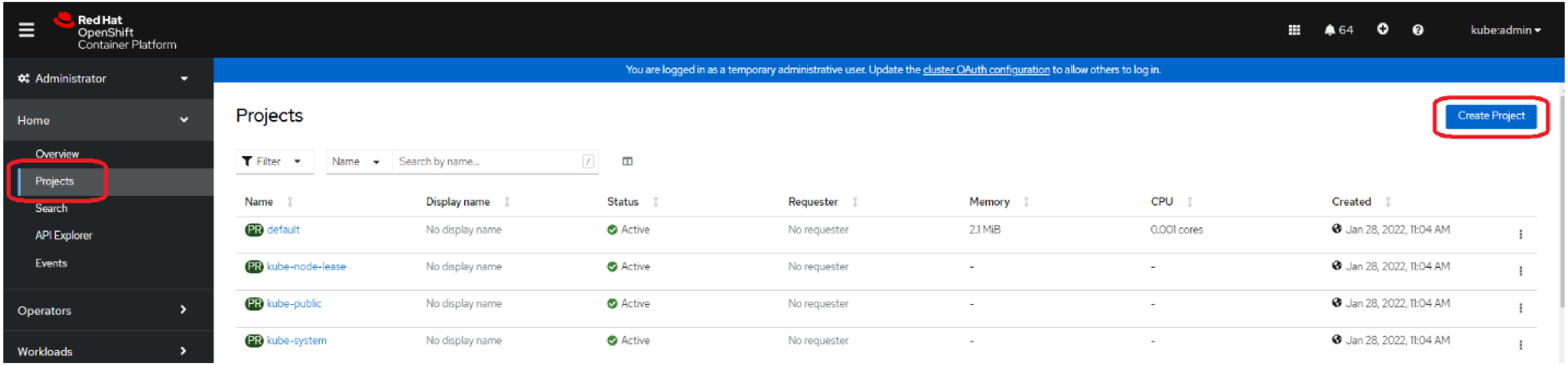
AI ENTERPRISE WITH OPENSHIFT

Step 1: Install the Node Feature Discovery (NFD) Operator

```
$ oc get nodes -l feature.node.kubernetes.io/pci-10de.present
NAME                         STATUS   ROLES    AGE   VERSION
nvaie-ocp-7rfr8-worker-7x5km   Ready    worker   20d   v1.22.3+e790d7f
nvaie-ocp-7rfr8-worker-jntsp   Ready    worker   11d   v1.22.3+e790d7f
```

