Run AI at the Edge: MicroK8s with Charmed Kubeflow on Lenovo ThinkEdge Servers

Planning / Implementation

Overview

Canonical MicroK8s (pronounced "micro kates") 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's EGX 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. In Addition, the Lenovo ThinkEdge server line is designed to virtualize traditional IT applications as well as new transformative AI systems, providing the processing power, storage, accelerator, and networking technologies required for today’s edge workloads.

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.

Solution components

The solution architecture includes Canonical Ubuntu running on Lenovo ThinkEdge Servers, MicroK8s, and Charmed Kubeflow to provide a comprehensive solution for developing, deploying and managing AI workloads in edge computing environments, using the NVIDIA EGX platform.

The NVIDIA EGX platform 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.

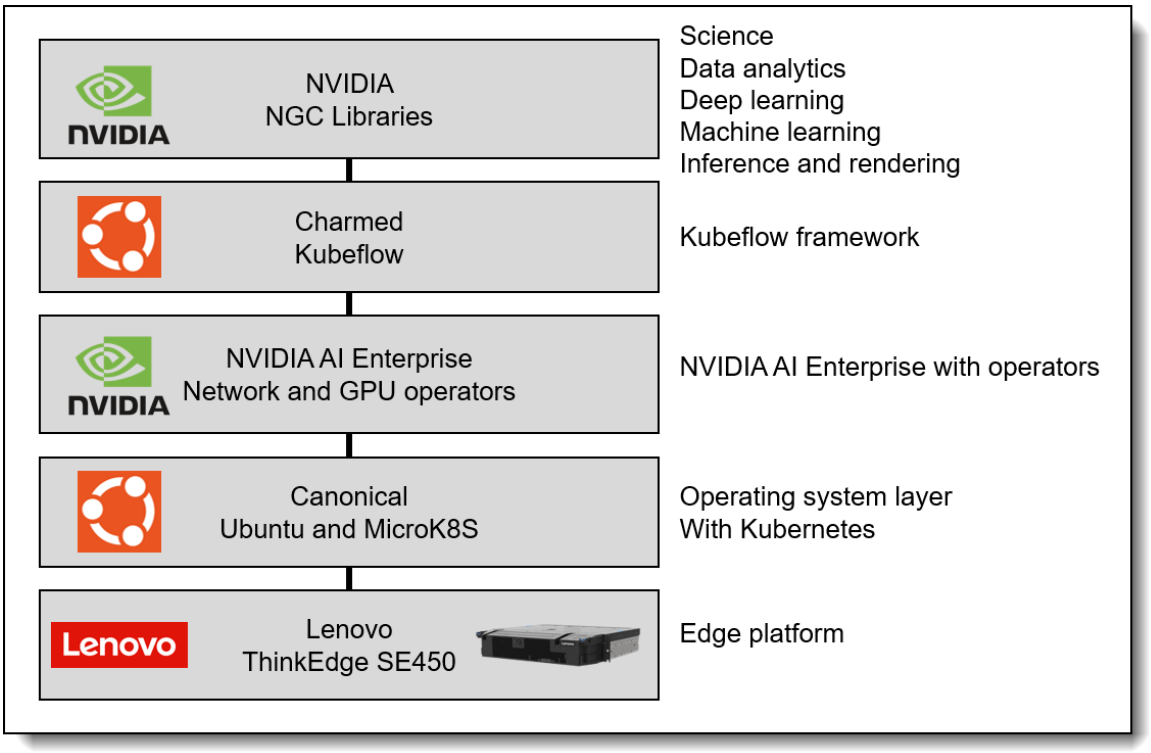

Figure 1. Solution components

By combining these components, this reference architecture enables organizations to leverage the power of Lenovo ThinkEdge Servers using 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.

Benefits

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

- Support for Lenovo ThinkEdge 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.

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.

The primary components of the solution are as follows:

- 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.

- 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.

- 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.

- 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.

Software versions

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

Table 1. Software versions

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

Lenovo hardware specifications

The reference architecture is based on testing the solution on the EGX-ready system represented by Lenovo ThinkEdge SE450.

Figure 2. Lenovo ThinkEdge SE450

The ThinkEdge SE450 is a purpose-built server that is significantly shorter than a traditional server, making it ideal for deployment in tight spaces. It can be mounted on a wall, placed vertically in a floor stand, or mounted in a rack.

