August 2023

# Run AI at the Edge: Use MicroK8s with Charmed Kubeflow on NVIDIA EGX platform

## Table of contents

# Overview

This document serves as a reference architecture guide for the infrastructure required for machine learning (ML) use cases. It focuses specifically on open-source ML software, including MicroK8s and Charmed Kubeflow, both delivered by Canonical and running on the NVIDIA EGX platform, which includes NVIDIA-Certified Systems provided by various hardware vendors.

Canonical MicroK8s is a Kubernetes distribution certified by the Cloud Native Computing Foundation (CNCF). Ongoing collaboration between NVIDIA and Canonical ensures continuous validation of test suites, enabling data scientists to benefit from infrastructure designed for AI at scale using their preferred MLOps (Machine Learning Operations) tooling, such as Charmed Kubeflow.

From a business use case perspective, this architecture offers several important advantages:

1. **Faster Iteration and experimentation:** the increased flexibility provided by this infrastructure allows data scientists to iterate faster on AI/ML models and accelerates the experimentation process.

2. **Scalability:** The architecture enables quick scaling of AI initiatives by providing infrastructure that is compatible and tested with various MLOps tooling options.

3. **Security:** Secure workloads can run on Ubuntu-optimized infrastructure, benefiting from regular patching, upgrades, and updates.

4. **AI-specific Requirements:** The architecture meets the specific needs of AI workloads by efficiently handling large datasets on an optimized hardware and software stack.

5. **End to end stack:** The architecture leverages NVIDIA EGX and NGC offerings and utilizes Canonical's MLOps platform, Charmed Kubeflow, to provide a stack for the end-to-end machine learning lifecycle.

6. **Reproducibility:** The solution offers a clear guide that can be used by professionals across the organization, expecting the same outcome.

While data scientists and machine learning engineers are the primary beneficiaries, as they can now easily run ML workloads on high-end hardware with powerful computing capabilities, other key stakeholders who can benefit from this architecture include infrastructure builders, solution architects, DevOps engineers, and CTOs who are looking to swiftly advance their AI initiatives while addressing the challenges that arise when working with AI at scale.

The guide covers hardware specifications, tools, services, and provides a step-by-step guide for setting up the hardware and software required to run ML workloads. It also delves into other tools used for cluster monitoring and management, explaining how all these components work together in the system. At the end of it, users will have a stack that is able to run AI at the edge.

# Executive summary

A Kubernetes cluster is now a common requirement for many organizations to execute cloud-native workloads. Kubeflow is an open, community-driven project that aims to simplify the deployment and management of a machine learning stack on any CNCF-compliant Kubernetes. Canonical's MicroK8s and Charmed Kubeflow which are based on upstream distributions, are comprehensive and reliable solutions. Charmed Kubeflow simplifies the deployment and management of AI workflows, providing an extensive ecosystem of tools and frameworks.

NVIDIA and Canonical have collaborated to test and build a reference architecture that outlines the software, hardware, and integration points of all solution components for deploying and operating Charmed Kubeflow on MicroK8s. This architecture serves as a starting point for setting up and deploying Kubeflow on on-premises infrastructure. Machine learning is a powerful tool gaining wide acceptance across all industry segments, solving various problems and achieving state-of-the-art results due to the abundance of available data and access to high-performance compute infrastructure. Kubeflow is a suitable choice for deploying and working with machine learning components as it enables a seamless transition using containers to deploy and scale ML code from developer systems to scale-out infrastructure without requiring any code modifications. Kubernetes and Kubeflow together truly democratize machine learning for organizations and make it possible for all organizations to accelerate their journey towards becoming AI-enabled companies.

Charmed Kubeflow, running on NVIDIA EGX platform, provides a comprehensive end-to-end stack that enables enterprises to maximize AI efficiency and seamlessly take models from concept to production in an automated and robust manner.

# 1 Solution overview

The reference architecture (Fig1) includes Ubuntu running on NVIDIA EGX platform, MicroK8s, and Charmed Kubeflow to provide a comprehensive solution for developing, deploying and managing AI workloads in edge computing environments.

