Deployment of Large Language Models (LLMs) in a private Cloud for a private version of ChatGPT

José Miguel Ferreira Carvalho
Master in Network and Information Systems Engineering
Department of Computer Science
Faculty of Sciences of the University of Porto
2025

Deployment of Large Language Models (LLMs) in a private Cloud for a private version of ChatGPT

José Miguel Ferreira Carvalho

Dissertation carried out as part of the Master in Network and Information Systems Engineering

Department of Computer Science

2025

Supervisor

Rolando da Silva Martins, Assistant Professor, Faculty of Sciences of the University of Porto

“ Any sufficiently advanced technology
is indistinguishable from magic. ”

Arthur C. Clarke

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my mother and boyfriend for their unwavering support, patience, and encouragement throughout this journey. Their constant presence and belief in me have been fundamental to reaching this milestone.

I am profoundly grateful to my supervisors, Professor Rolando Martins, Professor João Soares and Professor Håvard Dagenborg for their invaluable guidance, expertise, and dedication. Their insights, constructive feedback, and mentorship have been essential to the development of this work. I would also like to extend my sincere thanks to all the professors who have contributed to my academic growth and supported me along the way.

Finally, I wish to thank my colleagues at C3P for their collaboration, camaraderie, and stimulating discussions. Their support and friendship have made this experience both enriching and memorable.

To all who have been part of this journey, my heartfelt thanks.

Resumo

A gestão de cargas de trabalho de inteligência artificial em ambientes multi-nuvem apresenta desafios significativos devido à heterogeneidade de hardware, APIs específicas de fornecedores e complexidade operacional. Esta dissertação investiga a orquestração de IA multi-nuvem através da avaliação sistemática de três plataformas MLOps: Run:AI, Google Vertex AI e ClearML, considerando quatro princípios de design: orquestração inteligente de recursos, interoperabilidade multi-nuvem, governança automatizada e experiência de utilizador unificada. A investigação emprega uma metodologia bifásica: comparação abrangente de plataformas seguida de implementação de protótipo utilizando ClearML, Kubernetes e Argo Workflows. Os resultados demonstram que o Run:AI se destaca na optimização de GPU, o Vertex AI oferece integração gerida na nuvem, enquanto o ClearML proporciona flexibilidade multi-nuvem óptima e transparência de código aberto. O protótipo alcança 100% de sucesso na implementação ao longo de 10 execuções e valida capacidades de implementação inter-nuvem com fluxos de trabalho MLOps automatizados, acompanhamento abrangente de experiências e gestão de recursos. Esta investigação fornece evidência para a selecção de plataformas MLOps em cenários multi-nuvem e oferece um quadro prático para orquestração de cargas de trabalho de IA que aborda o vendor lock-in mantendo a eficiência operacional.

Palavras-chave: Ambiente Multi-Nuvem, Computação em Nuvem, Inteligência Artificial, Kubernetes, Optimização de Recursos de IA

Abstract

Multi-cloud AI workload management presents significant challenges due to hardware heterogeneity, vendor-specific APIs, and operational complexity. This thesis investigates multi-cloud AI orchestration through systematic evaluation of three MLOps platforms: Run:AI, Google Vertex AI, and ClearML across four design principles: intelligent resource orchestration, multi-cloud interoperability, automated governance, and unified user experience. The research employs a two-phase methodology: comprehensive platform comparison followed by prototype implementation using ClearML, Kubernetes, and Argo Workflows. Results demonstrate that Run:AI excels in GPU optimization, Vertex AI provides managed cloud integration, while ClearML offers optimal multi-cloud flexibility and open-source transparency. The prototype achieves 100% deployment success over 10 executions and validates cross-cloud deployment capabilities with automated MLOps workflows, comprehensive experiment tracking, and resource management. This research provides evidence for MLOps platform selection in multi-cloud scenarios and delivers a practical framework for AI workload orchestration that addresses vendor lock-in while maintaining operational efficiency.

Keywords: Multi-Cloud Environment,Cloud Computing,Artificial Intelligence (AI),Kubernetes,AI Resource Optimization

Table of Contents

List of Tables

List of Figures

1. Introduction

Artificial Intelligence is reshaping how organizations approach computational challenges and decision-making processes. From natural language processing systems that facilitate human-computer interaction to computer-vision applications enabling autonomous navigation, from medical diagnostic systems enhancing healthcare delivery to financial modeling algorithms optimizing investment strategies [1, 2], AI systems have become integral to contemporary technological infrastructure. This adoption has created computational demands that challenge the foundational assumptions of traditional computing architectures and operational methodologies.

The scale and complexity of AI applications requires computational resources and orchestration capabilities beyond conventional enterprise computing requirements. Training large-scale language models requires coordinated execution across various specialized processing units, often consuming weeks to months of continuous computational effort on thousands of GPUs with stringent synchronization requirements [3, 4]. Real-time inference systems must maintain consistent performance characteristics while adapting to demand patterns that can fluctuate within minutes. These operational demands represent a fundamental departure from traditional enterprise workloads, which were designed around more predictable resource consumption patterns and established performance baselines.

To address these computational demands, organizations commonly adopt cloud computing as a foundational approach, leveraging its elastic infrastructure capabilities to support the training phases and dynamic inference requirements characteristic of contemporary AI workloads[5, 6]. However, even with cloud computing adoption, challenges persist in optimizing resource utilization and ensuring seamless workflow orchestration. AI inference systems must accommodate dramatic variations in computational complexity based on input characteristics, model architecture, and real-time performance requirements. For example batch processing operations may require sustained high-throughput execution for hours, while interactive applications demand millisecond response times with guaranteed service level agreements. The nature of many AI applications, particularly those involving sequence processing or memory-augmented architectures, introduces session management complexities that traditional stateless cloud applications do not encounter. Furthermore, the resource intensity of AI inference operations, particularly for LLMs or

computer vision systems, requires dynamic scaling strategies that can rapidly provision computational resources while maintaining service continuity.

This leads to organizations often resort to utilizing multicloud environments, where they might leverage one provider for scalable data storage, another for specialized hardware acceleration, and a third for enterprise integration capabilities[7, 8]. Yet coordinating AI workloads across these providers still proves complex and largely manual, limiting both efficiency and innovation potential. This complexity is further amplified by the need for specialized tools and methodologies that can bridge the operational gaps between traditional cloud management approaches , whereas existing development methodologies and operational frameworks demonstrate significant limitations when applied to the context of multi-cloud AI deployments, considering traditional DevOps practices were, designed around stateless applications with predictable resource consumption patterns is expected that they would struggle to accommodate the requirements of distributed AI training operations.

The emergence of Machine Learning Operations (MLOps) as a discipline that seeks to bridge the gap between traditional operational practices and AI-specific requirements[9, 10]. Thus reflecting the broader tension facing AI practitioners today: while cloud computing provides the elastic infrastructure necessary for AI deployment, organizations increasingly require multi-cloud strategies to optimize costs, avoid vendor lock-in, and leverage specialized services. However, managing AI workloads across multiple cloud providers introduces significant operational complexity that current solutions have yet to comprehensively address[11, 12]. Tackling this operational complexity constitutes the primary motivation and objective of this thesis through the systematic evaluation of contemporary MLOps platforms and the development of unified orchestration approaches that can effectively manage AI workloads across heterogeneous cloud environments.

1.1. Research Objectives

This thesis will devise a Comprehensive Platform Analysis whose findings will lead to the development a multi-cloud orchestration framework that we will validate against the AI-specific constraints we highlighted earlier. The platform analysis will be conducted through a comparative evaluation of different MLOps platforms (Run:AI, Vertex AI, and ClearML), examining their architectural foundations, operational characteristics, and multi-cloud capabilities. This analysis seeks to identify specific limitations and trade-offs. The framework development will build upon the platform analysis findings in order to design

orchestration principles that hopefully bridge the gaps identified. The validation approach will implement prototype development and empirical testing to demonstrate whether our proposed framework effectively addresses the AI-specific operational complexities that necessitate specialized MLOps approaches, while revealing practical limitations and implementation considerations. Through our work, we aim to contribute by establishing a comprehensive understanding of current MLOps platform capabilities and their limitations in multi-cloud contexts, providing a framework for unified AI workload orchestration that addresses identified gaps, and delivering empirical validation that demonstrates the framework's effectiveness in solving real-world AI deployment challenges.

1.2. Thesis Organization and Structure

This thesis is structured to provide systematic progression from foundational concepts through empirical analysis to practical implementation and validation.

Chapter 2: Background establishes the theoretical foundation by examining AI workload characteristics, distributed computing fundamentals, GPU computing principles, cloud architectures, MLOps integration, container orchestration, and security considerations. This chapter provides the conceptual framework necessary for understanding multi-cloud AI workload orchestration complexities.

Chapter 3: State-of-the-Art provides comprehensive analysis of current research and industry practices in multi-cloud AI workload orchestration, examining the contemporary MLOps platform landscape, LLM infrastructure developments, emerging multi-cloud trends, and optimization strategies. This chapter identifies specific research gaps and positions the current work within the broader academic context.