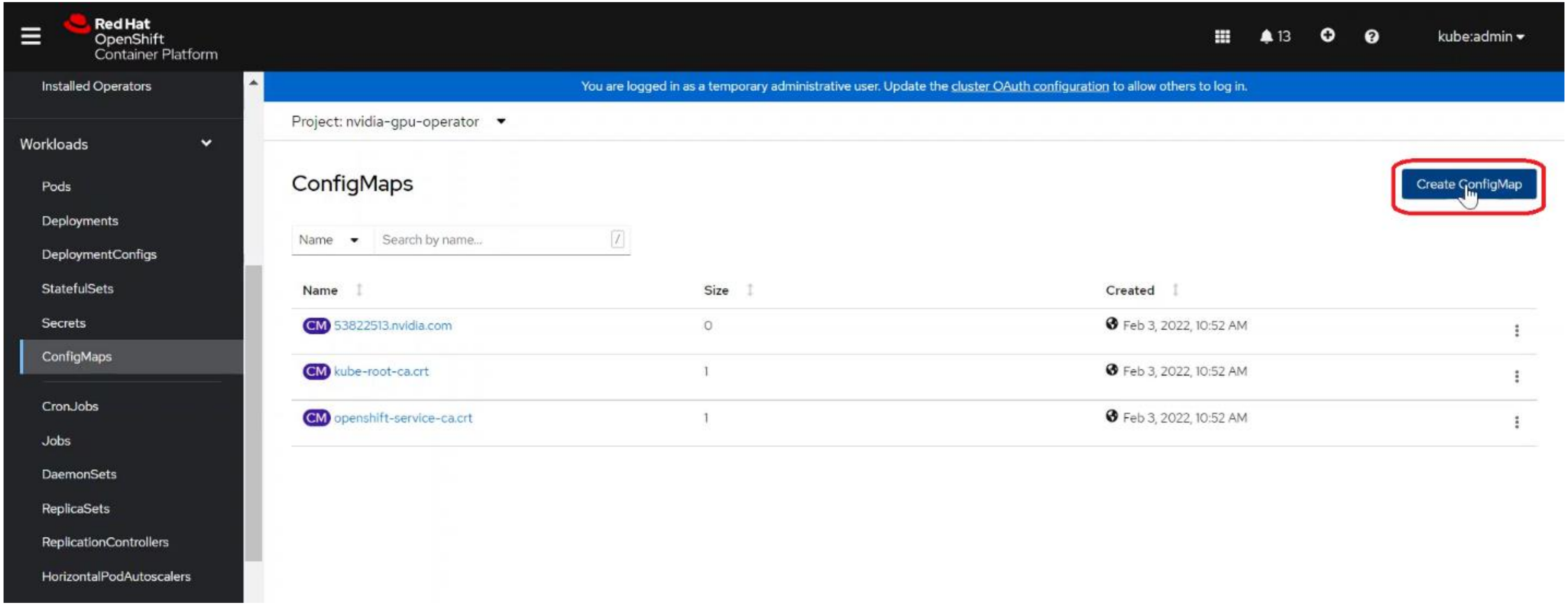
AI ENTERPRISE WITH OPENSHIFT

Step 2: Create NLS License Config Map



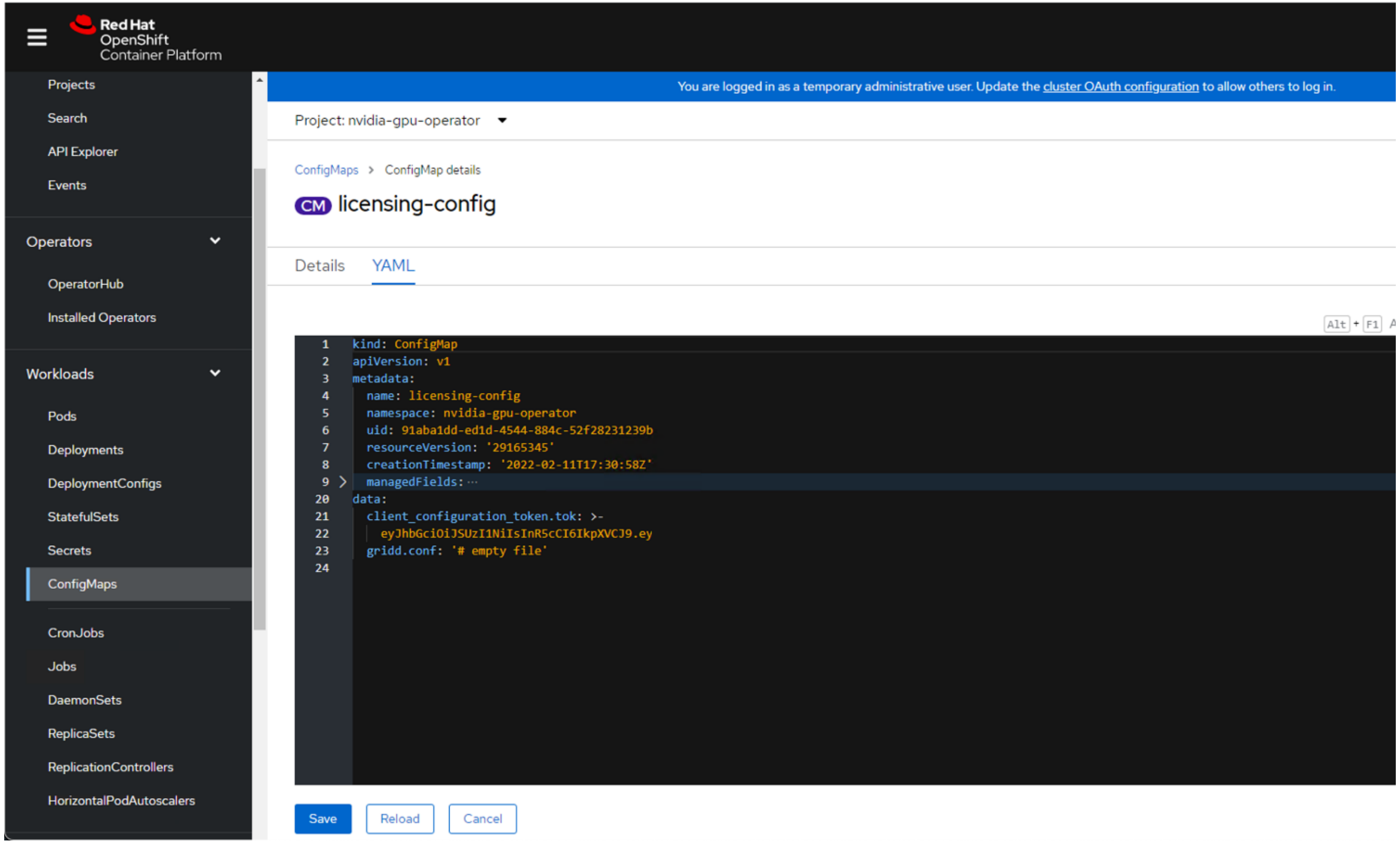
AI ENTERPRISE WITH OPENSHIFT

Step 2: Create NLS License Config Map



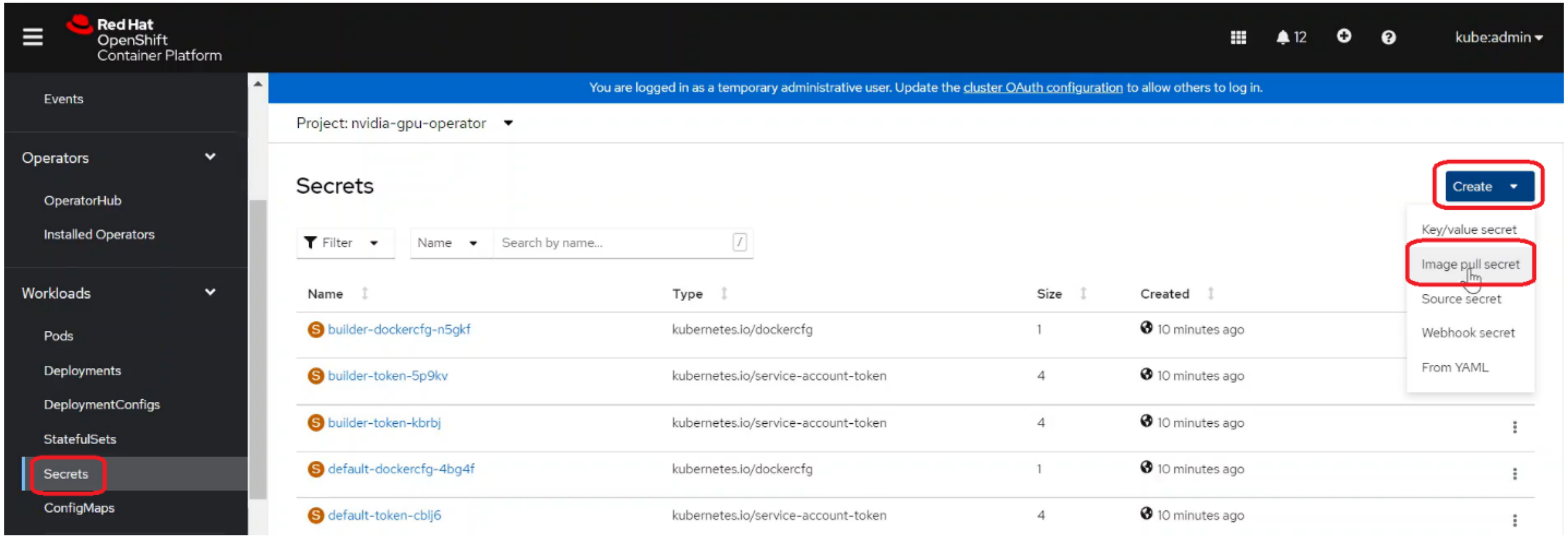
AI ENTERPRISE WITH OPENSHIFT

Step 2: Create NLS License Config Map



AI ENTERPRISE WITH OPENSHIFT

Step 3: Import NGC Secret



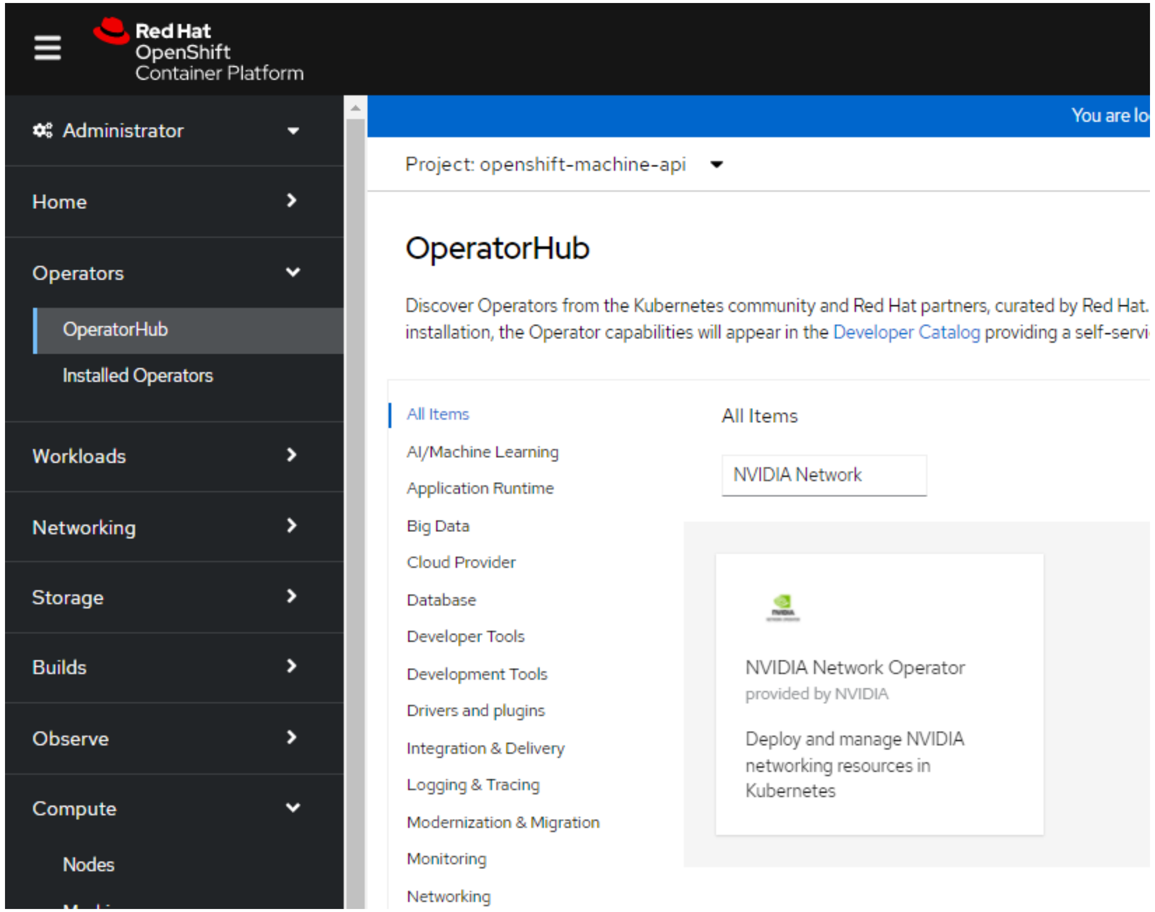
AI ENTERPRISE WITH OPENSHIFT

Step 3: Import NGC Secret

Secret name: `gpu-operator-secret`
Authentication type: `Image registry credentials`
Registry server address: `nvcr.io/nvaie`
Username: `$oauthtoken`
Password: `<API-KEY>`
Email: `<YOUR-EMAIL>`

AI ENTERPRISE WITH OPENSHIFT

Step 4: Install the NVIDIA Network Operator (Optional)



AI ENTERPRISE WITH OPENSHIFT

Step 4: Install the NVIDIA Network Operator (Optional)

NVIDIA Network Operator

1.1.0 provided by NVIDIA

Install

Latest version

1.1.0

Capability level

- Basic Install
- Seamless Upgrades
- Full Lifecycle
- Deep Insights
- Auto Pilot

Source

Certified

Provider

NVIDIA

Repository

https://github.com/Mellanox/network-operator/

Container image

N/A

Created at

N/A

Support

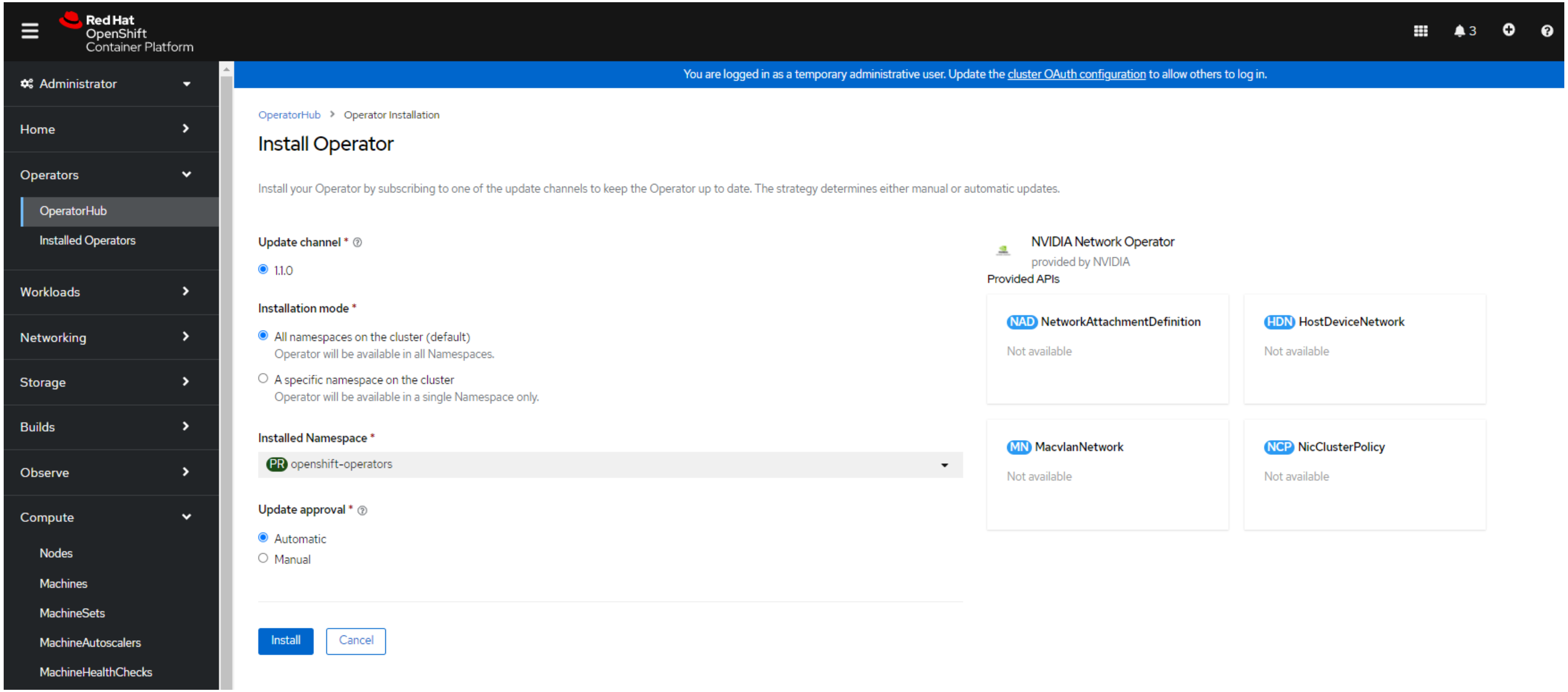
NVIDIA

NVIDIA Network Operator

The NVIDIA Network Operator simplifies the provisioning and management of NVIDIA networking resources in a Kubernetes cluster. The operator automatically installs the required host networking software - bringing together all the needed components to provide high-speed network connectivity. These components include the NVIDIA networking driver, Kubernetes device plugin, CNI plugins, IP address management (IPAM) plugin and others. The NVIDIA Network Operator works in conjunction with the NVIDIA GPU Operator to deliver high-throughput, low-latency networking for scale-out, GPU computing clusters. The Network Operator is open-source. For more information on contributions and release artifacts, see the GitHub repo

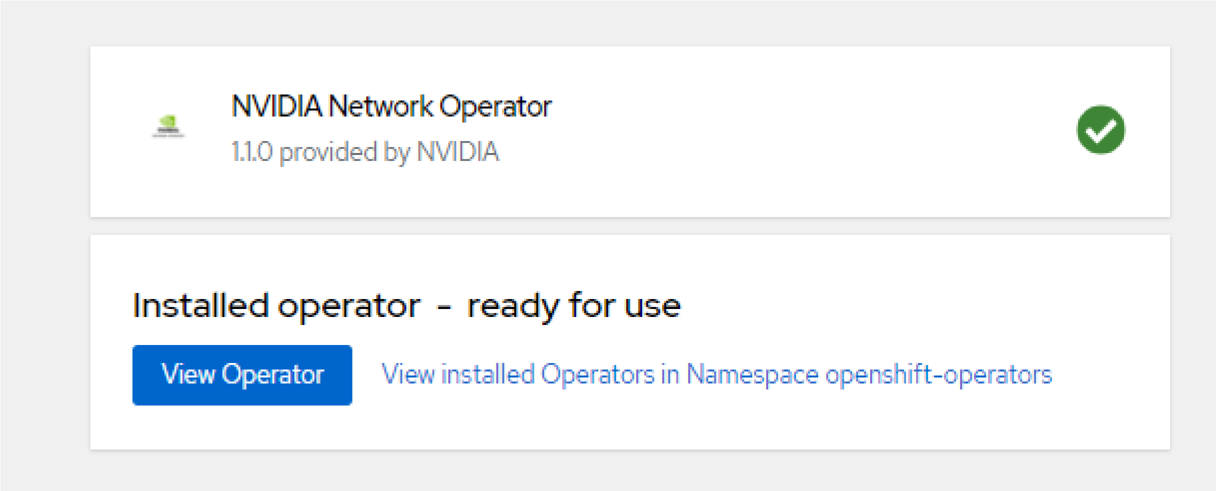
AI ENTERPRISE WITH OPENSHIFT

Step 4: Install the NVIDIA Network Operator (Optional)



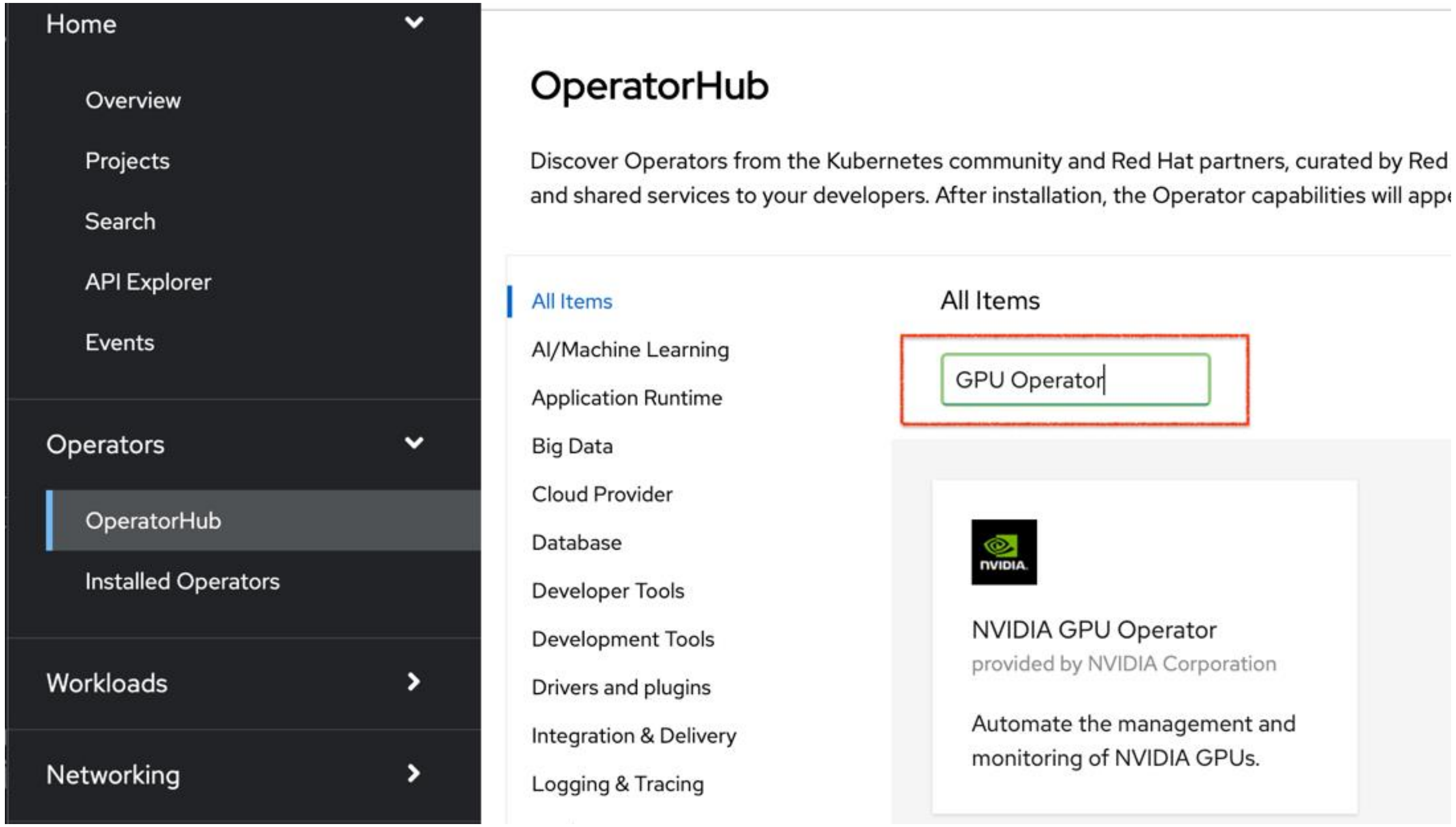
AI ENTERPRISE WITH OPENSHIFT

Step 4: Install the NVIDIA Network Operator (Optional)