The ThinkEdge SE450 puts increased processing power, storage and network closer to where data is generated, allowing actions resulting from the analysis of that data to take place more quickly.

Since these edge servers are typically deployed outside of secure data centers, they include technology that encrypts the data stored on the device if it is tampered with, only enabling authorized users to access it.

The server is equipped with a single Intel Xeon Gold processor, 128GB of RAM, a single 480GB drive, and two NVIDIA L40 GPUs.

NVIDIA software specifications

NVIDIA GPU Cloud

NVIDIA GPU Cloud (NGC) 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.

Triton Inference Server Software

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

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.

Tutorial: Deploying an object detection model

This section covers the steps for deploying the software components on the ThinkEdge SE450 Server. We assume that the user has experience on deploying, installing and setting up open-source frameworks.

1. Install Ubuntu 22.04 LTS on a machine with an NVIDIA GPU.
2. Update system

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

3. Install MicroK8s

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

4. 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
```

5. Log out and re-enter the session for the changes to take effect.
6. Enable MicroK8s add-ons for Charmed Kubeflow (replace IP addresses accordingly)

```
microk8s enable dns hostpath-storage ingress gpu metallb:192.168.1.10-1
92.168.1.16
```

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

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

8. Add alias for omitting microk8s when running commands

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

9. (Optional) Set forward IP address in CoreDNS:

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

10. Install Juju

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

11. Deploy Juju controller to MicroK8s

```
juju bootstrap microk8s
```

12. Add model for Kubeflow

```
juju add-model kubeflow
```

13. 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
```

14. Check Juju status until all statuses become active

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

15. 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/confi
g-changed; ./dispatch"
```

16. 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}')
```

