Use Azure Kubernetes Service to host GPU-based workloads

Azure Kubernetes Service (AKS)

This article describes how to efficiently run workloads that use GPU nodes on an Azure Kubernetes Service (AKS) cluster. Learn how to choose the right SKU, use GPU nodes to train machine learning models, and use GPU nodes to run inferences on AKS.

Scenarios

GPU workloads can be expensive to run. To avoid unnecessary cost, know when to deploy GPU-based nodes in your AKS clusters.

GPUs are purpose-built for graphics, AI and machine learning, and specialized tasks, which makes them ideal for compute-intensive workloads. CPUs effectively manage intricate logic and branching. GPUs are optimized for throughput. They can efficiently handle straightforward arithmetic and vector operations.

To determine when to use GPUs for AKS workloads, you must understand GPU optimization and compute intensity, but you should also consider other factors. To gain better insight into GPU usage for AKS workloads, consider the following workload examples that benefit from GPU nodes in an AKS cluster.

Data science and analytics

You can use GPUs to accelerate data preprocessing, feature engineering, and model training in data science workflows. To efficiently use GPUs, frameworks like RAPIDS and Dask GPU extend popular data-processing libraries, such as pandas and scikit-learn.

Open-source software (OSS)-accelerated SQL query engines and columnar databases like BlazingSQL and HeavyDB use GPUs to rapidly perform queries and analytics on large datasets.

Machine learning and deep learning

Popular machine learning and deep learning frameworks like TensorFlow and PyTorch benefit from GPUs because they can accelerate training and inference tasks.

Deep learning models have complex neural networks. Parallel processing on GPUs speeds up the model's computations. GPUs provide highly efficient matrix multiplication and convolutions, which are core operations in deep learning.

You can also use GPUs to accelerate tasks such as image classification, object detection, natural language processing, and speech recognition.

Computer vision and image processing

Computer vision tasks interpret visual data to extract meaningful information. These tasks are increasingly common in AI-powered applications, autonomous vehicles, medical imaging, surveillance systems, and augmented reality. GPUs use parallel processing, so they can efficiently handle large-scale image data and complex computations for tasks like object detection, image classification, and feature extraction.

Video processing and streaming

Videos are increasingly prevalent in business, so organizations need GPU-powered hardware rather than CPUs. Video-processing workloads, including transcoding, encoding, and streaming, are compute-intensive, especially if they have high-definition content or 4K content. GPUs provide an efficient platform that delivers high-performance, low-latency video experiences across diverse applications, like streaming sports events or corporate videos.

GPU-enabled agent nodes provide a rich customer experience in virtual desktop environments because they offload graphics-intensive tasks to the GPU. GPU-accelerated video encoding and decoding capabilities help improve real-time video streaming, video transcoding, and video analytics.

To accelerate computer vision tasks like object detection, object tracking, and image or video processing, you can use frameworks like OpenCV , OpenCL , NVIDIA CUDA , and NVIDIA cuDNN .

Gaming platforms and cloud gaming services rely on GPUs to deliver high-quality graphics and smooth gameplay experiences.

High-performance computing

High-performance computing (HPC) applications often require complex simulations, numerical analysis, and scientific computations. To quickly run these tasks, you can use GPUs to parallelize the workload across multiple cores. Examples of HPC applications that need massive parallel-processing power include scientific simulations, weather forecasting, computational fluid dynamics, and molecular modeling. GPUs are ideal for parallel computations and significantly accelerate HPC workloads. Scientific and research-driven endeavors benefit from GPUs.

To accelerate HPC applications, frameworks like NVIDIA CUDA , OpenCL , and OpenACC provide GPU-enabled APIs and libraries.

Genomic analysis and bioinformatics

Health and life sciences workloads, like genomic analysis and bioinformatics applications, are increasingly common. These workloads involve processing genetic data, such as DNA sequences and protein structures, and require complex algorithms for sequence alignment, variant calling, and genomic data mining. GPUs expedite genomic analysis workflows so that researchers can process data and uncover insights faster.

Consider cost implications before you deploy GPU nodes in AKS clusters. Understand GPU optimization for compute-intensive tasks like computer vision, video processing, HPC, and genomic analysis tasks. Consider these factors when you compare GPU resources versus CPU resources in AKS clusters.

Generative AI models

Large language models (LLMs), like OpenAI GPT , Meta Llama , Falcon , or Mistral , can take advantage of GPU parallel-processing capabilities. Use GPUs with these models to improve performance.

GPUs can speed up training and inference tasks, which involve complex computations and large amounts of data. GPUs have parallel-processing capabilities that divide the large computational tasks of a given model into smaller subtasks that run concurrently. This process delivers fast results and improves performance.

Language models often have complex neural networks with several layers and parameters, which can increase computational demand. GPUs accelerate key operations in language

processing, such as matrix multiplication and convolutions, which speeds up training and inference times.

GPUs provide sufficient memory capacity, bandwidth, and processing power to handle LLM-based applications that have conversational interfaces and text generation. For example, GPU enhancements provide fast response times for users that interact with chatbots and AI assistants.

