Deployment Guide: Cisco AI PODs with Canonical Kubernetes and Canonical MLOps Stack

Maximize the performance of Cisco AI PODs with Canonical Kubernetes and Canonical MLOps stack

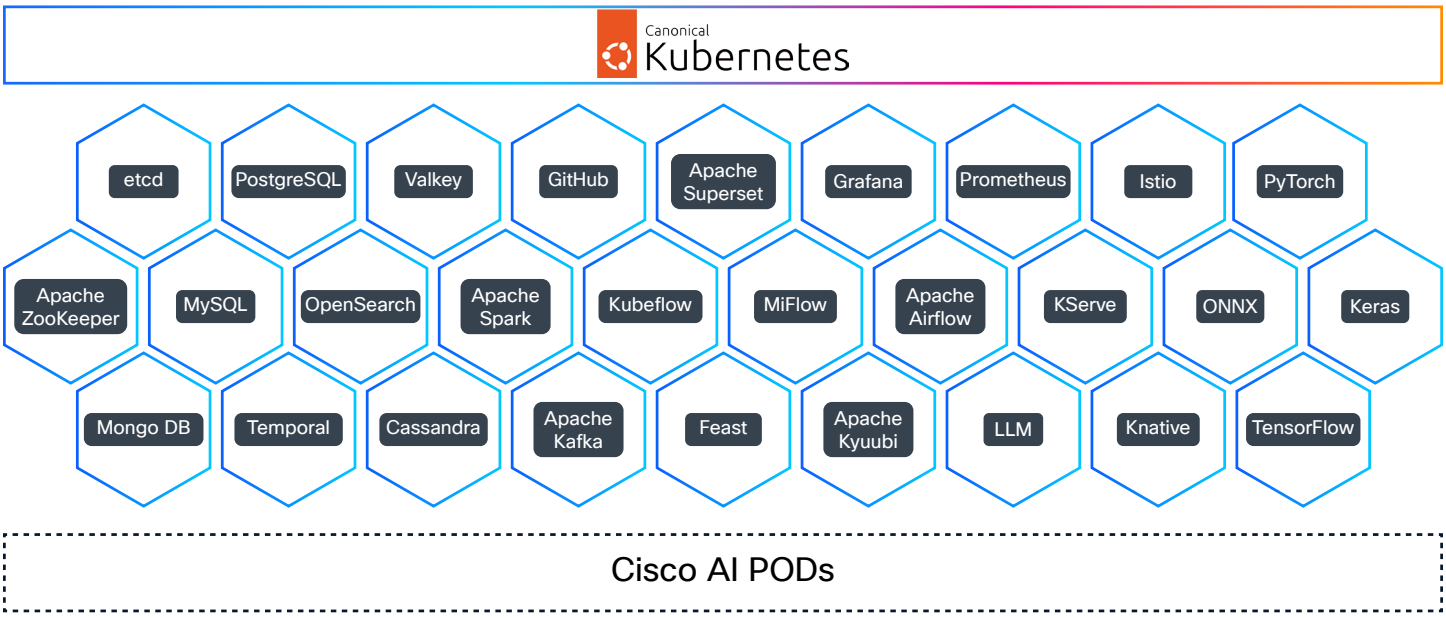
Contents

Accelerating AI/ML deployment with Canonical Kubernetes

This guide will show AI/ML solution architects, data scientists, and infrastructure engineers how to deploy Cisco AI PODs with Canonical Kubernetes and Canonical MLOps stack on Cisco AI PODs. Together, these solutions form a production-ready and scalable platform for ML workloads, delivering the cost and scale benefits of open source on a trusted enterprise platform:

- Pre-optimized resource use on Cisco AI PODs
- Streamlined Canonical MLOps lifecycle management for a quick transition from prototype to production
- Five years of support for LTS versions of Canonical Kubernetes and the Canonical MLOps stack, with the option of lengthening support to 15 years
- Security built into every layer of the stack through Cisco Secure AI Factory with NVIDIA, from infrastructure-level protection with Cisco Hypershield™ and BlueField DPUs to model-layer defense with Cisco AI Defense
- Cloud-managed operations through Cisco Intersight® and Nexus® Dashboard, enabling policy-driven GPU orchestration and zero-touch provisioning of AI infrastructure at scale

Solution overview



A powerful platform for MLOps, delivered by Canonical and Cisco

Canonical Kubernetes and Canonical MLOps integrate with Cisco AI PODs to deliver a turnkey environment for enterprise AI. The platform replaces fragmented tools with a unified platform that handles everything from GPU orchestration to model lifecycle management, significantly reducing the time it takes to scale AI/ML initiatives from pilot to production.

Cisco AI PODs: purpose-built infrastructure for enterprise AI

Cisco AI PODs are a pre-validated, modular AI infrastructure solution that combines Cisco UCS® servers with NVIDIA GPUs, Cisco Nexus high-performance Ethernet networking, and integrated storage into a single, rack-scale deployment unit. As a core building block of the Cisco Secure AI Factory with NVIDIA, each Cisco AI POD is engineered to compress deployment timelines from months to weeks, giving your teams a production-ready foundation for training, fine-tuning, and inference workloads without the burden of assembling and validating each component independently.

At the compute layer, Cisco UCS C880A M8 Rack Servers deliver up to 8 B300 Ultra Tensor Core GPUs interconnected through NVLink, providing the GPU-to-GPU bandwidth that large-scale model training demands. Dual Intel® Xeon® 6th Gen CPUs supply the host-side compute and memory capacity needed for data preprocessing, checkpointing, and serving workloads in parallel.

The networking fabric uses Cisco N9000 Series switches running at 400GbE/800GbE, built on Cisco® Silicon One® and NVIDIA Spectrum X. Cisco is the only networking vendor whose silicon is included in the NVIDIA Spectrum-X Ethernet platform through a shared-licensing partnership, combining Cisco's congestion-aware per-packet load balancing with NVIDIA's adaptive routing and out-of-order packet handling. This delivers the low-latency, lossless east/west traffic performance that distributed GPU training requires.

Security is built into Cisco AI PODs from the ground up rather than layered on after deployment. Cisco Hypershield provides AI-native, kernel-level enforcement through eBPFs, while NVIDIA BlueField-3 DPUs run the Cisco Hybrid Mesh Firewall at 400G line-rate stateful inspection without consuming CPU or GPU cycles. Cisco AI Defense adds model-layer protection: discovery of shadow AI assets, validation of model behavior, and runtime guardrails for prompt injection and data exfiltration. This security-first approach addresses the governance and compliance requirements that enterprise AI deployments face from day one.

Operationally, Cisco Intersight provides cloud-managed lifecycle management with policy-driven GPU orchestration, enabling real-time allocation of GPU resources to workloads and eliminating the manual provisioning overhead that slows down scaling. Cisco Nexus Dashboard delivers unified automation and observability across the network fabric, with topology-aware visualization and congestion analytics purpose-built for AI traffic patterns.

Canonical Kubernetes for GPU optimization:

Getting the most out of your hardware often entails a high degree of manual tweaking. Through Canonical's partnership with Cisco, Canonical Kubernetes is optimized to deliver the full potential of the NVIDIA GPUs included in Cisco AI PODs, right out of the box. This reduces the operational overhead of GPU configuration, allowing teams to partition and scale hardware resources dynamically while maintaining strict workload isolation.

Streamlining the AI lifecycle with Canonical's MLOps stack solutions:

Canonical's MLOps stack builds on this robust foundation by providing the essential tools and workflows for defining, managing, and automating the entire AI workload lifecycle. Canonical's Charmed Kubeflow, Charmed MLF, and Charmed Feast run seamlessly together to help ensure a smooth journey from data preparation and model training to deployment and monitoring.

Canonical's MLOps stack simplifies complex operations by:

- **Integrated tooling:** seamlessly combines best practice open-source MLOps tools into a cohesive platform
- **Workflow automation:** Modular charmed operators encapsulate complex operational logic. This creates a repeatable framework for model building and deployment, minimizing manual overhead and accelerating time to market
- **Unified management:** provides an integrated platform for managing ML experiments, artifacts, and production deployments

Enhanced networking with Cilium CNI – powered by Cisco Isovalent:

For high-performance, secure, and observable networking, this solution uses **Cilium** as the Container Network Interface (CNI), powered by Cisco Isovalent®. Cilium, based on eBPF technology, provides:

- **Superior performance:** offers high-throughput and low-latency networking crucial for distributed ML workloads that rely on fast communication between nodes and GPUs
- **Advanced security:** implements fine-grained network policies based on application identity, securing inter-service communication within your AI workloads
- **Deep observability:** provides rich network telemetry and tracing, simplifying troubleshooting and performance tuning for complex Kubernetes environments

This combination of Cisco's purpose-built AI infrastructure, Canonical's enterprise Kubernetes and MLOps expertise, and Cilium's networking delivers a production-grade platform for enterprises looking to move AI projects from prototype to production at scale.

Cisco acquired Isovalent in 2024, bringing the Cilium and Tetragon open-source projects under Cisco's engineering umbrella. Cilium is already the default CNI for Canonical Kubernetes, Google Kubernetes Engine, Amazon EKS, and Microsoft Azure AKS. With the Isovalent acquisition, Cisco now offers direct engineering support and enterprise-grade SLAs for Cilium deployments, tightly integrated with the broader Cisco Secure AI Factory networking and security stack. For AI workloads running on Cisco AI PODs, this means that the same vendor providing the physical network fabric, the GPU compute, and the infrastructure security also owns and maintains the Kubernetes networking layer, eliminating the multi-vendor finger-pointing that typically plagues troubleshooting in production AI clusters.

Deployment guide

1. Prerequisites and setup

1.1 Hardware and networking configuration

For this demo, SSH access was given to a pre-provisioned Ubuntu 24.04. machine on CiscoAI PODs.

The Cisco AI POD used in this deployment consists of a Cisco UCS C885A M8 Rack Server equipped with 8 NVIDIA HGX H200 Tensor Core GPUs and dual AMD EPYC processors (96 cores each). The backend network fabric uses Cisco Nexus 9364E-GX2A switches running at 400 GbE, configured for RoCE (RDMA over converged Ethernet) to support the low-latency GPU-to-GPU communication required by distributed training workloads. Ubuntu 24.04 LTS was pre-installed on the server through Cisco Intersight's OS provisioning workflow, which automates bare-metal deployment with policy-driven configuration profiles. Network configuration, including VLAN assignments, MTU settings for jumbo frames (9216 bytes), and LLDP discovery, was applied through Nexus Dashboard templates. The storage fabric was configured for NVMe over Fabrics (NVMeoF) to provide high-throughput data access to the training pipeline. All firmware versions for UCS BIOS, GPU vBIOS, and NIC firmware were validated against the Cisco Secure AI Factory with NVIDIA reference architecture compatibility matrix prior to deployment.

1.2 NVIDIA driver and CUDA installation

Install the recommended stable proprietary driver and the NVIDIA CUDA toolkit using the standard Ubuntu package manager.

```
sudo apt update
sudo ubuntu-drivers install --gpgpu nvidia:535-server
sudo apt install nvidia-cuda-toolkit
```

Next, perform a reboot to ensure that the new kernel module is loaded.

The installation can be verified by checking the driver version and running the “nvcc” compiler version check.

```
sudo nvidia-smi
nvcc --version
```

```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo nvidia-smi
[sudo] password for ubuntu:
Tue Jan 13 13:44:49 2026
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.274.02             Driver Version: 535.274.02   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA L40S                    Off | 00000000:71:00.0 Off |                    0 |
| N/A   31C    P8              35W / 350W |      0MiB / 46068MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
|   1  NVIDIA L40S                    Off | 00000000:91:00.0 Off |                    0 |
| N/A   32C    P8              35W / 350W |      0MiB / 46068MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Fri_Jan__6_16:45:21_PST_2023
Cuda compilation tools, release 12.0, V12.0.140
Build cuda_12.0.r12.0/compiler.32267302_0
ubuntu@cpn-ful-ai-c245-gpuwn4:~$
```

1.3 inotify setup

Canonical Kubernetes uses inotify to interact with the file system. This may lead to situations where large Canonical Kubernetes deployments (such as Charmed Kubeflow and Charmed MLflow) exceed the default inotify limits. Therefore, you can increase the limits, and retain the increased limits across the infrastructure, using the following commands:

```
sudo sysctl fs.inotify.max_user_instances=1280
sudo sysctl fs.inotify.max_user_watches=655360
echo “fs.inotify.max_user_instances=1280” >> sudo tee -a /etc/sysctl.conf
echo “fs.inotify.max_user_watches=655360” >> sudo tee -a /etc/sysctl.conf
```

2. Canonical Kubernetes deployment

The deployment uses Canonical's official Kubernetes snap, customized for the Cisco AI PODs' hardware capabilities.