17. 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
```

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

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

19. Create pipeline.yaml file on your local machine using the code listing in Appendix: pipeline.yaml
20. Click the **Create experiment** button

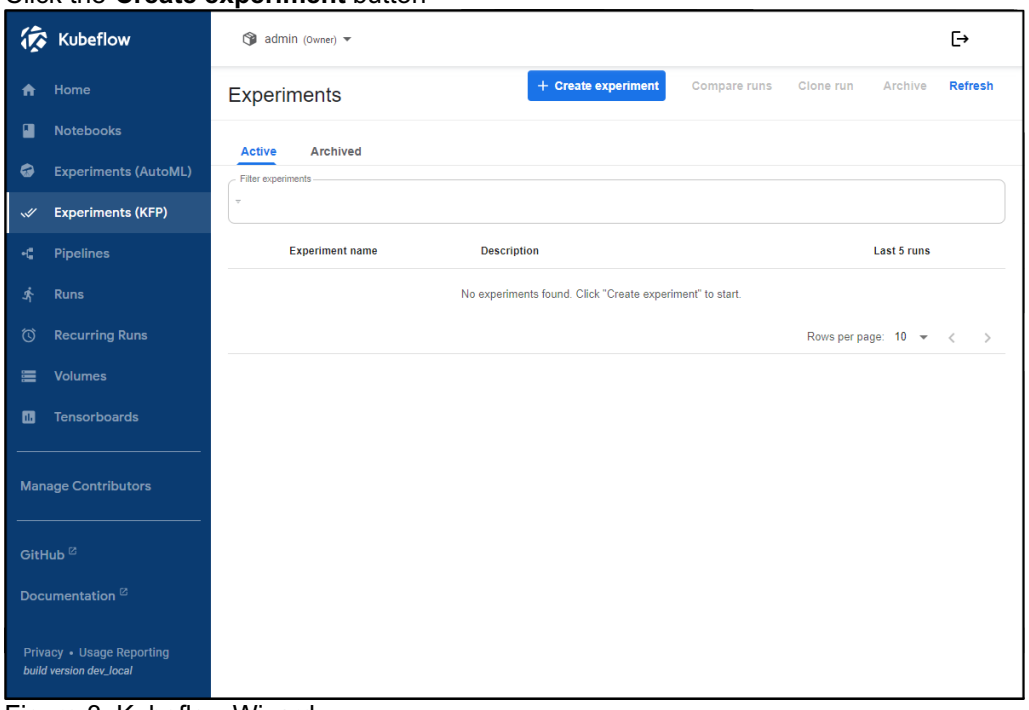

Figure 3. Kubeflow Wizard

21. Enter a name for the experiment and click Next.

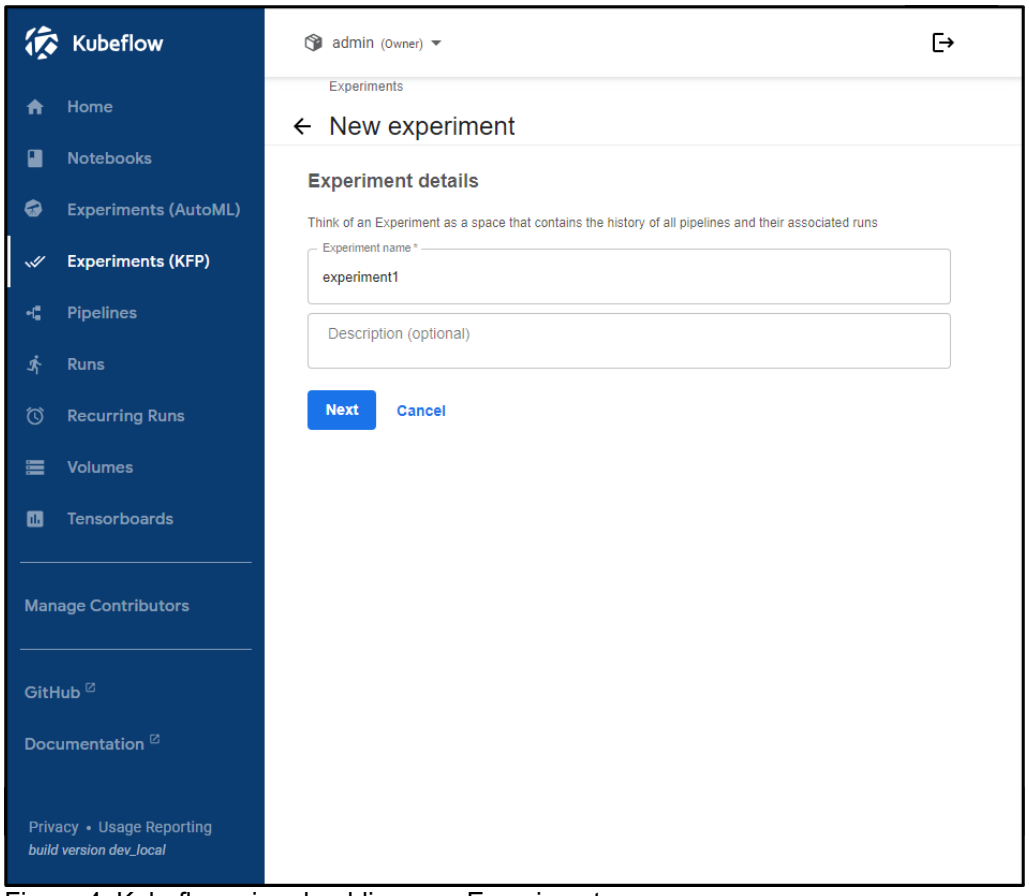