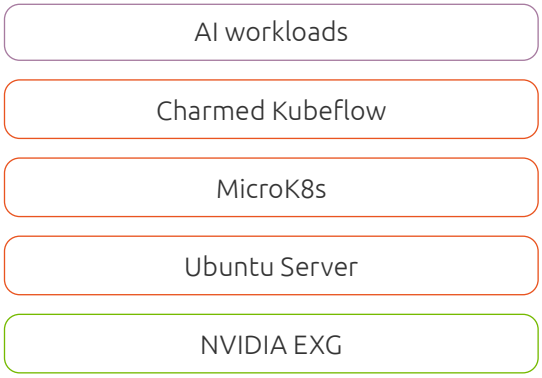



Fig 1. Reference architecture solution overview

NVIDIA EGX platform is the foundation of the architecture, offering high-performance server builds that are approved by NVIDIA. These servers are equipped with NVIDIA GPU cards, enabling powerful GPU-accelerated computing capabilities for AI workloads. It accelerates project delivery and allows professionals to iterate faster.

By combining these components, the reference architecture enables organizations to leverage the power of NVIDIA EGX platform for AI workloads at the edge. Ubuntu ensures a reliable and secure operating system, while MicroK8s provides efficient container orchestration. Charmed Kubeflow simplifies the deployment and management of AI workflows, providing an extensive ecosystem of tools and frameworks.

By leveraging enterprise support from both NVIDIA and Canonical, users of the stack can significantly enhance security and availability. The close engineering collaboration between the two companies ensures expedited bug fixes and prompt security updates, often before public patch releases.

This solution can also run on DGX platforms as shown in the full stack diagram below, check out the joint whitepaper of Run AI at Scale to learn more about Kubeflow on NVIDIA DGX. But for this particular implementation we will be running on EGX platform.

## 1.1 Benefits

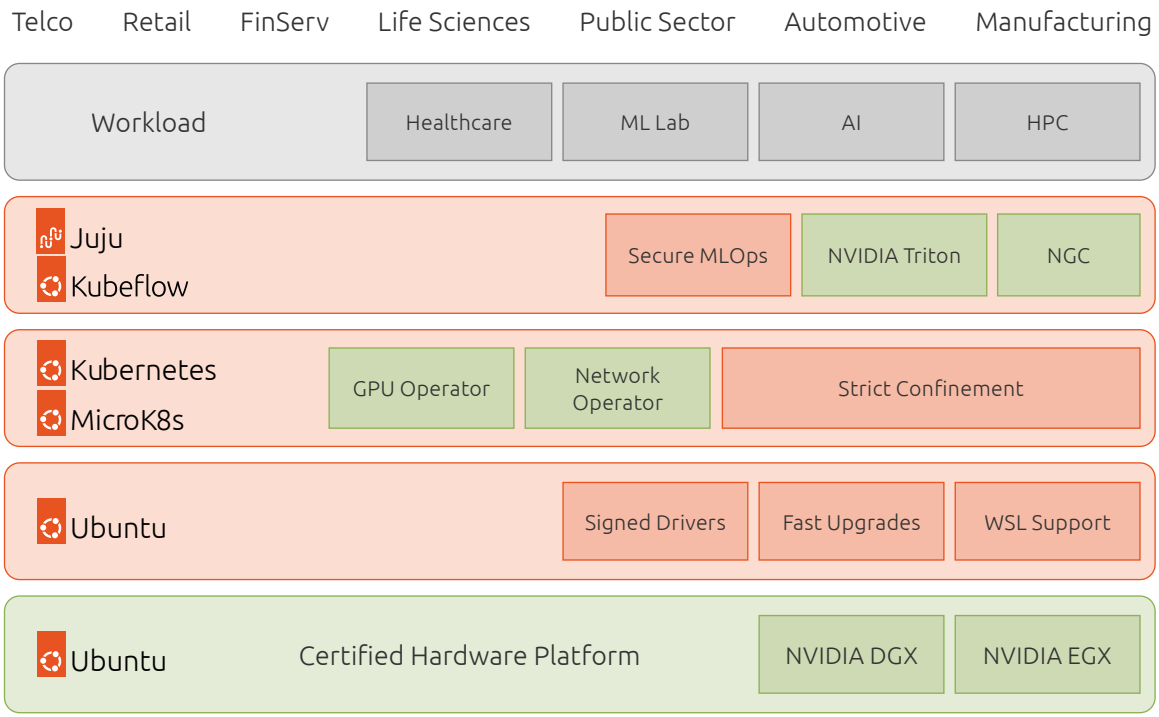



Fig 2. The enterprise-grade AI stack

NVIDIA and Canonical work together across the stack to get the best performance from your hardware, ensuring the fastest and most efficient operations.