2.1 Core cluster deployment

You can perform the Canonical Kubernetes deployment and bootstrap using the following snap command:

```
sudo snap install k8s --classic
sudo k8s bootstrap
```

2.2 Post-deployment verification

Upon completion of the bootstrap, you should check the status of the cluster:

```
sudo k8s status --wait-ready
```

```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s status --wait-ready
cluster status:           ready
control plane nodes:      192.133.164.24:6400 (voter)
high availability:        no
datastore:                etcd
network:                  enabled
dns:                      enabled at 10.152.183.44
ingress:                  disabled
load-balancer:            disabled
local-storage:            enabled at /var/snap/k8s/common/rawfile-storage
gateway                   enabled
```

```
sudo k8s kubectl get nodes
```

```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s kubectl get nodes
NAME                                STATUS   ROLES                AGE   VERSION
cpn-ful-ai-c245-gpuwn4.ai.ciscops.net   Ready    control-plane,worker   8d    v1.32.11
```

```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s kubectl version
Client Version: v1.32.11
Kustomize Version: v5.5.0
Server Version: v1.32.11
```

3. GPU and networking testing

3.1 NVIDIA GPU integration

Deploy the NVIDIA GPU Operator using its official Helm chart. This will automate the management and lifecycle of necessary components (such as drivers and device plug-ins) for GPU support in Kubernetes.

1. **Addition of the NVIDIA GPU Operator Helm Repository:** Add the official NVIDIA GPU Operator Helm Repository to the local Helm configuration:

```
sudo k8s helm repo add nvidia https://nvidia.github.io/gpu-operator
sudo k8s helm repo update
```

2. **NVIDIA GPU Operator installation:** Install the NVIDIA GPU Operator into the gpu-operator namespace:

```
sudo k8s helm install --generate-name --wait -n gpu-operator --create-namespace nvidia/gpu-operator --version=v24.9.2 --set mig.strategy=mixed
```

3. **Verification:** Monitor the cluster resources in order to confirm:

- Successful deployment of the driver and device plug-in pods
- Recognition of Kubernetes by the available NVIDIA GPUs

```
sudo k8s kubectl get pods -n gpu-operator
sudo k8s kubectl get nodes -o json | grep nvidia.com/gpu
```

```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s kubectl get pods -n gpu-operator
NAME                                                          READY   STATUS      RESTARTS        AGE
gpu-feature-discovery-g8sdb                                   1/1     Running     1 (6d23h ago)   8d
gpu-operator-1767620802-node-feature-discovery-gc-6ff54c5blcmbl   1/1     Running     1 (6d23h ago)   8d
gpu-operator-1767620802-node-feature-discovery-master-fd94k8mgt   1/1     Running     1 (6d23h ago)   8d
gpu-operator-1767620802-node-feature-discovery-worker-4c2cj       1/1     Running     1 (6d23h ago)   8d
gpu-operator-97655686-tvfgw                                   1/1     Running     1 (6d23h ago)   8d
nvidia-container-toolkit-daemonset-jshn6                      1/1     Running     1 (6d23h ago)   8d
nvidia-cuda-validator-m4vdf                                   0/1     Completed   0               6d23h
nvidia-dcgm-exporter-6t9h6                                    1/1     Running     1 (6d23h ago)   8d
nvidia-device-plugin-daemonset-j8txw                          1/1     Running     1 (6d23h ago)   8d
nvidia-operator-validator-pvvs9                               1/1     Running     1 (6d23h ago)   8d
```

```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s kubectl get nodes -o json | grep nvidia.com/gpu
                    "nfd.node.kubernetes.io/feature-labels": "cpu-cpuid.ADX,cpu-cpuid.AESNI,cpu-cpuid.AVX,cpu-cpuid.AVX2,cpu-cpuid.AVX512BF16,cpu-cpuid.AVX512BITALG,cpu-cpuid.AVX
512BW,cpu-cpuid.AVX512CD,cpu-cpuid.AVX512DQ,cpu-cpuid.AVX512F,cpu-cpuid.AVX512IFMA,cpu-cpuid.AVX512VBMI,cpu-cpuid.AVX512VBMI2,cpu-cpuid.AVX512VL,cpu-cpuid.AVX512VNNI,cpu-cpuid.AV
X512VP2INTERSECT,cpu-cpuid.AVX512VPOPCNTDQ,cpu-cpuid.AVXVNNI,cpu-cpuid.CETSS,cpu-cpuid.CLZERO,cpu-cpuid.CMPXCHG8,cpu-cpuid.CPBOOST,cpu-cpuid.CPPC,cpu-cpuid.EFER_LMSLE_UNS,cpu-cpu
id.FLUSH_L1D,cpu-cpuid.FMA3,cpu-cpuid.FSRM,cpu-cpuid.FXSR,cpu-cpuid.FXSROPT,cpu-cpuid.GFNI,cpu-cpuid.IBPB,cpu-cpuid.IBPB_BRTYPE,cpu-cpuid.IBRS,cpu-cpuid.IBRS_PREFERRED,cpu-cpuid.
IBRS_PROVIDES_SMP,cpu-cpuid.IBS,cpu-cpuid.IBSBRNTRGT,cpu-cpuid.IBSFETCHSAM,cpu-cpuid.IBSFFV,cpu-cpuid.IBSOPCNT,cpu-cpuid.IBSOPCNTEXT,cpu-cpuid.IBSOPSAM,cpu-cpuid.IBSRDWROPCNT,cpu
-cpuid.IBSRIPINVALIDCHK,cpu-cpuid.IBS_FETCH_CTLX,cpu-cpuid.IBS_OPFUSE,cpu-cpuid.IBS_PREVENTHOST,cpu-cpuid.IBS_ZEN4,cpu-cpuid.INT_WBINVD,cpu-cpuid.INVLPGB,cpu-cpuid.LAHF,cpu-cpuid
.LBRVIRT,cpu-cpuid.MCAOVERFLOW,cpu-cpuid.MOVBE,cpu-cpuid.MOVDIR64B,cpu-cpuid.MOVDIRI,cpu-cpuid.MOVU,cpu-cpuid.MSRIRC,cpu-cpuid.NRIPS,cpu-cpuid.OSXSAVE,cpu-cpuid.PPIN,cpu-cpuid.PS
FD,cpu-cpuid.RDPRU,cpu-cpuid.SBPB,cpu-cpuid.SEV,cpu-cpuid.SEV_64BIT,cpu-cpuid.SEV_ALTERNATIVE,cpu-cpuid.SEV_DEBUGSWAP,cpu-cpuid.SEV_ES,cpu-cpuid.SEV_RESTRICTED,cpu-cpuid.SEV_SNP,
cpu-cpuid.SHA,cpu-cpuid.SME,cpu-cpuid.SME_COHERENT,cpu-cpuid.SPEC_CTRL_SSBD,cpu-cpuid.SRSO_MSR_FIX,cpu-cpuid.SRSO_USER_KERNEL_NO,cpu-cpuid.SSE4A,cpu-cpuid.STIBP,cpu-cpuid.STIBP_A
LWAYSON,cpu-cpuid.SUCCOR,cpu-cpuid.SVM,cpu-cpuid.SVMDA,cpu-cpuid.SVMFBASID,cpu-cpuid.SVML,cpu-cpuid.SVMNP,cpu-cpuid.SVMPF,cpu-cpuid.SVMPFT,cpu-cpuid.SYSCALL,cpu-cpuid.SYSEE,cpu-c
puid.TLB_FLUSH_NESTED,cpu-cpuid.TOPEXT,cpu-cpuid.TSCRATEMSR,cpu-cpuid.VAES,cpu-cpuid.VMCBCLEAN,cpu-cpuid.VMPL,cpu-cpuid.VMSA_REGPROT,cpu-cpuid.VPCLMULQDQ,cpu-cpuid.VTE,cpu-cpuid.
WBNOINVD,cpu-cpuid.X87,cpu-cpuid.XGETBV1,cpu-cpuid.XSAVE,cpu-cpuid.XSAVEC,cpu-cpuid.XSAVEOPT,cpu-cpuid.XSAVES,cpu-hardware_multithreading,cpu-model.family,cpu-model.id,cpu-model.
vendor_id,kernel-config.NO_HZ,kernel-config.NO_HZ_FULL,kernel-version.full,kernel-version.major,kernel-version.minor,kernel-version.revision,memory-numa,memory-swap,network-sriov
.capable,nvidia.com/cuda.driver-version.full,nvidia.com/cuda.driver-version.major,nvidia.com/cuda.driver-version.minor,nvidia.com/cuda.driver-version.revision,nvidia.com/cuda.dri
ver.major,nvidia.com/cuda.driver.minor,nvidia.com/cuda.driver.rev,nvidia.com/cuda.runtime-version.full,nvidia.com/cuda.runtime-version.major,nvidia.com/cuda.runtime-version.minor
,nvidia.com/cuda.runtime.major,nvidia.com/cuda.runtime.minor,nvidia.com/gfd.timestamp,nvidia.com/gpu.compute.major,nvidia.com/gpu.compute.minor,nvidia.com/gpu.count,nvidia.com/gp
u.family,nvidia.com/gpu.machine,nvidia.com/gpu.memory,nvidia.com/gpu.mode,nvidia.com/gpu.product,nvidia.com/gpu.replicas,nvidia.com/gpu.sharing-strategy,nvidia.com/mig.capable,nv
idia.com/mig.strategy,nvidia.com/mps.capable,nvidia.com/vgpu.present,pci-10de.present,pci-10de.sriov.capable,pci-1137.present,pci-15b3.present,pci-15b3.sriov.capable,pci-1a03.pre
sent,rdma.capable,storage-nonrotationaldisk,system-os_release.ID,system-os_release.VERSION_ID,system-os_release.VERSION_ID.major,system-os_release.VERSION_ID.minor",
                    "nvidia.com/gpu-driver-upgrade-enabled": "true",
                    "nvidia.com/gpu-driver-upgrade-state": "upgrade-done",
                    "nvidia.com/gpu.compute.major": "8",
                    "nvidia.com/gpu.compute.minor": "9",
                    "nvidia.com/gpu.count": "2",
                    "nvidia.com/gpu.deploy.container-toolkit": "true",
                    "nvidia.com/gpu.deploy.dcgm": "true",
                    "nvidia.com/gpu.deploy.dcgm-exporter": "true",
                    "nvidia.com/gpu.deploy.device-plugin": "true",
                    "nvidia.com/gpu.deploy.driver": "pre-installed",
                    "nvidia.com/gpu.deploy.gpu-feature-discovery": "true",
                    "nvidia.com/gpu.deploy.node-status-exporter": "true",
                    "nvidia.com/gpu.deploy.operator-validator": "true",
                    "nvidia.com/gpu.family": "ampere",
                    "nvidia.com/gpu.machine": "UCSC-C245-M8SX",
                    "nvidia.com/gpu.memory": "46068",
                    "nvidia.com/gpu.mode": "compute",
                    "nvidia.com/gpu.present": "true",
                    "nvidia.com/gpu.product": "NVIDIA-L40S",
                    "nvidia.com/gpu.replicas": "1",
                    "nvidia.com/gpu.sharing-strategy": "none",
                    "nvidia.com/gpu": "2",
                    "nvidia.com/gpu": "2",
```

3.1.1 Basic CUDA verification

Perform the basic CUDA verification step by configuring a test pod to execute a simple CUDA operation.

As an output, you'll see that the test pod was deployed using the nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda10.2 image, requesting two GPU resources, which successfully ran and confirmed GPU functionality from within the containers.

```
apiVersion: v1
kind: Pod
metadata:
 name: cuda-gpu-test
spec:
 restartPolicy: OnFailure
 containers:
   - name: cuda-vector-add
     image: "nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda10.2"
     resources:
       limits:
         nvidia.com/gpu: 2
```

```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s kubectl logs cuda-gpu-test
[Vector addition of 50000 elements]
Copy input data from the host memory to the CUDA device
CUDA kernel launch with 196 blocks of 256 threads
Copy output data from the CUDA device to the host memory
Test PASSED
Done
```

3.1.1 Tensorflow benchmark

Perform the tensorflow benchmarking step by configuring a test pod to execute the tensorflow benchmarks.