Chapter 4: Platform Comparison presents systematic comparative evaluation of Run:AI, Vertex AI, and ClearML, examining architectural foundations, operational characteristics, and multi-cloud capabilities. This analysis employs systematic evaluation criteria to identify strengths, limitations, and trade-offs that inform both platform selection decisions and prototype development directions.

Chapter 5: Multi-Cloud Orchestration Framework details the design, implementation, and validation of the proposed orchestration framework through a dual-phase approach. This chapter describes the prototype architecture, implementation decisions, and empirical evaluation results, providing concrete evidence for the framework's feasibility and practical considerations.

Chapter 6: Conclusion synthesizes research findings by systematically addressing each research objective, connecting identified literature gaps to demonstrated solutions, and establishing strategic implications for multi-cloud AI infrastructure management. This chapter analyzes transformative industry developments that occurred during the research period, particularly the NVIDIA acquisition of Run:AI and subsequent open-sourcing initiatives, positioning these developments within the context of the findings. It also assesses research contributions while identifying directions for future investigation.

2. Background

In this section we provide the necessary foundational understandings with artificial intelligence workloads themselves, examining their unique computational characteristics, operational requirements, and fundamental challenges that distinguish them from traditional computing tasks. We examine the computational approaches that have emerged to address AI workload demands, including distributed computing principles, parallel processing architectures, and cloud computing models. The convergence of these computational demands with practical deployment constraints creates for topic that follows: multi-cloud. Finally, we investigate MLOps as the bridge between AI development practices and production deployment realities. Each section builds upon previous concepts to establish the baseline our research addresses.

2.1. Understanding AI Workloads

Artificial Intelligence workloads represent a paradigmatic shift in computational requirements, exhibiting characteristics that fundamentally distinguish them from traditional enterprise applications. These workloads encompass the entire machine learning lifecycle, from initial data preprocessing through model training, validation, deployment, and ongoing inference serving, each phase presenting distinct computational demands. AI workloads exhibit three fundamental characteristics: **Computational intensity**, **Data-centric processing patterns** and **Dynamic scalability** that create unique infrastructure requirements.

Computational intensity manifests through mathematical operations involving large matrices, tensor manipulations, and iterative optimization algorithms that demand substantial processing power and memory bandwidth. Data-centric processing patterns require systems to handle large datasets efficiently. Dynamic scalability requirements create infrastructure demands that vary across the AI lifecycle, from resource-intensive training phases requiring multiple processing units to lightweight inference serving that may need only fractional GPU resources but with stringent latency requirements.

To exemplify these demands consider the computational progression of developing a large language model for enterprise applications. The initial data preprocessing phase would require distributed systems to cleanse and tokenize multiple text documents, demanding high-throughput storage systems and parallel processing capabilities. The subsequent

training phase coordinates multiple processing units across multiple nodes for weeks, requiring high-bandwidth interconnects and fault-tolerant distributed training frameworks. Finally, the deployment phase serves the trained model through inference endpoints that must handle several concurrent requests with millisecond-level latency constraints, requiring entirely different infrastructure optimizations focused on throughput and response time rather than raw computational power.

2.1.1 AI Workload Classification

We can categorize AI Workloads into four primary types: **training workloads**, **inference workloads**, **fine-tuning workloads** and **data-processing workloads** where each exhibiting distinct operational patterns and infrastructure requirements.

Training workloads represent the most computationally intensive category, processing large datasets through iterative optimization algorithms following batch-oriented processing patterns with execution times normally ranging from hours to weeks, requiring sustained high resource utilization with horizontal scaling capabilities across multiple nodes. These workloads normally have high fault tolerance requirements, as node failures during multi-week training runs can result in significant computational waste and project delays.

Inference workloads prioritize response latency over computational throughput, serving trained models to end users through request-response patterns demanding real-time or near-real-time responses, typically measured in milliseconds, while handling potentially high request volumes with significant traffic variability. Inference workloads benefit from auto-scaling capabilities that can rapidly provision additional resources during traffic spikes and scale down during low-demand periods to optimize costs.

Fine-tuning workloads combine characteristics of both training and inference, adapting pre-trained models for specific domains or tasks, these workloads operate through semi-batch processing patterns, typically running for hours to days with high resource intensity but generally requiring fewer computational resources than full model training. Fine-tuning workloads often exhibit vertical scaling patterns, benefiting more from powerful individual nodes rather than distributed processing across many nodes.

Data-processing workloads support the entire AI pipeline through large-scale data ingestion, preprocessing, feature extraction, and pipeline orchestration activities, these

workloads run for minutes to hours with high resource intensity focused on I/O operations and data transformation rather than mathematical computation thus requiring elastic scaling capabilities to handle varying data volumes and benefit from storage systems optimized for machine learning access patterns.

The distinctive characteristics of these four AI workload types are summarized in Table 2.1, highlighting their differences in processing patterns, execution times, resource requirements, and optimization priorities.

Characteristic	**Training**	**Inference**	**Fine-tuning**	**Data Processing**
Processing Pattern	Batch-oriented iterative optimization	Request-response real-time serving	Semi-batch adaptation	Pipeline orchestration
Execution Time	Hours to weeks	Milliseconds	Hours to days	Minutes to hours
Resource Intensity	Highest computational intensity	Low-moderate computational needs	High intensity (less than training)	High I/O intensity
Scaling Approach	Horizontal scaling across nodes	Auto-scaling with traffic	Vertical scaling (powerful nodes)	Elastic scaling
Primary Focus	Mathematical computation	Response latency	Model adaptation	Data transformation
Traffic Pattern	Sustained utilization	Variable request volumes	Moderate resource bursts	Varying data volumes
Fault Tolerance	High (critical for long runs)	Moderate	Moderate	Moderate
Key Optimization	Computational throughput	Response time	Resource efficiency	I/O performance

Table 2.1: Distinctive characteristics of four AI workload types.

2.1.2 Infrastructure Challenges in AI Workloads

As seen the computational requirements of AI workloads depends on the type of workload and various other factors including *memory bandwidth*, *processing unit utilization*, *network communication* and *storage system requirements*.

Memory is often a bottleneck as modern AI models require rapid access to large parameter sets and training data, often exceeding the memory capacity of individual nodes and requiring distributed memory architectures with high-bandwidth interconnects. GPU utilization optimization presents complex scheduling challenges, as GPUs represent the most expensive components in AI infrastructure while exhibiting utilization patterns that vary dramatically across different workload phases. Efficient processing scheduling must account for model loading overhead, memory fragmentation, and the potential for resource sharing among multiple concurrent workloads without performance interference.

Network communication challenges originate from the need for high-bandwidth, low-latency interconnects between nodes with operations that combines data from all participating processes into a single result and distributes that result back to every process such as all-reduce operations for gradient synchronization require specialized network topologies and communication protocols optimized for collective operations rather than traditional point-to-point communication patterns. The communication-to-computation ratio varies significantly across different model architectures, requiring adaptive network provisioning strategies.

Storage system require high-throughput access for large training datasets and low-latency random access for real-time inference serving. AI workloads often require specialized storage architectures including distributed filesystems optimized for machine learning I/O patterns, caching systems for frequently accessed data, and tiered storage approaches that balance cost with performance across different access patterns. These rapidly changing and resource-intensive requirements create several infrastructure challenges that traditional computing environments do not typically encounter. Consequently, this necessitates the development of distinct infrastructure optimization approaches specifically tailored to AI workload management.Table 2.2 summarizes the key differences between traditional and AI workloads across multiple dimensions.

Dimension	Traditional Workloads	AI Workloads
Resource Intensity	Moderate, usually predictable utilization	High, variable utilization
Execution Duration	Continuous operation (days to years)	Batch execution (minutes to weeks)
Scaling Pattern	Reactive, user-demand driven	Proactive, computational requirement driven
Fault Tolerance	High availability, immediate failover	Checkpoint-based recovery, tolerance for delays
Performance Metrics	Throughput, latency, availability	Convergence speed, accuracy, inference latency
Resource Types	CPU, memory, storage	GPU, high-bandwidth memory, specialized accelerators
Communication Patterns	Client-server, request-response	Collective operations, parameter synchronization
Data Access Patterns	Transactional, random access	Sequential streaming, batch processing
Memory Requirements	Predictable, moderate bandwidth	Large parameter sets, high-bandwidth interconnects
Network Demands	Point-to-point communication	Specialized topologies, All-reduce operations
Storage Systems	General-purpose filesystems	ML-optimized distributed filesystems, tiered storage
Optimization Focus	Service availability and consistency	Resource utilization and computational efficiency

Table 2.2: Comparison of Traditional and AI Workloads

The fundamental differences between AI workloads and traditional computing environments manifest across multiple dimensions, creating distinct infrastructure optimization challenges as mentioned while traditional enterprise applications typically maintain predictable resource utilization patterns with established scaling strategies, AI workloads demand dynamic resource allocation that can adapt to varying computational intensities

throughout the machine learning lifecycle. These distinctive characteristics motivate the desideratum of computing architectures specifically designed to address AI workload requirements.

