WHITE PAPER

SUPERMICRO AND MICROK8S TOGETHER RUN AI AT THE EDGE

MicroK8s with Charmed Kubeflow on Supermicro E403 server

Contents

Overview

This document serves as a reference architecture guide for the infrastructure required for machine learning (ML) use cases. This whitepaper focuses on open-source ML software, including MicroK8s and Charmed Kubeflow, delivered by Canonical and running on NVIDIA-certified EGX platforms provided by Supermicro.

Canonical MicroK8s is a Kubernetes distribution certified by the Cloud Native Computing Foundation (CNCF). Ongoing collaboration between Supermicro and Canonical enables data scientists to benefit from an 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 significant 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's EGX offerings and utilizes Canonicals' MLOps platform, Charmed Kubeflow, to provide a stack for the end-to-end machine learning lifecycle.
6. **Reproducibility:** The solution offers a straightforward 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, and 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.

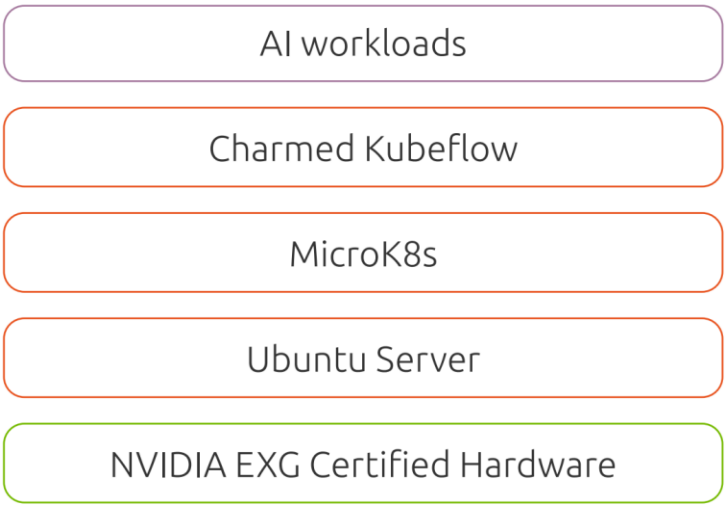
Executive Summary

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. Canonicals' 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.

Supermicro 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 toward becoming AI-enabled companies.

Charmed Kubeflow, running on NVIDIA EGX by Supermicro, 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.

Solution Overview



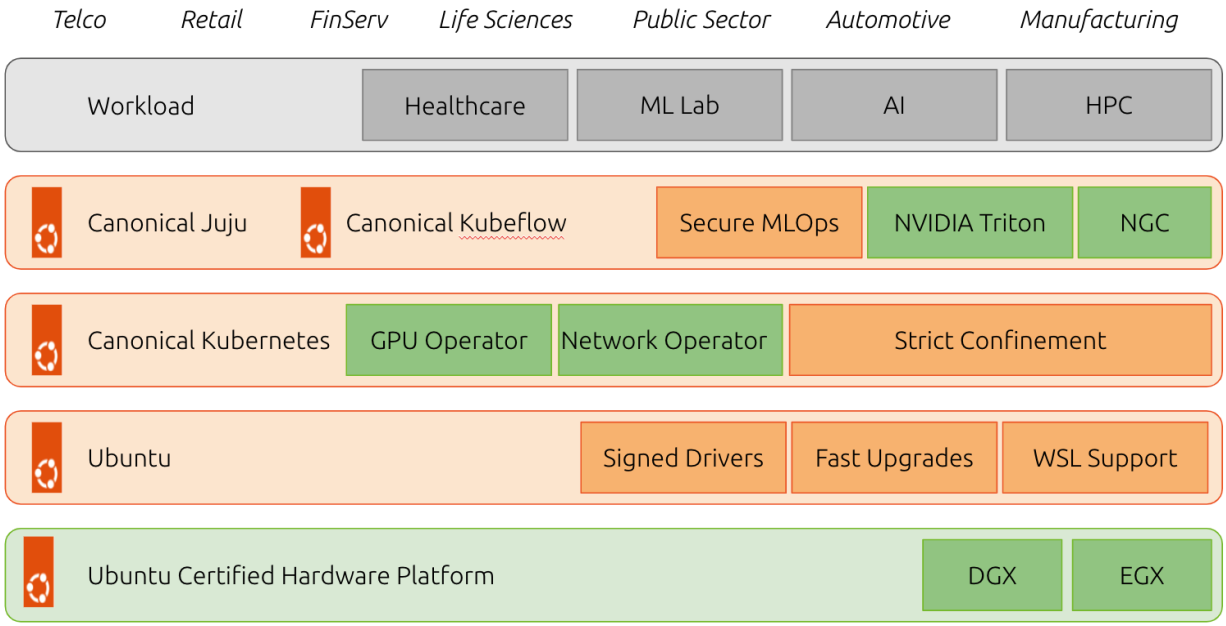
The reference architecture includes Ubuntu Server OS running on NVIDIA EGX hardware by Supermicro, MicroK8s, and Charmed Kubeflow to provide a comprehensive solution for developing, deploying, and managing AI workloads in edge computing environments.

