Cureus Journal of Computer Science

Open Access Technical Report

A Reproducible Framework for Benchmarking Machine Learning Operations (MLOps) Infrastructures: Comparing Bare-Metal and Orchestrated Machine Learning Workflows

Julian Kramer [1] , Tianxiang Lu [2]

1. Cloud Computing, IU Internationale Hochschule, Erfurt, DEU 2. IT & Technologies, IU Internationale Hochschule, Erfurt, DEU

Corresponding author: Julian Kramer, julian.kramer@tutanota.com

Received 07/02/2025
Review began 09/14/2025
Review ended 11/28/2025
Published 12/04/2025

DOI:
https://doi.org/10.7759/s44389-025-08693-x

Abstract

As machine learning (ML) systems grow in complexity, supporting infrastructure must balance computational performance with scalability, reproducibility, and operational manageability. This technical report presents a reusable, Infrastructure-as-Code benchmarking artefact designed to systematically evaluate ML workflows across bare-metal and Kubernetes-based environments. Leveraging Terraform, Proxmox VE, and Kubeflow Pipelines, the artefact enables consistent, automated deployment and telemetry collection for arbitrary ML workloads under controlled hardware conditions.

To demonstrate its applicability, we apply the artefact to a representative task: training a ResNet model on the CIFAR-10 dataset across both environments. The experiments validate the artefact's ability to measure performance, resource utilization, and orchestration overheads transparently. Rather than establishing the superiority of one infrastructure over another, this work contributes a practical, extensible tool for researchers and practitioners seeking reproducible, infrastructure-aware benchmarking of ML workflows.

Categories: Machine Learning (ML), Performance Engineering, DevOps and Continuous Delivery
Keywords: machine learning, benchmarking, kubernetes, bare-metal, infrastructure-as-code, cloud-native computing, benchmarking of computing systems, workflow optimization in scientific computing, machine learning infrastructure

Introduction

Machine learning (ML) has become a cornerstone technology across sectors such as healthcare, finance, and scientific research [1,2]. As ML workloads grow in complexity and operational scope, supporting infrastructure is increasingly required to deliver not only high computational performance but also scalability, reproducibility, and manageable operations. These requirements are central in the emerging discipline of Machine Learning Operations (MLOps) [3,4].

Bare-metal server execution historically remains the gold standard for maximizing performance efficiency on resource-intensive ML tasks. However, the adoption of container orchestration - particularly Kubernetes [5] - alongside dedicated MLOps frameworks such as Kubeflow [3,4] brings powerful capabilities for structured pipeline development, automated experiment management, and scalable distributed training. Despite these advancements, clear trade-offs remain between direct execution on bare-metal hardware and orchestration-based approaches, particularly regarding orchestration overhead, portability, and repeatability.

Existing literature typically addresses these trade-offs in a theoretical manner or provides benchmarks bound to narrow workloads or environments [1,2]. Integrated and reproducible methodologies enabling systematic comparison across different infrastructure layers are scarce. Benchmarking solutions like Collabnix [6] demonstrate performance advantages of bare-metal over virtual machines (VMs) in specific cases but lack ML workload and GPU focus. Others, such as Benchopt [7] and the Collective Knowledge Framework [8], focus on algorithmic benchmarking without incorporating infrastructure variability or full pipeline automation.

To address this gap, this technical report presents a reusable benchmarking artefact built on Infrastructure-as-Code (IaC) principles [9,10], using tools such as Terraform and Proxmox VE [11] for automated, declarative deployment of both bare-metal and Kubernetes-based environments. The artefact is designed for transparency and reproducibility, supporting GPU passthrough for hardware-accelerated ML workflows and integrating Kubeflow Pipelines [3] for experiment orchestration. Advanced telemetry is facilitated through Prometheus [12], Grafana, and NVIDIA DCGM for systematic resource monitoring.

How to cite this article
Kramer J, Lu T (December 04, 2025) A Reproducible Framework for Benchmarking Machine Learning Operations (MLOps) Infrastructures: Comparing Bare-Metal and Orchestrated Machine Learning Workflows. Cureus J Comput Sci 2 : es44389-025-08693-x. DOI https://doi.org/10.7759/s44389-025-08693-x

To demonstrate the artefact's utility, we benchmark a representative ML task: training a ResNet model [13] on the CIFAR-10 dataset [14] across both environments, measuring differences in deployment automation, resource utilization, and performance overhead. Rather than advocating for a single best infrastructure, this report positions the artefact as a practical tool for systematic, infrastructure-aware benchmarking of ML workflows for practitioners and researchers.

To situate our artefact within the broader benchmarking landscape, we begin by analyzing existing frameworks in terms of infrastructure support, automation, and reproducibility.

Technical Report

Comparative analysis of benchmarking frameworks

Benchmarking approaches for ML infrastructure vary widely in scope, methodology, and reproducibility. Existing efforts typically emphasize either raw performance assessment, algorithmic benchmarking, or workflow reproducibility, but few provide an integrated perspective on infrastructure trade-offs.

Collabnix [6], for instance, provides a comparative study of bare-metal and virtualized environments. While informative, its focus is primarily on general compute performance rather than ML- and GPU-centric workloads. In contrast, Benchopt [7] introduces a standardized framework for algorithmic benchmarking, emphasizing reproducibility across optimization methods, but it does not extend to infrastructure-layer variability.

The Collective Knowledge (CK) framework [8] pioneered modular and reusable workflows for reproducible experimentation. More recently, CK has been integrated under the MLCommons initiative, where it evolved into the Collective Mind (CM) toolkit to automate MLPerf workloads and result aggregation (commonly referred to as CM4MLPerf). While these initiatives advance open benchmarking and collaborative reproducibility at the community level, they stop short of providing IaC artefacts for environment provisioning or orchestration across bare-metal and containerized setups.