To provide a comprehensive understanding of AI workloads, it is essential to delineate the stages that comprise the ML lifecycle, from data acquisition to model serving. This lifecycle represents an iterative process that ensures the development, deployment, and maintenance of robust AI systems. According to a framework proposed by Google Cloud [13], the MLOps lifecycle integrates seven key stages: *ML development*, *training operationalization*, *continuous training*, *model deployment*, *prediction serving*, *continuous monitoring*, and *data and model management*. The ML development stage involves experimenting with data preparation, transformation, model training, and evaluation to create a reproducible training pipeline. Training operationalization automates the packaging, testing, and deployment of these pipelines, while continuous training executes them repeatedly in response to new data, code changes, or schedules. Model deployment focuses on preparing the model for production serving, and prediction serving handles inference requests. Continuous monitoring tracks model performance and efficiency, detecting issues like data drift or degradation. Finally, data and model management governs artifacts for auditability, traceability, and reusability, promoting compliance and collaboration. This structured lifecycle highlights the interdependencies between stages, where each phase builds upon the previous to address the unique challenges of AI systems, such as handling probabilistic outputs and evolving data distributions. Comparing these stages reveals trade-offs: for instance, the resource-intensive ML development and continuous training phases prioritize accuracy and adaptability, often requiring significant computational power, whereas deployment and serving emphasize efficiency and low latency to meet real-time demands.

Figure 2.1: The Machine Learning Lifecycle stages, adapted from [13].

This comparison underscores the need for MLOps frameworks to support seamless transitions between stages, ensuring that tools evaluated in subsequent chapters can fulfill these requirements across multi-cloud environments. The lifecycle is visualized in Figure 2.1, which illustrates the iterative flow and key components as a cyclical diagram. The figure shows the process as a loop, beginning with ML development on the left, where initial experiments occur, progressing clockwise through training operationalization (automating pipelines), continuous training (event-triggered retraining), model deployment (preparing for production), prediction serving (handling inferences), continuous monitoring (tracking performance), and central data/model management that supports all stages. Arrows indicate the iterative nature, with feedback loops for retraining based on monitoring results, emphasizing how MLOps enables ongoing improvement and adaptation in AI systems.

The following sections examine how distributed computing, parallel processing, GPU computing, and cloud architectures have evolved to support AI workloads, establishing the computational foundation necessary for effective multi-cloud AI orchestration and optimization strategies.

2.2. Common Computing Approaches

The unique characteristics of AI workloads necessitate specialized computing approaches to address the computational intensity, data-centric processing patterns, and dynamic scalability , several computing approaches have emerged as foundational technologies. In this section we will first describe the *distributed computing principles* that enable coordination across multiple nodes,followed by *GPU computing and parallel processing architectures* that provide the computational power required for AI operations. Finally, we describe*cloud computing* paradigms that offer the scalability and flexibility necessary for modern AI deployments.

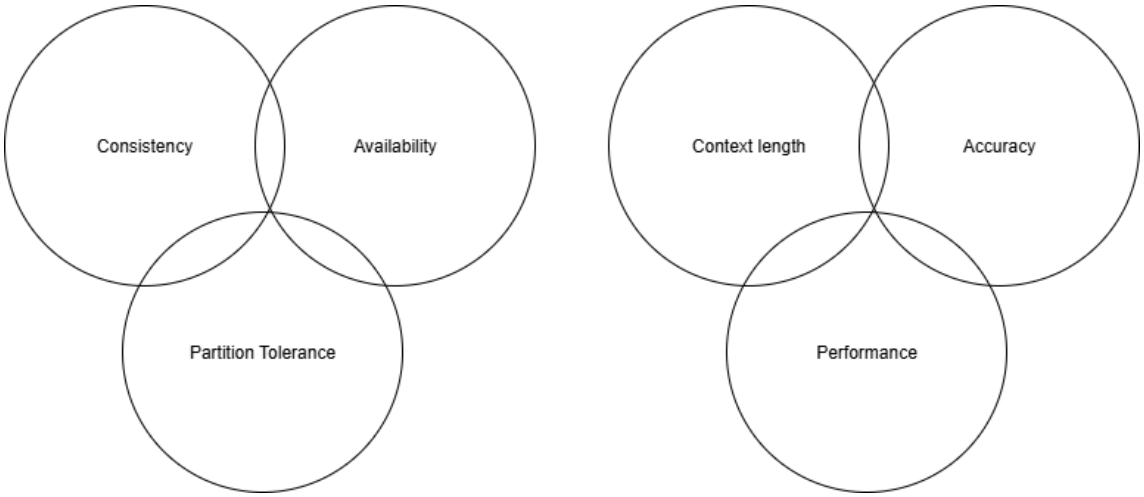

Figure 2.2: AI specific interpretation of CAP principles

As illustrated in Figure 2.2, AI workloads present a unique interpretation of the traditional CAP (Consistency, Availability, Partition Tolerance) theorem, balancing factors such as context length, accuracy, and performance alongside the classical distributed systems trade-offs. The left side of the figure depicts the classical CAP theorem from distributed systems theory, where a system can achieve at most two out of Consistency (all nodes see the same data at the same time), Availability (every request receives a response), and Partition Tolerance (the system continues operating despite network failures). On the right side, we present an AI-specific adaptation inspired by recent advancements in LLM serving. In this reinterpretation, the CAP principle stands for Context length (C: the effective input sequence length the model can process), Accuracy (A: the quality and precision of model outputs), and Performance (P: serving efficiency, including throughput, latency, and cost). As proposed in a comprehensive survey on long-context LLM serving [14], any optimization in LLM serving systems can improve at most two of these factors simultaneously due to inherent architectural constraints in transformer-based models,

such as quadratic computational complexity with increasing context length. For instance, techniques that extend context length (e.g., through advanced positional embeddings or memory mechanisms) often maintain or improve accuracy but degrade performance by increasing computational overhead. Conversely, performance optimizations like quantization or sparsity may enhance efficiency and support longer contexts indirectly but typically reduce accuracy. This trilemma guides the design of AI workloads, particularly in inference-heavy applications, highlighting why no single approach can achieve optimal results across all three dimensions without trade-offs.

Building upon these theoretical foundations, the practical implementation of distributed AI systems requires consideration of consistency models and their performance implications since consistency models for AI systems can range from strong consistency requirements for synchronized training to eventual consistency acceptance for many inference scenarios as we have seen in the previous section [15]. The choice of consistency model directly impacts system performance, with weaker consistency enabling higher throughput and availability at the cost of temporary inconsistencies [16]. The different AI workload types we examined earlier exhibit varying consistency requirements that influence system design decisions. Training workloads typically require strong consistency, as inconsistent updates can compromise model convergence. In contrast, inference workloads can often tolerate eventual consistency models that prioritize availability and response time over perfect consistency, since temporary inconsistencies in model serving typically have minimal impact on individual prediction accuracy.

Beyond consistency considerations, distributed AI systems must implement robust fault tolerance mechanisms since traditional distributed systems often rely on immediate failover mechanisms to maintain continuous service availability but AI workloads present different fault tolerance requirements due to their batch-oriented nature and tolerance for approximate solutions. To address this algorithms such as Byzantine Fault Tolerance provide robust consensus mechanisms for systems where nodes may exhibit arbitrary failures but may require adaptation for AI workloads to accommodate with their long-running processes, checkpoint-based recovery needs, and tolerance for approximate solutions [17]. The checkpoint-based recovery approach commonly used in AI training tries to restart from recent snapshots rather than requiring immediate failover, trading some computational progress for simplified fault tolerance mechanisms prioritizing computational efficiency over continuous availability. As as outlined the storage requirements must address unique I/O patterns including sequential access for training data, random access

for inference serving, and mixed patterns for feature engineering operations resulting in a diverse access patterns requiring storage systems that can optimize for different performance characteristics simultaneously. To solve this modern approaches implement disaggregated storage architectures that separate compute and storage resources, enabling elastic scaling while improving resource utilization and allowing independent scaling of computation and storage resources, optimizing costs while maintaining the high-throughput and low-latency access patterns required by different AI workload types [18]. These distributed computing foundations establish the coordination mechanisms necessary for multi-node AI systems, but the computational demands of AI workloads require specialized processing architectures that can efficiently execute the mathematical operations inherent in machine-learning algorithms.

2.2.1 GPU Computing and Parallel Processing

While distributed computing principles provide the teorems for multi-node AI systems, the actual execution of AI workloads demands specialized hardware architectures capable of handling intensive mathematical computations efficiently. Although specialized hardware exist for AI workloads, including Tensor Processing Units (TPUs), Field-Programmable Gate Arrays (FPGAs), and other Application-Specific Integrated Circuits (ASICs), GPUs have emerged as the dominant computational platform for AI workloads due to their massively parallel architecture, high memory bandwidth, and broad accessibility across cloud providers. This represents a fundamental shift from CPU-centric computing paradigms, driven by the superior performance characteristics of GPUs for the matrix operations and parallel computations that form the backbone of modern machine learning algorithms [19] since the parallel processing capabilities of GPUs align naturally with common requirements in machine learning, including matrix multiplications, convolutions, and element-wise operations that benefit from simultaneous execution across multiple processing cores that unlike traditional CPUs that optimize for sequential processing with several powerful cores, GPUs cores are designed for concurrent execution of similar operations, making them particularly suited for the data-parallel computations prevalent in AI workloads.Such computations employ three primary patterns that address different scaling challenges in AI workload execution: *data parallelism*, *model parallelism*, and *pipeline parallelism*.