NVIDIA EGX hardware by Supermicro forms 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 hardware by Supermicro 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.

Benefits



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

- Support across generations of EGX-ready systems by Supermicro 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 are getting integrated solutions that offer the highest performance and 'just-work' out of the box from Supermicro.
- Enterprise support, backed by NVIDIA. Deep engineering relationship means that Supermicro can often get bugs fixed with NVIDIA faster, and at times even before they are public.
- All this builds on Ubuntu's underlying security and LTS value proposition.
- 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 Canonicals' Observability stack.

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.

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 10 years and optional enterprise-grade phone and ticket support by Canonical.

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, allowing you to take advantage of the available GPU hardware.

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 for deploying the infrastructure and Kubernetes cluster - with the help of Juju.

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 a public or private cloud.

Canonical has developed and tested the Kubeflow charms to automate the delivery of its components.

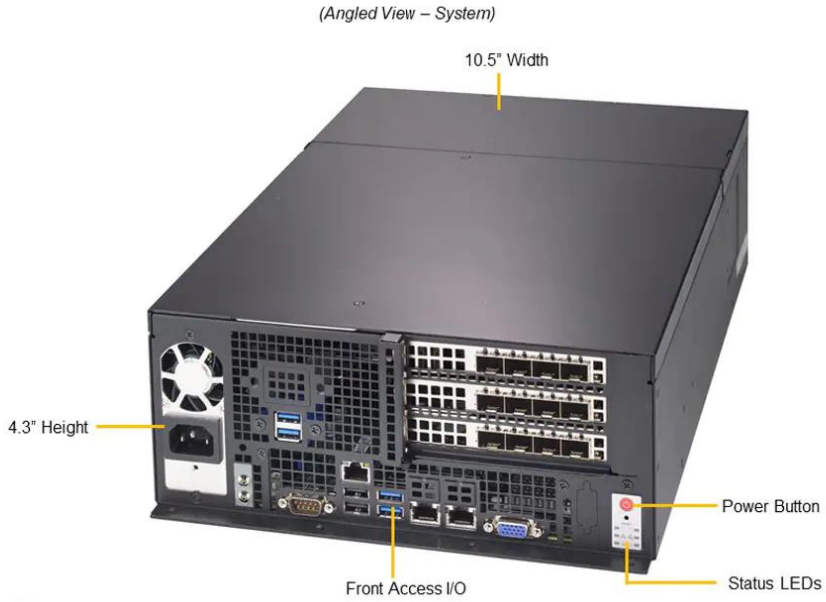
Software versions

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

Supermicro Specifications

The reference architecture is based on testing the solution on the EGX-ready system represented by Supermicro E403 with an NVIDIA A30 card.



Item	Description
System	E403-9D-4C-FN13TP
CPU	Intel(R) Xeon(R) D-2177NT CPU @ 1.90GHz
Core	16
Threads/Core	2
Memory	64 GB (16GB DDR4 1.2V 2666 ECC REG)
Storage	Kioxia XG6 512GB NVMe M.2
Networking	Intel® i350 (on-board) 1Gbps per port
GPU	Nvidia A30

Tutorial: Deploying an Object Detection Model

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

Update system

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

Install MicroK8s

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

Add the 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 hostpath-storage ingress gpu metallb:192.168.1.10-192.168.1.16
```

Note: Please replace the metallb IP with a static ranger of your servers LAN IP which can access from your client/browser, for example, Supermicro used 172.16.20.90-172.16.20.120 as the tested server IP is 172.16.20.49/24.

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

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

Add an 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
```

Note: To survive the system reboot, you can set the parameter below, otherwise you might find the system below response very slow.

```
echo "fs.inotify.max_user_watches=655360" | sudo tee -a  /etc/sysctl.d/99-kubeflow.conf
echo "fs.inotify.max_user_instances=1280" | sudo tee -a /etc/sysctl.d/99-kubeflow.conf
```

Check Juju status until all statuses become active

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

If the 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 the 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 the dashboard.