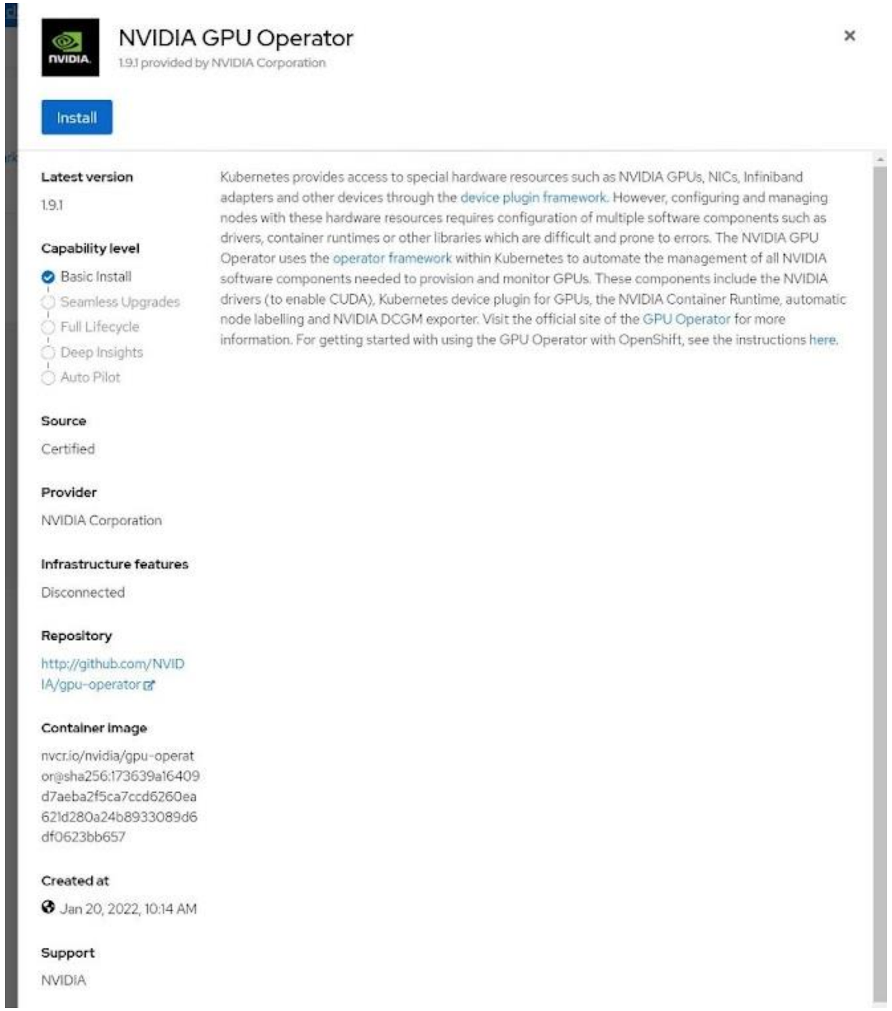
AI ENTERPRISE WITH OPENSHIFT

Step 5: Install GPU Operator



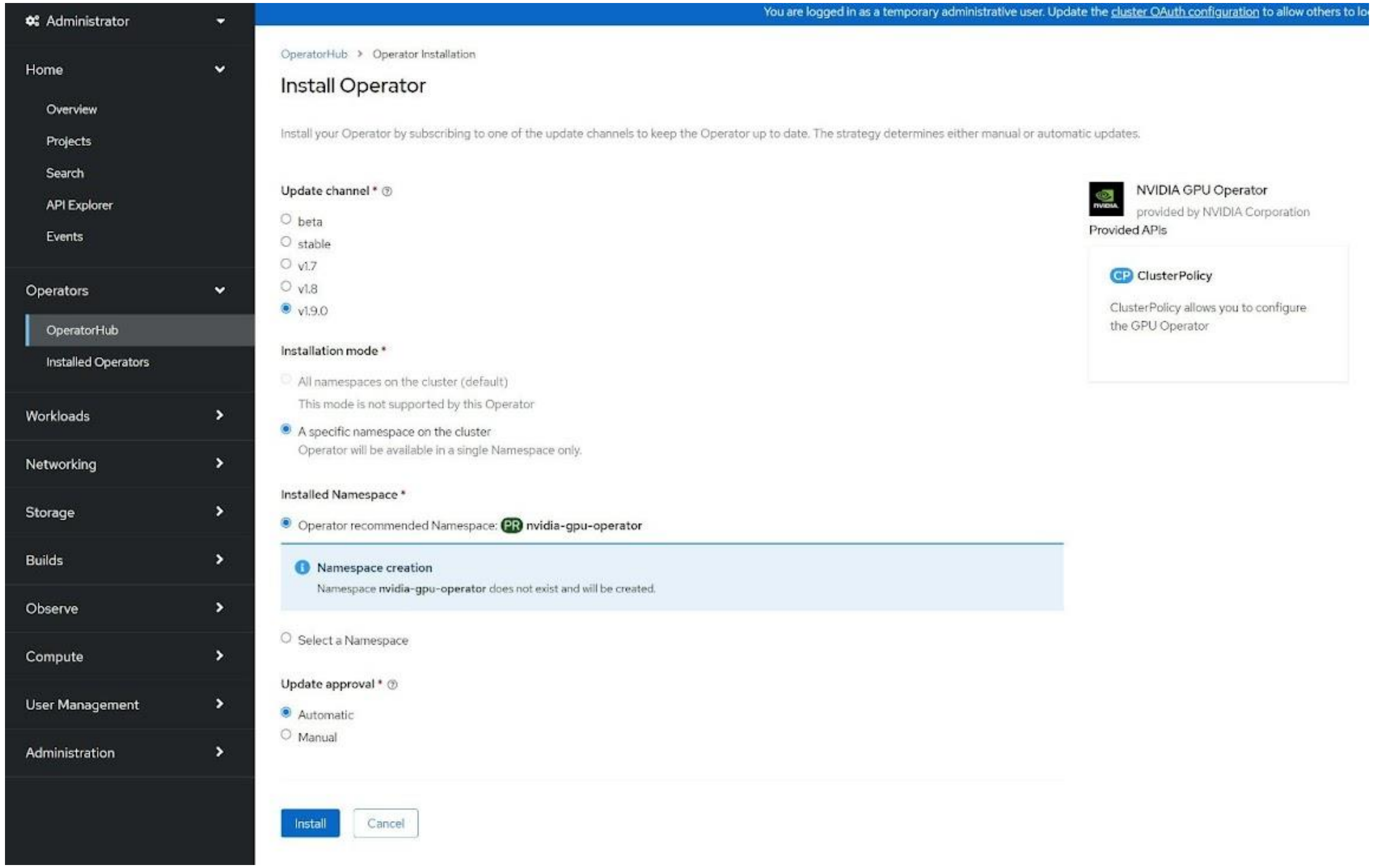
AI ENTERPRISE WITH OPENSHIFT

Step 5: Install GPU Operator



AI ENTERPRISE WITH OPENSHIFT

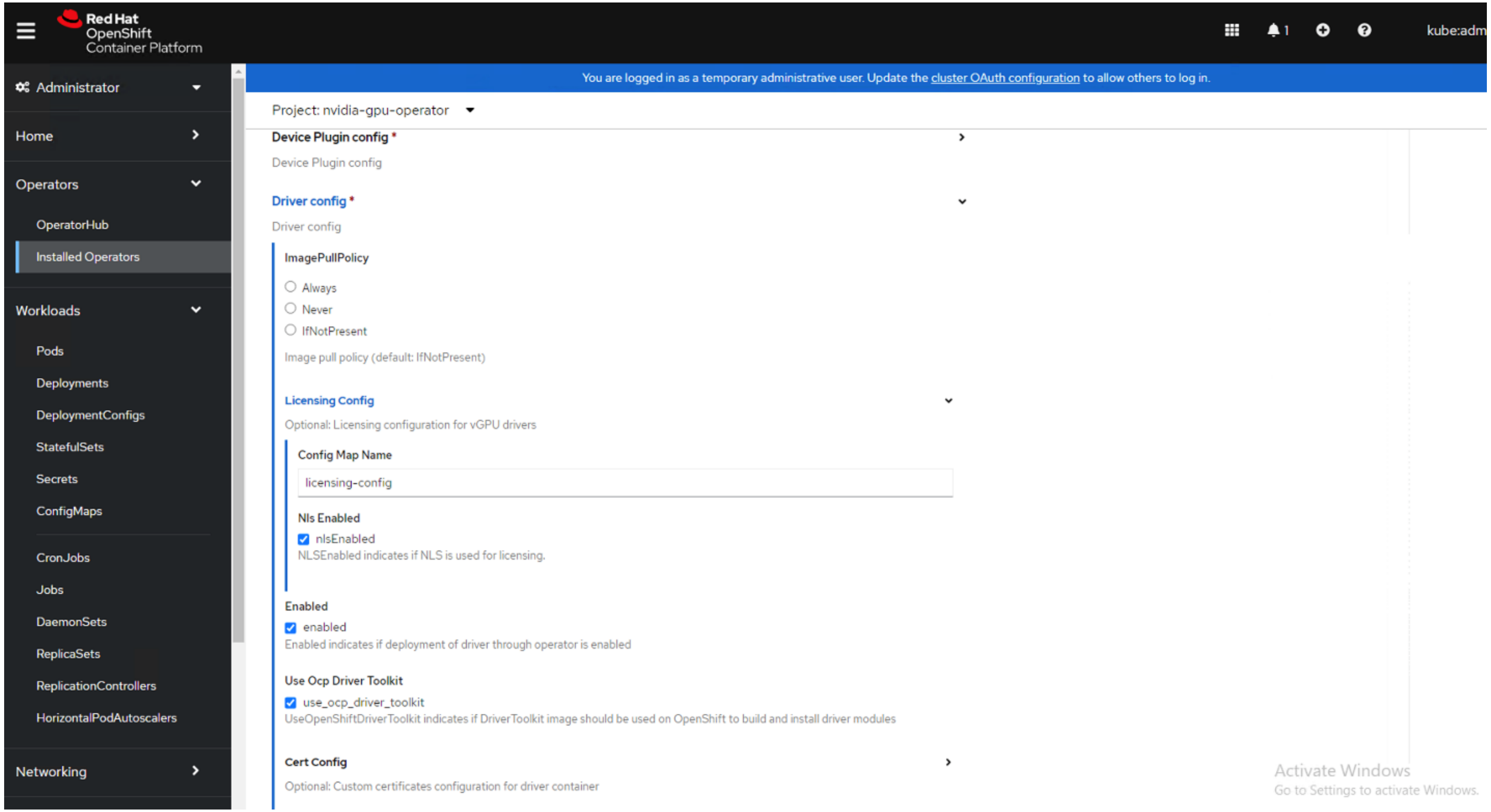
Step 5: Install GPU Operator




```
$ oc label ns/$NAMESPACE_NAME openshift.io/cluster-monitoring=true
```

AI ENTERPRISE WITH OPENSHIFT

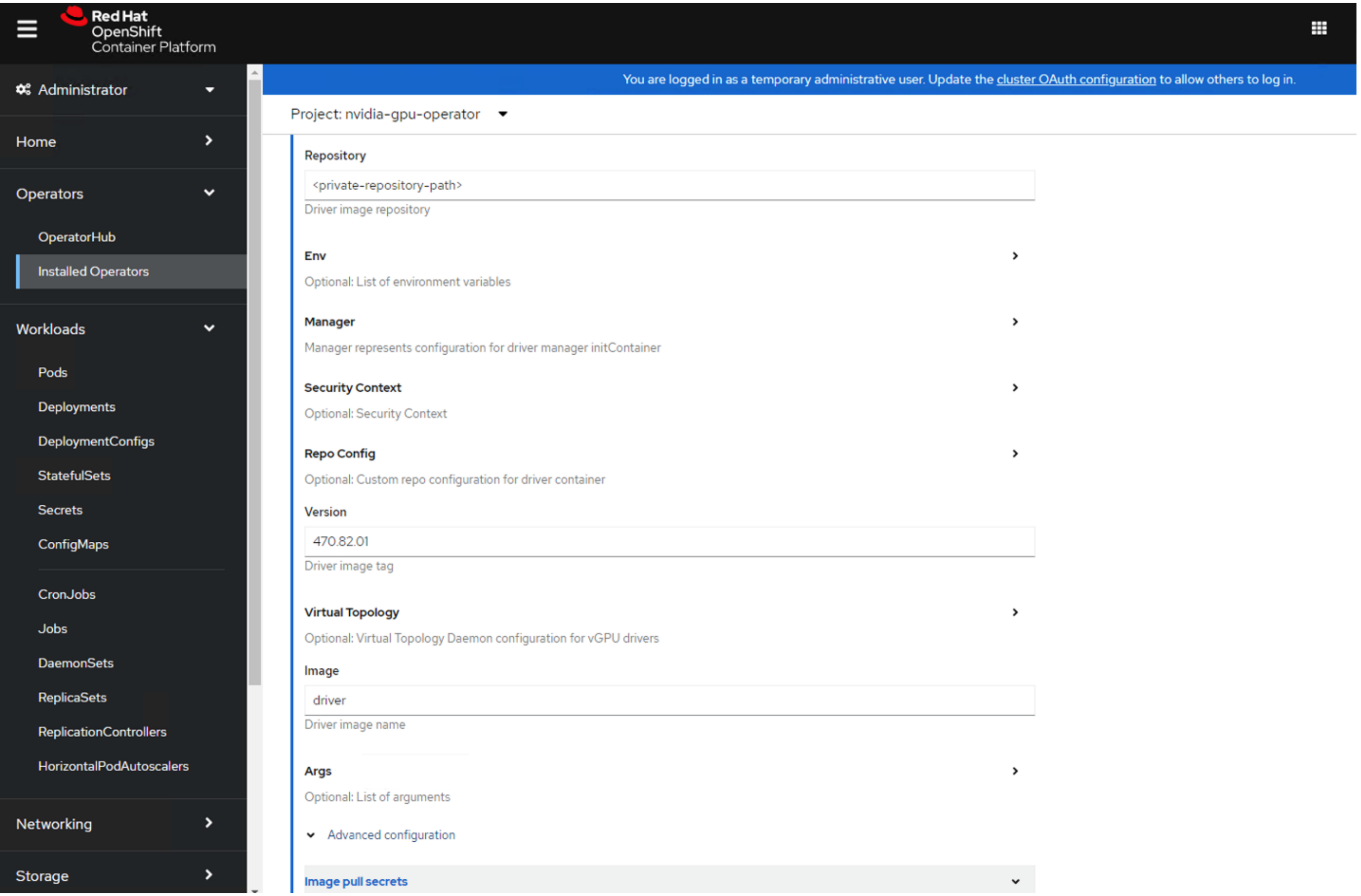
Step 6: Create the Cluster Policy Instance



NVIDIA GTC

AI ENTERPRISE WITH OPENSHIFT

Step 6: Create the Cluster Policy Instance




```
nlsEnabled: true
repository: nvcr.io/nvaie
version: 510.47.03
image: vgpu-guest-driver
```

Advanced configuration menu and specify the `imagePullSecret`. (eg: *gpu-operator-secret*

AI ENTERPRISE WITH OPENSHIFT

Step 6: Create the Cluster Policy Instance

Project: nvidia-gpu-operator

Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the Understanding Operators documentation. Or create an Operator and ClusterServiceVersion using the Operator SDK.