Data parallelism distributes training data across multiple GPUs while replicating the model,

enabling linear scaling for large datasets through efficient gradient aggregation and synchronization mechanisms [15]. This approach works effectively when the model fits within individual GPU memory but the dataset size requires distributed processing, allowing organizations to leverage multiple GPUs to accelerate training without architectural modifications to the model itself.

Model parallelism partitions large models across multiple devices when model size exceeds individual GPU memory capacity, introducing complex dependency management and communication patterns [20]. This approach becomes necessary for increasingly large models that cannot fit within the memory constraints of individual GPUs, but requires careful consideration of communication overhead and synchronization complexity.

Pipeline parallelism combines aspects of both approaches by partitioning models into stages and overlapping computation with communication [15], enabling efficient processing of large models while maintaining high throughput through concurrent execution of different pipeline stages.

Such computations employ three primary patterns that address different scaling challenges in AI workload execution: *data parallelism*, *model parallelism*, and *pipeline parallelism*, as summarized in Table 2.3.

Strategy	**Data Parallelism**	**Model Parallelism**	**Pipeline Parallelism**
Primary Use	Large datasets	Large models	Large models + datasets
Model Distribution	Full replication	Partitioned across GPUs	Staged partitions
Memory Constraint	Model fits single GPU	Exceeds single GPU	Variable by stage
Communication	Gradient aggregation	Inter-GPU dependencies	Sequential stages

Table 2.3: Comparison of GPU Parallelism Strategies for AI Workloads

Communication optimization becomes relevant as GPU count increases beyond single-node configurations, particularly in multi-cloud environments where network latency and bandwidth constraints can impact performance. All-reduce operations for gradient aggregation often become performance bottlenecks, requiring advanced techniques including gradient compression, communication scheduling, and topology-aware algorithms to maintain efficiency across hundreds of GPUs [16]. The communication overhead grows

with the number of participating nodes, necessitating sophisticated optimization strategies to prevent communication costs from overwhelming computational benefits.Multi-GPU coordination requires frameworks that manage computation across multiple GPUs within nodes and across multiple nodes in cluster environments, addressing the complexity of heterogeneous hardware configurations common in multi-cloud deployments. These frameworks must address gradient synchronization, fault tolerance, and dynamic resource allocation while maintaining computational efficiency across varying GPU architectures, memory configurations, and interconnect topologies offered by different cloud providers [15]. The heterogeneous nature of multi-cloud deployments introduces additional complexity as workloads may need to adapt to different hardware specifications and performance characteristics across providers.

The implementation of these parallel processing patterns requires programming frameworks that can effectively utilize GPU architectures. GPU programming frameworks such as CUDA and OpenCL provide the foundation for parallel computation, but present fundamental trade-offs between vendor optimization and portability that directly impact deployment strategies across heterogeneous hardware environments. CUDA, developed by NVIDIA, offers highly optimized performance for NVIDIA GPUs but creates vendor lock-in, while OpenCL provides cross-platform compatibility at the cost of potentially reduced performance optimization. Modern GPU architectures further complicate these trade-offs by incorporating specialized units such as Tensor Cores optimized for mixed-precision matrix operations common in deep learning applications [21]. These specialized units can dramatically accelerate AI workloads when properly utilized through vendor-specific optimizations, but may not be accessible through platform-agnostic programming approaches. The choice between vendor-specific optimizations and cross-platform compatibility becomes critical when deploying AI workloads across diverse hardware environments with varying architectural characteristics. While distributed computing provides coordination mechanisms and GPU computing delivers computational power, the practical deployment of AI workloads at scale requires infrastructure platforms that can dynamically provision these resources on demand, leading to the adoption of cloud computing paradigms.

2.2.2 Cloud Computing Paradigms and Multi-Cloud Strategies

While distributed computing provides coordination mechanisms and GPU computing delivers computational power, the practical deployment of AI workloads at scale requires infrastructure platforms that can dynamically provision these resources on demand, leading

to the adoption of cloud computing paradigms capable of supporting AI computational requirements while providing economic models that align costs with utilization patterns [22]. The elastic nature of cloud computing addresses the variable resource requirements of AI workloads that we examined earlier, enabling organizations to provision resources dynamically based on computational demands rather than maintaining fixed infrastructure capacity.However, the reliance on single cloud providers introduces several limitations that have motivated the evolution toward multi-cloud strategies. Vendor lock-in concerns arise when organizations become dependent on proprietary services and APIs, limiting their ability to migrate workloads or negotiate favorable terms. Performance limitations may occur when a single provider lacks optimal resources for specific workload types, geographic regions, or specialized hardware requirements.

Multi-cloud architectures address these limitations by enabling organizations to leverage services from multiple providers such as AWS, Google Cloud, and Azure, representing an evolution beyond single-provider cloud strategies. This approach enhances flexibility by aligning provider strengths with specific workload requirements, improves fault tolerance by mitigating vendor lock-in risks, and optimizes costs through dynamic resource allocation across providers [23, 24]. However, the implementation of multi-cloud strategies introduces several operational complexities that organizations must manage to realize the intended benefits. These complexities manifest across multiple dimensions of infrastructure management, each presenting unique challenges for AI workload deployment.

Resource provisioning and scaling represent the first major complexity area, requiring algorithms that predict resource needs and optimize allocation across multiple providers while accounting for conditions including warm-up times for model loading, GPU initialization overhead, and the stateful nature of training processes [25].

Network architecture design constitutes another challenge, as multi-cloud AI deployments require different design principles compared to traditional distributed systems due to the unique communication patterns, bandwidth requirements, and latency sensitivities inherent in AI workloads [26]. The network requirements for AI workloads often exceed those of traditional applications, particularly for distributed training scenarios that require high-bandwidth, low-latency communication between nodes, that can be particularly challenging when these communication patterns span multiple cloud providers.

Storage and memory management across cloud boundaries introduces further architectural considerations due to the distributed nature of multi-cloud deployments and the high-throughput data access requirements of AI workloads. Traditional storage architectures assume co-location of compute and storage resources within a single provider's infrastructure, but multi-cloud environments require new approaches that can maintain performance while enabling flexible resource placement across providers. Disaggregated memory systems address these challenges by providing horizontal scaling capabilities essential for elastic AI workload management across cloud boundaries while maintaining performance predictability [27]. These systems enable the separation of compute and memory resources, allowing independent scaling of each component based on workload requirements rather than forcing unified scaling decisions. In multi-cloud environments, disaggregated architectures become particularly valuable as they enable workload placement decisions based on optimal resource availability across providers rather than requiring co-location of all resources within a single provider's infrastructure.

Finally economic optimization presents a major complexity area, as multi-cloud deployments must account for data transfer costs, regional pricing variations, and the complexity overhead of managing heterogeneous infrastructure [28]. The cost optimization challenge becomes more complex in multi-cloud environments as organizations must balance the benefits of provider diversity against the increased operational complexity and potential data transfer costs. Effective multi-cloud cost management requires modeling of workload characteristics, provider pricing structures, and data movement patterns to optimize total cost of ownership while maintaining performance and availability requirements.

2.3. The Need for MLOps

While these computing approaches provide the foundation for AI workload deployment, their integration creates substantial operational complexity that extends beyond traditional infrastructure management. The convergence of distributed computing coordination, GPU computational power, and cloud infrastructure elasticity enables powerful AI capabilities, but simultaneously introduces challenges that become particularly acute when AI workloads must be deployed, monitored, and maintained in production environments where reliability, reproducibility, and continuous improvement are essential. Traditional DevOps practices need to adapt in order to bridge the gap between conventional operational practices and AI-specific requirements, thus motivating the development of Machine

Learning Operations (MLOps) as a specialized discipline that extends DevOps principles to address the complexities of managing AI systems throughout their lifecycle.

Traditional DevOps practices emerged from the need to bridge development and operations teams for conventional software applications, focusing on continuous integration, automated deployment, and infrastructure monitoring for stateless applications with predictable behavior patterns. AI systems fundamentally challenge all of those through three critical distinctions: *probabilistic outputs* that vary with input data rather than deterministic responses, *complex dependencies* between code, data, and model artifacts that traditional versioning cannot capture, and the need for *continuous model retraining* as data distributions evolve over time. Beyond these behavioral differences, multi-cloud environments amplify operational complexity by introducing heterogeneous infrastructure management across providers with varying APIs, resource allocation mechanisms, and monitoring interfaces therefore traditional DevOps tools designed for homogeneous environments cannot adequately address the abstraction layers and standardized interfaces across diverse cloud providers.

These differences between DevOps and MLOps are summarized in Table 2.4, highlighting how MLOps extends traditional practices to accommodate AI-specific requirements.