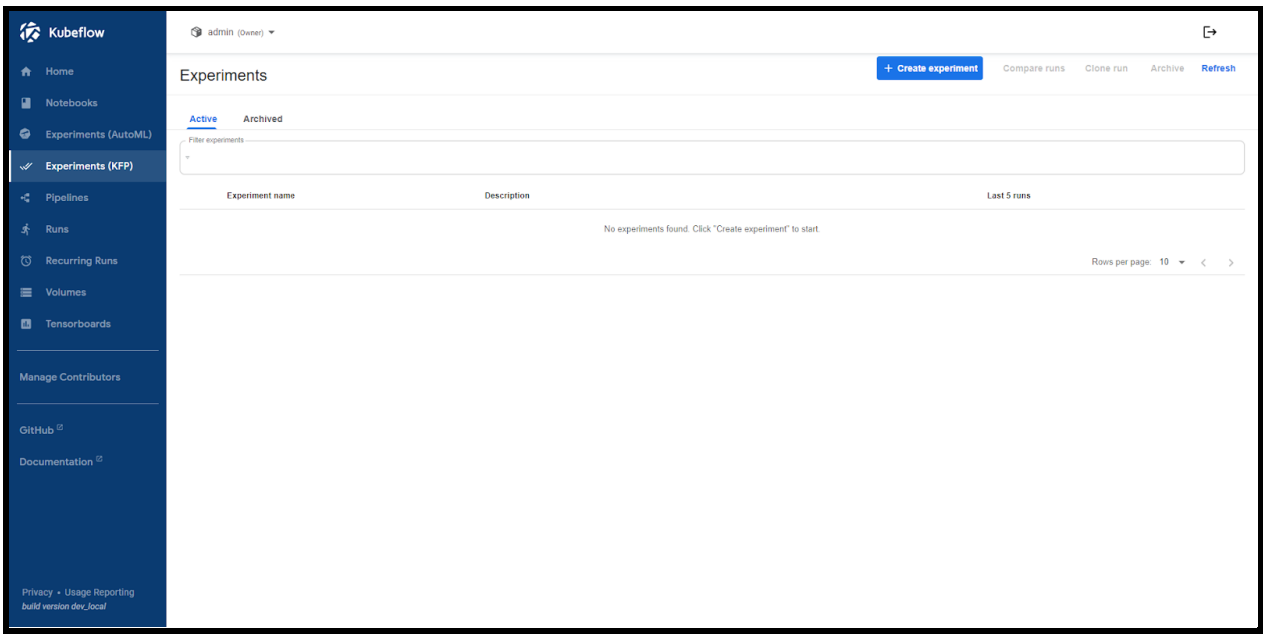
```
juju config dex-auth public-url=http://$IP.nip.io
juju config oidc-gatekeeper public-url=http://$IP.nip.io
juju config dex-auth static-username=admin
juju config dex-auth static-password=admin
```