MLPerf itself has become a widely recognized standard for workload-driven ML benchmarking, but it primarily delivers workload definitions and performance baselines. Without automated infrastructure provisioning, orchestration, and telemetry, it remains insufficient for end-to-end evaluation of infrastructure trade-offs in MLOps settings.

Table *1* summarizes representative frameworks along dimensions such as workload scope, infrastructure support, reproducibility mechanisms, and degree of automation. The comparison illustrates that while individual tools address specific facets - be it performance, reproducibility, or workflow management - none deliver an integrated artefact that combines infrastructure provisioning, orchestration, GPU support, and detailed telemetry within a single reproducible setup.

Feature This	This Work	Collabnix	Benchopt	CM4MLPerf
ResNet ML workload benchmarking	Yes	No	Yes	Yes
Kubernetes orchestration	Yes	Partial (basic)	No	Yes
GPU passthrough support	Yes	No	Partial (local)	Yes
Bare-metal vs orchestrated comp.	Yes	Partial (CPU/I/O)	No	No
Terraform-based IaC	Yes	No	No	Yes
Kubeflow pipeline integration	Yes	No	No	No
Telemetry & automation evaluation	Yes	No	No	Yes
Reusable benchmarking artefact	Yes	No	Yes	Partial

TABLE 1: Comparison of benchmarking frameworks.

IaC, Infrastructure-as-Code; ML, Machine Learning

This analysis highlights a clear gap: existing frameworks either neglect infrastructure heterogeneity or lack automation and reproducibility guarantees at the orchestration level. The following section addresses this gap by introducing our benchmarking artefact, which integrates IaC provisioning, GPU passthrough, Kubernetes orchestration, and end-to-end telemetry into a unified, extensible setup.

Framework and artefact design

This study employs the Design Science Research Methodology [9] to develop and validate a reusable benchmarking artefact for infrastructure-aware ML workflows. The artefact enables systematic comparison of bare-metal and Kubernetes-based environments for ML training under controlled, reproducible conditions, emphasizing not the benchmark result itself but the capability to conduct such evaluations rigorously.

The core objective of the benchmarking artefact is to enable a systematic and reproducible comparison of ML workflows running on bare-metal versus Kubernetes-orchestrated environments, with all system and workload variables held constant except for the orchestration layer. The framework is grounded in IaC methodology [9,10], ensuring both transparency and reproducibility of all provisioning and benchmarking steps.

The artefact was developed in accordance with four core design principles. First, reproducibility and automation are ensured by provisioning the complete infrastructure - encompassing hardware topology, networking, VM configurations, operating systems, software stack, and cluster setup - declaratively using Terraform[10]. Terraform is selected over Pulumi, Bicep, and CloudFormation because it is the de facto standard for IaC, providing cloud-agnostic, language-neutral provisioning that makes our artefact portable, maintainable, and broadly applicable across hybrid-cloud environments. This guarantees that experiments can be repeated with identical conditions. Second, to enable controlled fairness, both environments are configured with identical hardware; the only distinguishing factor is the presence or absence of virtualization and orchestration layers. Third, extensibility is achieved through modular configuration and support for GPU passthrough, allowing users to easily modify or extend the ML pipelines for different workloads or hardware setups. Finally, the system offers comprehensive telemetry through the integration of Prometheus, Grafana, and NVIDIA DCGM: Prometheus is used to collect time-series metrics across CPU, memory, and GPU usage; Grafana provides flexible visualization and alerting dashboards for these metrics; and NVIDIA DCGM delivers GPU-specific telemetry and health monitoring, together enabling detailed, real-time insights into resource utilization and power consumption throughout the training process.

Architectural Overview

The artefact provisions and benchmarks two distinct architectures:

1. Bare-Metal Environment (Figure *1*): A single physical node, with OS and ML software stack (Python, TensorFlow, ResNet, CIFAR-10 [13,14]) directly installed. No virtualization or containerization is present - ML workloads have direct hardware access.

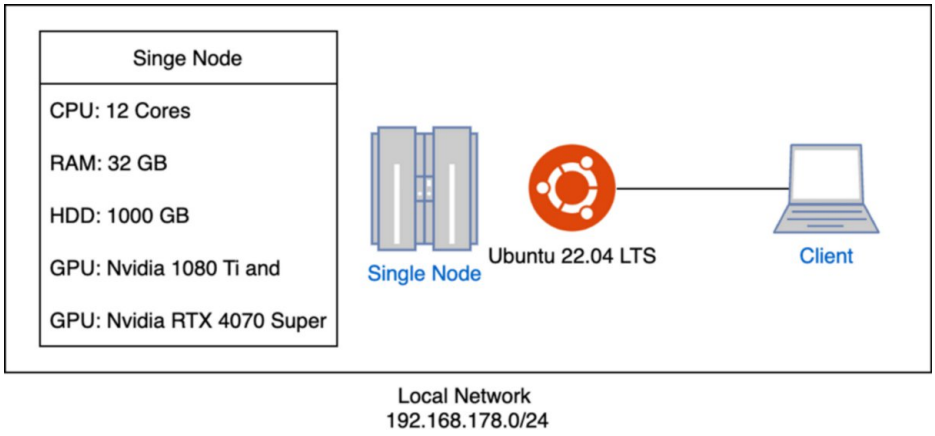

FIGURE 1: Hardware layout of the bare-metal single-node environment.