Not all workloads benefit from GPU-enabled agent nodes, and in some cases, CPUs are sufficient. For example, workloads that are primarily input and output-bound or don't require heavy computation might not improve with GPUs.

Customer stories

Many Microsoft customers take advantage of GPU workloads to innovate for their customers. Consider the following examples:

- NBA players improve performance with AI on Azure AI infrastructure .
- An AI company called Mr. Turing uses AI and AKS to unlock and retain company information—and make it searchable .
- OriGen accelerates energy reservoir simulations by 1,000 times by using Azure AI infrastructure .
- Sensyne Health aids the National Health Service in the COVID-19 struggle by using Microsoft HPC and AI technologies .
- Constellation Clearsight augments electrical infrastructure inspection by using automated machine learning for images from Azure Machine Learning .

GPU workload deployment best practices

AKS provides various options to deploy GPU-enabled Linux and Windows node pools and workloads. To ensure the smooth operation of your GPU workload, follow these best practices.

Linux workload deployment

- Create a node pool with a supported GPU-enabled virtual machine (VM) and manually install the NVIDIA device plugin to deploy GPU-enabled Linux node pools. This method doesn't support updating an existing node pool to add GPUs.
- View the supported GPU-enabled VMs in Azure. We recommend that you use a minimum size of *Standard_NC6s_v3* for AKS node pools. AKS doesn't support the NVv4 series based on AMD GPUs.
- Understand the limitations when you use an Azure Linux GPU-enabled node pool. Automatic security patches aren't applied and the default behavior for the cluster is *unmanaged*.
- Use Kubernetes node selectors , node affinity , taints, and tolerations when you schedule workloads on your GPU-enabled node pools.

Windows workload deployment

- Create a node pool with a supported GPU-enabled VM to deploy GPU-enabled Windows node pools. AKS automatically installs the drivers and necessary NVIDIA components. This method doesn't support updating an existing node pool to add GPUs.

  When you select a supported GPU-enabled VM, AKS automatically installs the appropriate NVIDIA CUDA or GRID driver. Some workloads depend on a specific driver, which can affect your deployment. For NC-series and ND-series VM sizes, AKS installs the CUDA driver. For NV-series VM sizes, AKS installs the GRID driver.
- View the supported GPU-enabled VMs in Azure. We recommend that you use a minimum size of *Standard_NC6s_v3* for AKS node pools. AKS doesn't support the NVv4 series based on AMD GPUs.
- Understand the limitations when you use a Windows node pool. Kubernetes versions 1.28 and below don't support Windows GPUs.
- Use Kubernetes node selectors , node affinity , taints, and tolerations when you schedule workloads on your GPU-enabled node pools.

> **Note**

Windows GPU is a preview feature. You need to **register the WindowsGPUPreview feature flag**.

NVIDIA GPU Operator

The NVIDIA GPU Operator is a tool that you can use to efficiently deploy and manage GPU resources within Kubernetes clusters. You can use the operator to automate the installation, configuration, and maintenance of software components. This approach ensures the optimal use of NVIDIA GPUs for demanding workloads, such as AI and machine learning workloads.

The NVIDIA GPU Operator automatically manages all NVIDIA software components that you require to deploy GPUs, like the NVIDIA device plugin for Kubernetes and the NVIDIA container runtime. The operator automatically installs the driver. For more information, see NVIDIA overview .

If you want increased control and flexibility for advanced GPU workloads, you can use the NVIDIA GPU Operator with your GPU-enabled nodes on AKS. The NVIDIA GPU Operator doesn't support Windows GPUs.

Consider the following best practices:

- Use the NVIDIA GPU Operator to do advanced GPU configurations, such as driver version selection and time-slicing.
- Skip the automatic driver installation before you use the GPU operator.
- Set the minimum count to 1 when you use the GPU operator with the cluster autoscaler.

> **Note**
>
> Microsoft doesn't support or manage the maintenance and compatibility of the NVIDIA drivers as part of the node image deployment when you use the GPU operator.

GPU workload deployment for LLMs

The Kubernetes AI toolchain operator (KAITO) is a Kubernetes operator that simplifies how you run open-source LLMs, like Falcon and Llama2, on your Kubernetes cluster. You can deploy KAITO on your AKS cluster as a managed add-on for AKS. KAITO uses Karpenter to automatically provision and deploy GPU nodes based on a specification in the workspace custom resource definition of your chosen model. KAITO creates the inference server as an endpoint for your LLM and reduces overall onboarding time so that you can do machine learning operations rather than infrastructure setup and maintenance.

To improve machine learning operations, KAITO provides the following capabilities:

- **Container image management**: Use container images to manage LLMs. KAITO provides an HTTP server so that you can use a supported model library to perform inference calls.
- **GPU hardware configuration**: KAITO provides preset configurations that are automatically applied based on model requirements. You don't have to manually tune deployment parameters to fit GPU hardware.
- **Automatic GPU node provisioning**: KAITO automatically provisions GPU nodes based on model requirements and recommends lower-cost GPU VM sizes to configure distributed inferencing.
- **Integration with Microsoft Artifact Registry**: If your LLM license allows, KAITO can host model images in the public Artifact Registry. This method simplifies access to and deployment of supported models. For open-source models with MIT or Apache2 licenses that the KAITO repository doesn't support, you can submit a request for new model onboarding.