As an output, you'll see that the test pod was deployed using the nvcr.io/nvidia/tensorflow:24.06-tf2-py3 image, requesting two GPU resources, which successfully ran and confirmed GPU functionality with tensorflow from within containers.

```
apiVersion: v1
kind: Pod
metadata:
 name: tf-gpu-bench-full
spec:
 restartPolicy: Never
 containers:
 - name: tf-benchmarks
   image: "nvcr.io/nvidia/tensorflow:24.06-tf2-py3" # Always check for latest OCI release tag: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/tensorflow/tags
   command: ["/bin/sh", "-c"]
   args: ["cd /workspace && git clone https://github.com/tensorflow/benchmarks/ && cd /workspace/benchmarks/scripts/tf _ cnn _ benchmarks && python tf _ cnn _ benchmarks.py --num _ gpus=1 --batch _ size=64 --model=resnet50 --use _ fp16"]
   resources:
     limits:
       nvidia.com/gpu: 2
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ cat cuda _ test.yaml
apiVersion: v1
kind: Pod
metadata:
 name: cuda-gpu-test
spec:
 restartPolicy: OnFailure
```

```
 containers:
   - name: cuda-vector-add
     image: "nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda10.2"
     resources:
       limits:
         nvidia.com/gpu: 2
```

```
sudo k8s kubectl logs tf-gpu-bench-full
```

```
INFO:tensorflow:Done running local_init_op.
I0105 13:54:18.541941 127618454127232 session_manager.py:548] Done running local_init_op.
2026-01-05 13:54:19.425922: I external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:465] Loaded cuDNN version 90100
TensorFlow:  2.16
Model:       resnet50
Dataset:     imagenet (synthetic)
Mode:        training
SingleSess:  False
Batch size:  64 global
             64 per device
Num batches: 100
Num epochs:  0.00
Devices:     ['/gpu:0']
NUMA bind:   False
Data format: NCHW
Optimizer:   sgd
Variables:   parameter_server
==========
Generating training model
Initializing graph
Running warm up
Done warm up
Step    Img/sec total_loss
1       images/sec: 1700.4 +/- 0.0 (jitter = 0.0)       7.603
10      images/sec: 1699.0 +/- 1.1 (jitter = 2.5)       7.849
20      images/sec: 1700.5 +/- 1.1 (jitter = 4.8)       8.015
30      images/sec: 1701.2 +/- 0.8 (jitter = 5.2)       7.939
40      images/sec: 1699.4 +/- 1.5 (jitter = 4.6)       8.133
50      images/sec: 1699.9 +/- 1.3 (jitter = 5.5)       8.053
60      images/sec: 1700.5 +/- 1.1 (jitter = 6.0)       7.786
70      images/sec: 1700.7 +/- 1.0 (jitter = 5.5)       7.859
80      images/sec: 1700.8 +/- 0.9 (jitter = 5.7)       8.005
90      images/sec: 1700.8 +/- 0.8 (jitter = 5.4)       7.857
100     images/sec: 1699.1 +/- 1.2 (jitter = 5.5)       8.079
----------------------------------------------------------------
total images/sec: 1696.69
----------------------------------------------------------------
```

3.1 Networking configuration

Cilium is the default Container Network Interface (CNI) for the Canonical Kubernetes distribution, providing integrated networking and network policy enforcement. Therefore, no additional installation or configuration steps are required.

The MetalLB plug-in should be enabled and configured to allow access to Kubeflow:

```
sudo k8s enable load-balancer
sudo k8s set load-balancer.cidrs=10.2.0.1-10.2.0.254
```

3.1.2 Cilium CNI testing

Perform the Cilium testing using the official Cilium single-node connectivity check.

This requires making adjustments to the yaml file, as coredns is used in Canonical Kubernetes instead of kube-dns.

wget https://raw.githubusercontent.com/cilium/cilium/refs/tags/v1.17.1/examples/kubernetes/connectivity-check/connectivity-check-single-node.yaml

```
sed -i 's/kube-dns/coredns/g' connectivity-check-single-node.yaml
sudo k8s kubectl create namespace cilium-test
sudo k8s kubectl apply -f connectivity-check-single-node.yaml
```

The previous command deploys the test pods and the necessary Cilium network policies:

```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s kubectl get ciliumnetworkpolicy -n cilium-test
NAME                                AGE     VALID
pod-to-a-allowed-cnp                3d23h   True
pod-to-a-denied-cnp                 3d23h   True
pod-to-external-fqdn-allow-google-cnp   3d23h   False
```

The test is successful if all created pods are in running state:

```
sudo k8s kubectl get pods -n cilium-test
```

```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s kubectl get pods -n cilium-test
NAME                                                READY   STATUS    RESTARTS   AGE
echo-a-54dcdd77c-9nfr4                              1/1     Running   0          3d23h
echo-b-549fdb8f8c-6f9vf                             1/1     Running   0          3d23h
echo-b-host-7cfdb688b7-27lgh                        1/1     Running   0          3d23h
pod-to-a-5f56dc8c9b-l5zw8                           1/1     Running   0          3d23h
pod-to-a-allowed-cnp-5dc859fd98-bcv4f               1/1     Running   0          3d23h
pod-to-a-denied-cnp-68976d7584-fsn26                1/1     Running   0          3d23h
pod-to-b-intra-node-nodeport-5884978697-6rgcg       1/1     Running   0          3d23h
pod-to-external-1111-797c647566-wqjd7               1/1     Running   0          3d23h
pod-to-external-fqdn-allow-google-cnp-5688c867dd-524x7   1/1     Running   0          3d23h
```

4. Canonical's MLOps stack

4.1 Introduction to Canonical's MLOps stack

Canonical's MLOps stack provides a robust, production-ready platform for machine learning and deep learning workloads on Kubernetes. The suite is based on two key components, both deployed and managed using Juju:

- **Charmed Kubeflow:** Canonical's enterprise-ready MLOps platform
- **Charmed MLflow:** a lightweight platform for managing the ML lifecycle, including experimentation, reproducibility, and deployment

Deploying these charmed operators through Juju brings you the power of Kubernetes while simplifying the complex operational tasks associated with running a full-featured MLOps environment.

4.2 Installation

In this guide, the installation of Canonical's MLOps stack is performed using the Juju operator lifecycle manager. This approach automates the deployment, scaling, integration, and management of the operators.

4.2.1 Juju setup

Install Juju configured Juju connect to the existing Canonical Kubernetes cluster:

```
sudo snap install juju
sudo k8s config | juju add-k8s ai-pod --client
juju bootstrap ai-pod
juju add-model kubeflow
```

4.2.2 Charmed Kubeflow deployment

Deploy the Charmed Kubeflow bundle from the 1.10/stable channel:

```
juju deploy kubeflow --trust --channel=1.10/stable
```

Wait until the deployed Juju applications are in an active state:

```
juju status --watch 5s
```

```
Model      Controller  Cloud/Region  Version  SLA          Timestamp
kubeflow   ai-pod      ai-pod        3.6.12   unsupported  15:25:33Z

App                     Version                  Status  Scale  Charm                    Channel          Rev  Address         Exposed  Message
admission-webhook                                active      1  admission-webhook        1.10/stable      467  10.152.183.197  no
argo-controller                                  active      1  argo-controller          3.5/stable       818  10.152.183.40   no
dex-auth                                         active      1  dex-auth                 2.41/stable      699  10.152.183.168  no
envoy                                            active      1  envoy                    2.4/stable       406  10.152.183.72   no
istio-ingressgateway                             active      1  istio-gateway            1.24/stable     1474  10.152.183.218  no
istio-pilot                                      active      1  istio-pilot              1.24/stable     1417  10.152.183.113  no
jupyter-controller                               active      1  jupyter-controller       1.10/stable     1301  10.152.183.175  no
jupyter-ui                                       active      1  jupyter-ui               1.10/stable     1298  10.152.183.112  no
katib-controller                                 active      1  katib-controller         0.18/stable     1163  10.152.183.147  no
katib-db                8.0.41-0ubuntu0.22.04.1  active      1  mysql-k8s                8.0/stable       255  10.152.183.123  no
katib-db-manager                                 active      1  katib-db-manager         0.18/stable     1123  10.152.183.178  no
katib-ui                                         active      1  katib-ui                 0.18/stable     1122  10.152.183.107  no
kfp-api                                          active      1  kfp-api                  2.5/stable      2240  10.152.183.95   no
kfp-db                  8.0.41-0ubuntu0.22.04.1  active      1  mysql-k8s                8.0/stable       255  10.152.183.163  no
kfp-metadata-writer                              active      1  kfp-metadata-writer      2.5/stable      1310  10.152.183.88   no
kfp-persistence                                  active      1  kfp-persistence          2.5/stable      2251  10.152.183.45   no
kfp-profile-controller                           active      1  kfp-profile-controller   2.5/stable      2211  10.152.183.219  no
kfp-schedwf                                      active      1  kfp-schedwf              2.5/stable      2261  10.152.183.188  no
kfp-ui                                           active      1  kfp-ui                   2.5/stable      2253  10.152.183.97   no
kfp-viewer                                       active      1  kfp-viewer               2.5/stable      2281  10.152.183.65   no
kfp-viz                                          active      1  kfp-viz                  2.5/stable      2197  10.152.183.83   no
knative-eventing                                 active      1  knative-eventing         1.16/stable      866  10.152.183.138  no
knative-operator                                 active      1  knative-operator         1.16/stable      827  10.152.183.96   no
knative-serving                                  active      1  knative-serving          1.16/stable      865  10.152.183.236  no
kserve-controller                                active      1  kserve-controller        0.14/stable      951  10.152.183.195  no
kubeflow-dashboard                               active      1  kubeflow-dashboard       1.10/stable      847  10.152.183.229  no
kubeflow-profiles                                active      1  kubeflow-profiles        1.10/stable      772  10.152.183.202  no
kubeflow-roles                                   active      1  kubeflow-roles           1.10/stable      357  10.152.183.213  no
kubeflow-volumes                                 active      1  kubeflow-volumes         1.10/stable      514  10.152.183.133  no
metacontroller-operator                          active      1  metacontroller-operator  4.11/stable      510  10.152.183.237  no
minio                   res:oci-image@7f2474f    active      1  minio                    ckf-1.10/stable  583  10.152.183.84   no
mlmd                                             active      1  mlmd                     ckf-1.10/stable  344  10.152.183.114  no
oidc-gatekeeper                                  active      1  oidc-gatekeeper          ckf-1.10/stable  556  10.152.183.184  no
pvcviewer-operator                               active      1  pvcviewer-operator       1.10/stable      293  10.152.183.101  no
resource-dispatcher                              active      1  resource-dispatcher      2.0/stable       402  10.152.183.253  no
tensorboard-controller                           active      1  tensorboard-controller   1.10/stable      557  10.152.183.68   no
tensorboards-web-app                             active      1  tensorboards-web-app     1.10/stable      543  10.152.183.134  no
training-operator                                active      1  training-operator        1.9/stable       653  10.152.183.165  no
```

4.2.3 Charmed MLflow deployment

Deploy Charmed MLflow separately and integrate it with Kubeflow:

```
juju deploy mlflow --channel=2.22/stable -trust
```

The integration with Kubeflow is done using the Resource dispatcher operator, which enables all Kubeflow users to access the MLflow model registry from their namespaces:

```
juju deploy resource-dispatcher --channel 2.0/stable --trust
#Integrate mlflow to the resource dispatcher
juju integrate mlflow-server:secrets resource-dispatcher:secrets
juju integrate mlflow-server:pod-defaults resource-dispatcher:pod-defaults
juju integrate mlflow-minio:object-storage kserve-controller:object-storage
juju integrate kserve-controller:service-accounts resource-dispatcher:service-accounts
juju integrate kserve-controller:secrets resource-dispatcher:secrets
#Integrate mlflow to the kubeflow dashboard
juju integrate mlflow-server:ingress istio-pilot:ingress
juju integrate mlflow-server:dashboard-links kubeflow-dashboard:links
```