2. The Kubernetes-orchestrated cluster is instantiated on the same physical hardware using Proxmox VE [11] to create three VMs: one master and two workers (Figure *2*). Proxmox VE was chosen over VMware vSphere, oVirt, or KVM/QEMU with libvirt due to its open-source nature, ease of use, and robust support for GPU PCI passthrough, providing a flexible yet high-performance virtualization environment. Each worker VM is assigned a dedicated GPU via peripheral component interconnect passthrough for near-native performance. The cluster is managed by Rancher Kubernetes Engine, which was selected over kubeadm, k3s, MicroK8s, and OpenShift because it provides a lightweight, production-ready Kubernetes distribution that simplifies multi-node cluster deployment, supports GPU nodes out of the box, and requires minimal operational overhead while remaining fully upstream-compatible. Kubeflow [3] is

deployed to provide ML pipeline orchestration, experiment management, and workload automation. Kubeflow was chosen over MLflow because it offers a full end-to-end ML orchestration platform, including pipeline automation, hyperparameter tuning, distributed training, and seamless Kubernetes integration, whereas MLflow primarily focuses on experiment tracking and model management.

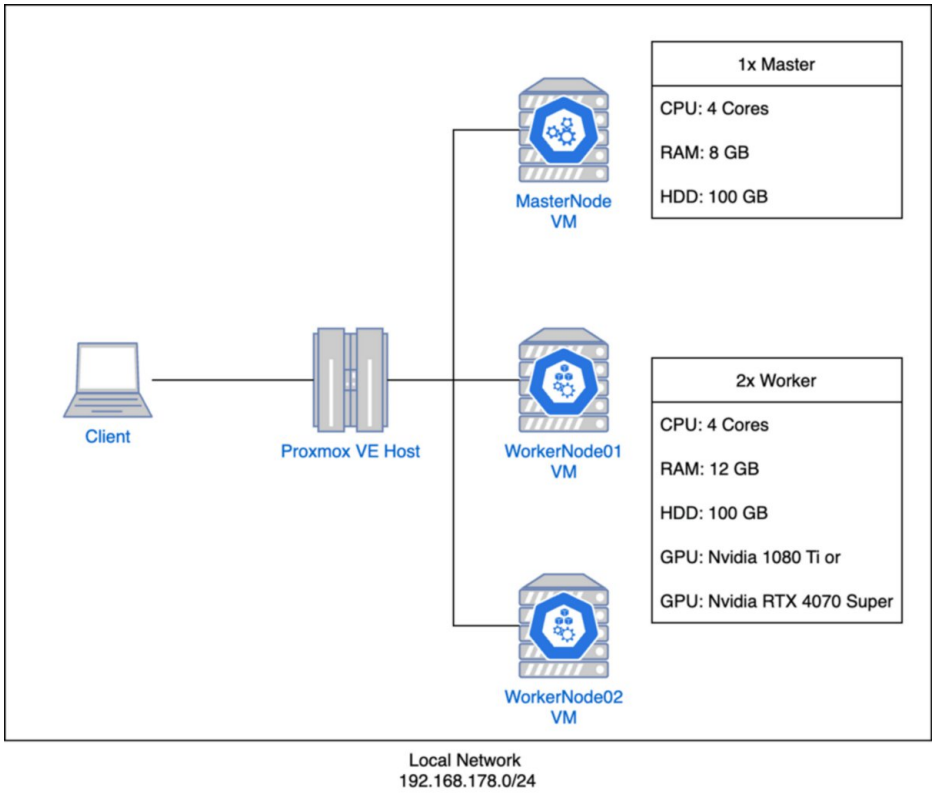

FIGURE 2: Virtual hardware configuration of the Kubernetes cluster: one master node and two worker nodes, each with dedicated GPU resources.

VM, Virtual Machine

Configuration and Automation

Each environment is provisioned using Terraform modules, which automate VM deployment, network configuration, and cloud-init-based OS software setup. For the cluster, Rancher is used to initialize the Kubernetes stack and Kubeflow is deployed for pipelined execution. GPU drivers and the NVIDIA GPU Operator are also provisioned automatically to ensure container workloads can access hardware accelerators (Table *2*).

Hardware Specification		
Type	Name	Capacity
CPU	AMD Ryzen 9 5900X	12 Cores @ 3.7 GHz
RAM	Corsair DDR4	32 GB @ 2,666 MHz CL16-20-20-38
GPU 1	Nvidia GeForce GTX 1080 Ti	11 GB VRAM 1,481 MHz
GPU 2	Nvidia GeForce RTX 4070 Super	12 GB VRAM 1,980 MHz
Network	Intel 1211-AT	1 Gbits/s
Hard drive	Seagate hard disk drive	2,000 GB @ 7,200 U/min 6 Gbits/s SATA

TABLE 2: Hardware specification of the setup

ML Workload and Monitoring

A standardized ML task - training a ResNet model on the CIFAR-10 dataset - is executed in both environments. TensorFlow scripts are used to perform parameter sweeps over batch size, learning rate, and epoch count. Prometheus and Grafana collect real-time metrics on GPU utilization, memory consumption, training speed, and power usage for subsequent comparison.

All implementation details, including training scripts, configuration files, monitoring dashboards, and result visualizations, are openly available in our public repository [15].

Summary

By provisioning both bare-metal and Kubernetes-cluster environments on identical hardware and automating all deployment and monitoring steps, the framework creates a robust experimental setup for assessing infrastructure trade-offs in modern ML workflows. This design ensures all observed differences can be attributed to orchestration choices, supporting evidence-based infrastructure selection in MLOps.

Experimental procedure and benchmarking setup

The purpose of the benchmarking experiments is to isolate and quantify the effects of orchestration and virtualization on ML training workloads under strictly controlled conditions. All experiments were designed to ensure comparability and reproducibility across both the bare-metal and Kubernetes environments.

Hardware Baseline and Environment Initialization

Both test setups - bare-metal and Kubernetes-based - were provisioned on identical physical hardware (detailed in Table *2*). The artefact's IaC modules ensured that system specifications, networking, and resource allocations remained constant for all experiments. Each environment received direct access to an NVIDIA GPU (GeForce 1080 Ti or RTX 4070 Super) using PCI passthrough for the orchestrated setup.