For more information about KAITO, see the following resources:

- Explore the KAITO open-source project
- Deploy an AI model on AKS with KAITO
- Fine tune your language models with open-source KAITO
- Deploy KAITO on AKS by using Terraform

Workload and cluster scaling

For AI and machine learning scenarios, you must differentiate between training workloads and inferencing with pretrained models. To build and train your machine learning model, consider using GPU compute that's designed for deep learning and parallelizing AI

computations. Training often requires gradual scaling and the distribution of large quantities of data across GPUs to achieve high accuracy with data parallelism.

Model sharding is a common advanced technique that you can use to divide stages of model training. You can assign GPUs to distinct tasks and maximize their use. GPUs can scale up and scale out HPC workloads, such as NV-series or ND-series VMs on Azure. This capability helps maintain high resource usage and reduce user intervention for machine learning training processes that are lengthy and expensive.

Alternatively, you can use pretrained, open-source AI and machine learning models for inferencing only. Get started with popular models like Llama, Falcon, or Mistral as a faster and more cost-effective option compared to building and training an LLM from scratch. For more information, see Language models on AKS.

When you use pretrained models for inferencing, you might experience dynamic and fluctuating resource usage, depending on the volume of data that you process. When you run live data through your chosen model, spikes in traffic sometimes occur depending on the model size and requirements. You must maintain an acceptable, low level of latency throughout the inferencing process. To effectively use your GPUs for high performance and low latency, you can conduct distributed inference with the models that KAITO supports. This approach expands your compute options to include lower GPU-count SKUs that have one or two GPUs each, provides high availability across Azure regions, and reduces maintenance costs.

GPU workload cost management

GPUs can increase cost. Properly monitor workloads to help understand what drives GPU costs, and identify optimization opportunities. To increase cost visibility, you can use the AKS cost analysis add-on.

The following scenarios benefit from cost visibility.

GPU-enabled VM size cost

Select the right GPU-enabled VM size to optimize the cost of running GPUs. Daily costs can vary depending on the VM size that you choose. A100 GPUs are costly. You should avoid them unless your workload has specific requirements. AKS cost analysis shows the daily cost for each of your VMs and breaks down the associated costs of each workload that runs

on the GPU-enabled VM. Use this data to evaluate whether you have a proper VM size or if you need a more cost-effective option.

Idle cost

After you create a GPU-enabled node pool, you incur costs on the Azure resource even if you don't run a GPU workload. Idle costs represent the cost of available resource capacity that workloads don't use. This cost adds up quickly if you have several unused nodes. To avoid high idle costs, only create node pools when you run your workload, and use methods such as the cluster stop feature when you don't run your workload. AKS cost analysis shows idle costs for each of your nodes.

Overprovisioning and underuse

Overprovisioning is when you allocate more resources than necessary for a pod. Overprovisioning leads to resource waste and underuse. Excess resources remain reserved for the node even if you don't use them. To reduce overprovisioning, use the vertical pod autoscaler to set accurate requests and limits based on previous usage patterns.

Underuse can occur when your workloads don't use GPUs fully. Consider advanced GPU sharing and partitioning techniques. Rather than using multiple nodes, you might use a single node with partitioning to maximize GPU usage. These techniques can help you allocate the appropriate amount of GPU acceleration for each workload, which can enhance usage and lower the operational costs of deployment.

Linux GPU workload deployments on AKS support multiple-instance GPUs. Use this feature to partition a NVIDIA A100 GPU into up to seven independent instances. Each instance has its own memory and stream multiprocessor.

NVIDIA supports other partitioning techniques, such as time-slicing and the multi-process service implementation. To manually apply these configurations, use the NVIDIA GPU Operator.

For advanced scenarios, you can improve resource bin packing on your nodes. You can set scheduler configurations, and run a second scheduler. Configure and maintain a secondary scheduler to use workload placement strategies that differ from the default AKS scheduler. For more information, see Configure multiple schedulers on Kubernetes .

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Ayobami Ayodeji | Senior Program Manager

Other contributors:

- Steve Buchanan | Principal Program Manager
- Sachi Desai | Product Manager
- Ally Ford | Product Manager 2
- Paolo Salvatori | Principal Service Engineer
- Erin Schaffer | Content Developer 2

To see nonpublic LinkedIn profiles, sign in to LinkedIn.

Next steps

- Bring your own AI models to intelligent apps on AKS with KAITO
- Deploy an AI model on AKS with KAITO
- Deploy an application that uses OpenAI on AKS
- Deploy KAITO on AKS by using Terraform
- Deploy the Azure Machine Learning extension on AKS or Azure Arc-enabled Kubernetes clusters
- Learn about the model catalog and collections on Azure
- Use GPUs for Windows node pools on AKS
- Use GPUs on AKS

Related resource

- Baseline architecture for an AKS cluster

Feedback

Was this page helpful? Yes No