Name | Search by name...

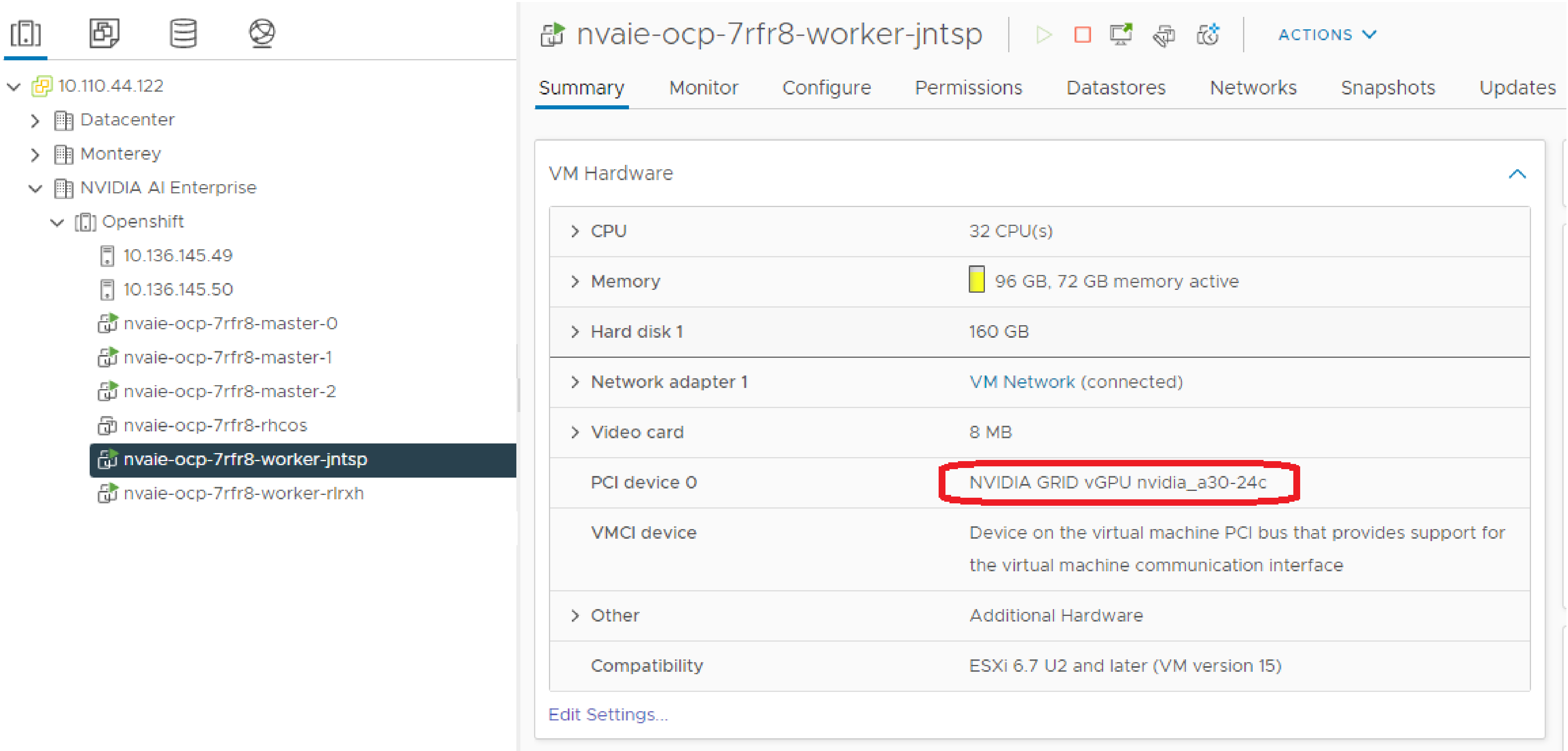
Name	Managed Namespaces	Status	Last updated	Provided APIs
NVIDIA GPU Operator 1.9.0 provided by NVIDIA Corporation	NS nvidia-gpu-operator	Succeeded Up to date	1 minute ago	ClusterPolicy

```
$ oc get nodes -o=custom-
columns='Node:metadata.name,GPUs:status.capacity.nvidia\.com/gpu'
Node                           GPUs
nvaie-ocp-7rfr8-master-0       <none>
nvaie-ocp-7rfr8-master-1       <none>
nvaie-ocp-7rfr8-master-2       <none>
nvaie-ocp-7rfr8-worker-7x5km   1
nvaie-ocp-7rfr8-worker-9jgmk   <none>
nvaie-ocp-7rfr8-worker-jntsp   1
nvaie-ocp-7rfr8-worker-zkggt   <none>
```

Orchestration with OCP and vSphere

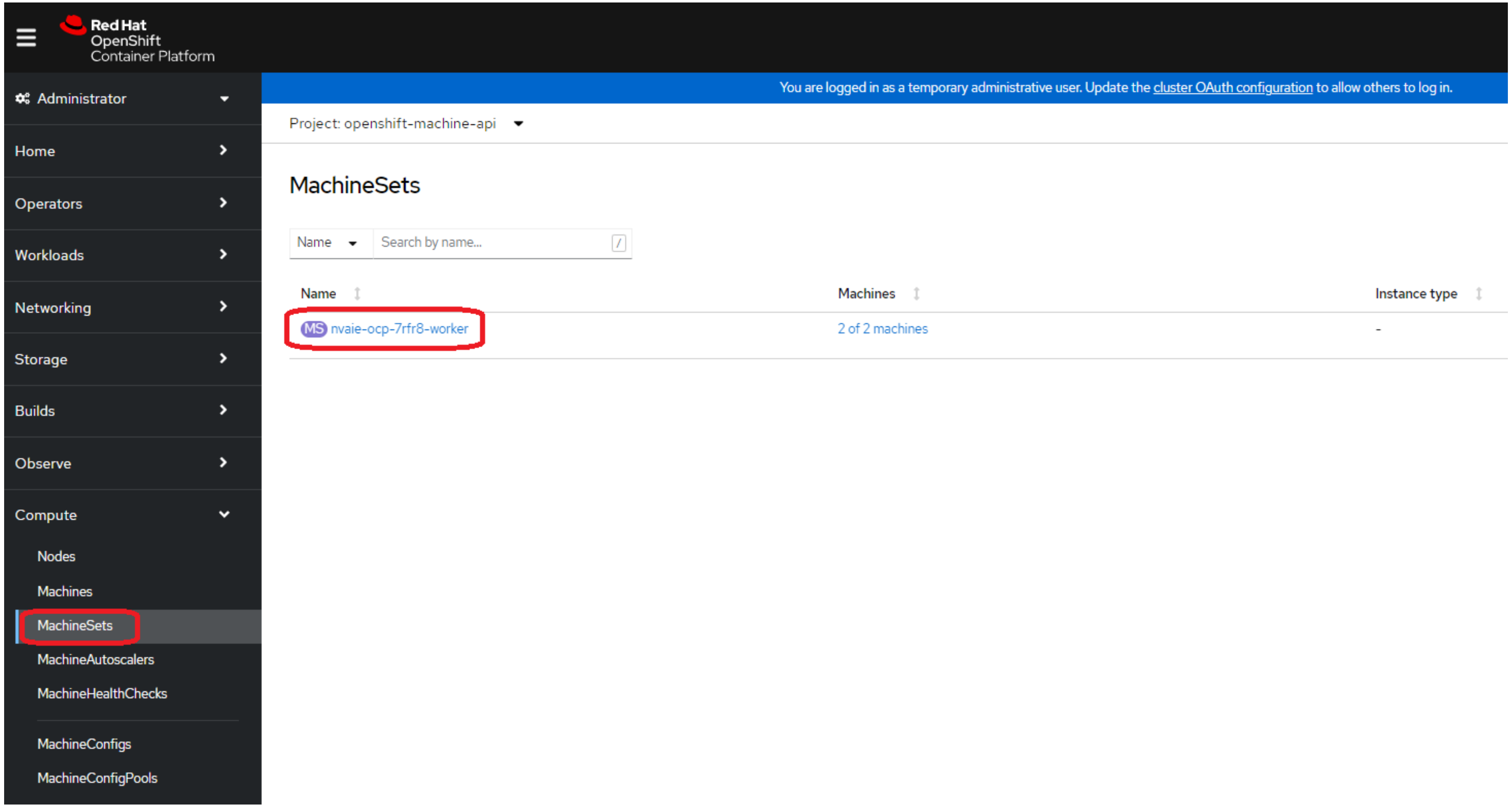
AI ENTERPRISE WITH OPENSHIFT AND VSPHERE

Scaling OpenShift with MachineSets



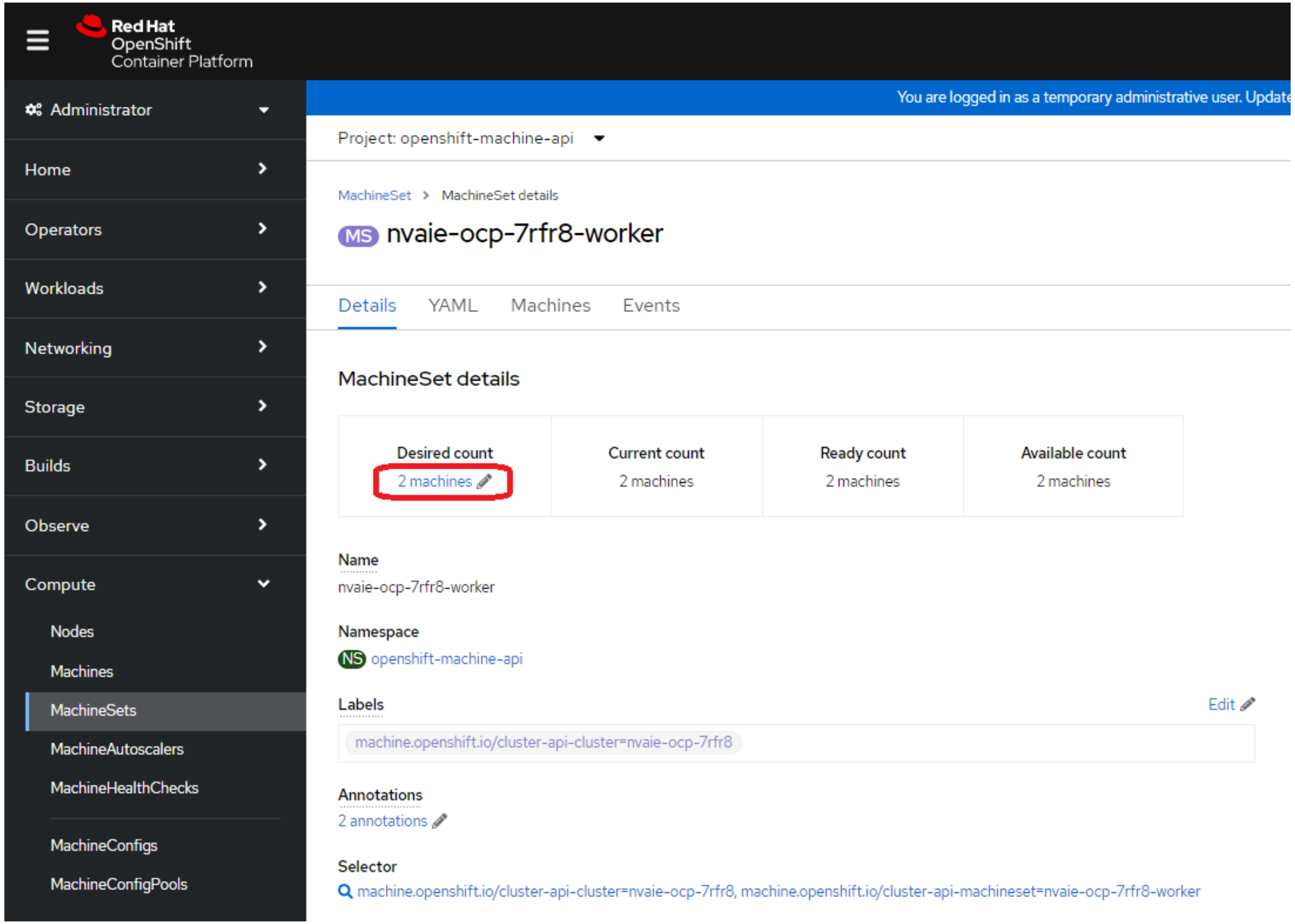
AI ENTERPRISE WITH OPENSHIFT AND VSPHERE