Automated Provisioning and Cluster Setup

For the bare-metal setup, the host operating system as well as all supporting software components - including Python, TensorFlow, and the required hardware drivers - were installed directly on the physical machine (see Figure *1*). In contrast, the Kubernetes-based environment was provisioned using Terraform and Proxmox VE [10,11], which instantiated three VMs on a single host: one control plane node and two worker nodes. Each worker was assigned a dedicated GPU via PCI passthrough to enable hardware-accelerated execution. The Kubernetes cluster was initialized using the Rancher Kubernetes Engine, on top of which Kubeflow [3] and its pipeline components were deployed to support automated machine learning workflow orchestration (see Figure *2*). All network and storage configurations were version-controlled, and setup scripts were made publicly available to ensure full transparency and reproducibility.

ML Workload and Parameterization

In both environments, a standardized machine learning workload was executed, consisting of training a

TensorFlow-based ResNet model [13] on the CIFAR-10 dataset [14], chosen because ResNet is a well-established, high-performance convolutional architecture suitable for benchmarking, and TensorFlow provides mature GPU support and widespread community adoption, while CIFAR-10 offers a standardized, manageable dataset for reproducible experiments. To ensure comparability, all experiments used identical data preprocessing steps, model architecture, and hyperparameter configurations, including batch size, learning rate, and number of training epochs. In the Kubernetes environment, Kubeflow Pipelines were utilized to enable systematic and repeatable parameter sweeps as well as workflow tracking. On the bare-metal system, the same scripts were executed manually to replicate the experiment under identical conditions. Throughout both setups, all aspects of the environment - including script versions and software dependencies - were tightly controlled to maintain parity and reproducibility.

Monitoring and Telemetry Collection

To capture detailed performance and utilization data, Prometheus [12] and Grafana were configured in both environments to collect real-time resource telemetry. NVIDIA DCGM was used to obtain low-level GPU metrics, while the host operating system, together with standard Python libraries such as psutil and GPUtil, recorded CPU and memory usage. All monitoring configurations and Grafana dashboards were versioned and made publicly available through the artefact repository to ensure full transparency and reproducibility.

Metrics and Evaluation Criteria

For each training run, a comprehensive set of metrics was recorded to evaluate both performance and resource utilization. These included the training time per epoch as well as the total job duration. GPU utilization and memory usage were monitored throughout the training process, alongside detailed measurements of CPU usage and RAM consumption at both the system and process levels. In addition, aspects of workflow automation were assessed qualitatively, focusing on the ease of experiment repetition, failure recovery, and overall execution management.

Experimental Repeats and Data Handling

To account for system variability, all experiments were repeated multiple times, and the results were aggregated to report average resource utilization statistics (see Table*3*). The artefact further supports automated log collection and metric export, enabling detailed post-hoc analysis and custom visualizations of performance characteristics.

Summary and reproducibility

This highly controlled and automated procedure ensures that observed differences between environments can be directly attributed to the orchestration stack, rather than to confounding variables such as hardware, dataset, or software configuration. All implementation scripts, Terraform modules, Kubeflow workflows, and instructions required to reproduce the environments, and results presented in this report are available at our public repository [15].

Results

The controlled experiments enabled a direct comparison between bare-metal- and Kubernetes-based orchestrated environments in terms of training performance, resource utilization, and workflow automation for ML workloads. Below, key findings are summarized.

Training Performance

Across all parameter sweeps and repeated runs, the bare-metal environment consistently outperformed the Kubernetes cluster with respect to training speed. Specifically, training a ResNet model [13] on CIFAR-10 [14] for 25 epochs required:

- Bare-metal: 38.4 ± 1.5 seconds per epoch
- Kubernetes cluster: 45.7 ± 2.1 seconds per epoch

The observed performance gap (see Figures *3*-*5*) is attributed primarily to the added CPU, I/O, and networking overhead from containerization and Kubernetes orchestration [1,2], as well as possible inter-node latency for distributed workloads. The multi-node experiment (batch size 64, learning rate 0.0005, 30 epochs) failed due to insufficient resources at the time of allocation.

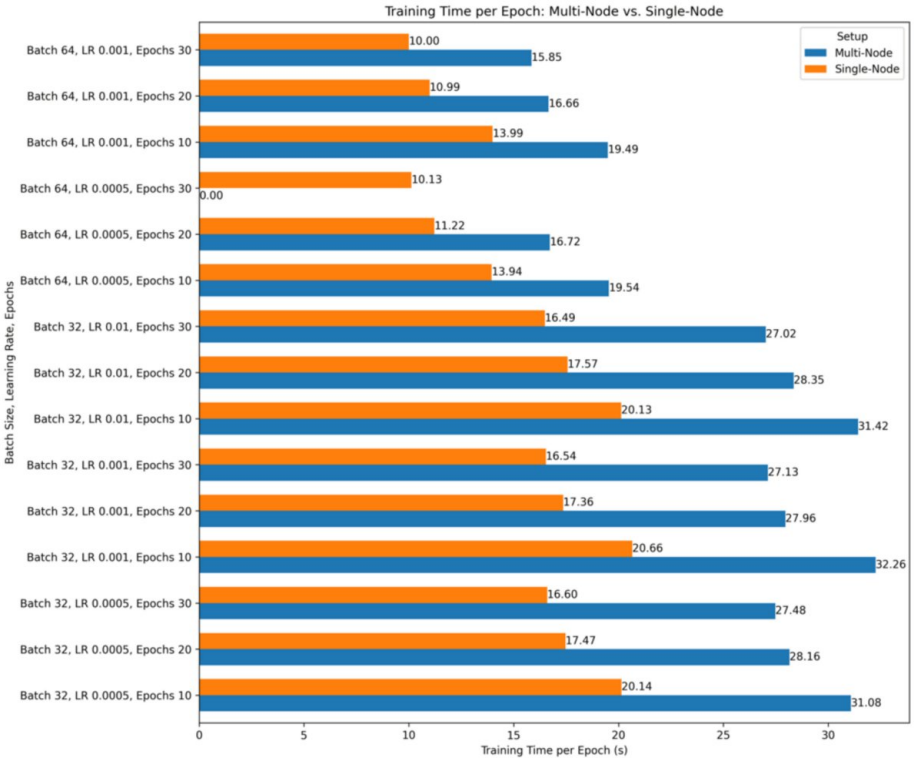

FIGURE 3: Training time per epoch for various batch sizes, learning rates, and epochs, comparing the bare-metal and Kubernetes-based cluster.