- Support across generations of NVIDIA-Certified Systems and Ubuntu LTS releases
- Wide range of GPU driver support for NVIDIA GPUs - whether you want a robust production-ready version, or the bleeding-edge experimental latest versions.
- Enterprise-ready GPU Drivers from NVIDIA, signed by Canonical. Secure boot.
- Deep integrations with NVIDIA engineering getting integrated solutions that offer highest performance and 'just-work' out of the box.
- Enterprise support, backed by NVIDIA. Deep engineering relationship means that we are often able to get bugs fixed with NVIDIA faster, and at times even before they are public.
- All of this builds on the underlying security and LTS value proposition of Ubuntu.
- Leverage the familiarity and efficiency of Ubuntu, already embraced by AI/ML developers, by adopting it as your unified production environment.
- Take advantage of Canonical's comprehensive support offerings to meet all your AI/ML requirements with confidence.
- Secure open source software for machine learning operations as part of a growing portfolio of applications that include Charmed Kubeflow, Charmed MLFlow, or Spark.
- Monitor production-grade infrastructure using Canonical's Observability stack.

# 2 Canonical software components

The standards-based APIs are the same between all Kubernetes deployments, and they enable customer and vendor ecosystems to operate across multiple clouds. The site specific infrastructure combines open and proprietary software, NVIDIA and Canonical certified hardware, and operational processes to deliver cloud resources as a service.

The implementation choices for each cloud infrastructure are highly specific to the requirements of each site. Many of these choices can be standardized and automated using the tools in this reference architecture. Conforming to best practices helps reduce operational risk by leveraging the accumulated experience of NVIDIA and Canonical.

## 2.1 Ubuntu Server

Ubuntu Pro is a subscription-based offering that extends the standard Ubuntu distribution with additional features and support for enterprise environments. With Ubuntu Pro,  organizations gain access to an expanded security maintenance coverage that spans over 30,000 packages for a duration of 10 years, and optional enterprise-grade phone and ticket support by Canonical.

## 2.2 MicroK8s

Canonical MicroK8s is a CNCF-certified Kubernetes distribution that offers a lightweight and streamlined approach to deploying and managing Kubernetes clusters. It is delivered in the form of a snap - the universal Linux app packaging format which dramatically simplifies the installation and upgrades of its components. MicroK8s installs the NVIDIA operator which allows you to take advantage of the GPU hardware available.

## 2.3 Charmed Kubeflow

Charmed Kubeflow is an enterprise-grade distribution of Kubeflow, a popular open-source machine learning toolkit built for Kubernetes environments. Developed by Canonical, Charmed Kubeflow offers a comprehensive and reliable solution for deploying and managing machine learning workflows.

Charmed Kubeflow is a full set of Kubernetes operators to deliver the 30+ applications and services that make up the latest version of Kubeflow, for easy operations anywhere, from workstations to on-prem, to public cloud and edge.

Canonical delivers Kubeflow components in an automated fashion, using the same approach and toolset as for deploying the infrastructure and Kubernetes cluster - with help of Juju.

## 2.4 Juju

Juju is an open-source framework that helps you move from configuration management to application management across your hybrid cloud estate through sharable, reusable, tiny applications called Charmed Operators.

A Charmed Operator is Juju's expansion and generalization of the Kubernetes notion of an operator. In the Kubernetes tradition, an Operator is an application packaged with all the operational knowledge required to install, maintain and upgrade it on a Kubernetes cluster, container, virtual machine, or bare metal machine, running on public or private cloud.

Canonical has developed and tested the Kubeflow charms for automating the delivery of its components.