Wait until the deployed Juju applications are in an active state (notice how now also mlflow-* applications are deployed):

```
juju status --watch 5s
```

```
Model     Controller  Cloud/Region  Version  SLA          Timestamp
kubeflow  ai-pod      ai-pod        3.6.12   unsupported  15:39:35Z

App                       Version                  Status  Scale  Charm                     Channel          Rev  Address          Exposed  Message
admission-webhook                                  active      1  admission-webhook         1.10/stable      467  10.152.183.197   no
argo-controller                                    active      1  argo-controller           3.5/stable       818  10.152.183.40    no
dex-auth                                           active      1  dex-auth                  2.41/stable      699  10.152.183.168   no
envoy                                              active      1  envoy                     2.4/stable       406  10.152.183.72    no
istio-ingressgateway                               active      1  istio-gateway             1.24/stable     1474  10.152.183.218   no
istio-pilot                                        active      1  istio-pilot               1.24/stable     1417  10.152.183.113   no
jupyter-controller                                 active      1  jupyter-controller        1.10/stable     1301  10.152.183.175   no
jupyter-ui                                         active      1  jupyter-ui                1.10/stable     1298  10.152.183.112   no
katib-controller                                   active      1  katib-controller          0.18/stable     1163  10.152.183.147   no
katib-db                  8.0.41-0ubuntu0.22.04.1  active      1  mysql-k8s                 8.0/stable       255  10.152.183.123   no
katib-db-manager                                   active      1  katib-db-manager          0.18/stable     1123  10.152.183.178   no
katib-ui                                           active      1  katib-ui                  0.18/stable     1122  10.152.183.107   no
kfp-api                                            active      1  kfp-api                   2.5/stable      2240  10.152.183.95    no
kfp-db                    8.0.41-0ubuntu0.22.04.1  active      1  mysql-k8s                 8.0/stable       255  10.152.183.163   no
kfp-metadata-writer                                active      1  kfp-metadata-writer       2.5/stable      1310  10.152.183.88    no
kfp-persistence                                    active      1  kfp-persistence           2.5/stable      2251  10.152.183.45    no
kfp-profile-controller                             active      1  kfp-profile-controller    2.5/stable      2211  10.152.183.219   no
kfp-schedwf                                        active      1  kfp-schedwf               2.5/stable      2261  10.152.183.188   no
kfp-ui                                             active      1  kfp-ui                    2.5/stable      2253  10.152.183.97    no
kfp-viewer                                         active      1  kfp-viewer                2.5/stable      2281  10.152.183.65    no
kfp-viz                                            active      1  kfp-viz                   2.5/stable      2197  10.152.183.83    no
knative-eventing                                   active      1  knative-eventing          1.16/stable      866  10.152.183.138   no
knative-operator                                   active      1  knative-operator          1.16/stable      827  10.152.183.96    no
knative-serving                                    active      1  knative-serving           1.16/stable      865  10.152.183.236   no
kserve-controller                                  active      1  kserve-controller         0.14/stable      951  10.152.183.195   no
kubeflow-dashboard                                 active      1  kubeflow-dashboard        1.10/stable      847  10.152.183.229   no
kubeflow-profiles                                  active      1  kubeflow-profiles         1.10/stable      772  10.152.183.202   no
kubeflow-roles                                     active      1  kubeflow-roles            1.10/stable      357  10.152.183.213   no
kubeflow-volumes                                   active      1  kubeflow-volumes          1.10/stable      514  10.152.183.133   no
metacontroller-operator                            active      1  metacontroller-operator   4.11/stable      510  10.152.183.237   no
minio                     res:oci-image@7f2474f    active      1  minio                     ckf-1.10/stable  583  10.152.183.84    no
mlflow-minio              res:oci-image@7f2474f    active      1  minio                     ckf-1.10/stable  583  10.152.183.131   no
mlflow-mysql              8.0.41-0ubuntu0.22.04.1  active      1  mysql-k8s                 8.0/stable       255  10.152.183.186   no
mlflow-server                                      active      1  mlflow-server             2.22/stable     1252  10.152.183.115   no
mlmd                                               active      1  mlmd                      ckf-1.10/stable  344  10.152.183.114   no
oidc-gatekeeper                                    active      1  oidc-gatekeeper           ckf-1.10/stable  556  10.152.183.184   no
pvcviewer-operator                                 active      1  pvcviewer-operator        1.10/stable      293  10.152.183.101   no
resource-dispatcher                                active      1  resource-dispatcher       2.0/stable       402  10.152.183.253   no
tensorboard-controller                             active      1  tensorboard-controller    1.10/stable      557  10.152.183.68    no
tensorboards-web-app                               active      1  tensorboards-web-app      1.10/stable      543  10.152.183.134   no
training-operator                                  active      1  training-operator         1.9/stable       653  10.152.183.165   no
```

4.2.4 Accessing the Kubeflow dashboard

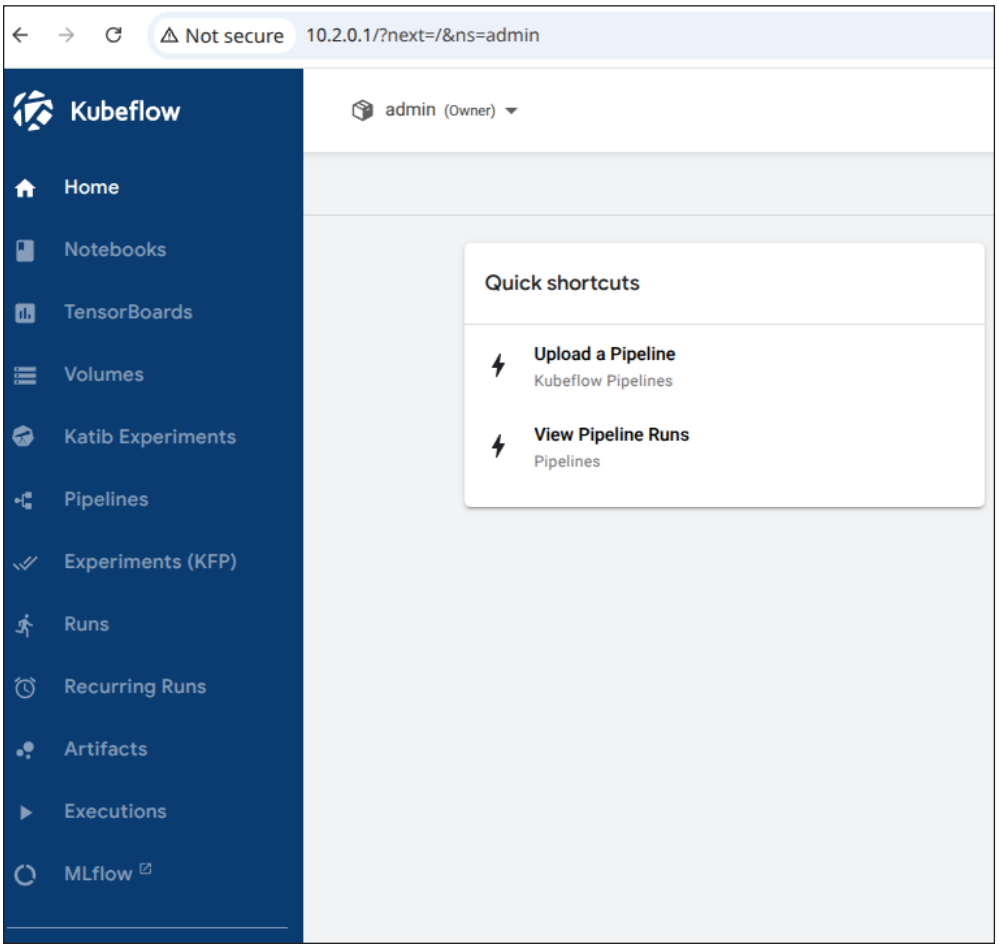
Expose the Kubeflow dashboard using the existing MetalLB load balancer configuration. You can determine the access IP by checking the external IP of the istio ingress gateway service.

Once that is complete, set up Dex authentication to allow access to a first user:

```
juju config dex-auth static-username=admin
juju config dex-auth static-password=admin
sudo k8s kubectl get svc -n kubeflow | grep istio-ingressgateway
```

You can verify that access to the dashboard has been successful through the assigned load-balancer IP.

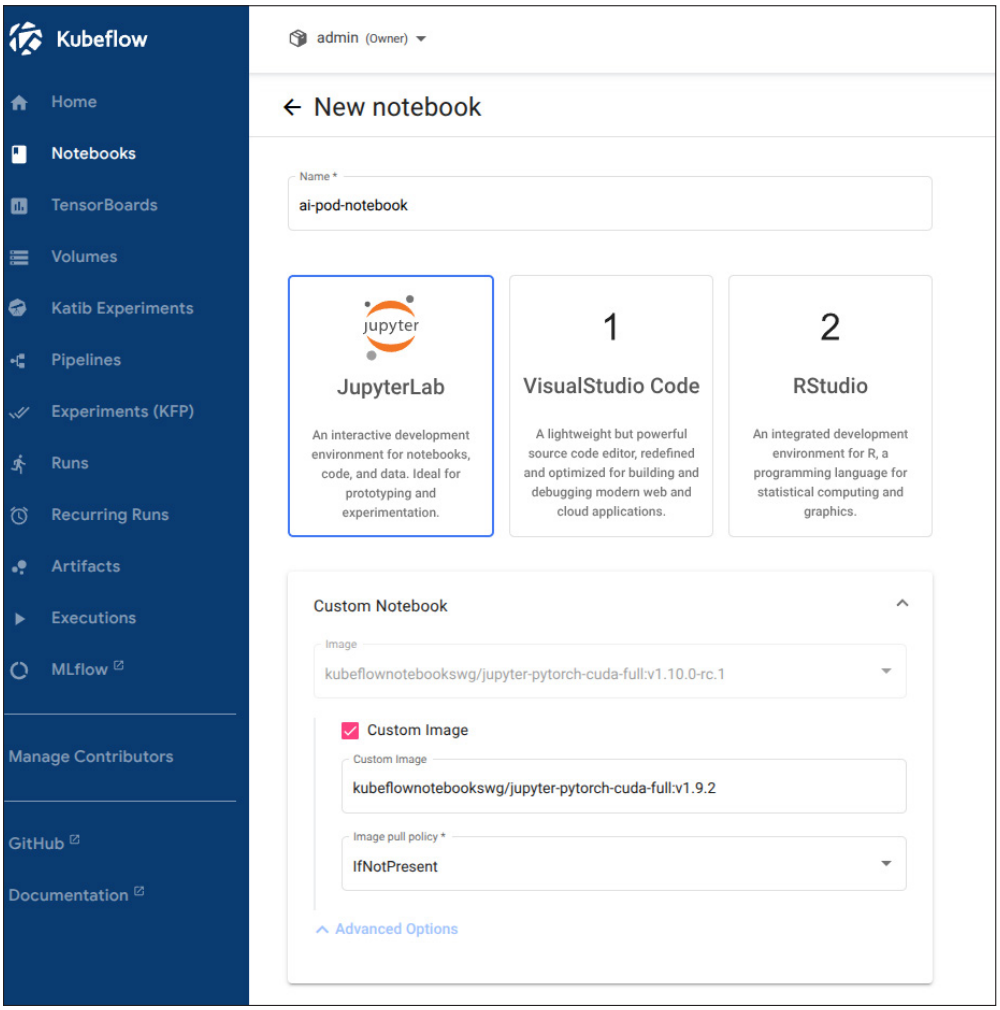
```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s kubectl get svc -n kubeflow | grep istio-ingressgateway
[sudo] password for ubuntu:
istio-ingressgateway              ClusterIP      10.152.183.218   <none>          65535/TCP                    4d
istio-ingressgateway-endpoints    ClusterIP      None             <none>          <none>                       4d
istio-ingressgateway-workload     LoadBalancer   10.152.183.207   10.2.0.1        80:31876/TCP,443:30723/TCP   4d
```



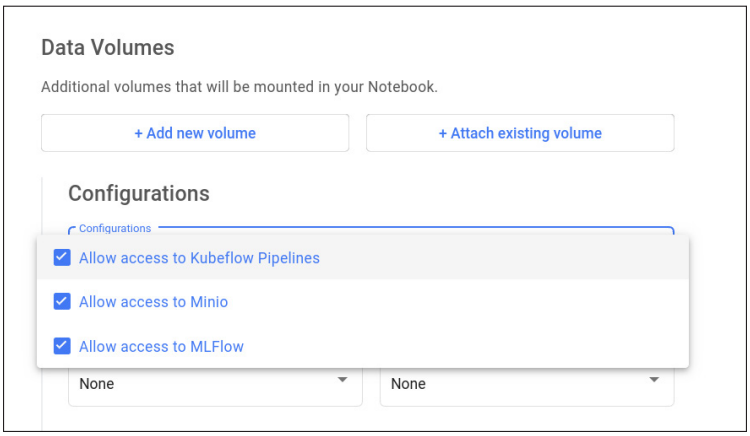
4.2.5 Jupyter notebook creation

From the Notebooks tab, select New Notebook to create a new notebook.