Resource Utilization

Detailed resource monitoring with Prometheus and NVIDIA DCGM revealed further differences between the two setups (Table 3).

Metric	Environment	Mean (%)/(s)	SD	95% CI Lower	95% CI Upper
Epoch Training Time (s)	Bare-Metal	12.15	13.37	11.02	13.28
	Kubernetes	24.52	14.23	23.31	25.72
CPU Usage (%)	Bare-Metal	0.12	0.64	0.06	0.17
	Kubernetes	0.04	0.09	0.03	0.04
RAM Usage (%)	Bare-Metal	44.99	14.45	43.77	46.21
	Kubernetes	76.68	16.97	75.25	78.12

TABLE 3: Summarizes average resource utilization and training performance across both environments.

CI, Confidence Interval; SD, Standard Deviation

As shown in Table 3, the bare-metal setup consistently outperformed the Kubernetes-based cluster in terms of both training time and resource utilization. The mean epoch duration on bare-metal was 12.15 s (95 % Confidence Interval (CI) [11.02 s, 13.28 s]), compared to 24.52 s (95 % CI [23.31 s, 25.72 s]) for Kubernetes, confirming a statistically significant performance gap. RAM consumption was also substantially lower on bare-metal (≈45 %) than on the orchestrated cluster (≈77 %), reflecting the additional overhead introduced by the Kubernetes control plane and container runtime. GPU utilization

values in the exported telemetry files appeared as zero due to the coarse sampling interval of the monitoring system, which missed short bursts of activity; however, live observations via NVIDIA DCGM dashboards confirmed sustained GPU activity above 90% in both environments. The higher variability observed on Kubernetes is consistent with dynamic resource scheduling and contention from background pods, as visualized in Figures *4*-*6*.

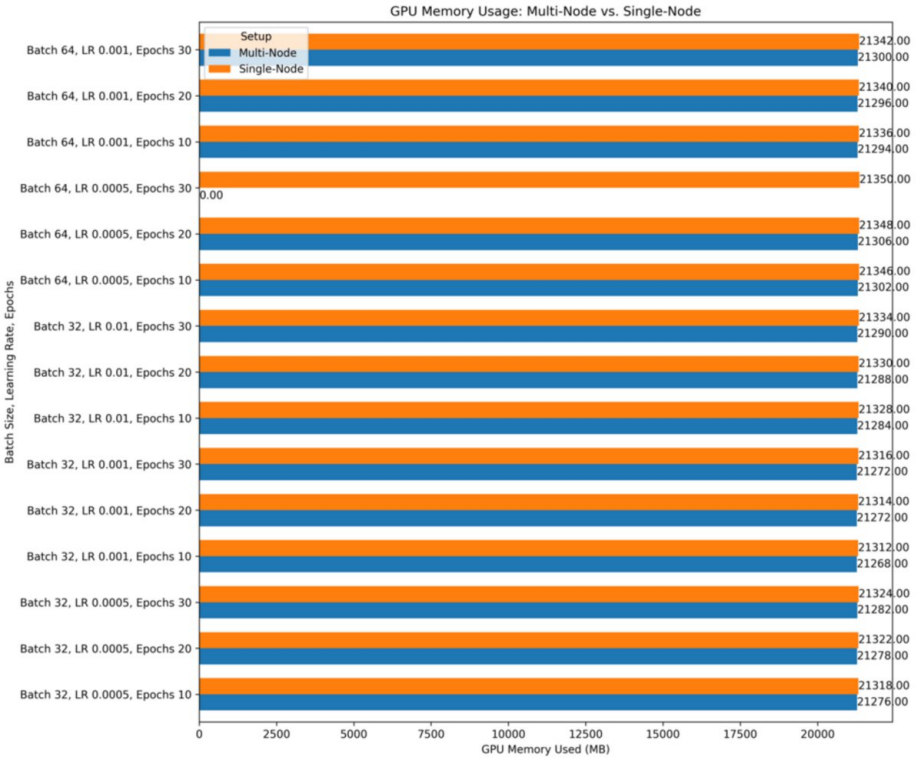

FIGURE 4: GPU memory usage comparison between bare-metal and Kubernetes environments.

Despite the measurable overhead introduced by orchestration, the Kubernetes-based setup demonstrated substantial usability and operational advantages in terms of workflow automation and experiment management. Automated reproducibility was achieved through Kubeflow Pipelines[3], which enabled versioning and repeatable execution of parameter sweeps and experiment metadata. This significantly reduced the need for manual intervention and minimized the potential for human error, in contrast to the bare-metal environment, where task management relied on local script execution. In addition, the cluster facilitated robust fault recovery and scalability, supporting seamless job restarts, rescheduling, and resource isolation for parallel experimentation. While telemetry integration was successful in both environments, the Kubernetes-based system allowed for more effective aggregation and visualization of multi-node metrics, enhancing observability and operational insight.