## 2.5 Software versions

The following versions of software are part of this reference architecture:

| Component | Version |
|---|---|
| Ubuntu Server | 22.04 LTS (kernel 5.15.0-73-generic) |
| MicroK8s | 1.24.13 |
| Charmed Kubeflow | 1.7 |
| Juju | 2.9.42 |
| Triton | 23.05 |

# 3 NVIDIA Specifications

The reference architecture is based on testing the solution on NVIDIA-certified systems represented by Lenovo SE350 edge server.

The machine is equipped with a single Intel Xeon-D CPU, 192GB of RAM, a single drive with 650GB of capacity and a single Nvidia T4 GPU adapter

## 3.1 NVIDIA-Certified Systems

NVIDIA-Certified Systems  are NVIDIA tested and validated high performance servers that brings accelerated AI computing capabilities to the edge of the network. It is designed to enable organizations to deploy AI workloads closer to the data source, reducing latency and improving real-time processing capabilities. NVIDIA-Certified Systems combine NVIDIA's high-performance GPUs, advanced networking technologies, and optimized software stack to deliver AI computing power directly at the edge devices

## 3.2 NGC

NVIDIA NGC (NVIDIA GPU Cloud) is a comprehensive platform that provides a hub for GPU-optimized software, tools, and pre-trained models for deep learning, machine learning, and accelerated computing. It offers a curated collection of software containers, models, and industry-specific SDKs, enabling developers and researchers to accelerate their AI and data science workflows.
NVIDIA Triton is part of NVIDIA NGC catalog and within the scope of this reference architecture is used as an enhancement of Charmed Kubeflow platform, allowing to perform inference at the edge in a more robust way compared to the other  inference servers.

## 3.3 Triton Inference Server

NVIDIA Triton Inference Server, included in NVIDIA AI Enterprise, is a high-performance inference serving software that simplifies the deployment of AI models in production environments.

Triton's automatic compatibility with diverse model frameworks and formats, including PyTorch, TensorFlow, ONNX, and others, simplifies the integration of models developed using different frameworks. This compatibility ensures smooth and efficient deployment of models without the need for extensive conversion or compatibility adjustments.

It delivers enhanced efficiency in terms of response time and throughput by utilizing dynamic batching and load balancing techniques. Triton's ability to employ multiple model instances enables efficient distribution of the workload, ensuring effective load balancing and maximizing performance. Written in C and optimized by NVIDIA, Triton delivers exceptional performance and resource efficiency without the need for additional optimizations. It is designed to be faster, consume less memory, and require fewer GPU and CPU resources compared to other alternatives.

Triton provides a streamlined and lightweight solution for serving models. Unlike installing multiple dependencies and the entire runtime framework, Triton is a single implementation often packaged as a container. This packaging approach makes it smaller, more lightweight, and easier to manage. By focusing solely on serving models, Triton eliminates unnecessary components and provides a dedicated environment specifically designed for efficient and effective model serving.

Canonical and NVIDIA have worked together to deliver full integration between NVIDIA Triton and Charmed Kubeflow for an end-to-end AI workflow. It uses KServe as the main component to perform model serving using NVIDIA’s framework.

# 4 Tutorial: Deploying an object detection model

Install Ubuntu Server 22.04 LTS on a machine with an NVIDIA GPU.

Update system

```
sudo apt update -y && sudo apt upgrade -y
```

Install MicroK8s

```
sudo snap install microk8s --classic --channel=1.24/stable
```

Add current user to the microk8s group and give access to the .kube directory

```
sudo usermod -a -G microk8s $USER
sudo chown -f -R $USER ~/.kube
```

Log out and re-enter the session for the changes to take effect.