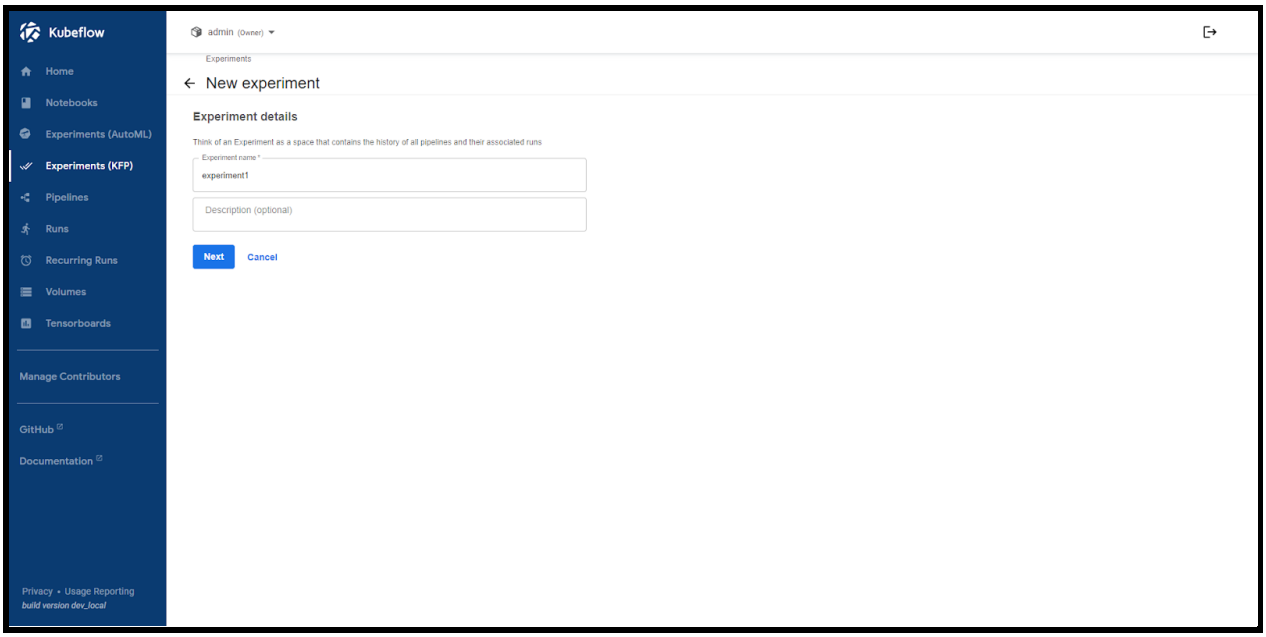
Login to the Charmed Kubeflow dashboard with a browser and accept default settings.

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

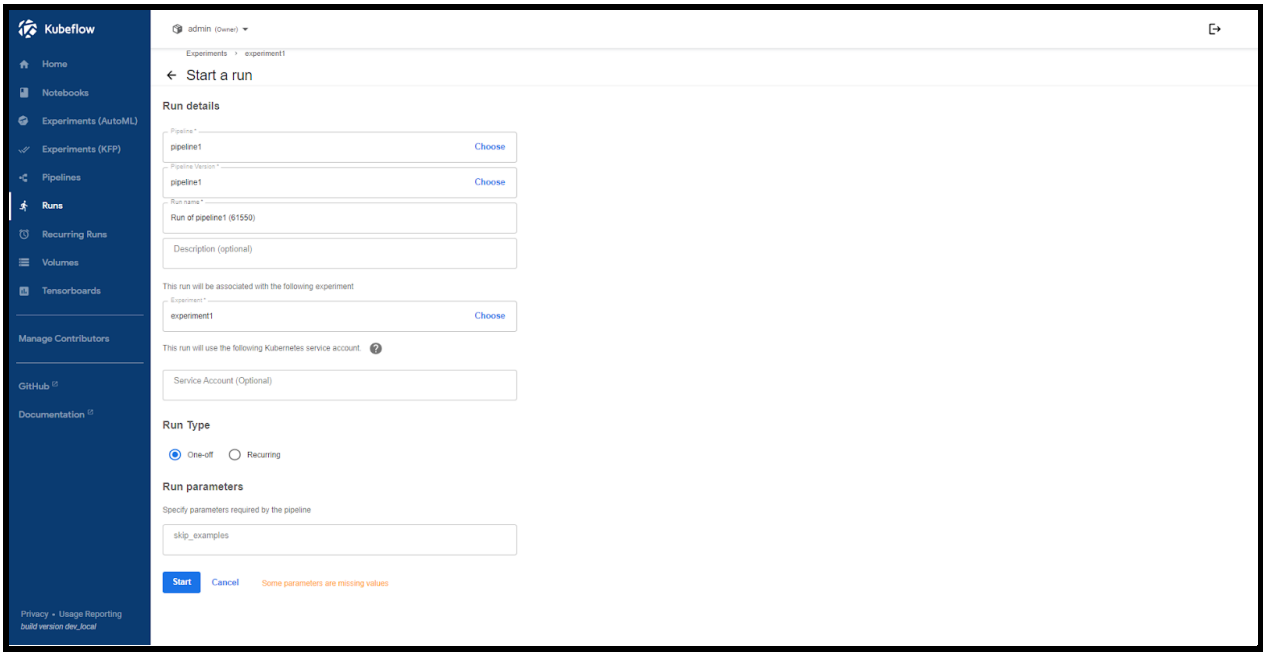
Create pipeline.yaml file on your local machine (refer to Appendix A.1)

Create experiment.