Figure 4. Kubeflow wizard, adding new Experiment

22. Enter the run details as shown below.

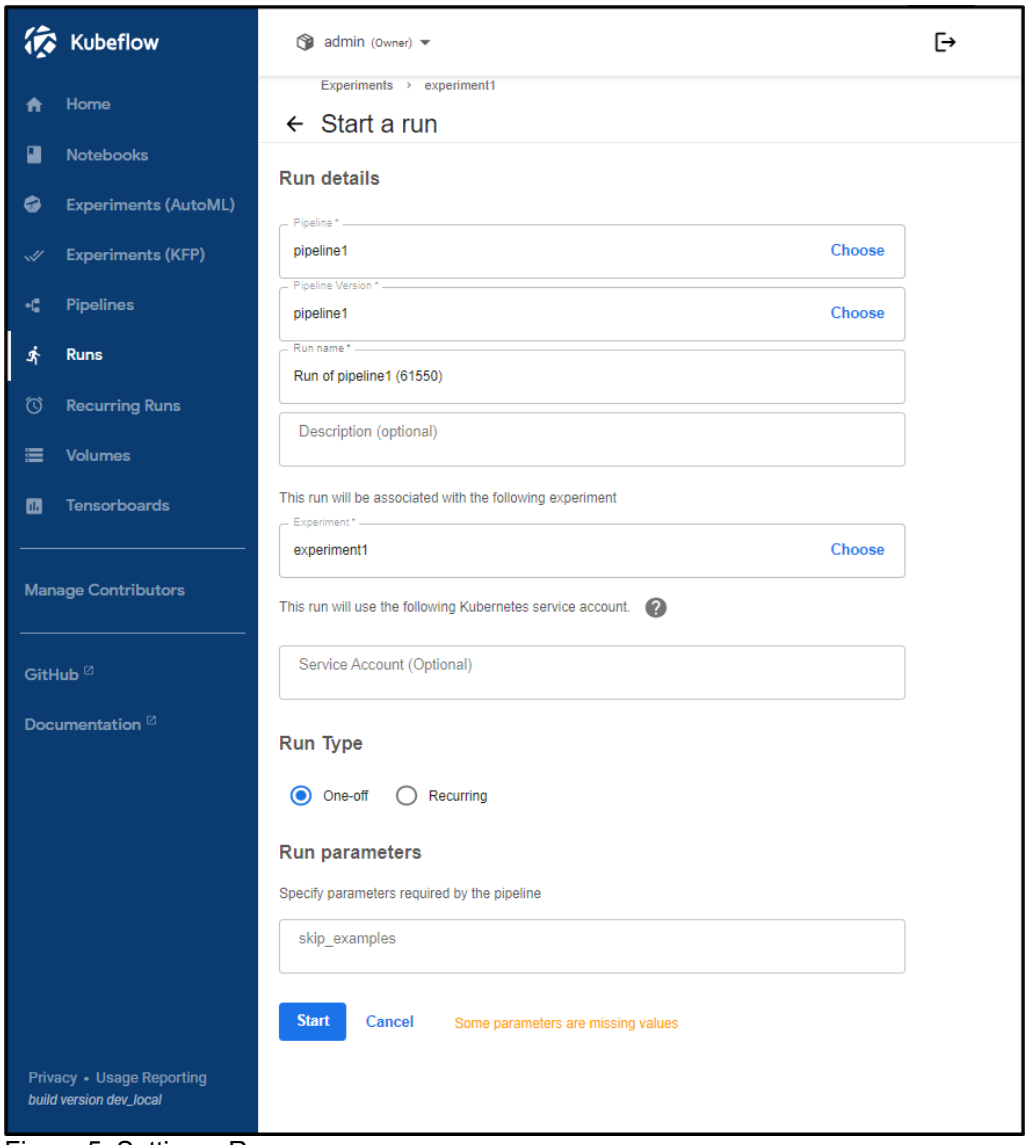

Figure 5. Setting a Run

23. Create run and upload pipeline.yaml

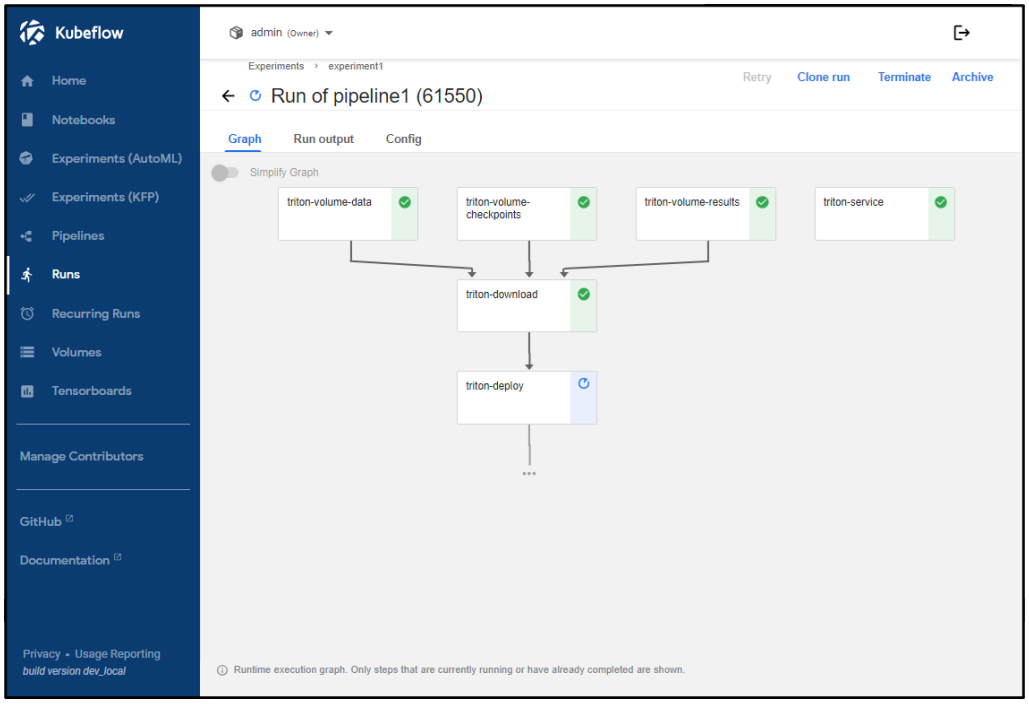

Figure 6. Modeling Pipeline

24. Let the pipeline run and download the Triton container. It will eventually look like this
25. Verify the Triton Inference server can be reached and has loaded the models