Enable MicroK8s add-ons for Charmed Kubeflow (replace IP addresses accordingly)

```
microk8s enable dns storage ingress gpu metallb:192.168.1.10-192.168.1.16
```

Check MicroK8s status until the output shows “microk8s is running” and the add-ons installed are listed under “enabled”

```
microk8s status --wait-ready
```

Add alias for omitting microk8s when running commands

```
alias kubectl='microk8s kubectl'
echo "alias kubectl='microk8s kubectl'" > ~/.bash_aliases
```

(Optional) Set forward IP address in CoreDNS:

```
microk8s kubectl -n kube-system edit configmap coredns
```

Install Juju

```
sudo snap install juju --classic --channel=2.9/stable
```

Deploy Juju controller to MicroK8s

```
juju bootstrap microk8s
```

Add model for Kubeflow

```
juju add-model kubeflow
```

Deploy Charmed Kubeflow

We need to run the first two commands because MicroK8s uses inotify to interact with the filesystem, and in kubeflow the default inotify limits may be exceeded.

```
sudo sysctl fs.inotify.max_user_instances=1280
sudo sysctl fs.inotify.max_user_watches=655360
juju deploy kubeflow --trust  --channel=1.7/stable
```

Check Juju status until all statuses become active

```
watch -c 'juju status --color | grep -E "blocked|error|maintenance|
waiting|App|Unit"'
```

If tensorboard-controller is stuck with the status message “Waiting for gateway relation”, run the following command. This is a known issue, see tensorboard-controller GitHub issue for more info.

```
juju run --unit istio-pilot/0 -- "export JUJU_DISPATCH_PATH=hooks/config-
changed; ./dispatch"
```

Get the IP address of Istio ingress gateway load balancer

```
IP=$(microk8s kubectl -n kubeflow get svc istio-ingressgateway-workload -o
jsonpath='{.status.loadBalancer.ingress[0].ip}')
```

Configure authentication for dashboard

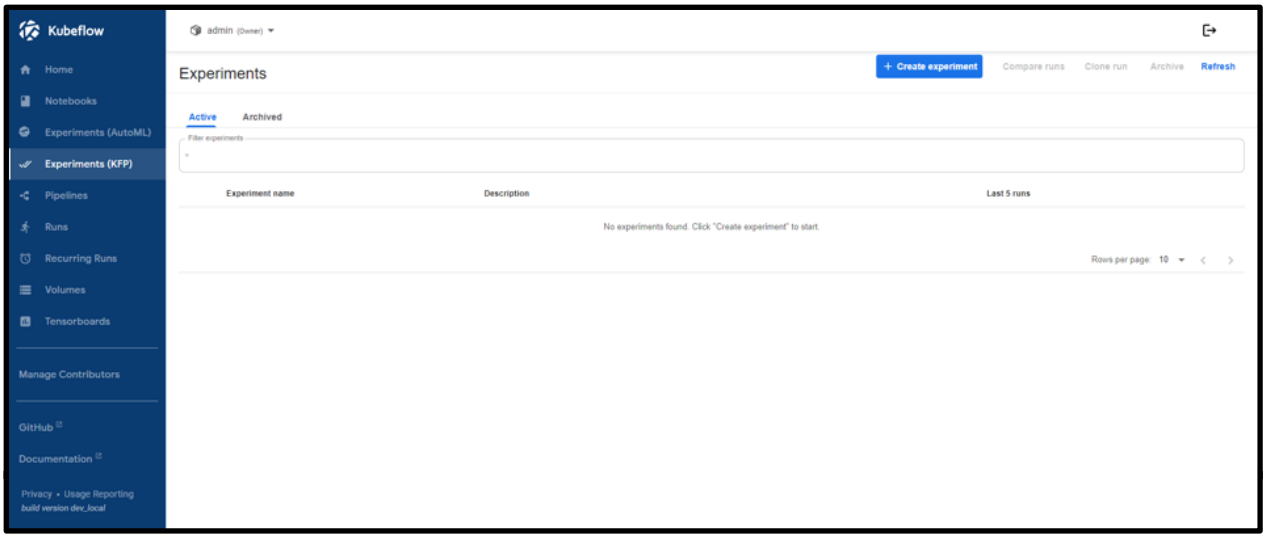
```
juju config dex-auth public-url=http://$IP.nip.io
juju config oidc-gatekeeper public-url=http://$IP.nip.io
juju config dex-auth static-username=admin
juju config dex-auth static-password=admin
```