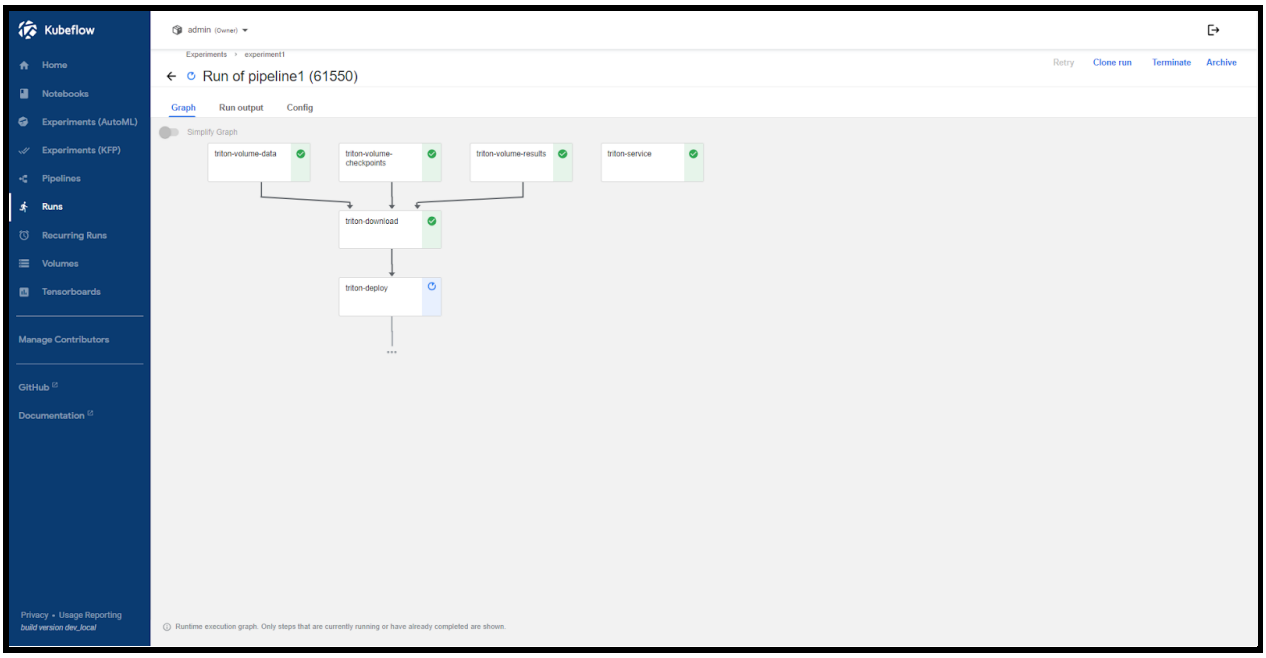




Create run and upload pipeline.yaml



Let the pipeline run and download the Triton container. It will eventually look like this.



Verify the Triton Inference server can be reached and has loaded the models.

```
IP=$(kubectl get service triton-kubeflow -n admin -o jsonpath='{.spec.clusterIP}')
curl http://$IP:8000/v2/models/densenet_onnx
```

Successful output will look like the following.

```
{"name":"densenet_onnx","versions":["1"],"platform":"onnxruntime_onnx","inputs":[{"name":"data_0","datatype":"FP32","
shape":[3,224,224]}],"outputs":[{"name":"fc6_1","datatype":"FP32","shape":[1000]}]}
```

Test image classification

```
sudo snap install docker
sudo docker run -it --rm nvcr.io/nvidia/tritonserver:23.05-py3-sdk /workspace/install/bin/image_client -u $IP:8000 -m
densenet_onnx -c 3 -s INCEPTION /workspace/images/mug.jpg
```

Successful output will look like the following.

```
Request 0, batch size 1
Image '/workspace/images/mug.jpg':
   15.349561 (504) = COFFEE MUG
   13.227463 (968) = CUP
   10.424892 (505) = COFFEEPOT
```

References:

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

Conclusion

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 Supermicro hardware, 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, Supermicro, and NVIDIA, organizations can enhance data analytics, optimize decision-making processes, and revolutionize customer experiences. 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. To learn more about Kubeflow on NVIDIA, check out https://ubuntu.com/engage/run-ai-at-scale

Learn more about Supermicro Edge Solutions at: www.supermicro.com/

Appendix

A.1 pipeline.yaml

```
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
```