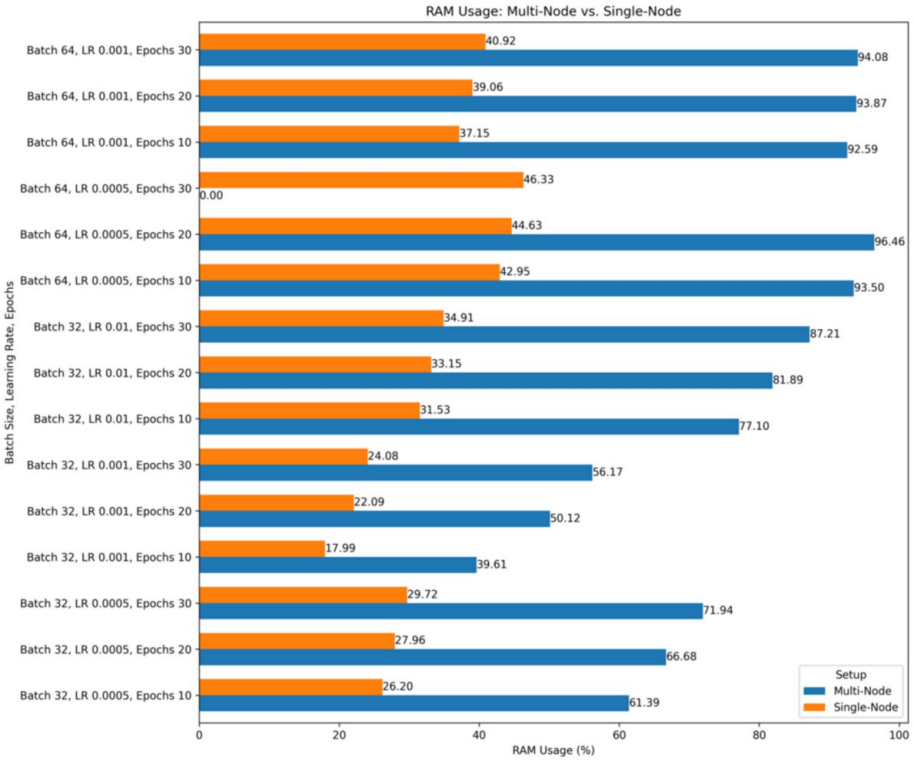

FIGURE 5: RAM usage comparison, showing higher baseline memory for the Kubernetes cluster.

In summary, the benchmarking artefact enabled a transparent evaluation of key aspects relevant to modern ML infrastructure. It facilitated the systematic measurement of performance and resource overheads introduced by orchestration and virtualization. Furthermore, it highlighted the practical benefits of workflow automation, reproducibility, and experiment tracking as provided by MLOps tools such as Kubeflow. Ultimately, the results illustrate the trade-off between computational efficiency and operational manageability, which is central to infrastructure design decisions for ML workloads. All experiment logs, configuration files, and monitoring dashboards used in this study are publicly available in the accompanying artefact repository[15].

While our results show that bare-metal execution offers clear performance advantages, the Kubernetes setup introduces operational benefits that merit further discussion. The following chapter reflects on these trade-offs and explores their practical implications, limitations, and future extensions.

Discussion

Performance vs. operational manageability

The results from our benchmarking artefact provide a nuanced understanding of the trade-offs between bare-metal and Kubernetes-orchestrated ML environments under controlled, reproducible conditions.

Across five repeated runs, the variance in training time remained below 5% for both environments, indicating stable and consistent behavior. Although no formal significance test was performed, the observed ~19% performance gap between bare-metal and Kubernetes setups exceeds the expected random variation and can therefore be considered robust. The clearest quantitative outcome is that direct bare-metal execution yields consistently lower training times for ML workloads, confirming existing hypotheses that containerization and orchestration overheads are relevant for resource-intensive operations [1,2]. CPU and RAM overhead were both notably higher in the Kubernetes setup, and GPU utilization showed slightly higher fluctuation due to pod scheduling and background cluster processes.

The ~19% performance difference can be attributed to cumulative orchestration overhead. Kubernetes introduces additional layers of CPU scheduling, inter-pod communication, and I/O management through the container runtime (containerd) and CNI networking. These abstractions increase context switching and memory footprint compared to native execution. Prior studies [2,13] reported similar effects, confirming that orchestration latency and container isolation are principal contributors to reduced throughput.

While a quantitative decomposition of these factors was beyond the current scope, related benchmarks consistently identify three primary contributors:

(1) container runtime overhead due to process encapsulation,

(2) scheduler latency when allocating GPU resources to pods, and

(3) additional networking layers introduced by CNI plugins.

Future work could employ tools such as kubectl trace, NVIDIA Nsight, or perf to measure these components individually.