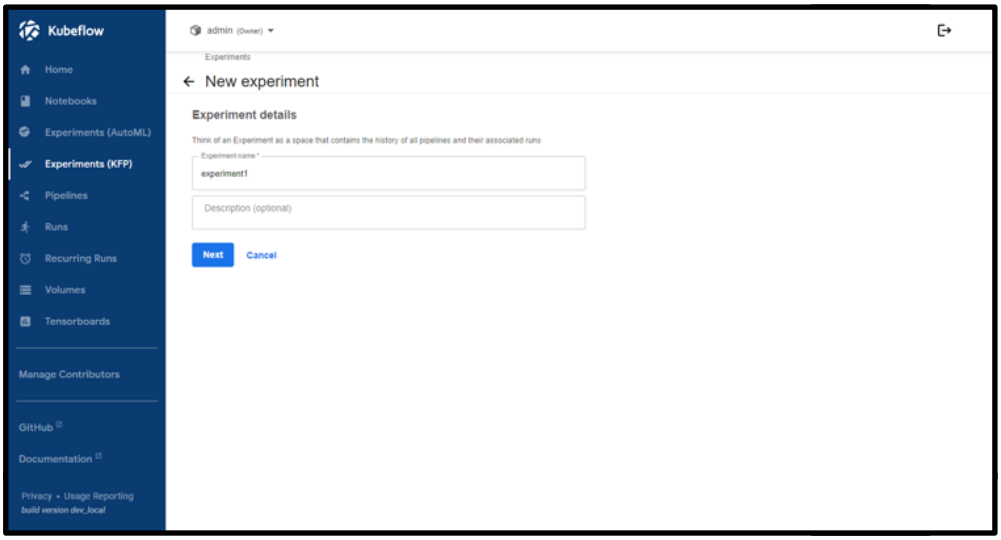
Login to Charmed Kubeflow dashboard with a browser and accept default settings

```
http://$IP.nip.io
```

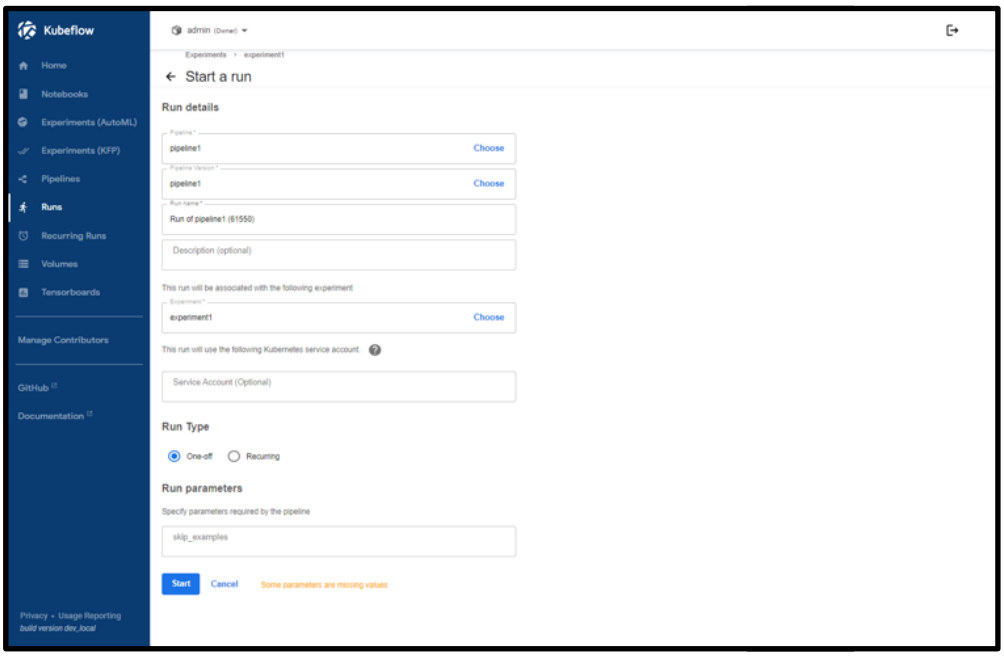
Create pipeline.yaml file on your local machine (refer to Appendix A.1)

Create experiment




Create run and upload pipeline.yaml



References:

https://github.com/NVIDIA/deepops/tree/master/workloads/examples/k8s/kubeflow-pipeline-deploy

# 5 Conclusion

Finally and still important, Canonical not only provides a reference architecture that can be used as the blueprint to build a private cloud, we also use that same architecture for our own internal cloud, open source compute and storage in practice.