Scaling OpenShift with MachineSets



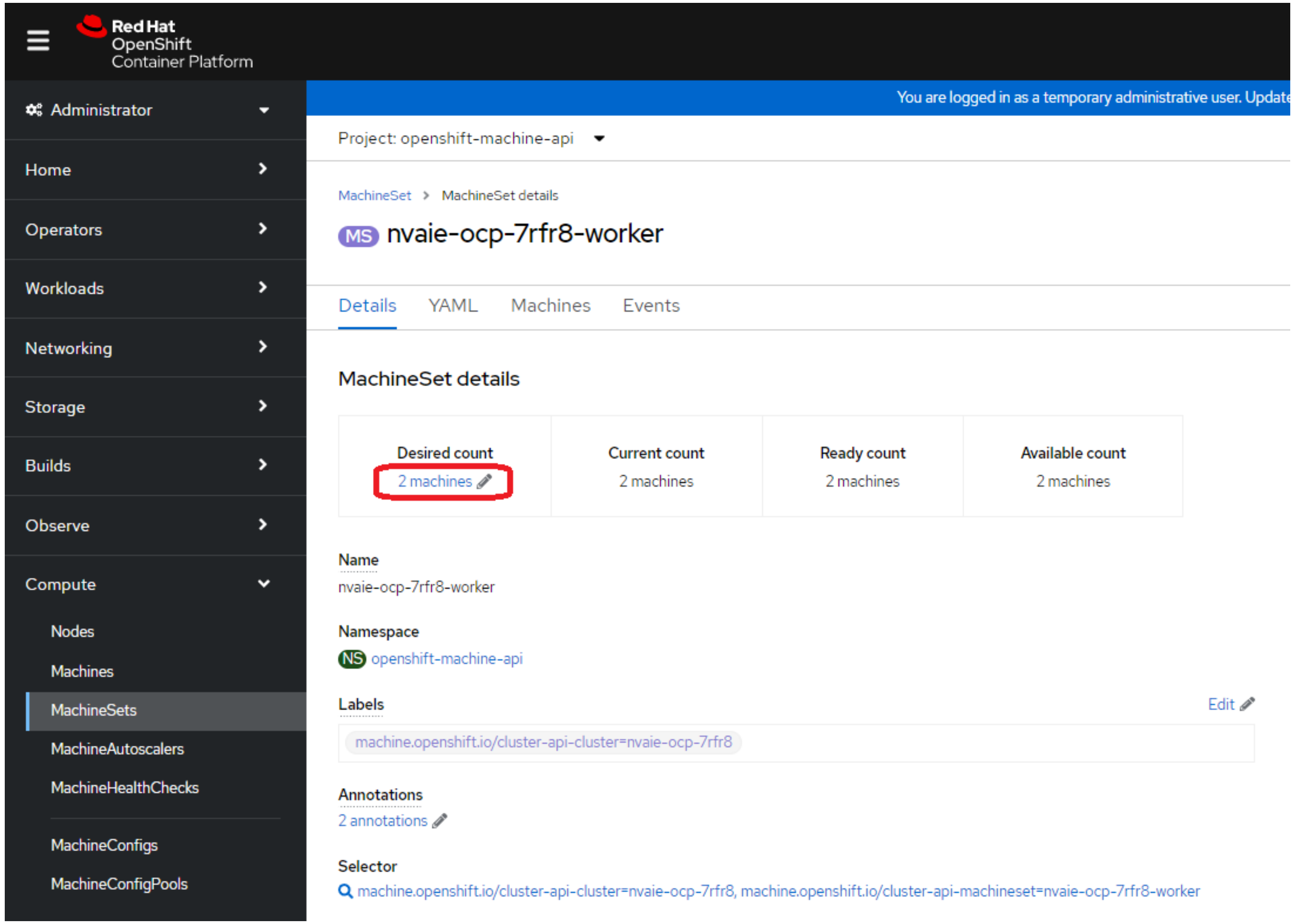
AI ENTERPRISE WITH OPENSHIFT AND VSPHERE