For this showcase, we're creating a Jupyter notebook using the kubeflownotebookswg/jupyter-pytorch-cuda-full:v1.9.2 image, for compatibility with the available Cuda version on the machine (12.2).

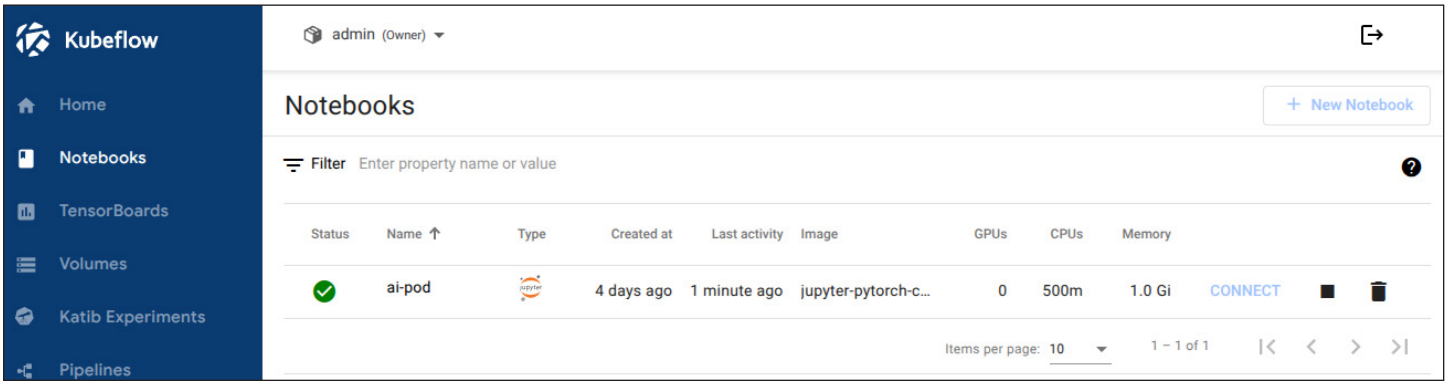


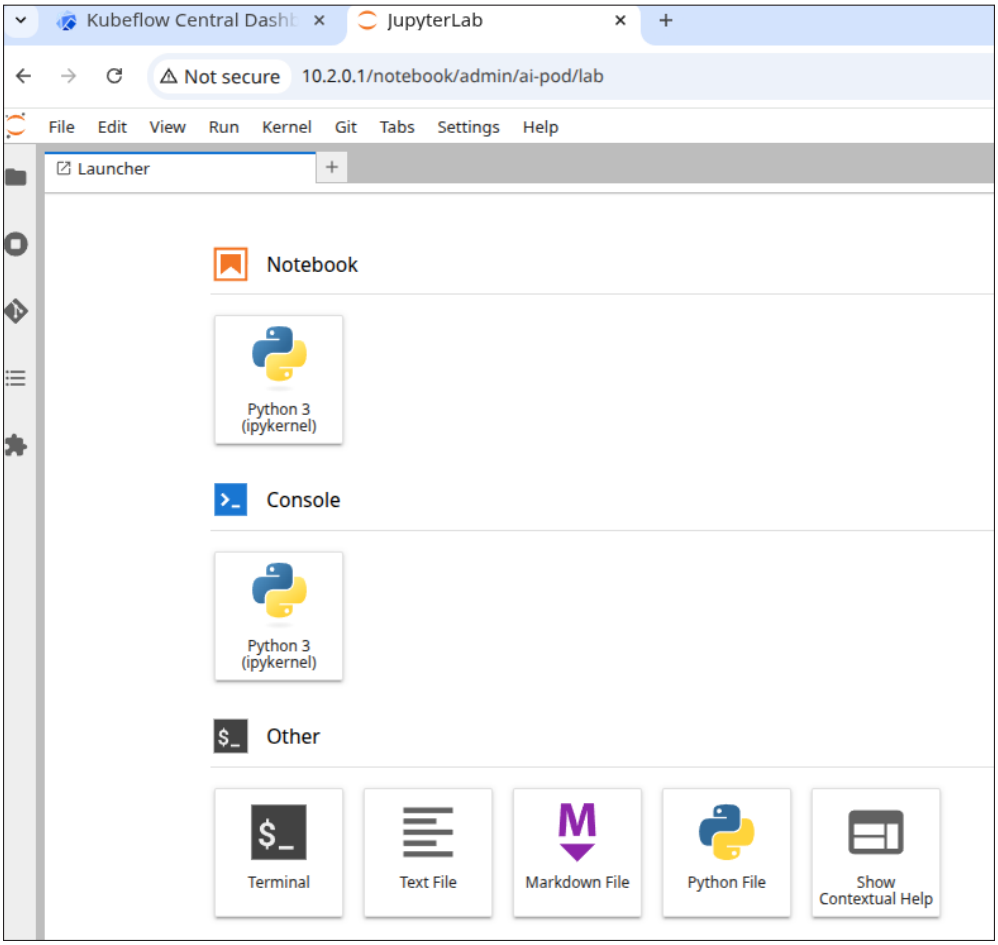
As our notebook needs to be integrated with MLFlow, from Advanced Options, go to Configurations and allow access to Kubeflow Pipelines, MinIO, and MLflow from the dropdown menu:



This will properly setup environment variables in the created jupyter notebook.

After clicking on the Launch button and waiting for the notebook status to become green, clicking on Connect will redirect to the notebook.





4.3 Test showcase: MLOps pipeline setup

4.3.1 Kubeflow + MLflow model training

In the Jupyter notebook, install the prerequisites:

```
pip install mlflow==2.15.1 kserve==0.13.1 tenacity
```

Import the necessary packages:

```
import kfp
import mlflow
import os
import requests


from kfp.dsl import Input, Model, component
from kfp.dsl import InputPath, OutputPath, pipeline, component
from kserve import KServeClient
from mlflow.tracking import MlflowClient
from tenacity import retry, stop_after_attempt, wait_exponential
```

Create a pipeline component that downloads and saves the model as .csv:

```
@component(
    base_image="python:3.11",
    packages_to_install=["requests==2.32.3", "pandas==2.2.2"]
)
def download_dataset(url: str, dataset_path: OutputPath('Dataset')) -> None:
    import requests
    import pandas as pd
    response = requests.get(url)
    response.raise_for_status()
    from io import StringIO
    dataset = pd.read_csv(StringIO(response.text), header=0, sep=";")
    dataset.to_csv(dataset_path, index=False)
```

Create a pipeline component that processes the dataset:

```
@component(
    base_image="python:3.11",
    packages_to_install=["requests==2.32.3", "pandas==2.2.2"]
)
def download_dataset(url: str, dataset_path: OutputPath('Dataset')) -> None:
    import requests
    import pandas as pd
```

```
    response = requests.get(url)
    response.raise_for_status()
    from io import StringIO
    dataset = pd.read_csv(StringIO(response.text), header=0, sep=";")
    dataset.to_csv(dataset_path, index=False)
```

Create a pipeline component that trains the model. This starts an MLFlow training session, which will result in the model being trained and stored on MinIO, ready to be deployed:

```
@component(
    base_image="python:3.11",
    packages_to_install=["pandas==2.2.2", "scikit-learn==1.5.1", "mlflow==2.15.1",
"pyarrow==15.0.2", "boto3==1.34.162"]
)
def train_model(dataset: InputPath('Dataset'), run_name: str, model_name: str) -> str:
    import os
    import mlflow
    import pandas as pd
    from sklearn.linear_model import ElasticNet
    from sklearn.model_selection import train_test_split
    df = pd.read_parquet(dataset)
    target_column = "quality"
    train_x, test_x, train_y, test_y = train_test_split(
        df.drop(columns=[target_column]),
        df[target_column], test_size=0.25,
        random_state=42, stratify=df[target_column]
    )
    mlflow.sklearn.autolog()
    with mlflow.start_run(run_name=run_name) as run:
        mlflow.set_tag("author", "kf-testing")
        lr = ElasticNet(alpha=0.5, l1_ratio=0.5, random_state=42)
        lr.fit(train_x, train_y)
        mlflow.sklearn.log_model(lr, "model", registered_model_name=model_name)
        model_uri = f"{run.info.artifact_uri}/model"
        print(model_uri)
        return model_uri
```

Now create a pipeline component that deploys the trained model and serves it using Kserve, reserving a GPU for the served model:

```
@component(
    base _ image="python:3.11",
    packages _ to _ install=["kserve==0.13.1", "kubernetes==26.1.0", "tenacity==9.0.0"]
)
def deploy _ model _ with _ kserve(model _ uri: str, isvc _ name: str) -> str:
    from kubernetes.client import V1ObjectMeta, V1ResourceRequirements
    from kserve import (
        constants,
        KServeClient,
        V1beta1InferenceService,
        V1beta1InferenceServiceSpec,
        V1beta1PredictorSpec,
        V1beta1SKLearnSpec,
    )
    from tenacity import retry, wait _ exponential, stop _ after _ attempt
    isvc = V1beta1InferenceService(
        api _ version=constants.KSERVE _ V1BETA1,
        kind=constants.KSERVE _ KIND,
        metadata=V1ObjectMeta(
            name=isvc _ name,
            annotations={"sidecar.istio.io/inject": "false"},
        ),
        spec=V1beta1InferenceServiceSpec(
            predictor=V1beta1PredictorSpec(
                service _ account _ name="kserve-controller-s3",
                sklearn=V1beta1SKLearnSpec(
                    storage _ uri=model _ uri
                ),
                resources=V1ResourceRequirements(
                    limits={"nvidia.com/gpu": "1"}
                )
            )
        )
    )
    client = KServeClient()
    client.create(isvc)
    @retry(
```

```
        wait=wait_exponential(multiplier=2, min=1, max=10),
        stop=stop_after_attempt(30),
        reraise=True,
    )
    def assert_isvc_created(client, isvc_name):
        assert client.is_isvc_ready(isvc_name), f"Failed to create Inference Service {isvc_name}."
    assert_isvc_created(client, isvc_name)
    isvc_resp = client.get(isvc_name)
    isvc_url = isvc_resp['status']['address']['url']
    print("Inference URL:", isvc_url)
    return isvc_url
```

The pipeline can be configured to enable GPU allocation requests for a specific task:

```
def add_gpu_request(task: dsl.PipelineTask) -> dsl.PipelineTask:
    """Add a request field for a GPU to the container created by the PipelineTask object."""
    return task.add_node_selector_constraint(accelerator="nvidia.com/gpu").
set_accelerator_limit(
        limit=1
    )
```

Now, put all components together and define the pipeline:

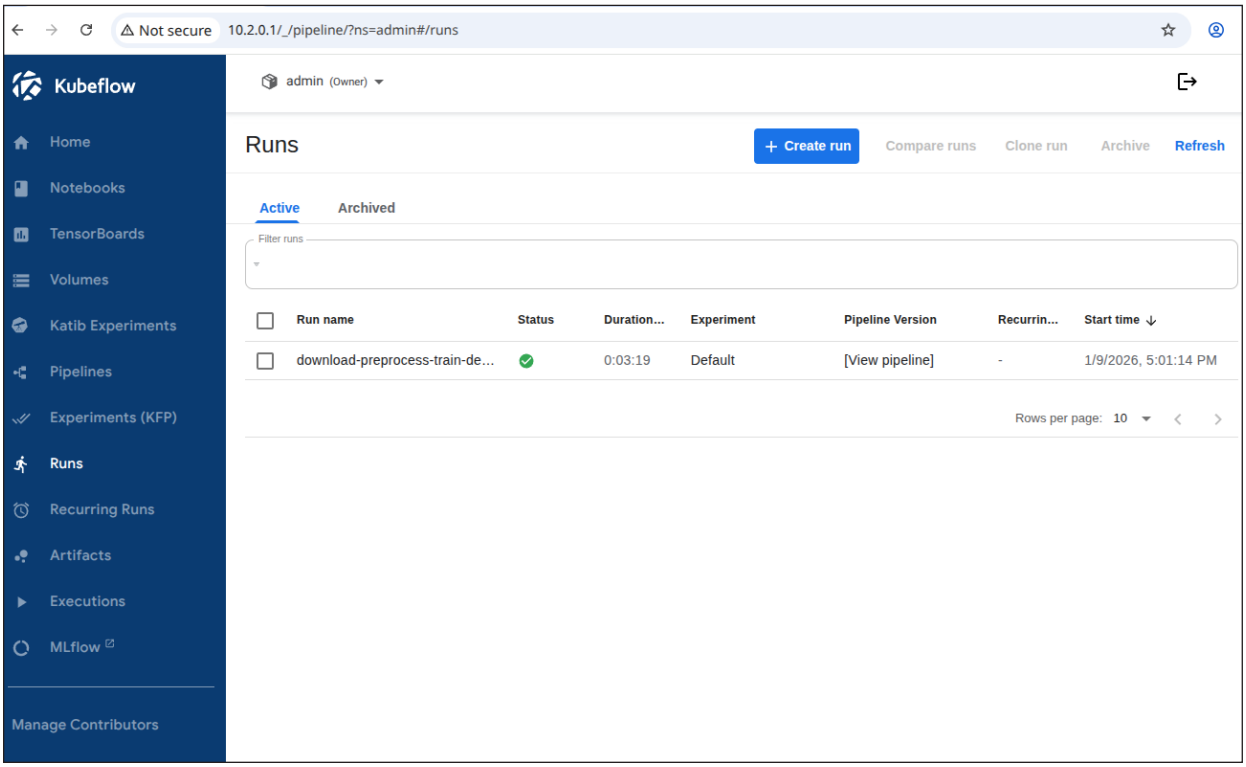
```
ISVC_NAME = "wine-regressor5"
MLFLOW_RUN_NAME = "elastic_net_models"
MLFLOW_MODEL_NAME = "wine-elasticnet"

mlflow_tracking_uri = os.getenv('MLFLOW_TRACKING_URI')
mlflow_s3_endpoint_url = os.getenv('MLFLOW_S3_ENDPOINT_URL')
aws_access_key_id = os.getenv('AWS_ACCESS_KEY_ID')
aws_secret_access_key = os.getenv('AWS_SECRET_ACCESS_KEY')
@pipeline(name='download-preprocess-train-deploy-pipeline')
def download_preprocess_train_deploy_pipeline(url: str):
    download_task = download_dataset(url=url)
    preprocess_task = preprocess_dataset(
        dataset=download_task.outputs['dataset_path']
    )
    train_task = train_model(
        dataset=preprocess_task.outputs['output_file'], run_name=MLFLOW_RUN_NAME,
model_name=MLFLOW_MODEL_NAME
    ).set_env_variable(name='MLFLOW_TRACKING_URI', value=mlflow_tracking_uri)\
    .set_env_variable(name='MLFLOW_S3_ENDPOINT_URL', value=mlflow_s3_endpoint_url)\
```

```
    .set_env_variable(name='AWS_ACCESS_KEY_ID', value=aws_access_key_id)\
    .set_env_variable(name='AWS_SECRET_ACCESS_KEY', value=aws_secret_access_key)
    train_task = add_gpu_request(train_task)
    deploy_task = deploy_model_with_kserve(
        model_uri=train_task.output, isvc_name=ISVC_NAME
    ).set_env_variable(name='AWS_SECRET_ACCESS_KEY', value=aws_secret_access_key)
```

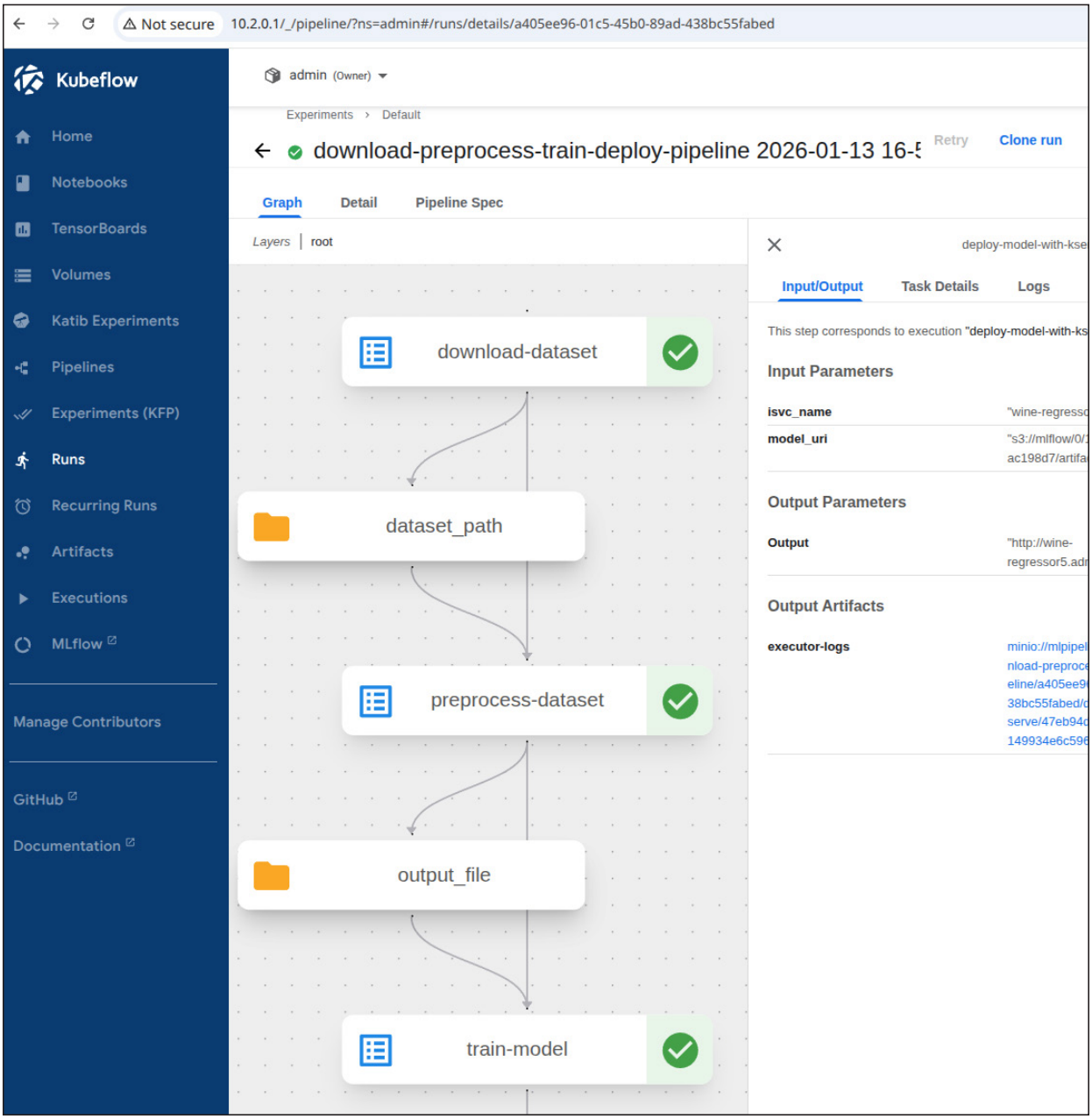
Now, let's run it:

```
client = kfp.Client()
url = 'https://raw.githubusercontent.com/canonical/kubeflow-examples/main/e2e-wine-kfp-mlflow/winequality-red.csv'
kfp.compiler.Compiler().compile(download_preprocess_train_deploy_pipeline, 'download_preprocess_train_deploy_pipeline.yaml')
run = client.create_run_from_pipeline_func(download_preprocess_train_deploy_pipeline, arguments={'url': url}, enable_caching=False)
```

The pipeline run will now appear in the Run tab in the Kubeflow UI:



Once the pipeline is completed, the output of the last step will be the URL of the deployed model using Kserve:



Which we can query from the Jupyter notebook:

```
kserve _ client = KServeClient()

isvc _ resp = kserve _ client.get(ISVC _ NAME)
inference _ service _ url = isvc _ resp['status']['address']['url']
print("Inference URL:", inference _ service _ url)
input _ data = {
    "instances": [
        [7.4, 0.7, 0.0, 1.9, 0.076, 11.0, 34.0, 0.9978, 3.51, 0.56, 9.4],
        [7.8, 0.88, 0.0, 2.6, 0.098, 25.0, 67.0, 0.9968, 3.2, 0.68, 9.8]
    ]
}
response = requests.post(f"{inference _ service _ url}/v1/models/{ISVC _ NAME}:predict",
json=input _ data)
print(response.text)
```

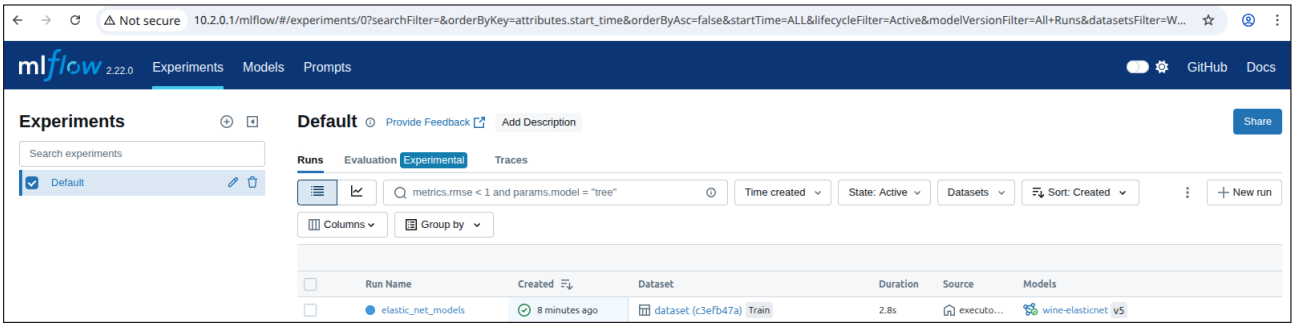
```
[26]: kserve_client = KServeClient()

      isvc_resp = kserve_client.get(ISVC_NAME)
      inference_service_url = isvc_resp['status']['address']['url']
      print("Inference URL:", inference_service_url)

      input_data = {
          "instances": [
              [7.4, 0.7, 0.0, 1.9, 0.076, 11.0, 34.0, 0.9978, 3.51, 0.56, 9.4],
              [7.8, 0.88, 0.0, 2.6, 0.098, 25.0, 67.0, 0.9968, 3.2, 0.68, 9.8]
          ]
      }

      response = requests.post(f"{inference_service_url}/v1/models/{ISVC_NAME}:predict", json=input_data)
      print(response.text)

      Inference URL: http://wine-regressor5.admin.svc.cluster.local
      {"predictions":[5.575510497546586,5.527143590500911]}
```

Check that the GPU has correctly been assigned to the Kserve pod:

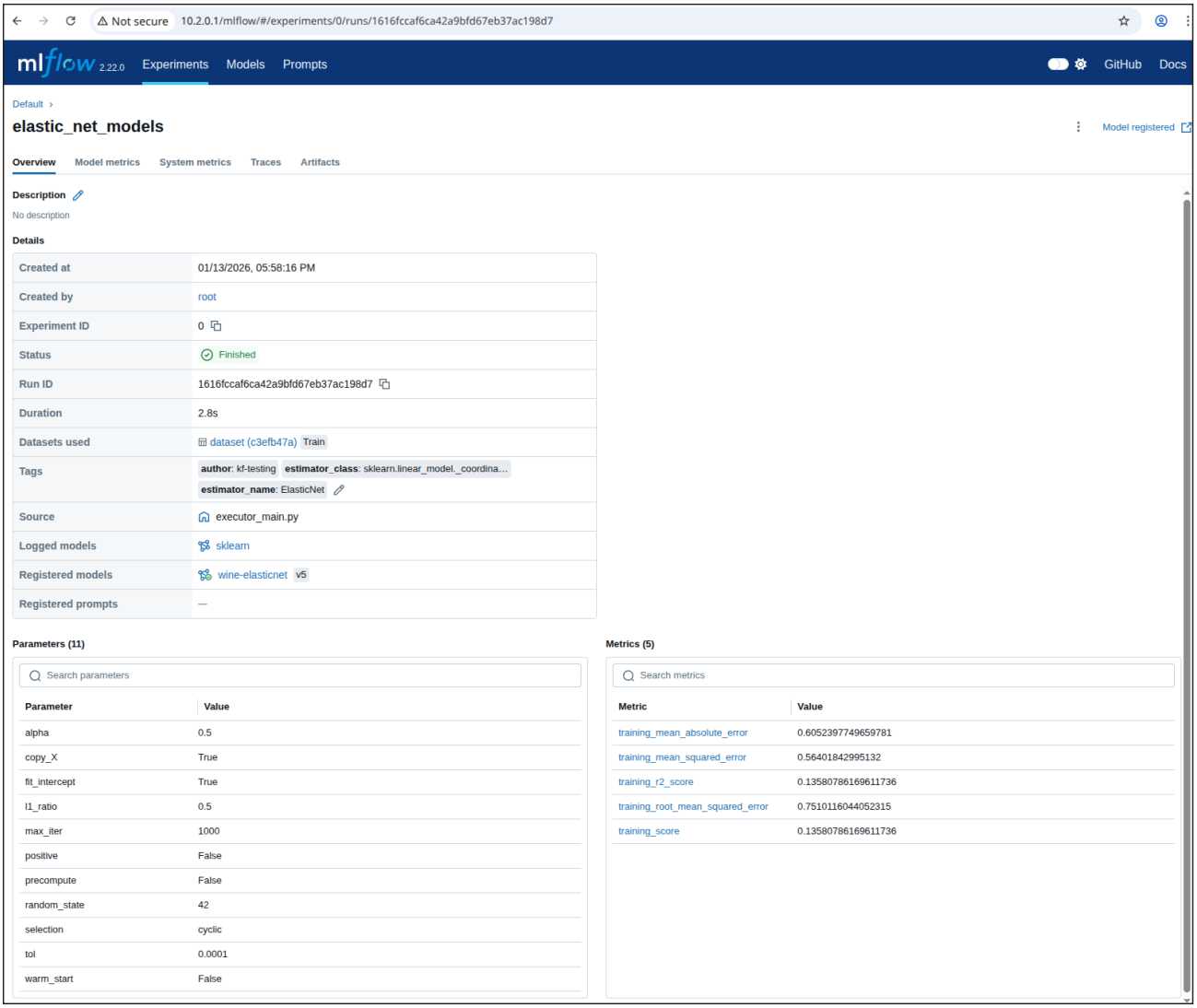
```
ubuntu@cpn-ful-ai-c245-gpuwn4:~$ sudo k8s kubectl exec wine-regressor5-predictor-00001-deployment-79668c7bfd-gqlbr -n admin -- nvidia-smi -L
Defaulted container "kserve-container" out of: kserve-container, queue-proxy, storage-initializer (init)
GPU 0: NVIDIA L40S (UUID: GPU-78311ddd-7ec6-d098-8a7a-7b850e1d82e5)
```

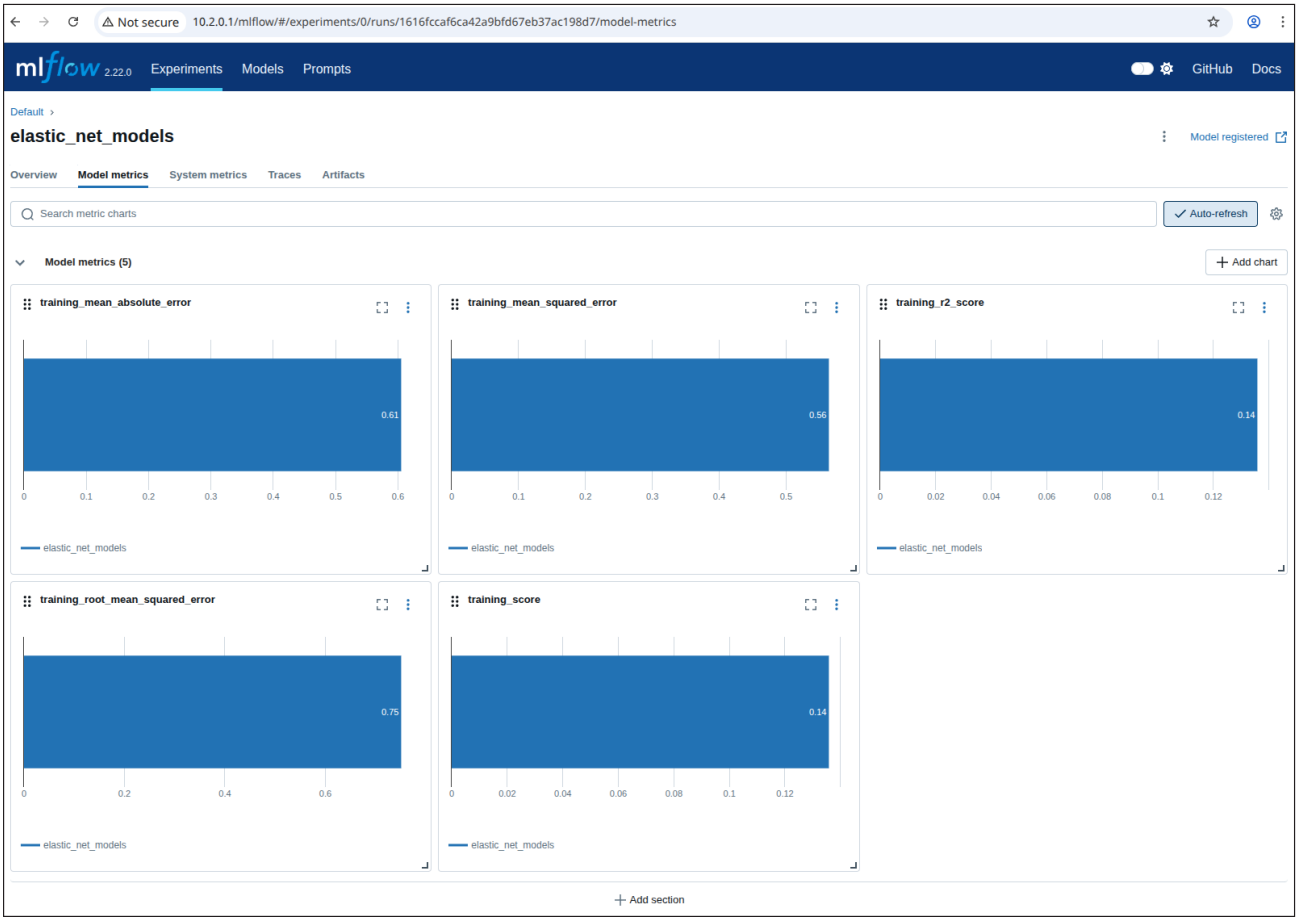
The pipeline compiled in the Jupyter notebook is shown in the UIs as a Kubeflow run and as an MLflow experiment, used for tracking parameters, metrics, artifacts, data, and environment configuration. Additionally, the ElasticNet regression model is also stored in the MLflow model registry, which enables model versioning, aliasing, tracking, and annotations.

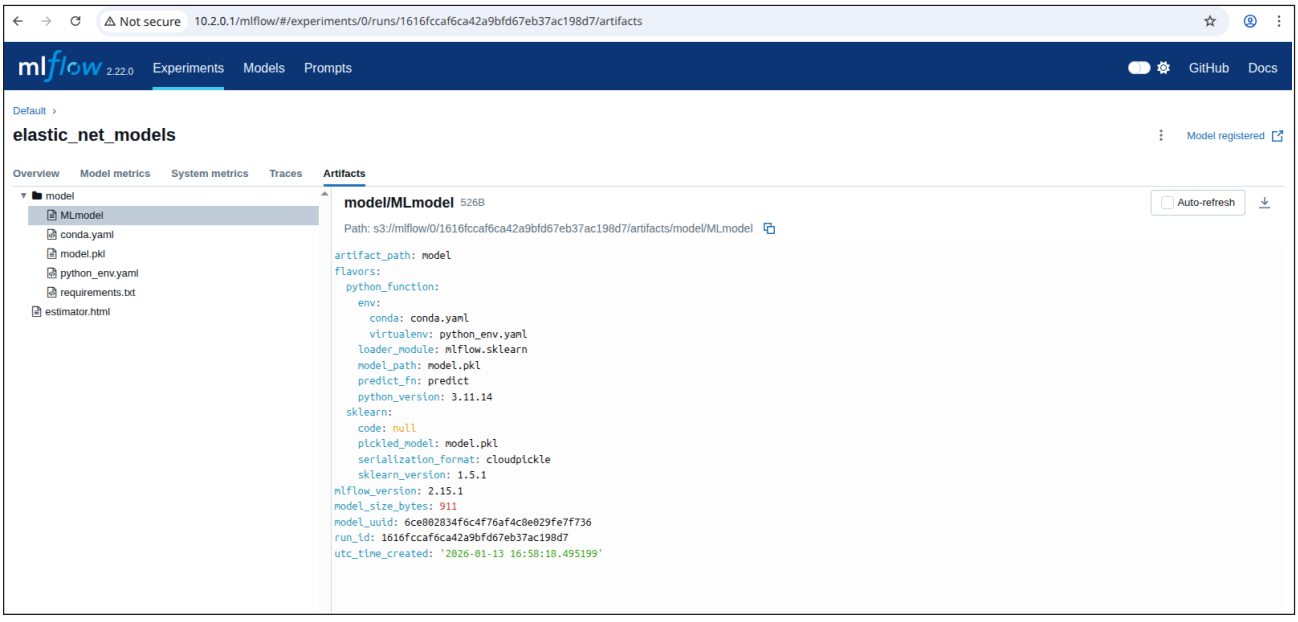
To view the MLflow tracking UI, select MLflow from the Kubeflow central dashboard sidebar. Within Experiments, information about each experiment is available, including used datasets, hyperparameters, and model metrics:



Clicking on the model, you can see information related to registered models, including descriptions, tags, and versions:







Conclusion

When scaling your AI/ML workloads, it's important you have a platform that empowers you to move quickly and efficiently.

Cisco AI PODs offer a robust, pre-integrated hardware layer including networking, built on a validated reference architecture. Canonical Kubernetes and Canonical's MLOps stack make it easy to set up your system quickly and helps to ensure your GPUs are working at their full potential.