Dimension	**DevOps**	**MLOps**
Focus	Software code and infrastructure for deterministic applications	ML models, data pipelines, and infrastructure for probabilistic systems
Key Artifacts	Code and configuration files	Code, data, models, and experiment metadata
Versioning	Code versioning (e.g., Git)	Comprehensive versioning of code, data, models, and hyperparameters
Testing	Unit, integration, and performance tests for code	Model validation, data quality checks, bias detection, and performance metrics
Deployment	Continuous integration/deployment (CI/CD) for applications	CI/CD/CT (continuous training) for models with retraining triggers
Monitoring	System metrics (e.g., uptime, latency)	Model metrics (e.g., accuracy, drift, bias) alongside system metrics
Team Collaboration	Developers and operations engineers	Data scientists, ML engineers, developers, and operations teams
Environment Handling	Homogeneous infrastructure	Heterogeneous multi-cloud setups with varying APIs

Table 2.4: Comparison of DevOps and MLOps Practices, adapted from [29] and [30].

MLOps addresses these challenges by extending DevOps principles to encompass AI system requirements. The core principles of MLOps include *comprehensive versioning* that tracks relationships between code, data, and model artifacts; *automated testing frameworks* that validate not only code correctness but also model behavior, data quality, and performance characteristics; *continuous integration and deployment pipelines* that accommodate the iterative nature of model development and the complexity of multi-component AI systems; and *monitoring approaches* that track both traditional system metrics and AI-specific indicators including model accuracy, prediction latency, and data drift detection. MLOPs practices also differ at an architectural level, that reflect different priorities and operational contexts , such as *Cloud-native managed services*, *Infrastructure focused platforms* and *Comprehensive open-source frameworks*.Cloud native managed services such as Google Vertex AI, integrate tightly with specific cloud provider ecosystems such as Google Cloud, offering streamlined operations and reduced infrastructure management overhead while potentially creating dependencies on proprietary services and APIs that may limit portability across providers [31]. Infrastructure-focused platforms prioritize resource optimization and orchestration capabilities namely Run:AI , particularly for GPU-intensive workloads, while requiring integration with external tools for comprehensive MLOps functionality [32]. Comprehensive open-source frameworks attempt to provide end-to-end MLOps capabilities while maintaining deployment flexibility and vendor independence, enabling organizations to customize their operational practices according to specific requirements in particular ClearML [33].

The selection of appropriate MLOps approaches becomes critical for organizations planning to deploy AI workloads across multi-cloud environments, as different platform architectures present trade-offs between operational simplicity, flexibility, and vendor independence. On one hand cloud-native approaches may offer reduced operational complexity and seamless integration within single-provider ecosystems, but also may limit the ability to leverage multi-cloud strategies for cost optimization, risk mitigation, and capability diversification. On the other hand infrastructure-focused platforms provide greater control over resource allocation and optimization but require more operational expertise and integration effort to achieve comprehensive MLOps functionality. Furthermore open-source frameworks offer maximum flexibility and vendor independence but demand significant implementation and maintenance effort to achieve production-ready operational capabilities.These trade-offs are summarized in Table 2.5.

Characteristic	Cloud-Native Managed Services (e.g., Vertex AI)	Infrastructure-Focused Platforms (e.g., Run:AI)	Comprehensive Open-Source Frameworks (e.g., ClearML)
Integration	Tight with specific cloud ecosystem	Focus on resource orchestration	Vendor-agnostic and customizable
Operational Complexity	Low (managed services)	Medium (requires integration)	High (custom implementation)
Flexibility	Limited by provider	High for resources, lower for full pipeline	Maximum across environments
Vendor Independence	Low (proprietary dependencies)	Medium	High
Primary Strength	Streamlined operations	GPU/resource optimization	End-to-end customization
Key Trade-Off	Portability limitations	Additional tooling needed	Higher maintenance effort

Table 2.5: Comparison of MLOps Architectural Approaches, adapted from [34] and [35].

hese considerations directly influence the evaluation criteria for multi-cloud AI orchestration platforms, so the platform comparison and prototype development described in subsequent chapters build upon these MLOps foundations to examine how different orchestration approaches address the operational challenges of managing AI workloads across heterogeneous multi-cloud environments while maintaining the reliability, reproducibility, and scalability requirements that characterize production AI systems.

The foundational technologies examined in this chapter: distributed computing architectures, GPU acceleration, cloud infrastructure, and MLOps principles establish the building blocks for modern AI systems. However, understanding individual technologies alone proves insufficient for navigating contemporary AI workload orchestration. The state-of-the-art represents the dynamic convergence of technological capabilities, industry practices, and research innovations that define how organizations currently approach multi-cloud AI deployment challenges.

3. State-of-the-Art

Our chapter examines how the AI research community and industry have responded to the computational demands and operational complexities identified in our background analysis. We trace the evolution from isolated infrastructure components to integrated platform solutions, exploring how recent advances in large language model serving, distributed training frameworks, and regulatory requirements have reshaped the field during 2024-2025. By analyzing current platforms on their architectural philosophies, deployment patterns, and real-world implementations we showcase both remarkable progress and persistent limitations in multi-cloud AI orchestration.

We examine five interconnected dimensions: contemporary AI infrastructure demands driven by large language models, multi-cloud adoption patterns addressing these demands, platform approaches that have emerged (managed services, infrastructure optimization, and open-source frameworks exemplified respectively by Vertex AI, Run:AI, and ClearML), the broader MLOps ecosystem including regulatory frameworks and industry practices, and finally the motivation behind our systematic platform comparison in Chapter 3 and prototype development in Chapter 4. This analysis serves two critical purposes: enabling systematic evaluation of platforms against multi-cloud requirements, and identifying specific limitations in unified management, predictive orchestration, and cross-cloud interoperability that our work addresses. The chapter thus bridges foundational concepts with empirical investigation, positioning our contributions within the broader evolution of AI infrastructure.

3.1. Contemporary AI Infrastructure Demands

The artificial intelligence landscape has undergone a fundamental transformation in computational requirements, driven primarily by the emergence of large language models and multi-modal systems that dwarf the resource demands of earlier AI approaches. While Chapter 2 established the foundational characteristics of AI workloads: computational intensity, data-centric processing, and dynamic scaling, the period spanning 2024-2025 has witnessed a qualitative shift in infrastructure needs that challenges conventional deployment strategies. Understanding these evolving demands proves essential for comprehending why organizations increasingly turn to multi-cloud solutions despite the operational complexity such architectures introduce.

3.1.1 The LLM Infrastructure Revolution

The serving framework revolution of 2024-2025 has introduced technologies that achieve dramatic performance improvements while simultaneously introducing new complexity dimensions for infrastructure orchestration. These advances fundamentally alter the relationship between model serving and infrastructure management, requiring orchestration capabilities that extend beyond traditional machine learning deployment patterns.

vLLM has emerged as the dominant inference serving framework, with version 0.6.0 demonstrating how architectural innovation can dramatically reshape performance expectations. The framework achieves 2.7x higher throughput and 5x faster time-per-output-token through process separation and multi-step scheduling innovations that enable dynamic request handling adapted to varying sequence lengths [36]. Most significantly, the continuous batching implementation achieves 3.5x throughput improvements over naive batching approaches by treating attention computations similar to virtual memory paging. The PagedAttention mechanism represents a fundamental rethinking of memory management for LLM serving. By enabling efficient sharing of key-value caches across multiple requests, the approach reduces memory consumption by up to 80% without compromising computational performance [37]. This innovation addresses one of the critical bottlenecks in large-scale LLM deployment, but introduces orchestration challenges when coordinating these optimizations across heterogeneous cloud environments with different hardware configurations and memory architectures.

SGLang has emerged as another major competitor through its RadixAttention technology, demonstrating that the serving framework landscape remains highly dynamic and competitive enabling up to 5x throughput improvements through automatic key-value cache reuse and intelligent request routing, achieving deployment on over 1 million GPUs worldwide [38]. This massive adoption scale illustrates both the critical importance of efficient LLM serving and the fragmentation challenges organizations face when selecting and integrating serving technologies across multiple cloud providers. TensorRT-LLM provides a third example of specialized optimization via NVIDIA-optimized inference capabilities specifically designed for transformer architectures, achieving significant performance improvements on NVIDIA hardware through kernel fusion, quantization optimization, and specialized attention implementations [39]. However, this hardware-specific optimization creates deployment challenges in multi-cloud environments where different providers offer varying GPU configurations and optimization tool chains, therefore organizations must

often maintain multiple serving implementations to leverage provider-specific optimizations, increasing operational complexity and potentially fragmenting model deployment workflows. These serving framework advances demonstrate the rapid evolution of LLM infrastructure while highlighting the complexity of managing these systems across multiple cloud providers with different hardware configurations, optimization capabilities, and API paradigms. While performance benefits prove substantial, achieving them requires sophisticated orchestration that current platforms inadequately address.