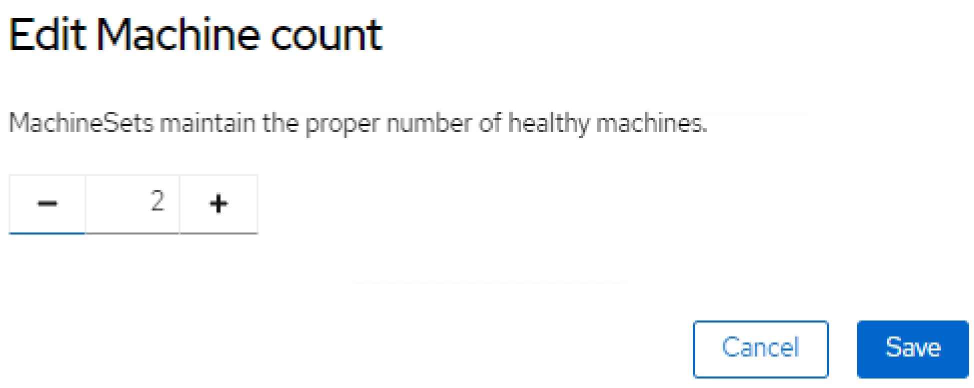
Scaling OpenShift with MachineSets



AI ENTERPRISE WITH OPENSHIFT AND VSPHERE

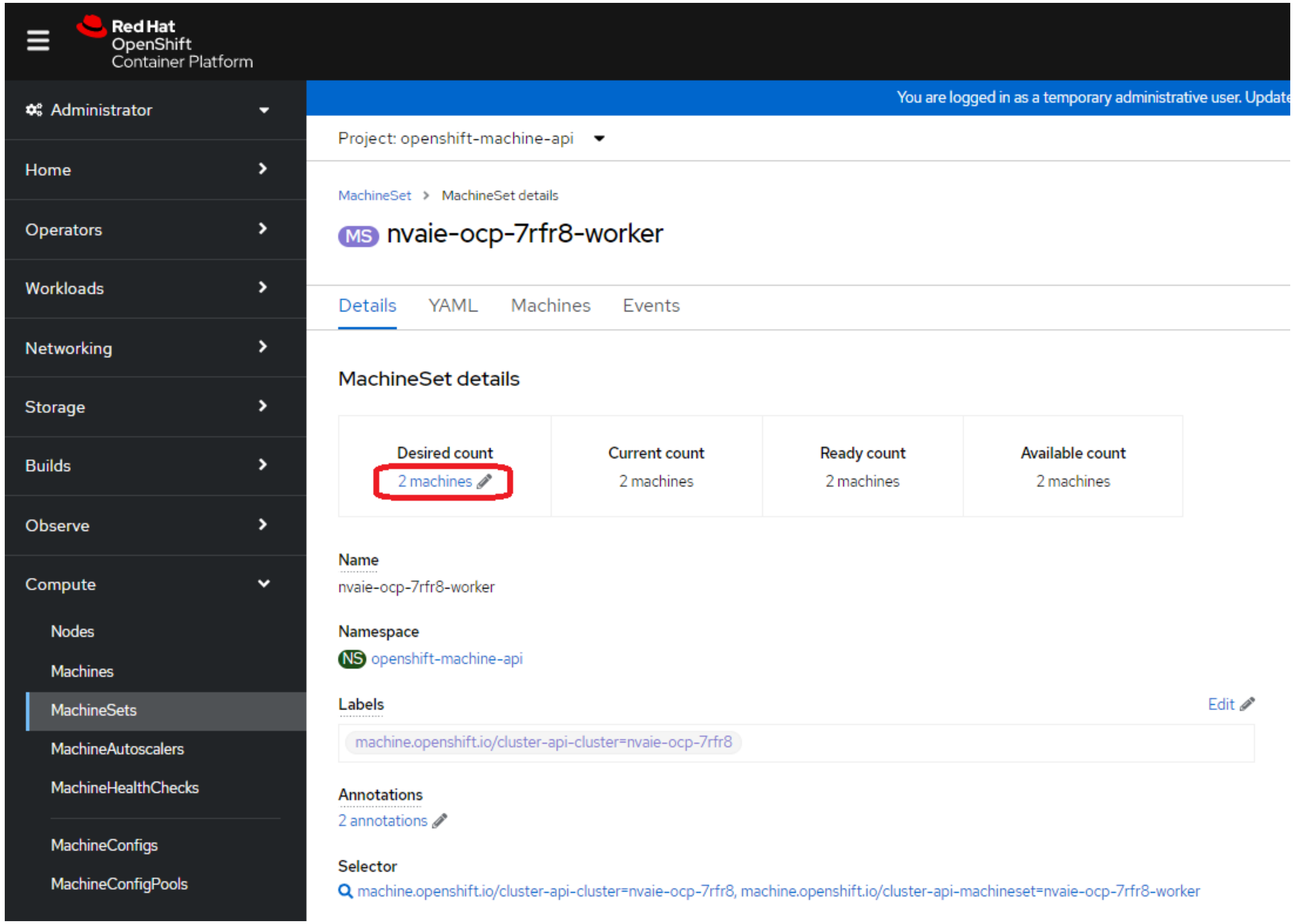
Scaling OpenShift with MachineSets

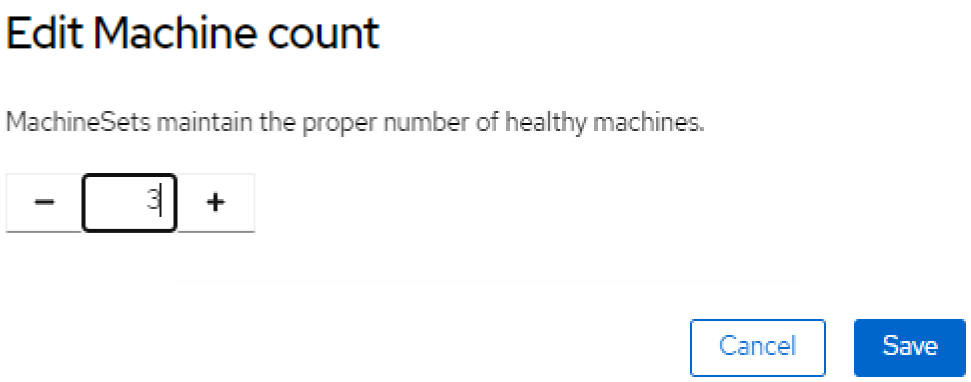




AI ENTERPRISE WITH OPENSHIFT AND VSPHERE

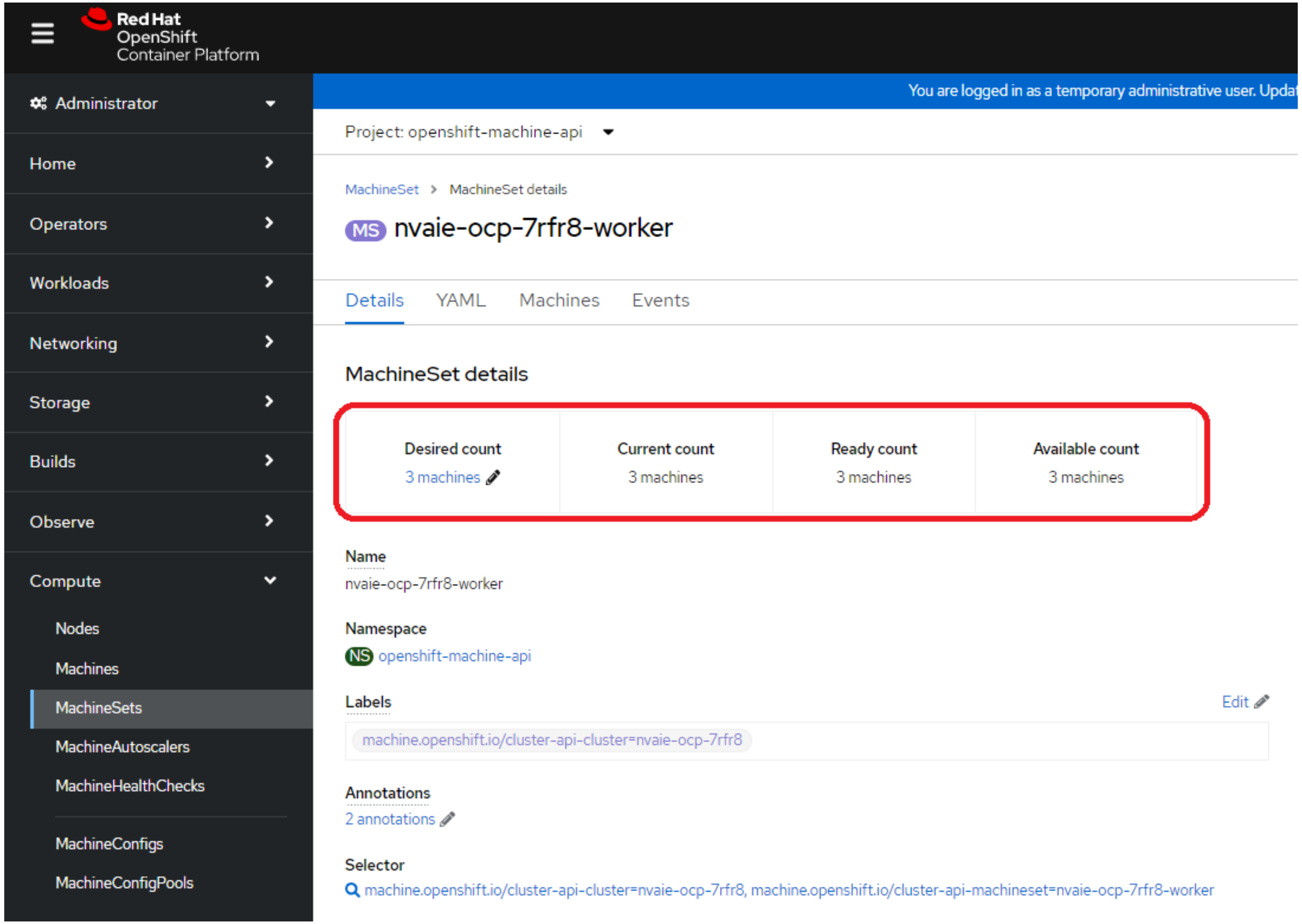
Scaling OpenShift with MachineSets





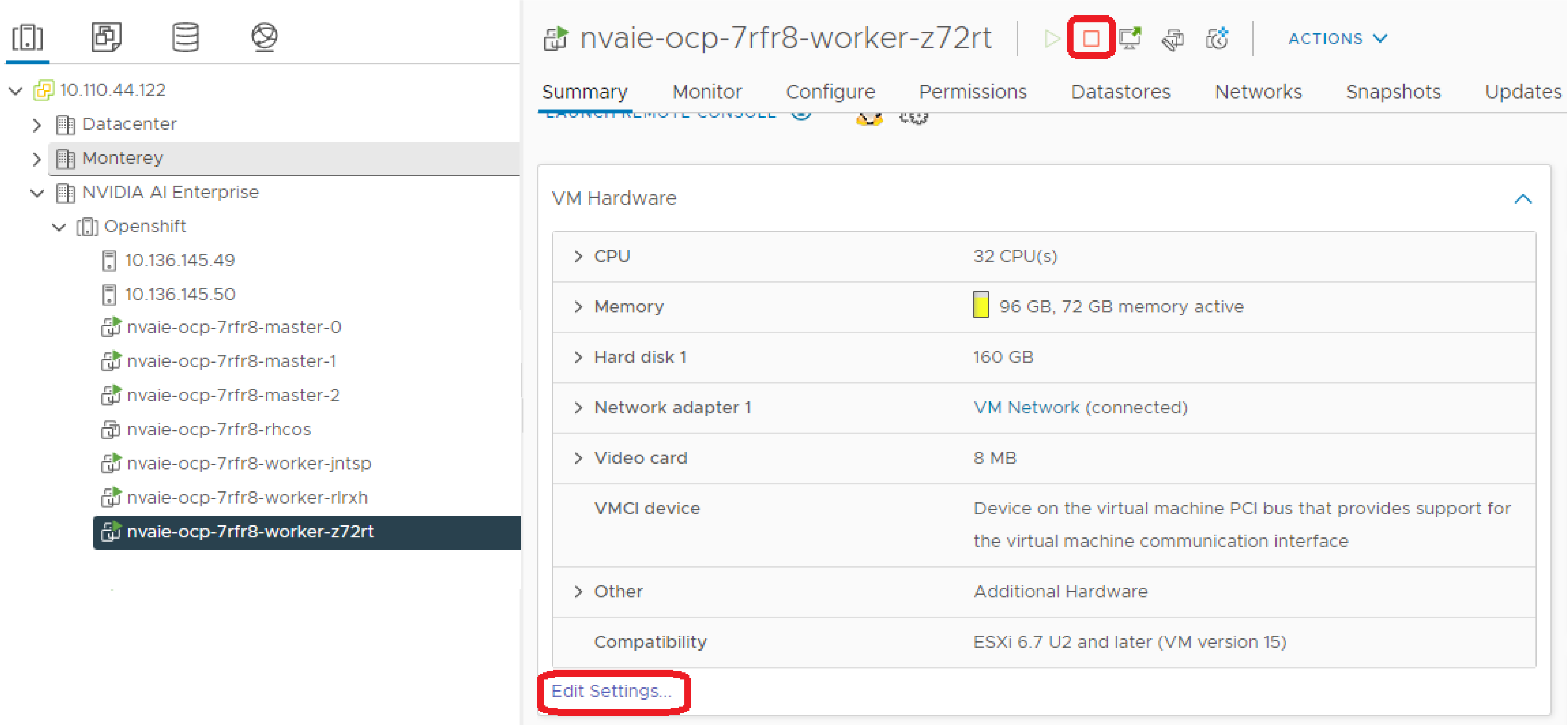
AI ENTERPRISE WITH OPENSHIFT AND VSPHERE

Scaling OpenShift with MachineSets



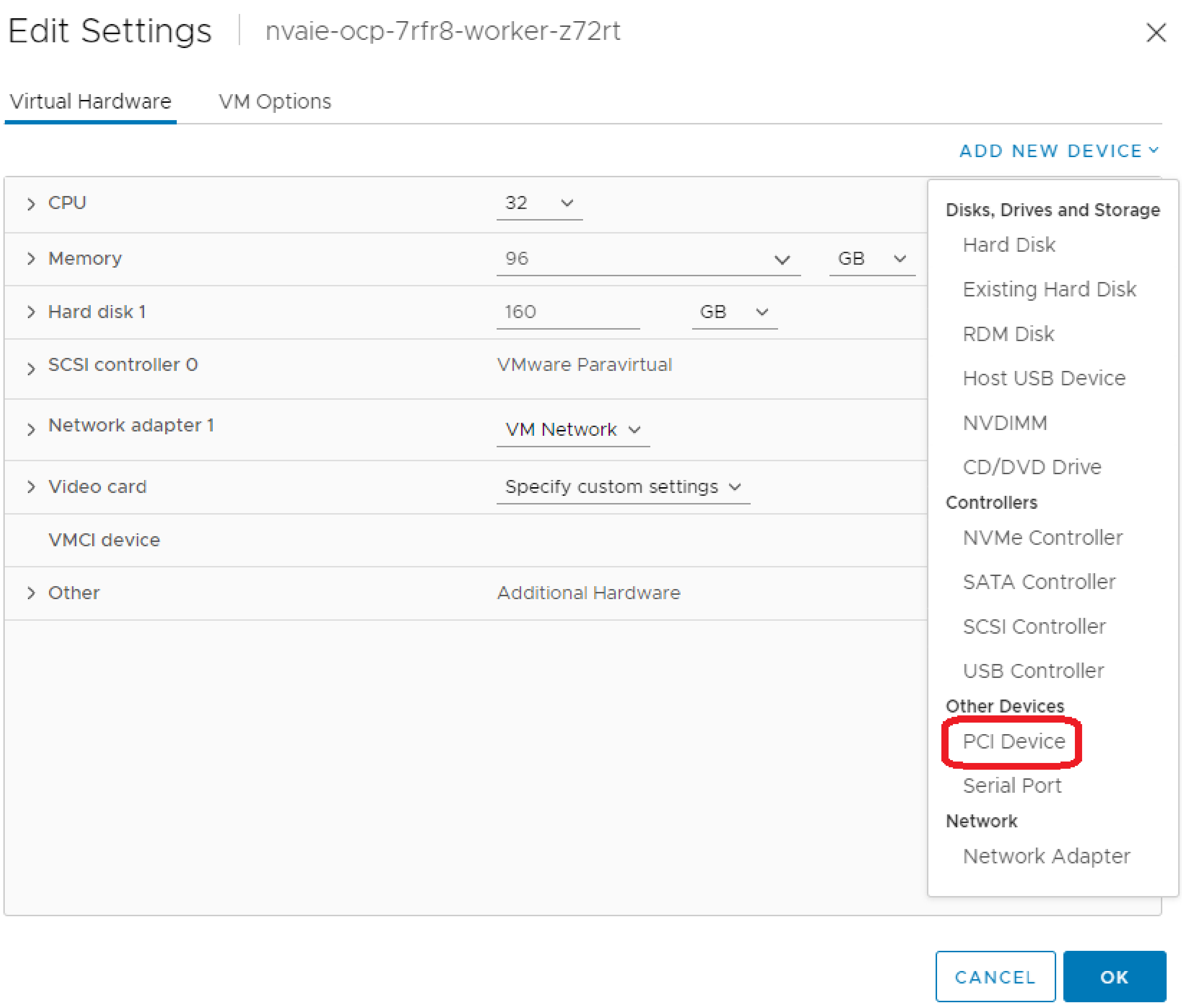
AI ENTERPRISE WITH OPENSHIFT AND VSPHERE

Scaling OpenShift with MachineSets



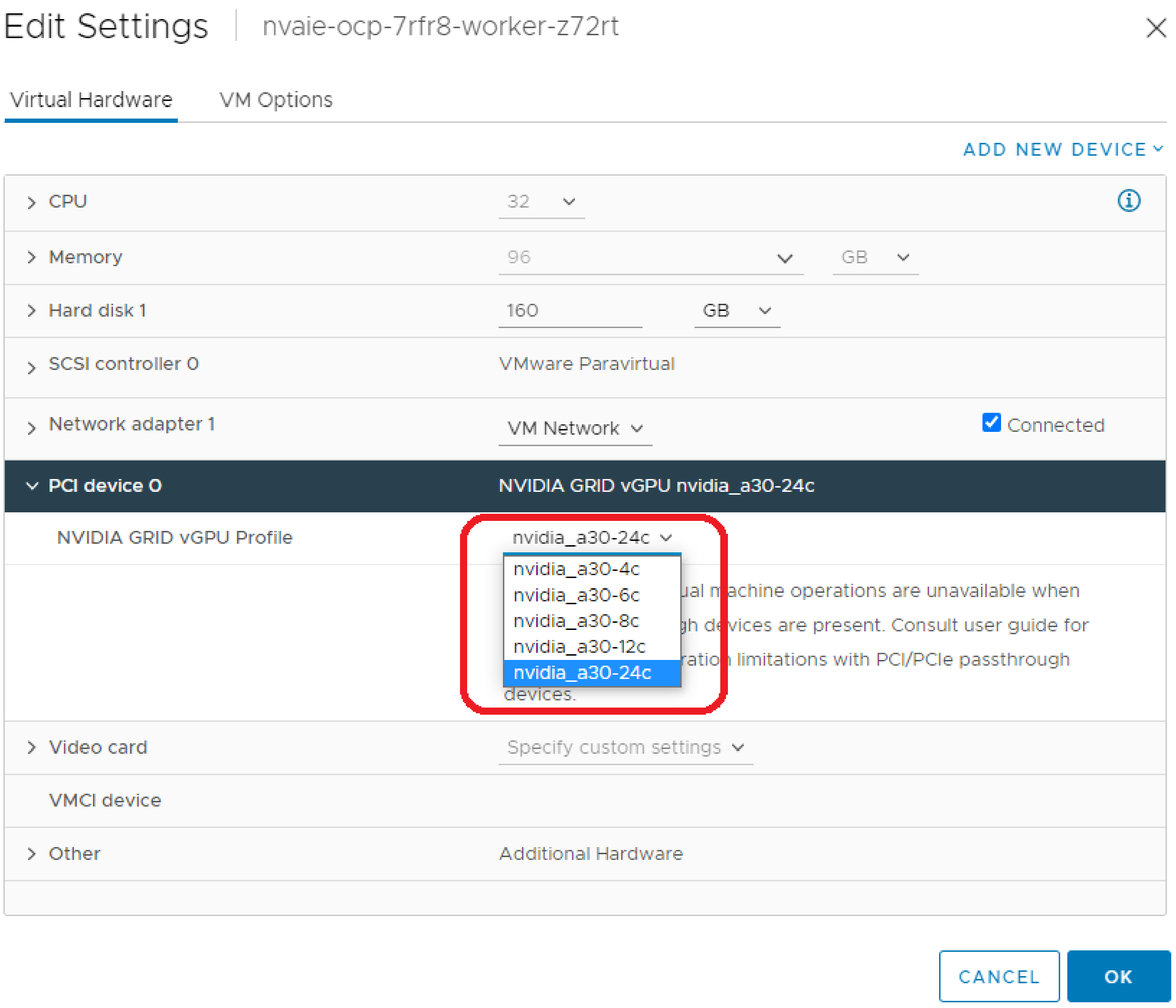
AI ENTERPRISE WITH OPENSHIFT AND VSPHERE

Scaling OpenShift with MachineSets



AI ENTERPRISE WITH OPENSHIFT AND VSPHERE

Scaling OpenShift with MachineSets



Machine Learning Workflows in Practice

NVIDIA END-TO-END AI SOFTWARE SUITE

Typical AI Workflow | How They're Deployed

DATA PREP	TRAIN	INFERENCE	SCALE
NVIDIA AI Enterprise	NVIDIA AI Enterprise	NVIDIA AI Enterprise	NVIDIA AI Enterprise
RAPIDS	TensorFlow PyTorch	TensorRT	Triton Inference Server

Full or Multi GPU Scale up or scale out with-multi node clusters	**Fractional GPU** Optimized GPU Resources
A100*, A30*	A100*, A30* with MIG

https://docs.nvidia.com/datacenter/tesla/mig-user-guide/

** Recommended GPU for NVIDIA AI Enterprise. Other NVIDIA GPUs are supported as well.*

DATA PREPROCESSING

In a GPU machine learning pipeline, the data never leaves the GPU.

cuDF has pandas like APIs for data wrangling.

Additional helper functions for Tokenization (for language models) on GPU which is 30X faster than preprocessing on CPU. Can be scaled on multiple GPUs with Dask.