The solution outlined above is suitable for running AI at the edge, helping enterprises that leverage workloads in a broad set of industries, from Telco to Healthcare to HPC. Open source machine learning tooling, such as MicroK8s with Charmed Kubeflow, deployed as part of an accelerated computing stack with NVIDIA EGX platform helps professionals to deliver projects faster, reduce operational costs and have an end-to-end experience within the same tool. This reference architecture is only an example of the larger implementation that may solve challenges related to running and ensuring tool compatibility between ecosystem tools and frameworks, thus maintaining security features and optimizing across compute efficiencies.

Furthermore, by leveraging the combined expertise of Canonical and NVIDIA, organizations can enhance data analytics, optimize decision-making processes, and revolutionize customer experiences. With Canonical and NVIDIA as trusted partners, organizations can confidently embrace this solution to drive innovation, accelerate AI adoption, and unlock new opportunities in their respective domains.

For more information, please visit https://ubuntu.com/ai, https://microk8s.io/, and https://ubuntu.com/ai/what-is-kubeflow.

## Apendix

Create pipeline.yaml file on your local machine (Appendix A.1)

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: tritonpipeline-
  labels: {pipelines.kubeflow.org/kfp_sdk_version: 1.8.22}
spec:
  entrypoint: tritonpipeline
  templates:
  - name: condition-skip-examples-download-1
    dag:
      tasks:
      - {name: triton-download, template: triton-download}
  - name: triton-deploy
    container:
      args: ['echo Deploying: /results/model_repository;ls /data; ls /
results; ls
          /checkpoints; tritonserver --model-store=/results/model_
repository']
      command: [/bin/bash, -cx]
      image: nvcr.io/nvidia/tritonserver:23.05-py3
      ports:
      - {containerPort: 8000, hostPort: 8000}
      - {containerPort: 8001, hostPort: 8001}
      - {containerPort: 8002, hostPort: 8002}
      resources:
        limits: {nvidia.com/gpu: 1}
      volumeMounts:
      - {mountPath: /results/, name: triton-results, readOnly: false}
     - {mountPath: /data/, name: triton-data, readOnly: true}
      - {mountPath: /checkpoints/, name: triton-checkpoints, readOnly:
true}
    metadata:
      labels:
        app: triton-kubeflow
        pipelines.kubeflow.org/kfp_sdk_version: 1.8.22
        pipelines.kubeflow.org/pipeline-sdk-type: kfp
        pipelines.kubeflow.org/enable_caching: "true"
    volumes:
    - name: triton-checkpoints
      persistentVolumeClaim: {claimName: triton-checkpoints, readOnly:
false}
    - name: triton-data
      persistentVolumeClaim: {claimName: triton-data, readOnly: false}
    - name: triton-results
      persistentVolumeClaim: {claimName: triton-results, readOnly: false}
  - name: triton-download
container:
      args: ['cd /tmp; git clone https://github.com/triton-inference-server/
server.git;
          cd server/docs/examples; ./fetch_models.sh; cd model_repository;
cp -a .
```

```
          /results/model_repository']
command: [/bin/bash, -cx]
      image: nvcr.io/nvidia/tritonserver:23.05-py3
      volumeMounts:
      - {mountPath: /results/, name: triton-results, readOnly: false}
      - {mountPath: /data/, name: triton-data, readOnly: true}
      - {mountPath: /checkpoints/, name: triton-checkpoints, readOnly:
true}
    metadata:
  labels:
        app: triton-kubeflow
        pipelines.kubeflow.org/kfp_sdk_version: 1.8.22
        pipelines.kubeflow.org/pipeline-sdk-type: kfp
        pipelines.kubeflow.org/enable_caching: "true"
volumes:
    - name: triton-checkpoints
      persistentVolumeClaim: {claimName: triton-checkpoints, readOnly:
false}
- name: triton-data
      persistentVolumeClaim: {claimName: triton-data, readOnly: false}
    - name: triton-results
      persistentVolumeClaim: {claimName: triton-results, readOnly: false}
  - name: triton-service
    resource:
      action: create
      manifest: |
        apiVersion: v1
        kind: Service
        metadata:
          name: triton-kubeflow
        spec:
          ports:
          - name: http
            nodePort: 30800
            port: 8000
            protocol: TCP
            targetPort: 8000
          - name: grpc
            nodePort: 30801
            port: 8001
            targetPort: 8001
          - name: metrics
            nodePort: 30802
            port: 8002
            targetPort: 8002
          selector:
            app: triton-kubeflow
          type: NodePort
outputs:
      parameters:
      - name: triton-service-manifest
        valueFrom: {jsonPath: '{}'}
      - name: triton-service-name
        valueFrom: {jsonPath: '{.metadata.name}'}
```