3.1.2 Multi-Tenant and Multi-Modal Optimization Challenges

The infrastructure challenges extend beyond single-model serving to encompass scenarios where organizations must simultaneously serve multiple models for different tenants or handle workloads that combine multiple data modalities. These multi-tenant and multi-modal requirements introduce additional orchestration complexity that compounds the serving framework challenges identified above.

Recent advances in multi-tenant optimization have produced significant improvements in resource efficiency for LLM serving. S-LoRA enables serving thousands of concurrent LoRA adapters with unified memory management, achieving substantial improvements in resource utilization while maintaining performance isolation between different tenants [40]. Punica demonstrates complementary multi-tenant LoRA serving capabilities that enable organizations to optimize resource utilization while serving multiple models and clients simultaneously [41].

However, these multi-tenant serving approaches introduce new orchestration requirements. Organizations must dynamically allocate resources based on tenant priorities, service-level agreements, and workload characteristics coordination that becomes exponentially more complex in multi-cloud environments. Current platforms typically handle multi-tenancy within single-cloud contexts, requiring manual coordination when tenant isolation, resource allocation, and performance guarantees must span heterogeneous infrastructure with varying capabilities and cost structures.

The evolution toward multi-modal AI applications adds another dimension to these orchestration challenges. Vision-language models like Qwen2.5-Omni require efficient multi-modal batching techniques that can handle simultaneous text and image processing while

maintaining computational efficiency [42]. Unlike single-modality workloads, these applications demand heterogeneous hardware resources: GPUs for visual processing, specialized accelerators for language models, and high-bandwidth memory for cross-modal attention mechanisms. Coordinating such diverse computational requirements creates routing and scheduling challenges that single-cloud managed services cannot optimally address, particularly when optimal resources for different modalities reside on different cloud providers.

While serving challenges dominate production deployments, the computational demands of training these sophisticated models create equally significant orchestration requirements at unprecedented scale. The evolution from serving optimization to training coordination reveals how infrastructure complexity compounds across the AI lifecycle.

3.1.3 Distributed Training at Unprecedented Scale

Meta's MAST system manages over 16,000 H100 GPUs across geo-distributed datacenters through temporal and scope decoupling strategies [43]. This scale demonstrates how training workloads now routinely span multiple geographic locations and infrastructure providers, but effective coordination requires more than simple resource aggregation. With this in mind, network topology becomes critical when coordinating multiple GPUs across different datacenters and cloud regions. The MAST system's success depends on sophisticated scheduling that considers both computational resources and network characteristics. Most current MLOps platforms lack these capabilities, particularly in multi-cloud contexts where network paths traverse multiple provider boundaries.

Modern scheduling techniques incorporate workload prediction to optimize resource allocation proactively, making use of machine learning-based resource management models to improve accuracy in predicting workload requirements, enabling better provisioning and cost optimization [44]. For large-scale training workloads, resource allocation decisions made hours or days in advance significantly impact both cost and completion time. An example is Microsoft Research's SuperBench framework that achieves 22.61x improvement in Mean Time Between Infrastructure failures through intelligent reliability monitoring and predictive maintenance [45], providing standardized methodologies for evaluating infrastructure reliability across different cloud providers, though implementing such monitoring across multiple providers requires orchestration capabilities that most platforms lack.

These distributed training advances reveal both remarkable progress and persistent limitations. While individual cloud providers offer increasingly sophisticated training capabilities, coordinating these capabilities across multiple providers to achieve optimal performance, cost, and reliability remains largely unsolved. This gap between single-provider excellence and multi-provider coordination typifies the broader infrastructure challenge facing AI deployments.

3.2. Multi-Cloud Adoption in Practice

The infrastructure demands identified in the previous section have driven measurable shifts in organizational deployment strategies. Industry analysis reveals that 92% of enterprises have adopted multi-cloud strategies, with the global AI orchestration market reaching $9.33 billion in 2024 and growing at 23% annually [46, 47]. Meanwhile, 98% of organizations are exploring generative AI capabilities, and overall AI adoption has jumped from 50% to 72% globally [48]. However, multi-cloud adoption patterns vary significantly across organizational contexts, reflecting different priorities, constraints, and strategic objectives. Real-world implementations demonstrate both the potential benefits and persistent challenges of distributed AI infrastructure, providing concrete evidence for how organizations navigate these trade-offs in production environments.

Automotive sector implementations reveal the impact of effective resource orchestration on development velocity and cost efficiency. Wayve, an autonomous vehicle company, confronted severe GPU utilization challenges that threatened their competitive position in the rapidly evolving AV market [49]. Their infrastructure suffered from 15% baseline GPU utilization, with data scientists experiencing multi-day queue times and development teams struggling with resource contention that severely impacted productivity. Addressing these challenges required specialized infrastructure orchestration platforms designed specifically for GPU-intensive workloads. Wayve adopted Run:AI, a Kubernetes-native orchestration platform focused on maximizing GPU utilization through intelligent scheduling and resource sharing. The implementation transformed their infrastructure performance: GPU utilization increased from 15% to 85% while simultaneously reducing training time for critical models by 60% through features including gang scheduling for distributed training, fractional GPU allocation enabling multiple workloads to share resources, and dynamic priority management coordinating development across autonomous vehicle projects [49]. This improvement in resource efficiency directly accelerated Wayve's development cycles and reduced infrastructure costs, demonstrating how infrastructure-focused orchestration

platforms can unlock substantial value in AI-intensive industries where GPU availability constrains innovation velocity.

Manufacturing sector adoption demonstrates different priorities, with emphasis on comprehensive MLOps capabilities and intellectual property control rather than pure resource optimization. Volkswagen's autonomous vehicle development program faced challenges managing thousands of machine learning experiments across multiple manufacturing facilities and research centers, requiring not just computational resources but reproducible workflows, rigorous model versioning, and comprehensive audit trails across distributed development teams [50]. These requirements drove adoption of comprehensive open-source MLOps platforms that could provide end-to-end experiment management while maintaining organizational control over proprietary data and models. Volkswagen implemented ClearML, an open-source platform offering full-stack MLOps capabilities from experiment tracking through model deployment. The platform's agent-based architecture enabled distributed execution while preserving centralized experiment tracking, allowing coordination across multiple facilities without compromising data sovereignty or regulatory compliance requirements [50]. This implementation illustrates how large enterprises with complex organizational structures leverage comprehensive open-source frameworks to balance extensive functionality with the operational control necessary for protecting intellectual property and meeting compliance obligations.

Technology sector implementations demonstrate diverse strategies reflecting varying cloud provider relationships and architectural priorities. Organizations with substantial existing cloud infrastructure investments often seek platforms that integrate seamlessly with their established ecosystems, prioritizing operational simplicity and rapid deployment over deployment flexibility. These organizations pursue fully managed service approaches that abstract infrastructure complexity, accepting tighter cloud provider coupling in exchange for reduced operational overhead and accelerated development cycles. Google Cloud-oriented organizations exemplify this pattern through adoption of Vertex AI, a comprehensive managed MLOps platform deeply integrated within Google's cloud ecosystem [51]. These implementations leverage native integration with services including BigQuery for data processing, Cloud Storage for artifact management, and Tensor Processing Units for optimized training, enabling teams to focus on model development rather than infrastructure management. The managed service approach proves particularly valuable for organizations lacking dedicated infrastructure expertise or those prioritizing rapid iteration over customization capabilities.

Despite these successful implementations, persistent challenges constrain effective multi-cloud AI orchestration. Organizations consistently cite infrastructure complexity as the primary barrier, with heterogeneous environments, inconsistent APIs, and fragmented management interfaces creating significant operational overhead. Data orchestration across cloud boundaries presents particular challenges, as moving large training datasets between providers incurs substantial costs while maintaining data lineage and access controls requires extensive manual coordination.

Resource optimization across providers remains limited by current platform capabilities. While individual clouds offer increasingly sophisticated resource management, coordinating allocations across providers to minimize cost while meeting performance requirements exceeds most orchestration platforms' capabilities. Organizations often resort to conservative over-provisioning or accept suboptimal utilization rather than implementing dynamic cross-cloud optimization strategies.

These persistent challenges have driven platform specialization along three distinct architectural philosophies, each addressing different aspects of multi-cloud complexity through fundamentally different design approaches. The following section examines how managed service platforms like Vertex AI, infrastructure-focused solutions like Run:AI, and comprehensive open-source frameworks like ClearML address organizational needs through their respective architectural choices, revealing both significant progress and persistent limitations in multi-cloud AI orchestration.

3.3. Platform Architectural Approaches

The multi-cloud challenges and organizational priorities identified in the previous section have driven the emergence of distinct platform architectures, each representing fundamentally different trade-offs between operational simplicity, deployment flexibility, and infrastructure control. The MLOps market has evolved beyond simple cloud service abstraction toward specialized solutions addressing specific organizational pain points, with platform selection critically depending on existing infrastructure investments, operational capabilities, and strategic priorities.

3.3.1 Managed Service Integration: Cloud-Native Simplification

Organizations deeply invested in specific cloud ecosystems often prioritize platforms that maximize integration with existing infrastructure while minimizing operational complexity. The managed service approach exemplifies this philosophy, abstracting infrastructure

management to enable teams to focus on model development rather than platform operations. This approach has gained substantial traction, with major cloud providers reporting 40-60% of enterprise AI workloads running on managed ML services [52].