Types of data

Static

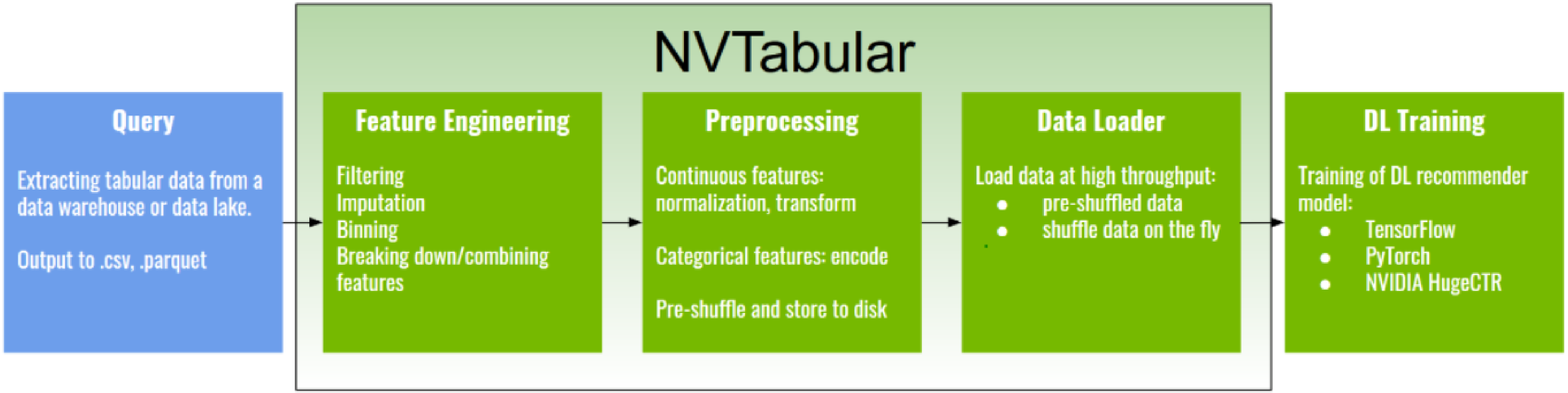
- NV Tabular

Streaming

- cuStreamz

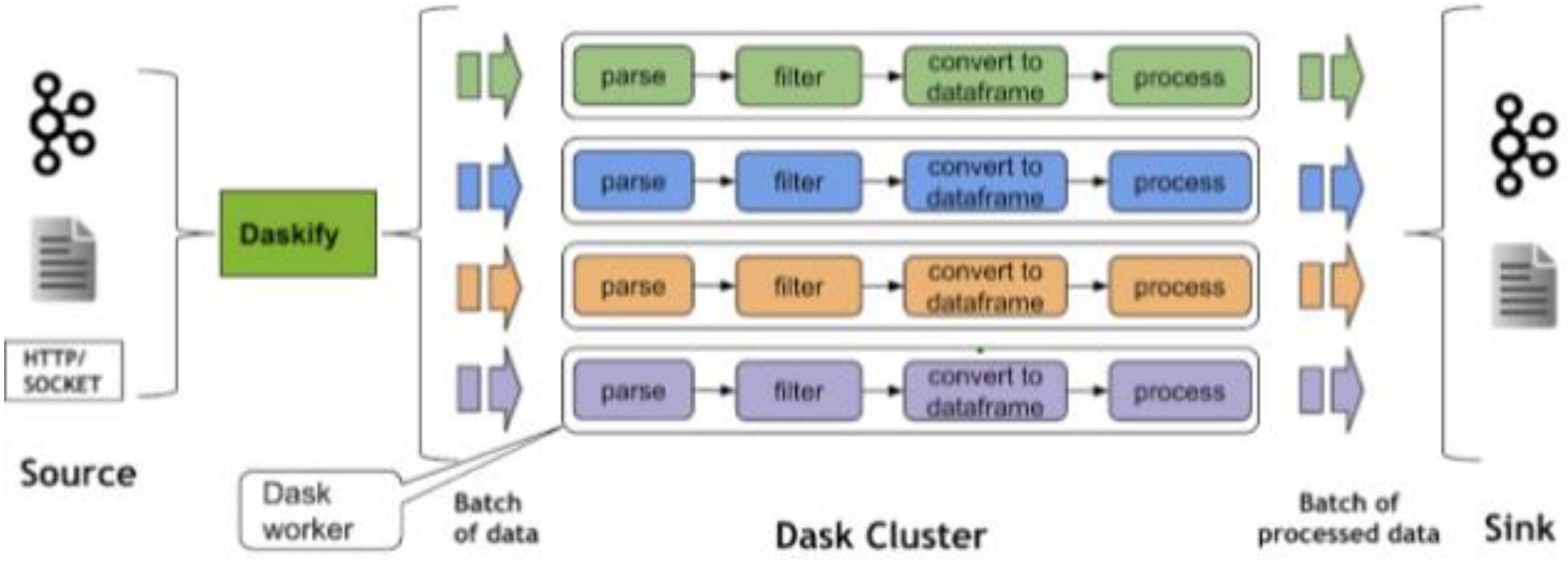
Available with support as part of the NVIDIA AI Enterprise software suite

NVTABULAR



Available with support as part of the NVIDIA AI Enterprise software suite

CUSTREAMZ

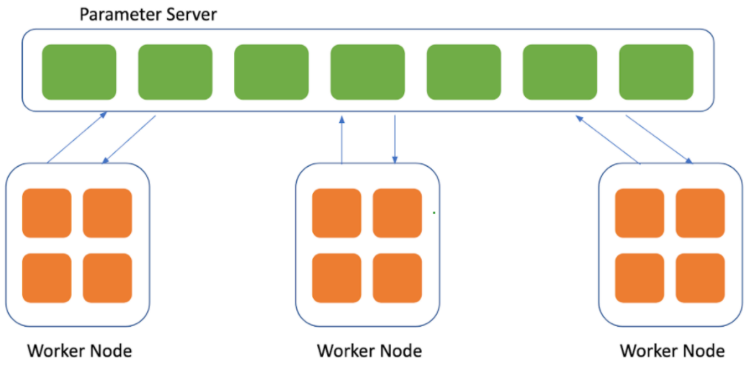


Distributed cuStreamz workstream using Dask

MODEL TRAINING

Horovod and MPI

- Horovod is a distributed deep learning training framework for TensorFlow, Keras, PyTorch, and Apache MXNet. The goal of Horovod is to make distributed deep learning fast and easy to use.
- MPI with NCCL can launch process on remote machines which can communicate to each other over TCP sockets or infiniband using NCCL. Horovod is the application layer on top of Tensorflow that works with MPI to make distributed training possible.



MODEL TRAINING

MultiGPU and Multinode Training

Magnum IO: The NVIDIA MAGNUM IO software development kit (SDK) enables developers to remove input/output (IO) bottlenecks in AI training, high performance computing (HPC), data science.

Components

- NVIDIA Collective Communication Library (NCCL)
  - Implements multi-GPU and multi-node communication primitives optimized for NVIDIA GPUs and Networking.
- MOFED
  - Drivers to enable Infiniband and RoCE for Multinode communications.
- GPU Direct RDMA (GDRDMA) and GPU Direct Storage (GDS)
  - Read data directly from Disk to GPU (GDS) and access the address space of a remote GPU (GDRDMA) without CPU Intervention.

MODEL TRAINING

Kubernetes operator support for MPI

- The MPI Operator makes it easy to run allreduce-style distributed training on Kubernetes.
- Different from Tensorflow or Pytorch operator. It is decoupled from the underlying machine learning framework and has support for Horovod.
- Has a CRD to specify the the master and worker pods. Mpi commands with NCCL flags can be specified in the command spec of the CRD.

```yaml
apiVersion: kubeflow.org/v1alpha2
kind: MPIJob
metadata:
  name: tensorflow-benchmarks
spec:
  slotsPerWorker: 1
  cleanPodPolicy: Running
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
         spec:
           containers:
           - image: mpioperator/tensorflow-benchmarks:latest
             name: tensorflow-benchmarks
             command:
             - mpirun
             - python
             - scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py
             - --model=resnet101
             - --batch_size=64
             - --variable_update=horovod
    Worker:
      replicas: 2
      template:
        spec:
          containers:
          - image: mpioperator/tensorflow-benchmarks:latest
            name: tensorflow-benchmarks
            resources:
              limits:
                nvidia.com/gpu: 1
```

MODEL TRAINING

NVIDIA AI Enterprise with MPI Operator

- Start with the base Dockerfile available at https://github.com/kubeflow/mpi-operator/blob/master/build/base/Dockerfile

- Add NVIDIA AI Enterprise Tensorflow container as the base container.

- Create the new Docker image and upload to the private registry.

- Create a Persistent volume to hold the dataset

- Install MPI operator

git clone https://github.com/kubeflow/mpi-operator
cd mpi-operator
kubectl apply -f deploy/v2beta1/mpi-operator.yaml

```
FROM nvcr.io/nvaie/tensorflow-1-1:21.08-nvaie1.1-tf1-py3

ARG port=2222

RUN apt update && apt install -y --no-install-recommends \
                         openssh-server \
                         openssh-client \
                         libcap2-bin \
                 && rm -rf /var/lib/apt/lists/*
# Add priviledge separation directoy to run sshd as root.
RUN mkdir -p /var/run/sshd
# Add capability to run sshd as non-root.
RUN setcap CAP_NET_BIND_SERVICE=+eip /usr/sbin/sshd

# Allow OpenSSH to talk to containers without asking for confirmation
# by disabling StrictHostKeyChecking.
# mpi-operator mounts the .ssh folder from a Secret. For that to work, we need
# to disable UserKnownHostsFile to avoid write permissions.
# Disabling StrictModes avoids directory and files read permission checks.
RUN sed -i "s/[ #]\(.*StrictHostKeyChecking \).*/ \1no/g" /etc/ssh/ssh_config \
    && echo "    UserKnownHostsFile /dev/null" >> /etc/ssh/ssh_config \
    && sed -i "s/[ #]\(.*Port \).*/ \1$port/g" /etc/ssh/ssh_config \
    && sed -i "s/#\(StrictModes \).*/\1no/g" /etc/ssh/sshd_config \
    && sed -i "s/#\(Port \).*/\1$port/g" /etc/ssh/sshd_config

RUN useradd -m mpiuser
WORKDIR /home/mpiuser
# Configurations for running sshd as non-root.
COPY --chown=mpiuser sshd_config .sshd_config
RUN echo "Port $port" >> /home/mpiuser/.sshd_config
```

MODEL TRAINING

MPI Operator with Network Operator and GPU Direct

- Setup you Ethernet Switch 100/200G to enable RoCE on a separate VLAN

- Install Network Operator (as covered in previous slides)
  - Make sure to specify the VLAN ID in the NetworkAttachmentDefinition

- Specify the overlay network in the MPI operator CRD (mlnxrdma) in the graphic on right

- Add Mellanox resource to the CRD (nvidia.com/sriov_rdma:1) in the graphic on the right

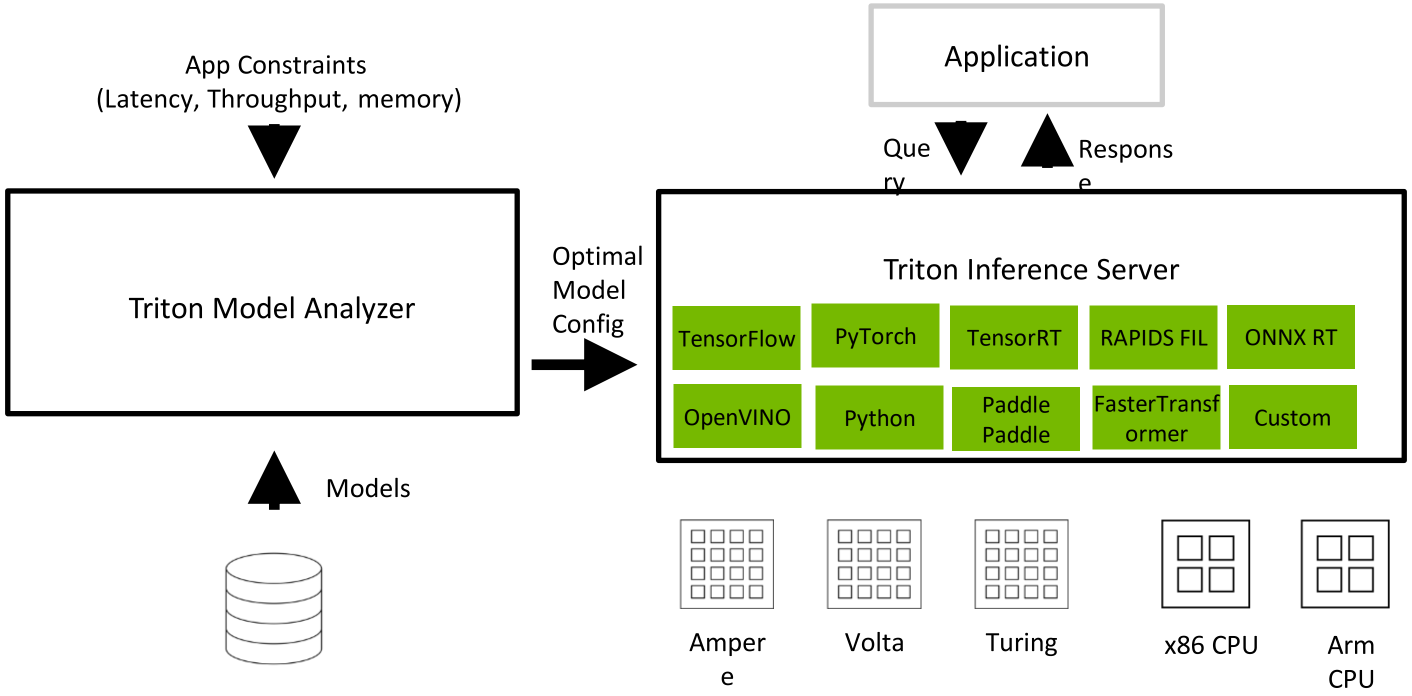
- The Network operator comes with nv_peer_mem pod for GPUdirect and NCCL should make use of it by default

```yaml
Worker:
  replicas: 2
  template:
    metadata:
      annotations:
        k8s.v1.cni.cncf.io/networks: mlnxrdma
    spec:
      volumes:
        - name: task-pv-storage
          persistentVolumeClaim:
            claimName: dataset-pv-claim
      containers:
      - image: nvcr.io/nvaie-tme/mpi-operator:latest
        name: tensorflow-benchmarks
        volumeMounts:
          - mountPath: "/data"
            name: task-pv-storage
        securityContext:
          capabilities:
            add: [ "IPC_LOCK" ]
        resources:
          limits:
            nvidia.com/gpu: 1
            nvidia.com/sriov_rdma: 1
```

MODEL INFERENCE WITH TRITON INFERENCE SERVER

Bringing Fast and Scalable AI to Applications

All Major Frameworks, Major Clouds, AI Platforms
Diverse query types – Real time, Offline batch, Video/Audio streaming, Ensembles
Model Analyzer Optimizes For App Constraints
Distributed Multi-GPU Multi-Node Inference



DEPLOY AT SCALE

NVIDIA
TRITON INFERENCE SERVER

Available with support as part of the NVIDIA AI Enterprise software suite
https://developer.nvidia.com/nvidia-triton-inference-server

MODEL INFERENCE

Model Repository

The Triton Inference Server serves models from one or more model repositories that are specified when the server is started. While Triton is running, the models being served can be modified .