```
    metadata:
      labels:
        pipelines.kubeflow.org/kfp_sdk_version: 1.8.22
        pipelines.kubeflow.org/pipeline-sdk-type: kfp
        pipelines.kubeflow.org/enable_caching: "true"
  - name: triton-volume-checkpoints
    resource:
      action: apply
      manifest: |
        apiVersion: v1
        kind: PersistentVolumeClaim
metadata:
          name: triton-checkpoints
        spec:
          accessModes:
          - ReadWriteMany
          resources:
            requests:
              storage: 10Gi
          storageClassName: microk8s-hostpath
    outputs:
      parameters:
      - name: triton-volume-checkpoints-manifest
        valueFrom: {jsonPath: '{}'}
      - name: triton-volume-checkpoints-name
        valueFrom: {jsonPath: '{.metadata.name}'}
    metadata:
      labels:
        pipelines.kubeflow.org/kfp_sdk_version: 1.8.22
        pipelines.kubeflow.org/pipeline-sdk-type: kfp
        pipelines.kubeflow.org/enable_caching: "true"
  - name: triton-volume-data
    resource:
      action: apply
      manifest: |
        apiVersion: v1
        kind: PersistentVolumeClaim
        metadata:
          name: triton-data
        spec:
          accessModes:
          - ReadWriteMany
          resources:
            requests:
              storage: 10Gi
          storageClassName: microk8s-hostpath
    outputs:
      parameters:
      - name: triton-volume-data-manifest
        valueFrom: {jsonPath: '{}'}
      - name: triton-volume-data-name
        valueFrom: {jsonPath: '{.metadata.name}'}
    metadata:
      labels:
        pipelines.kubeflow.org/kfp_sdk_version: 1.8.22
        pipelines.kubeflow.org/pipeline-sdk-type: kfp
        pipelines.kubeflow.org/enable_caching: "true"
```

```
  - name: triton-volume-results
    resource:
      action: apply
      manifest: |
        apiVersion: v1
        kind: PersistentVolumeClaim
 metadata:
          name: triton-results
        spec:
          accessModes:
          - ReadWriteMany
          resources:
            requests:
              storage: 10Gi
          storageClassName: microk8s-hostpath
outputs:
      parameters:
 - name: triton-volume-results-manifest
        valueFrom: {jsonPath: '{}'}
      - name: triton-volume-results-name
        valueFrom: {jsonPath: '{.metadata.name}'}
    metadata:
      labels:
        pipelines.kubeflow.org/kfp_sdk_version: 1.8.22
        pipelines.kubeflow.org/pipeline-sdk-type: kfp
        pipelines.kubeflow.org/enable_caching: "true"
  - name: tritonpipeline
    inputs:
      parameters:
      - {name: skip_examples}
    dag:
      tasks:
      - name: condition-skip-examples-download-1
        template: condition-skip-examples-download-1
        when: '"{{inputs.parameters.skip_examples}}" == ""'
        dependencies: [triton-volume-checkpoints, triton-volume-data,
triton-volume-results]
      - name: triton-deploy
        template: triton-deploy
        dependencies: [condition-skip-examples-download-1]
      - {name: triton-service, template: triton-service}
      - {name: triton-volume-checkpoints, template: triton-volume-
checkpoints}
      - {name: triton-volume-data, template: triton-volume-data}
      - {name: triton-volume-results, template: triton-volume-results}
  arguments:
    parameters:
    - {name: skip_examples}
  serviceAccountName: pipeline-runner
```