Vertex AI demonstrates this approach through comprehensive integration within Google Cloud's ecosystem, offering access to 100+ foundation models through its Model Garden while leveraging native services including BigQuery for data processing, Cloud Storage for artifacts, and Tensor Processing Units for optimized training [51]. The platform automates infrastructure provisioning, hyperparameter tuning through Vertex Vizier, and deployment scaling, achieving documented uptime exceeding 99.96% while reducing operational overhead. Organizations adopting this approach accept tighter cloud coupling in exchange for rapid deployment capabilities, particularly valuable for teams lacking dedicated infrastructure expertise.

Alternative managed service implementations demonstrate similar patterns with provider-specific optimizations. AWS SageMaker provides comprehensive AWS ecosystem integration including seamless data lake access and automated deployment pipelines, while Azure Machine Learning offers advanced governance features and visual pipeline designer tools that reduce development complexity for enterprise teams. These variations illustrate how managed services optimize for their respective cloud ecosystems while maintaining the fundamental philosophy of operational simplicity over deployment flexibility.

3.3.2 Infrastructure Optimization: Resource-Focused Orchestration

The infrastructure optimization approach prioritizes maximum hardware utilization and scheduling efficiency. GPU resources typically achieve only 15-30% utilization in traditional environments due to static allocation and inefficient scheduling. Platforms following this philosophy provide sophisticated resource management while requiring deeper infrastructure expertise.

Run:AI exemplifies this approach through Kubernetes-native GPU orchestration that extends container scheduling with AI-specific capabilities [53]. Gang scheduling coordinates distributed training by ensuring all required pods receive simultaneous GPU allocation, preventing partial deployments that waste resources. Fractional GPU allocation enables resource sharing down to 0.1 GPU granularity, allowing multiple workloads to efficiently utilize hardware that would otherwise remain idle. The platform also integrates

with NVIDIA Multi-Instance GPU technology, providing hardware-level partitioning into up to seven independent instances, each with dedicated compute and memory resources ensuring complete isolation between workloads.

Wayve's deployment demonstrated this philosophy's effectiveness, achieving 15% to 85% GPU utilization improvements while reducing training times by 60% [49]. These performance improvements directly translate to reduced infrastructure costs and accelerated development cycles, particularly valuable for resource-constrained environments where GPU availability constrains research velocity.

The platform's enterprise licensing model and initially proprietary codebase represented deployment considerations distinct from open-source alternatives, though the project has since transitioned toward open-source availability. This evolution reflects broader industry trends toward transparent, community-driven infrastructure development while maintaining enterprise-grade support and deployment capabilities. The approach proves particularly valuable where GPU costs dominate infrastructure expenses and maximizing hardware ROI justifies the platform deployment complexity.

3.3.3 Comprehensive Open-Source: End-to-End Flexibility

The comprehensive open-source approach attempts to provide full MLOps lifecycle capabilities while maintaining deployment flexibility across diverse infrastructure environments. This philosophy prioritizes vendor independence and customization capabilities over operational simplicity, requiring greater internal expertise but enabling complete platform control and intellectual property protection.

ClearML demonstrates this approach through unified experiment tracking, dataset management, pipeline orchestration, and model serving capabilities deployable across cloud, on-premises, and hybrid environments [54]. The platform's agent-based architecture enables distributed execution while maintaining centralized tracking, supporting deployment from simplified Docker Compose configurations for development to production Kubernetes clusters via Helm charts. This flexibility extends to GPU management through fractional allocation implemented via container-based memory limits, though at the container level rather than through deep scheduler integration like specialized infrastructure platforms.

Volkswagen's implementation managing various autonomous vehicle experiments across multiple facilities illustrates this approach's value for complex organizational structures [50].

The automotive manufacturer leveraged ClearML's open-source foundation to maintain complete infrastructure control while achieving comprehensive MLOps capabilities spanning experiment reproducibility, model versioning, and audit trail maintenance across geographically distributed teams. Organizations adopting this approach accept higher operational complexity and infrastructure management responsibilities in exchange for complete platform control, particularly valuable for regulated industries requiring strict data sovereignty and comprehensive compliance capabilities.

3.3.4 Platform Selection and Multi-Cloud Implications

Vertex AI, Run:AI, and ClearML demonstrate three fundamentally different approaches to multi-cloud AI orchestration. Vertex AI prioritizes operational simplicity through deep cloud integration, accepting vendor coupling for cohesive ecosystem benefits. Run:AI focuses on GPU utilization optimization, sacrificing comprehensive capabilities for specialized infrastructure control. ClearML attempts end-to-end lifecycle management across heterogeneous infrastructure, trading operational complexity for deployment flexibility. These three platforms exemplify the core architectural philosophies currently addressing multi-cloud challenges, each revealing distinct trade-offs between simplicity, specialization, and flexibility.

However, the MLOps ecosystem extends beyond these dominant approaches to include diverse specialized solutions. Data-centric platforms like Databricks emphasize Apache Spark integration for organizations with substantial data engineering investments, with Unity Catalog providing unified governance across AWS, Azure, and Google Cloud [55]. Modular tools including MLflow for experiment tracking, Kubeflow for Kubernetes-native orchestration, and Weights & Biases for collaborative research enable organizations to assemble customized stacks, though requiring significant integration effort to coordinate multiple specialized components into cohesive workflows.

This ecosystem diversity underscores why systematic platform analysis proves essential for understanding multi-cloud orchestration challenges. These trade-offs extend beyond individual platform limitations to reflect deeper challenges in multi-cloud AI orchestration. Evolving workload requirements, regulatory constraints, and rapid technological advancement continue driving platform specialization and ecosystem fragmentation. The detailed comparative analysis in Chapter 3 systematically evaluates how Vertex AI, Run:AI, and

ClearML navigate these tensions, revealing both their significant accomplishments and the persistent limitations that our research addresses through prototype development.

3.4. Emerging MLOps Ecosystem Technologies

The MLOps landscape has evolved beyond individual platform capabilities to encompass transversal technologies that fundamentally reshape how organizations approach AI infrastructure. These developments influence platform selection, deployment strategies, and operational practices across diverse organizational contexts.

3.4.1 Container Orchestration and GPU Virtualization

Kubernetes has become the standard for AI workload orchestration, with the NVIDIA GPU Operator automating deployment of GPU drivers, container toolkit, device plugins, and monitoring tools [56]. GPU Feature Discovery automatically labels nodes with hardware characteristics, the Device Plugin registers GPU resources with kubelet for scheduler allocation, and DCGM Exporter provides Prometheus-compatible metrics for utilization monitoring [57, 58].

GPU virtualization technologies have advanced significantly beyond simple exclusive allocation. Time-slicing enables round-robin context switching among containers sharing GPU hardware, allowing organizations to increase resource utilization from typical 15-30% baselines to 80-90% through concurrent workload execution [59, 60]. Multi-Instance GPU technology provides hardware-level partitioning on datacenter GPUs like A100 and H100, creating up to seven independent instances with dedicated streaming multiprocessors and memory controllers [61].

The NVIDIA KAI Scheduler, introduced in 2024, extends Kubernetes-native GPU scheduling for thousand-node deployments through topology-aware placement, gang scheduling for distributed training, and intelligent preemption policies [62]. This evolution from basic resource counting to workload-aware orchestration reflects broader industry recognition that effective AI infrastructure requires understanding workload characteristics, not merely tracking resource availability.

3.4.2 Cross-Cloud Communication and Service Mesh

Service mesh technologies have achieved widespread adoption among organizations deploying AI applications, providing essential capabilities for cross-cloud communication

patterns [48, 63]. These solutions implement circuit breaking, retry policies, and distributed tracing that enhance reliability across heterogeneous cloud environments. Service meshes enable consistent traffic management, security policies, and observability regardless of underlying cloud provider, addressing fragmentation that otherwise requires provider-specific implementations.

However, service mesh adoption introduces operational complexity through sidecar proxies, control plane management, and configuration overhead that organizations must balance against reliability benefits. The technology proves particularly valuable for applications spanning multiple clouds where consistent networking and security policies justify deployment complexity.

3.4.3 Regulatory Compliance and Governance

The regulatory landscape for AI systems has transformed dramatically in 2024-2025. The European Union AI Act reached full force in August 2024, with prohibition provisions active since February 2025 and comprehensive requirements for high-risk AI systems scheduled for August 2026 [64]. US regulations introduced in January 2025 require identity verification for foreign cloud access and computing cluster registration for training runs exceeding specified computational thresholds [65].

These regulations directly impact multi-cloud AI orchestration by necessitating identity management, access control, and audit trail capabilities across heterogeneous cloud environments. Confidential computing technologies have emerged as essential capabilities for meeting regulatory requirements while enabling multi-cloud deployments, providing cryptographic guarantees about data protection and computational integrity [66].