The corresponding repository layout must be:

```
<model-repository-path>/
  <model-name>/
    [config.pbtxt]
    [<output-labels-file> ...]
    <version>/
      <model-definition-file>
    <version>/
      <model-definition-file>
```

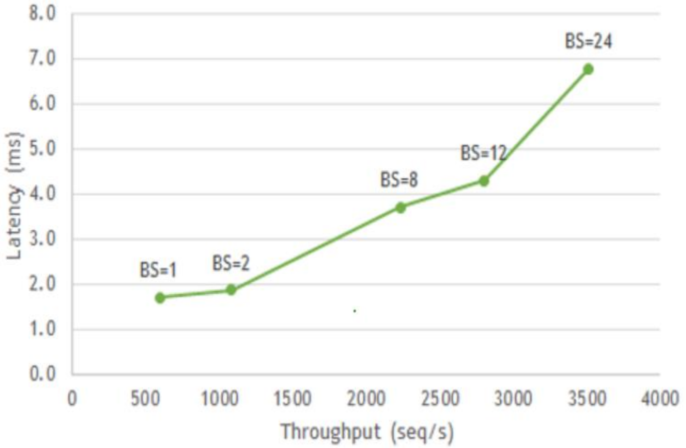
Start the server by pulling the NVIDIA AI Enterprise Triton Inference server container and pointing it to the model repository

```
$ tritonserver --model-repository=/path/to/model/repository
```

MODEL INFERENCE

Triton Inference Server on Kubernetes

- Triton Inference server can be deployed as a Kubernetes service inside the cluster.
- It is preferable to setup the model repository on a volume mount with a Persistent Volume Claim.
- **Latency vs Batching**
  - Triton supports batch inferencing by allowing individual inference requests to specify a batch of inputs.
  - The inferencing for a batch of inputs is performed at the same time which is especially important for GPUs since it can greatly increase inferencing throughput.
  - In many use cases the individual inference requests are not batched, therefore, they do not benefit from the throughput benefits of batching.
  - The inference server contains multiple scheduling and batching algorithms that support many different model types and use-cases.
  - A balance between the latency and throughput requirements must be maintained and the correct value depends on the individual use case.
  - Dynamic batching is a feature of Triton that allows inference requests to be combined by the server, so that a batch is created dynamically.



MODEL INFERENCE

Triton Forest Inference Library (FIL)

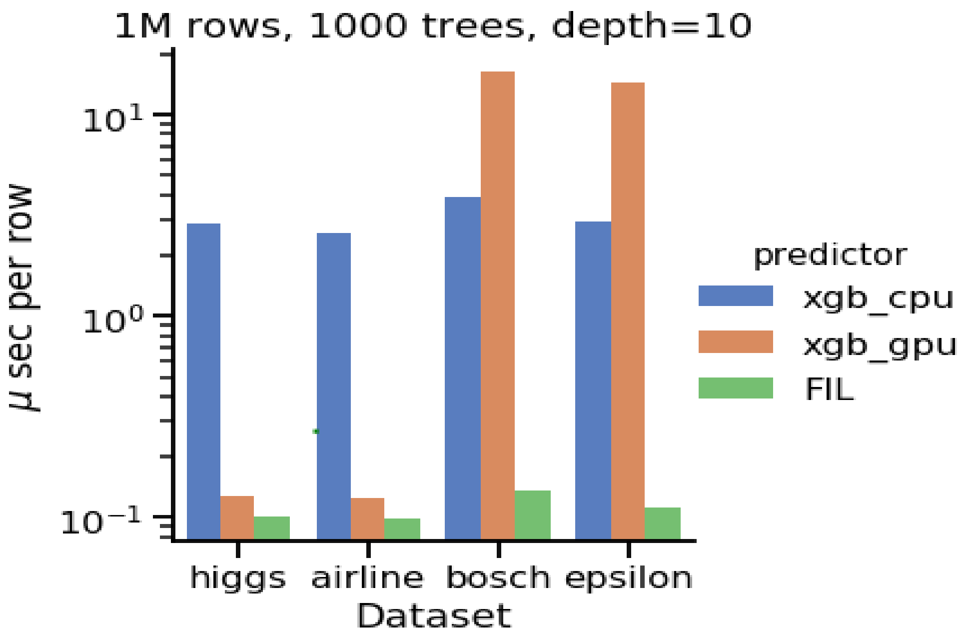
Triton Inference server in addition to serving Deep learning models also has libraries to host XGBoost and Random Forest Models through the FIL backend.

Models are served in a similar manner as regular deep learning models in a model repository.

```
model_repository/
`-- fil
    |-- 1
    |   `-- xgboost.model
    `-- config.pbtxt
```

Model Configuration file (config.pbtxt) needs to be specified which has information like Model batch size, input and output shapes and threads per tree etc.

The Latency is much better than using XGBoost Inference directly performed on Python.



CPU and GPU performance across datasets

MODEL INFERENCE

Autoscaling Triton Inference Server on Kubernetes

As your service starts becoming more popular over time, the number of inference requests increase. It then becomes important for the service to make use of more compute(GPU) power.

Traditionally, the devops admin gauges server load (requests per second) and then adds additional resources depending on the load and scales down the resources when the load decreases.

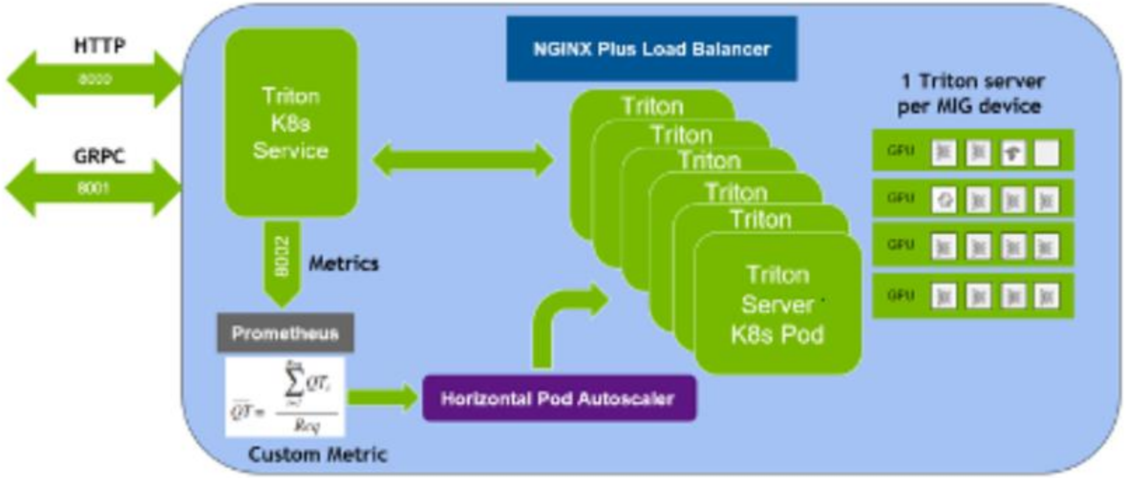
Kubernetes automates this workflow using Horizontal Pod autoscaler (HPA).

The Kubernetes Horizontal Pod auto scaler automatically scales the number of Pods in a Deployment, replication controller, or replica set based on that metrics like CPU utilization.

By providing custom metrics like GPU Utilization, Duty cycle etc to the HPA, the Triton Inference server pods can autoscale on demand based on these Metrics

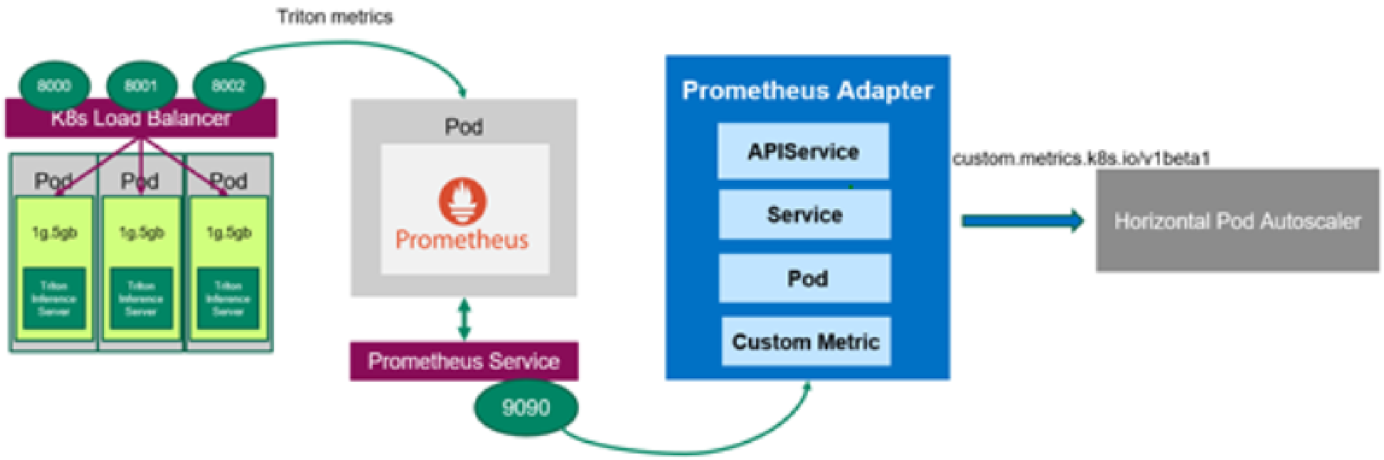
MODEL INFERENCE

Autoscaling with Kubernetes



MODEL INFERENCE

Autoscaling with Kubernetes



MODEL INFERENCE

Steps to Autoscale Triton Inference Server

Custom Metrics Server

The custom metrics server exposes custom metrics for Horizontal Pod auto scaler to the API server. Custom metrics server can be deployed on Kubernetes as follows.

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

NVIDIA DCGM Exporter Service

To gather GPU telemetry in Kubernetes, the Nvidia Data Center GPU Manager (DCGM) is used. This suite of data center management tools allows you to manage and monitor GPU resources in an accelerated data center. Since the DevOps Engineer already installed the GPU Operator, the NVIDIA DCGM exporter service is already installed onto the cluster.

MODEL INFERENCE

Steps to Autoscale Triton Inference Server

Prometheus Server

To expose cluster-level and node-level metrics, Prometheus is used. Prometheus, a Cloud Native Computing Foundation project, is a systems and service monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts when specific conditions are observed. Refer to the guide on the GPU Operator website to set up Prometheus on your cluster.

Install Prometheus Adapter

The Prometheus adapter exposes the Prometheus metrics from the DCGM exporter to the custom metrics server we deployed. Therefore, this adapter is suitable for use with the autoscaling/v2 Horizontal Pod auto scaler in Kubernetes 1.16+. It can also replace the metrics server on clusters that already run Prometheus and collect the appropriate metrics.

MODEL INFERENCE

Steps to Autoscale Triton Inference Server

Verify if the Custom Metrics are Available to the Metrics Server

```
nvidia@node1:~/yaml$ kubectl get --raw /apis/custom.metrics.k8s.io/v1beta1 | jq -r . |
  grep DCGM_FI_DEV_MEM_COPY_UTIL
```

```
"name": "pods/DCGM_FI_DEV_MEM_COPY_UTIL",
"name": "jobs.batch/DCGM_FI_DEV_MEM_COPY_UTIL",
"name": "namespaces/DCGM_FI_DEV_MEM_COPY_UTIL",
```

MODEL INFERENCE

Steps to Autoscale Triton Inference Server

Create a yaml file for Horizontal Pod Autoscaler.

```yaml
kind: HorizontalPodAutoscaler
apiVersion: autoscaling/v2beta1
metadata:
name: gpu-hpa
spec:
scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: bert-qa
minReplicas: 1
maxReplicas: 3
metrics:
- type: Pods
    pods:
    metricName: DCGM_FI_DEV_GPU_UTIL # Average GPU usage of the pod.
    targetAverageValue: 40
```

minReplicas is the lower bound to the number of pods to scale down to in the pod auto scaler deployment, with **maxReplicas** being the upper bound. The custom metric on which the pods are to be auto scaled is `DCGM_FI_DEV_GPU_UTIL` (which is the average GPU Utilization). If it exceeds the average target value of 40 percent, a new pod is scheduled.

FAST TRACK YOUR AI JOURNEY WITH NVIDIA LAUNCHPAD

With Curated Labs for NVIDIA AI Enterprise

- Immediate, short-term, remote access to accelerated compute infrastructure
- Hands-on experience tailored for AI Practitioners, Data Scientists, and IT Administrators
- Available globally

▶ APPLY NOW TO GET STARTED: nvidia.com/try-ai

AI

- Train and Deploy an AI Support Chatbot
- Train an AI model for Image Classification of Online Products

DATA SCIENCE

- Accelerate Data Processing and Train an AI Model to Predict Prices
- Accelerate Data Processing, Tokenization, and Train an AI Model to Perform Sentiment Analysis
- Scale Data Science with Domino Enterprise MLOps Platform

INFRASTRUCTURE OPTIMIZATION

- Configure and Optimize VMware vSphere for AI and Data Science Workloads
- Configure, Optimize, and Orchestrate Resources for AI and Data Science Workloads with VMware Tanzu
- Configure, Optimize, and Orchestrate Resources for AI and Data Science Workloads with Red Hat OpenShift

Curated Labs for AI Practitioners/Data Scientists

Curated Lab for Both

Curated Labs for IT Administrators

NVIDIA LAUNCHPAD

NVIDIA AI ENTERPRISE SESSIONS TO CHECK OUT AT GTC SPRING 2022

Democratizing AI for the Enterprise

SESSIONS (Data Center & Virtualization)	
Session ID	**Title**
S41858	What Every Business Leader Needs to Know to be Successful with AI
S41894	Containers or VMs: Deploy AI Workloads with Ease
S41871	NVIDIA AI Enterprise 101: Technology Session
S41864	Developing AI with Enterprise-Ready Kubernetes
S41876	Running Cloud Native Apps in NVIDIA AI Enterprise
S41867	Virtualize GPU-accelerated Data Science and AI Workflows in Your Data Center with Enterprise MLOps
S41877	How to Implement AI Across the Enterprise
S41551	Architecting the Next Generation Accelerated Data Center
S42061	Medical Image Reconstruction with Memory-Efficient Neural Networks
S41308	Scaling Remote Healthcare & Wellness: Virtualized GPU Acceleration of Mixed AI Workloads
S41382	Start Your AI Journey in a VMware Data Center
S41838	Tuning Virtualized GPUs for Optimal Performance on ML/AI Workloads
S41883	Manage hyper-converged and accelerated workloads in edge virtual data centers
S41307	A Modern Approach to End-to-End AI/ML: Learn How to Deliver Self-Service MLOps
S42535	AI Your Way: Solutions for Every Organization and VMware

NVIDIA GTC