```
IP=$(kubectl get service triton-kubeflow -n admin -o jsonpath='{.spec.c
lusterIP}')
curl http://$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]}],"outpu
ts":[{"name":"fc6_1","datatype":"FP32","shape":[1000]}]}
```

26. Test the image classification

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

A 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
```

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 Lenovo 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, Lenovo 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

For more information, visit these pages:

- kubeflow-pipeline-deploy on Github:
  https://github.com/NVIDIA/deepops/tree/master/workloads/examples/k8s/kubeflow-pipeline-deploy
- Take your models to production with open source AI
  https://ubuntu.com/ai
- Kubernetes MicroK8s
  https://microk8s.io/
- What is Kubeflow
  https://ubuntu.com/ai/what-is-kubeflow
- Kubeflow on NVIDIA:
  https://ubuntu.com/engage/run-ai-at-scale

Appendix: pipeline.yaml

This is the contents of the file **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
```

Authors

David Ellison is the Chief Data Scientist for Lenovo ISG. Through Lenovo's US and European AI Discover Centers, he leads a team that uses cutting-edge AI techniques to deliver solutions for external customers while internally supporting the overall AI strategy for the Worldwide Infrastructure Solutions Group. Before joining Lenovo, he ran an international scientific analysis and equipment company and worked as a Data Scientist for the US Postal Service. Previous to that, he received a PhD in Biomedical Engineering from Johns Hopkins University. He has numerous publications in top tier journals including two in the Proceedings of the National Academy of the Sciences.

Carlos Huescas is the Worldwide Product Manager for NVIDIA software at Lenovo. He specializes in High Performance Computing and AI solutions. He has more than 15 years of experience as an IT architect and in product management positions across several high-tech companies.

Mircea Troaca is a AI Technical Engineer leading the benchmarking and reference architectures for Lenovo's AI Discover Lab. He consistently delivers #1 results in MLPerf, the leading AI Benchmark, and is also leading Lenovo and Intel's efforts in the TPCx-AI benchmark.

Related product families

Product families related to this document are the following:

- Artificial Intelligence
- Edge Servers
- ThinkEdge SE450 Edge Server

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service. Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
8001 Development Drive
Morrisville, NC 27560
U.S.A.
Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION ”AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary. Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk. Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This document, LP1863, was created or updated on January 17, 2024.

Send us your comments in one of the following ways:

- Use the online Contact us review form found at:
  https://lenovopress.lenovo.com/LP1863
- Send your comments in an e-mail to:
  comments@lenovopress.com

This document is available online at https://lenovopress.lenovo.com/LP1863.

Trademarks

Lenovo and the Lenovo logo are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. A current list of Lenovo trademarks is available on the Web at https://www.lenovo.com/us/en/legal/copytrade/.

The following terms are trademarks of Lenovo in the United States, other countries, or both:
Lenovo®
ThinkEdge®

The following terms are trademarks of other companies:

Intel® and Xeon® are trademarks of Intel Corporation or its subsidiaries.

Linux® is the trademark of Linus Torvalds in the U.S. and other countries.

Other company, product, or service names may be trademarks or service marks of others.