For more information

- Cisco Secure AI Factory with NVIDIA
- Cisco AI PODs Data Sheet
- Cisco UCS C885A M8 Rack Server Data Sheet
- Cisco Nexus Hyperfabric AI Infrastructure Data Sheet
- Cisco Hypershield Solution Overview
- Cisco AI Defense Data Sheet
- Cisco Intersight Platform

To find out more about what you can do with Canonical's solutions, see the following further reading:

- Canonical Kubernetes documentation
- Charmed Kubeflow documentation
- Charmed MLflow documentation

Annex A: Cilium CNI testing yaml

```
---
metadata:
  name: echo-a
  labels:
    name: echo-a
    topology: any
    component: network-check
    traffic: internal
    quarantine: “false”
    type: autocheck
spec:
  template:
    metadata:
      labels:
        name: echo-a
    spec:
      hostNetwork: false
      containers:
      - name: echo-a-container
        env:
        - name: PORT
          value: “8080”
        ports:
        - containerPort: 8080
        image: quay.io/cilium/json-mock:v1.3.2@
sha256:bc6c46c74efadb135bc996c2467cece6989302371ef4e3f068361460abaf39be
        imagePullPolicy: IfNotPresent
        terminationMessagePolicy: FallbackToLogsOnError
        readinessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - “5”
            - -o
```

```
            - /dev/null
            - localhost:8080
        livenessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - localhost:8080
  selector:
    matchLabels:
      name: echo-a
  replicas: 1
apiVersion: apps/v1
kind: Deployment
---
metadata:
  name: echo-b
  labels:
    name: echo-b
    topology: any
    component: services-check
    traffic: internal
    quarantine: "false"
    type: autocheck
spec:
  template:
    metadata:
      labels:
        name: echo-b
    spec:
      hostNetwork: false
      containers:
      - name: echo-b-container
```

```
        env:
        - name: PORT
          value: "8080"
        ports:
        - containerPort: 8080
          hostPort: 40000
        image: quay.io/cilium/json-mock:v1.3.2@
sha256:bc6c46c74efadb135bc996c2467cece6989302371ef4e3f068361460abaf39be
        imagePullPolicy: IfNotPresent
        terminationMessagePolicy: FallbackToLogsOnError
        readinessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - localhost:8080
        livenessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - localhost:8080
  selector:
    matchLabels:
      name: echo-b
  replicas: 1
apiVersion: apps/v1
```

```
kind: Deployment
---
metadata:
  name: echo-b-host
  labels:
    name: echo-b-host
    topology: any
    component: services-check
    traffic: internal
    quarantine: “false”
    type: autocheck
spec:
  template:
    metadata:
      labels:
        name: echo-b-host
    spec:
      hostNetwork: true
      containers:
      - name: echo-b-host-container
        env:
        - name: PORT
          value: “21000”
        ports: []
        image: quay.io/cilium/json-mock:v1.3.2@
sha256:bc6c46c74efadb135bc996c2467cece6989302371ef4e3f068361460abaf39be
        imagePullPolicy: IfNotPresent
        terminationMessagePolicy: FallbackToLogsOnError
        readinessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - “5”
            - -o
            - /dev/null
            - localhost:21000
```

```
        livenessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - localhost:21000
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: name
                operator: In
                values:
                - echo-b
            topologyKey: kubernetes.io/hostname
  selector:
    matchLabels:
      name: echo-b-host
  replicas: 1
apiVersion: apps/v1
kind: Deployment
---
metadata:
  name: pod-to-a
  labels:
    name: pod-to-a
    topology: any
    component: network-check
    traffic: internal
    quarantine: "false"
    type: autocheck
```

```
spec:
  template:
    metadata:
      labels:
        name: pod-to-a
    spec:
      hostNetwork: false
      containers:
      - name: pod-to-a-container
        ports: []
        image: quay.io/cilium/alpine-curl:v1.5.0@
sha256:7b286939730d8af1149ef88dba15739d8330bb83d7d9853a23e5ab4043e2d33c
        imagePullPolicy: IfNotPresent
        command:
        - /bin/ash
        - -c
        - sleep 1000000000
        terminationMessagePolicy: FallbackToLogsOnError
        readinessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - echo-a:8080/public
        livenessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
```

```
            - /dev/null
            - echo-a:8080/public
  selector:
    matchLabels:
      name: pod-to-a
  replicas: 1
apiVersion: apps/v1
kind: Deployment
---
metadata:
  name: pod-to-external-1111
  labels:
    name: pod-to-external-1111
    topology: any
    component: network-check
    traffic: external
    quarantine: "false"
    type: autocheck
spec:
  template:
    metadata:
      labels:
        name: pod-to-external-1111
    spec:
      hostNetwork: false
      containers:
      - name: pod-to-external-1111-container
        ports: []
        image: quay.io/cilium/alpine-curl:v1.5.0@
sha256:7b286939730d8af1149ef88dba15739d8330bb83d7d9853a23e5ab4043e2d33c
        imagePullPolicy: IfNotPresent
        command:
        - /bin/ash
        - -c
        - sleep 1000000000
        terminationMessagePolicy: FallbackToLogsOnError
        readinessProbe:
          timeoutSeconds: 7
          exec:
            command:
```

```
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - https://1.1.1.1
        livenessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - https://1.1.1.1
  selector:
    matchLabels:
      name: pod-to-external-1111
  replicas: 1
apiVersion: apps/v1
kind: Deployment
---
metadata:
  name: pod-to-a-denied-cnp
  labels:
    name: pod-to-a-denied-cnp
    topology: any
    component: policy-check
    traffic: internal
    quarantine: "false"
    type: autocheck
```

```
spec:
  template:
    metadata:
      labels:
        name: pod-to-a-denied-cnp
    spec:
      hostNetwork: false
      containers:
      - name: pod-to-a-denied-cnp-container
        ports: []
        image: quay.io/cilium/alpine-curl:v1.5.0@
sha256:7b286939730d8af1149ef88dba15739d8330bb83d7d9853a23e5ab4043e2d33c
        imagePullPolicy: IfNotPresent
        command:
        - /bin/ash
        - -c
        - sleep 1000000000
        terminationMessagePolicy: FallbackToLogsOnError
        readinessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - ash
            - -c
            - '! curl -s --fail --connect-timeout 5 -o /dev/null echo-a:8080/private'
        livenessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - ash
            - -c
            - '! curl -s --fail --connect-timeout 5 -o /dev/null echo-a:8080/private'
  selector:
    matchLabels:
      name: pod-to-a-denied-cnp
  replicas: 1
apiVersion: apps/v1
kind: Deployment
---
```

```
metadata:
  name: pod-to-a-allowed-cnp
  labels:
    name: pod-to-a-allowed-cnp
    topology: any
    component: policy-check
    traffic: internal
    quarantine: “false”
    type: autocheck
spec:
  template:
    metadata:
      labels:
        name: pod-to-a-allowed-cnp
    spec:
      hostNetwork: false
      containers:
      - name: pod-to-a-allowed-cnp-container
        ports: []
        image: quay.io/cilium/alpine-curl:v1.5.0@
sha256:7b286939730d8af1149ef88dba15739d8330bb83d7d9853a23e5ab4043e2d33c
        imagePullPolicy: IfNotPresent
        command:
        - /bin/ash
        - -c
        - sleep 1000000000
        terminationMessagePolicy: FallbackToLogsOnError
        readinessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - “5”
            - -o
            - /dev/null
            - echo-a:8080/public
        livenessProbe:
```

```
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - echo-a:8080/public
  selector:
    matchLabels:
      name: pod-to-a-allowed-cnp
  replicas: 1
apiVersion: apps/v1
kind: Deployment
---
metadata:
  name: pod-to-external-fqdn-allow-google-cnp
  labels:
    name: pod-to-external-fqdn-allow-google-cnp
    topology: any
    component: policy-check
    traffic: external
    quarantine: "false"
    type: autocheck
spec:
  template:
    metadata:
      labels:
        name: pod-to-external-fqdn-allow-google-cnp
    spec:
      hostNetwork: false
      containers:
      - name: pod-to-external-fqdn-allow-google-cnp-container
        ports: []
        image: quay.io/cilium/alpine-curl:v1.5.0@
sha256:7b286939730d8af1149ef88dba15739d8330bb83d7d9853a23e5ab4043e2d33c
        imagePullPolicy: IfNotPresent
```

```
        command:
        - /bin/ash
        - -c
        - sleep 1000000000
        terminationMessagePolicy: FallbackToLogsOnError
        readinessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - www.google.com
        livenessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - www.google.com
  selector:
    matchLabels:
      name: pod-to-external-fqdn-allow-google-cnp
  replicas: 1
apiVersion: apps/v1
kind: Deployment
---
```

```
metadata:
  name: pod-to-b-intra-node-nodeport
  labels:
    name: pod-to-b-intra-node-nodeport
    topology: intra-node
    component: nodeport-check
    traffic: internal
    quarantine: “false”
    type: autocheck
spec:
  template:
    metadata:
      labels:
        name: pod-to-b-intra-node-nodeport
    spec:
      hostNetwork: false
      containers:
      - name: pod-to-b-intra-node-nodeport-container
        ports: []
        image: quay.io/cilium/alpine-curl:v1.5.0@
sha256:7b286939730d8af1149ef88dba15739d8330bb83d7d9853a23e5ab4043e2d33c
        imagePullPolicy: IfNotPresent
        command:
        - /bin/ash
        - -c
        - sleep 1000000000
        terminationMessagePolicy: FallbackToLogsOnError
        readinessProbe:
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - “5”
            - -o
            - /dev/null
            - echo-b-host-headless:31414/public
        livenessProbe:
```

```
          timeoutSeconds: 7
          exec:
            command:
            - curl
            - -sS
            - --fail
            - --connect-timeout
            - "5"
            - -o
            - /dev/null
            - echo-b-host-headless:31414/public
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: name
                operator: In
                values:
                - echo-b
            topologyKey: kubernetes.io/hostname
  selector:
    matchLabels:
      name: pod-to-b-intra-node-nodeport
  replicas: 1
apiVersion: apps/v1
kind: Deployment
---
metadata:
  name: echo-a
  labels:
    name: echo-a
    topology: any
    component: network-check
    traffic: internal
    quarantine: "false"
    type: autocheck
```

```
spec:
  ports:
  - name: http
    port: 8080
  type: ClusterIP
  selector:
    name: echo-a
apiVersion: v1
kind: Service
---
metadata:
  name: echo-b
  labels:
    name: echo-b
    topology: any
    component: services-check
    traffic: internal
    quarantine: “false”
    type: autocheck
spec:
  ports:
  - name: http
    port: 8080
    nodePort: 31414
  type: NodePort
  selector:
    name: echo-b
apiVersion: v1
kind: Service
---
metadata:
  name: echo-b-headless
  labels:
    name: echo-b-headless
    topology: any
    component: services-check
    traffic: internal
    quarantine: “false”
    type: autocheck
```

```
spec:
  ports:
  - name: http
    port: 8080
  type: ClusterIP
  selector:
    name: echo-b
  clusterIP: None
apiVersion: v1
kind: Service
---
metadata:
  name: echo-b-host-headless
  labels:
    name: echo-b-host-headless
    topology: any
    component: services-check
    traffic: internal
    quarantine: "false"
    type: autocheck
spec:
  ports: []
  type: ClusterIP
  selector:
    name: echo-b-host
  clusterIP: None
apiVersion: v1
kind: Service
---
metadata:
  name: pod-to-a-denied-cnp
  labels:
    name: pod-to-a-denied-cnp
    topology: any
    component: policy-check
    traffic: internal
    quarantine: "false"
    type: autocheck
```

```
spec:
  endpointSelector:
    matchLabels:
      name: pod-to-a-denied-cnp
  egress:
  - toPorts:
    - ports:
      - port: “53”
        protocol: ANY
    toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: kube-system
        k8s:k8s-app: coredns
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: kube-system
        k8s:k8s-app: node-local-dns
  - toPorts:
    - ports:
      - port: “5353”
        protocol: UDP
    toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: openshift-dns
        k8s:dns.operator.openshift.io/daemonset-dns: default
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
---
metadata:
  name: pod-to-a-allowed-cnp
  labels:
    name: pod-to-a-allowed-cnp
    topology: any
    component: policy-check
    traffic: internal
    quarantine: “false”
    type: autocheck
```

```
spec:
  endpointSelector:
    matchLabels:
      name: pod-to-a-allowed-cnp
  egress:
  - toPorts:
    - ports:
      - port: “8080”
        protocol: TCP
    toEndpoints:
    - matchLabels:
        name: echo-a
  - toPorts:
    - ports:
      - port: “53”
        protocol: ANY
    toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: kube-system
        k8s:k8s-app: coredns
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: kube-system
        k8s:k8s-app: node-local-dns
  - toPorts:
    - ports:
      - port: “5353”
        protocol: UDP
    toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: openshift-dns
        k8s:dns.operator.openshift.io/daemonset-dns: default
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
---
metadata:
  name: pod-to-external-fqdn-allow-google-cnp
  labels:
    name: pod-to-external-fqdn-allow-google-cnp
    topology: any
```

```
    component: policy-check
    traffic: external
    quarantine: “false”
    type: autocheck
spec:
  endpointSelector:
    matchLabels:
      name: pod-to-external-fqdn-allow-google-cnp
  egress:
  - toFQDNs:
    - matchPattern: ‘*.google.com’
  - toPorts:
    - ports:
      - port: “53”
        protocol: ANY
      rules:
        dns:
        - matchPattern: ‘*’
    toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: kube-system
        k8s:k8s-app: coredns
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: kube-system
        k8s:k8s-app: node-local-dns
  - toPorts:
    - ports:
      - port: “5353”
        protocol: UDP
      rules:
        dns:
        - matchPattern: ‘*’
    toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: openshift-dns
        k8s:dns.operator.openshift.io/daemonset-dns: default
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
```