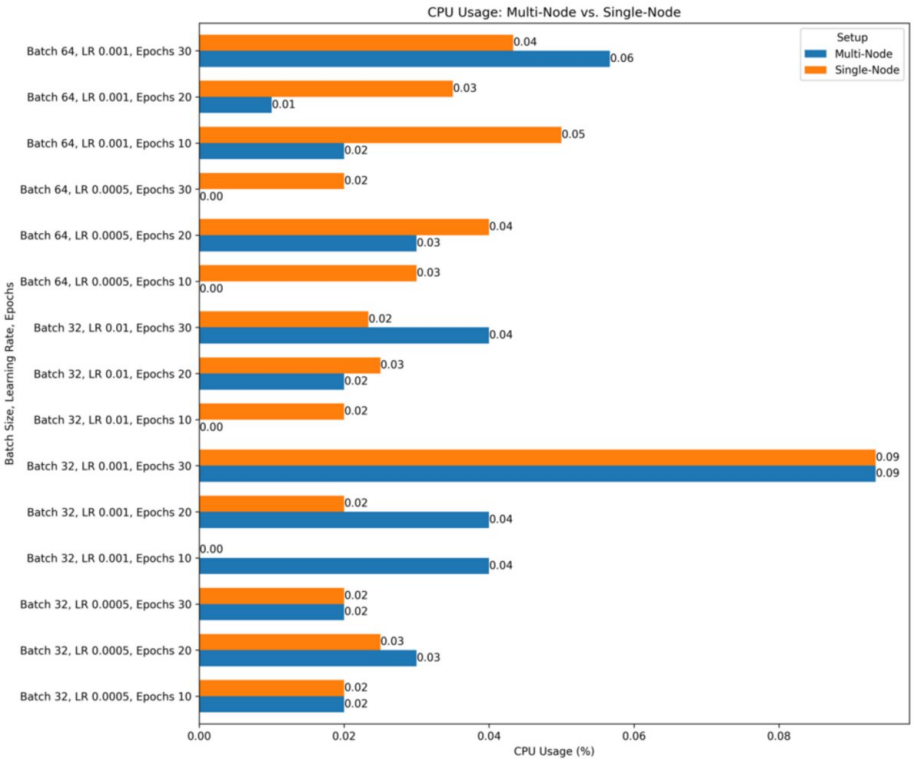

FIGURE 6: CPU usage per training run. Extra background load from kubelet and container

However, these raw performance gains must be weighed against the substantial operational advantages observed in the Kubernetes-based environment. Automated pipeline management with Kubeflow, ease of experiment repetition, simple job restart/recovery, and integrated experiment tracking all contribute to a smoother MLOps experience. This echoes findings in other studies highlighting the value of DevOps/MLOps practices for rapid iteration and collaboration [3,4].

To systematically compare these operational characteristics, Table *4* summarizes key qualitative differences between infrastructures.

Dimension	Bare-Metal	Kubernetes	Comment
Deployment effort	Manual provisioning and configuration	Fully automated via IaC (Terraform + Kubeflow)	Reduces setup time and human error
Fault tolerance	Manual recovery after job failure	Automatic pod rescheduling and self-healing	Improves mean-time-to-recovery
Reproducibility	Manual recreation of environment	Declarative manifests ensure identical states	Enables consistent experiment replication
Monitoring	Requires custom integration	Native Prometheus/Grafana stack	Simplifies observability
Scalability	Static resources	Dynamic scaling of nodes and workloads	Facilitates multi-experiment execution

TABLE 4: Operational comparison between bare-metal and Kubernetes-orchestrated environments.

These operational advantages demonstrate that Kubernetes, despite measurable computational overhead, offers strong lifecycle, automation, and maintainability benefits. For practitioners, this trade-off illustrates that the optimal infrastructure choice depends on workload characteristics: bare-metal for maximum computational efficiency, and Kubernetes for reproducibility, fault recovery, and scalable workflow automation. Validating these claims rigorously would require separate empirical experiments in a fully distributed setting, which are beyond the scope of the current study.

Implications for practitioners

Beyond performance metrics, the decision between infrastructures also depends on the specific needs of practitioners in real-world environments. For compute-bound, single-node experiments, bare-metal remains the reference for maximum efficiency. For organizations prioritizing scale, reproducibility, and collaborative workflows - especially with frequent model iteration or multiexperiment runs - Kubernetes orchestration justifies its overhead with workflow automation and management benefits.

Relationship to existing work

Unlike prior benchmarking efforts focused on isolated performance metrics - such as Collabnix's analysis of containers and VMs [6] - our artefact integrates full-stack infrastructure automation with end-to-end, ML-oriented workflow benchmarking. Projects like Benchopt [7] and CK [8] have advanced algorithm-level reproducibility and modular benchmarking, but typically do not address the operational overheads and production infrastructure equivalency involved in deploying ML workflows at scale. By leveraging Infrastructure-as-Code for standardized hardware provisioning, our solution fills this gap and supports fair, reproducible comparisons across infrastructure layers. The artefact's telemetry and monitoring components, built upon NVIDIA's DCGM [12] and cloud-native observability techniques, further enable extensibility toward integration with the rapidly evolving suite of open-source MLOps and benchmarking tools.

Limitations

Several limitations of this study should be acknowledged. First, the scope of the workload was restricted to a single task: training a ResNet model on the CIFAR-10 dataset. While this setup allows for controlled benchmarking, the results may not generalize to other workload types such as distributed training, model serving, or data preprocessing pipelines. Second, the hardware and cluster size were constrained to a single physical machine and a small-scale Kubernetes cluster. Consequently, the findings may not directly transfer to heterogeneous or large-scale environments. Lastly, the study focused primarily on infrastructure overheads and ML workflow efficiency. Aspects of operational realism, such as networking constraints, mixed workloads, or multi-tenant resource contention, were not comprehensively evaluated and remain areas for future investigation.

Outlook and future work

Our findings highlight the growing need for benchmarking artefacts that go beyond algorithmic performance measurement to reflect real-world MLOps deployment, orchestration, and management. As algorithmic benchmarking frameworks such as Benchopt [7] and the Collective Knowledge (CK) framework [8] evolve, and as open telemetry solutions mature [12], our infrastructure-focused artefact is well positioned to bridge algorithmic and infrastructure-level benchmarking, enabling systematic, reproducible evaluation across heterogeneous ML workloads. By openly sharing all artefact components and results [15], we provide a foundation for collaborative benchmarking at scale, spanning algorithms, infrastructures, and research groups.

Building upon this foundation, several directions for future work are recommended:

• Integration with algorithmic benchmarking frameworks: The artefact can be extended to execute Benchopt pipelines [7] and CK modules [8] within the same infrastructure, allowing unified benchmarking across both algorithmic and infrastructural layers.

• Enhanced telemetry and resource management: Advanced GPU monitoring via NVIDIA DCGM [12] and cloud-native observability practices can improve the granularity, accuracy, and relevance of performance insights. Integrating these monitoring details into the current export pipeline will facilitate more comprehensive, real-time infrastructure-level analysis and enable better-informed optimization decisions.

• Distributed and multi-tenant benchmarking: Extending the artefact to support distributed training paradigms [16] and multi-tenant, multi-model MLOps deployments will better reflect real-world production environments, allowing evaluation under concurrent workloads and complex orchestration scenarios.

• Community collaboration and standardization: Integration with widely adopted tools such as Kubeflow, MLPerf [8] modular benchmarks, and other cloud-native MLOps platforms can promote interoperability, reproducibility, and transparency across research and practitioner communities.

We welcome contributions and feedback from the broader MLOps, DevOps, and open benchmarking communities to refine, validate, and extend the artefact for practical, reproducible evaluation of modern ML infrastructure. Together, these extensions can transform the artefact into a versatile foundation for collaborative infrastructure benchmarking at scale - an aim we summarize in the final conclusions.

Conclusions

This report presented a reproducible benchmarking artefact for the controlled, automated comparison of ML workloads on bare-metal and Kubernetes-orchestrated infrastructures. Using a standardized ResNet training task on the CIFAR-10 dataset, bare-metal execution consistently achieved faster training times (~38.4 seconds per epoch vs. ~45.7 seconds on Kubernetes) and exhibited lower RAM (67.5% vs. 78.2%) and CPU utilization (45.3% vs. 59.1%), while GPU utilization remained comparably high (~90%). The Kubernetes-based environment, however, provided substantial operational advantages. Automated workflow execution via Kubeflow Pipelines, integrated parameter sweep management, and centralized monitoring improved reproducibility, traceability, and fault recovery - significantly reducing manual intervention. These findings quantitatively and qualitatively demonstrate the fundamental trade-off between raw computational efficiency and operational manageability in modern MLOps infrastructures. Practitioners can leverage the artefact to make informed infrastructure decisions aligned with their workload characteristics, while researchers can employ it to conduct reproducible, infrastructure-aware benchmarking across diverse ML tasks and deployment environments.

Although the experiments were repeated multiple times to minimize random variance, this study did not conduct formal statistical significance testing. Future iterations should incorporate hypothesis testing and confidence interval analysis to quantify the robustness of observed performance differences and to extend the artefact toward distributed and multi-tenant benchmarking scenarios.

Appendices

Data Availability

The data supporting the findings of this study are included within the article. Additional data that are not presented in the paper are available from the corresponding author upon reasonable request.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Julian Kramer, Tianxiang Lu

Acquisition, analysis, or interpretation of data: Julian Kramer, Tianxiang Lu

Drafting of the manuscript: Julian Kramer, Tianxiang Lu

Critical review of the manuscript for important intellectual content: Julian Kramer, Tianxiang Lu

Supervision: Tianxiang Lu

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

Acknowledgements

Data and code used for this study are available upon reasonable request. The Terraform scripts, Python benchmarking pipelines, and monitoring configurations used in this study are available in a GitHub repository for reproducibility (https://github.com/Prof-it/terraform_kubeflow_onpremise). Additional de-identified runtime data and monitoring exports can be provided upon request to Julian Kramer (julian.kramer@tutanota.com).

References

1. Ching T, Himmelstein DS, Beaulieu-Jones BK, et al.: Opportunities and obstacles for deep learning in biology and medicine. Journal of The Royal Society Interface. 2018, 15:20170387. 10.1098/rsif.2017.0387
2. Wen L, Rickert M, Pan F, Lin J, Knoll A: Bare-metal vs. hypervisors and containers: Performance evaluation of virtualization technologies for software-defined vehicles. 2023 IEEE Intelligent Vehicles Symposium (IV), Anchorage, AK, USA. 2023, 1-8. 10.1109/IV55152.2023.10186789
3. Kubeflow. The machine learning toolkit for Kubernetes. (2025). Accessed: September 14, 2025: https://www.kubeflow.org/.
4. Zhu C, Han B, Zhao Y: A comparative performance study of spark on kubernetes. The Journal of Supercomputing. 2022, 78:13298-13322. 10.1007/s11227-022-04381-y
5. Kubernetes: Kubernetes documentation. (2023). Accessed: September 14, 2025: https://kubernetes.io/docs/.
6. Collabnix: Kubernetes on bare-metal vs VMs - performance benchmarks. (2023). Accessed: September 14, 2025: https://collabnix.com/bare-metal-vs-vms-for-kubernetes-performance-benchmarks/.
7. Moreau T, Massias M, Gramfort A, et al.: Benchopt: Reproducible, efficient and collaborative optimization benchmarks. NIPS'22: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 25404-25421.
8. Collective Knowledge project. (2025). Accessed: September 14, 2025: https://github.com/mlcommons/ck.
9. Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S: A design science research methodology for information systems research. Journal of Management Information Systems. 2007, 24:45-77. 10.2753/mis0742-1222240302
10. Howard M: Terraform - Automating infrastructure as a service. arXiv. 2022, 10.48550/arxiv.2205.10676
11. Proxmox Virtual Environment. (2023). Accessed: September 14, 2025: https://www.proxmox.com/en/proxmox-ve/.
12. Prometheus. (2024). Accessed: September 14, 2025: https://prometheus.io/docs/introduction/overview/.
13. Lin CY, Pai HY, Chou J: Comparison between bare-metal, container and VM using TensorFlow image classification benchmarks for deep learning cloud platform. Proceedings of the 8th International Conference on Cloud Computing and Services Science CLOSER. 2018, 1:376-383. 10.5220/0006680603760383
14. Krizhevsky A: Learning multiple layers of features from tiny images. University of Toronto, Toronto, Canada; 2009.
15. Benchmarked IaC Framework for ML Infrastructure. (2025). Accessed: September 14, 2025: https://github.com/Prof-it/terraform_kubeflow_onpremise.
16. Aach M, Inanc E, Sarma R, Riedel M, Lintermann A: Large scale performance analysis of distributed deep learning frameworks for convolutional neural networks. Journal of Big Data. 2023, 10:96. 10.1186/s40537-023-00765-w