3.4.4 Sustainability and Energy Efficiency

AI workload energy consumption has become increasingly significant as model complexity and training requirements grow. Large-scale transformer model training can consume hundreds of thousands of kilowatt-hours [67], driving organizations to implement energy-efficient computing strategies including specialized hardware utilization, dynamic resource scaling, and scheduling optimizations that align computational requirements with renewable energy availability [68].

However, measuring and optimizing energy efficiency in multi-cloud AI workloads remains challenging due to limited visibility into actual energy consumption across providers

and complexity in attributing costs to specific workloads [69, 70]. Standardized energy monitoring protocols are essential for enabling consistent measurement and optimization across heterogeneous cloud environments.

3.4.5 Industry Adoption Patterns

Platform migration experiences reveal consistent patterns across organizational contexts [71]. Companies transitioning from proprietary platforms to open-source solutions report improved cost predictability and reduced vendor dependency, though acknowledge increased operational complexity. Migrations toward managed platforms cite operational burden reduction and improved development velocity while accepting reduced customization capabilities and increased long-term costs. Hybrid approaches utilizing multiple platforms for different use cases emerge as pragmatic solutions balancing diverse requirements while introducing integration complexity across systems.

These adoption patterns demonstrate that platform selection involves continuous trade-offs rather than optimal solutions. Organizations must evaluate platforms within specific operational contexts, considering existing infrastructure investments, team capabilities, regulatory requirements, and strategic priorities that evolve over time.

3.5. Research Synthesis: Four Fundamental Challenges

Our analysis of contemporary AI infrastructure reveals four challenges that consistently emerge across the MLOps ecosystem. These challenges drive platform architectural decisions and shape organizational adoption patterns, ultimately determining the effectiveness of multi-cloud AI deployments.

3.5.1 Challenge 1: Intelligent Resource Orchestration

Modern AI workloads require unprecedented computational resources. Large language models demand various GPUs for distributed training and millisecond-latency inference serving, creating complex resource management requirements. Current platforms reveal a troubling pattern: specialized solutions like Run:AI achieve dramatic GPU utilization improvements yet require extensive expertise, while managed platforms like Vertex AI provide operational simplicity at the cost of optimization visibility and customization capabilities.

The LLM infrastructure revolution demonstrates that serving frameworks can achieve substantial performance improvements, yet coordinating these optimizations across heterogeneous cloud environments remains largely manual. Distributed training advances managing thousands of GPUs show what is possible, but orchestration complexity increases exponentially when spanning multiple cloud providers with varying capabilities.

Research Question 1 (Q1): *Can unified orchestration frameworks provide intelligent resource allocation approaching specialized platform efficiency while maintaining operational simplicity and multi-cloud portability?*

3.5.2 Challenge 2: Multi-Cloud Interoperability

Despite 92% enterprise multi-cloud adoption, achieving effective interoperability requires addressing hardware heterogeneity, API inconsistencies, and data coordination challenges when resources span cloud boundaries. Current platforms demonstrate fundamental tensions: cloud-native solutions provide excellent single-ecosystem integration while creating vendor lock-in, infrastructure-focused platforms support multi-cloud deployment yet require configurations, and comprehensive frameworks offer flexibility but demand significant operational expertise.

Ecosystem fragmentation extends beyond platforms to specialized tools for experiment tracking, orchestration, serving, and data management. Organizations attempting best-of-breed integration face substantial complexity coordinating multiple components across heterogeneous infrastructure.

Research Question 2 (Q2): *How can multi-cloud orchestration architectures achieve seamless interoperability across diverse providers without sacrificing provider-specific optimizations or introducing unacceptable operational complexity?*

3.5.3 Challenge 3: Automated Governance

Regulatory transformation in 2024-2025 fundamentally altered AI deployment requirements. Organizations must implement comprehensive audit trails, risk assessment frameworks, and compliance monitoring operating consistently across distributed multi-cloud infrastructure. Current platforms demonstrate varying governance maturity: managed services inherit provider security frameworks while limiting customization, specialized platforms focus on optimization while delegating governance externally, and comprehensive frameworks provide foundational capabilities requiring substantial production hardening.

Emerging requirements for confidential computing, data sovereignty, and cross-border compliance create complexity layers that current orchestration platforms inadequately address.

Research Question 3 (Q3): *Can automated governance frameworks provide consistent policy enforcement and compliance monitoring across multi-cloud environments while accommodating provider-specific capabilities and evolving regulatory requirements?*

3.5.4 Challenge 4: Unified User Experience

Operational fragmentation across MLOps ecosystems creates significant cognitive overhead. Researchers must master different interfaces for experiment tracking, resource allocation, model deployment, and monitoring, often requiring distinct expertise for each cloud provider. Platform adoption patterns reveal user experience as critical: managed platforms achieve high satisfaction through polished interfaces yet sacrifice flexibility, infrastructure-focused solutions provide powerful capabilities but demand specialized expertise, and comprehensive frameworks attempt unified experiences while introducing their own complexity.

The challenge extends beyond interface consistency to workflow integration: seamlessly transitioning from experimentation to production, from single-GPU development to distributed training, from one provider to another without relearning operations or reconfiguring toolchains.

Research Question 4 (Q4): *Can unified user experiences maintain operational consistency across complex multi-cloud deployments while preserving specialized capabilities and avoiding lowest-common-denominator abstractions?*

3.5.5 Research Objectives and Investigation Structure

These four research questions frame our systematic investigation into multi-cloud AI workload orchestration. Chapter 4 evaluates how current leading solutions address these challenges, revealing architectural trade-offs and identifying limitations that inform our prototype development. Chapter 5 demonstrates practical unified orchestration approaches through empirical implementation, providing evidence regarding the feasibility of addressing these challenges within realistic deployment constraints.

Our investigation acknowledges that perfect solutions may not exist. The tensions between specialization and comprehensiveness, simplicity and flexibility, optimization and portability may represent fundamental trade-offs rather than engineering challenges awaiting resolution. However, evaluation of current approaches and principled prototype development hopefully can help understanding.

Chapter 3 established that modern MLOps platforms must address four fundamental challenges: *intelligent resource orchestration*, *multi-cloud interoperability*, *automated governance*, and *unified user experience*. Three architectural approaches emerged: *cloud-native managed services*, *infrastructure-focused orchestration*, and *comprehensive frameworks*. This chapter evaluates three representative platforms exemplifying these approaches: *Google Vertex AI* (managed service), *Run:AI* (infrastructure-focused), and *ClearML* (comprehensive framework). Section 4.1 examines each platform's design principles. Section 4.2 synthesizes their architectures. Section 4.3 validates the analysis through industry adoption patterns and performance benchmarks. Finally, Section 4.5 presents our selection framework and rationale for choosing ClearML as our research prototype foundation.

4. Platform Comparison

This chapter compares three AI computing platform, each with its own unique design philosophy: **Google Vertex AI** represents the managed service approach, emphasizing seamless integration within Google Cloud Platform's ecosystem. **Run:AI** embodies the infrastructure-focused orchestration strategy, delivering fine-grained resource control through Kubernetes-native implementation. **ClearML** demonstrates the comprehensive framework philosophy, providing end-to-end MLOps capabilities with deployment flexibility across diverse environments. The analysis progresses through four stages: Section 4.1 examines each platform's foundational design principles and implementation strategies. Section 4.2 synthesizes individual platform architectures. Section 4.3 validates architectural analysis through empirical evidence, examining industry adoption patterns and performance benchmarks that demonstrate real-world platform effectiveness. Finally, Section 4.5 presents our selection framework and provides detailed rationale for choosing ClearML as the foundation for our research prototype.

4.1. Platform Architectural Foundations and Design Philosophy

Each platform addresses AI workload management differently, reflecting distinct design philosophies that determine their strengths and limitations.As we have seen AI workloads require platforms have various operational patterns, table 4.1 categorizes these workload types by execution pattern, latency requirements, scaling approach, and resource demands.

Workload Type	Pattern	Latency	Scaling	Resources
Training	Batch	Hours-weeks	Horizontal	Very High
Inference	Request-response	Milliseconds	Auto	Moderate
Fine-tuning	Semi-batch	Hours-days	Vertical	High
Data Processing	Pipeline	Minutes-hours	Elastic	High

Table 4.1: AI Workload Characteristics Matrix

These diverse operational patterns create competing architectural requirements that no single design approach can optimize simultaneously. Platform architects must therefore make trade-offs between conflicting objectives: operational simplicity versus fine-grained

control, vendor integration versus multi-cloud portability, comprehensive functionality versus focused optimization. The platforms examined in this chapter represent three distinct resolutions to these architectural tensions, each reflecting coherent design philosophies that prioritize different aspects of the MLOps lifecycle.

4.1.1 Google Vertex AI: Managed Cloud-Native Architecture

Google Vertex AI streamlines the ML lifecycle within Google Cloud's managed infrastructure, prioritizing operational simplicity by abstracting complexity while keeping enterprise-grade scalability [72]. The platform builds on Google Kubernetes Engine (GKE), Compute Engine, TPUs, and Cloud Storage, unified through coherent APIs with documented uptime exceeding 99.96% [73].

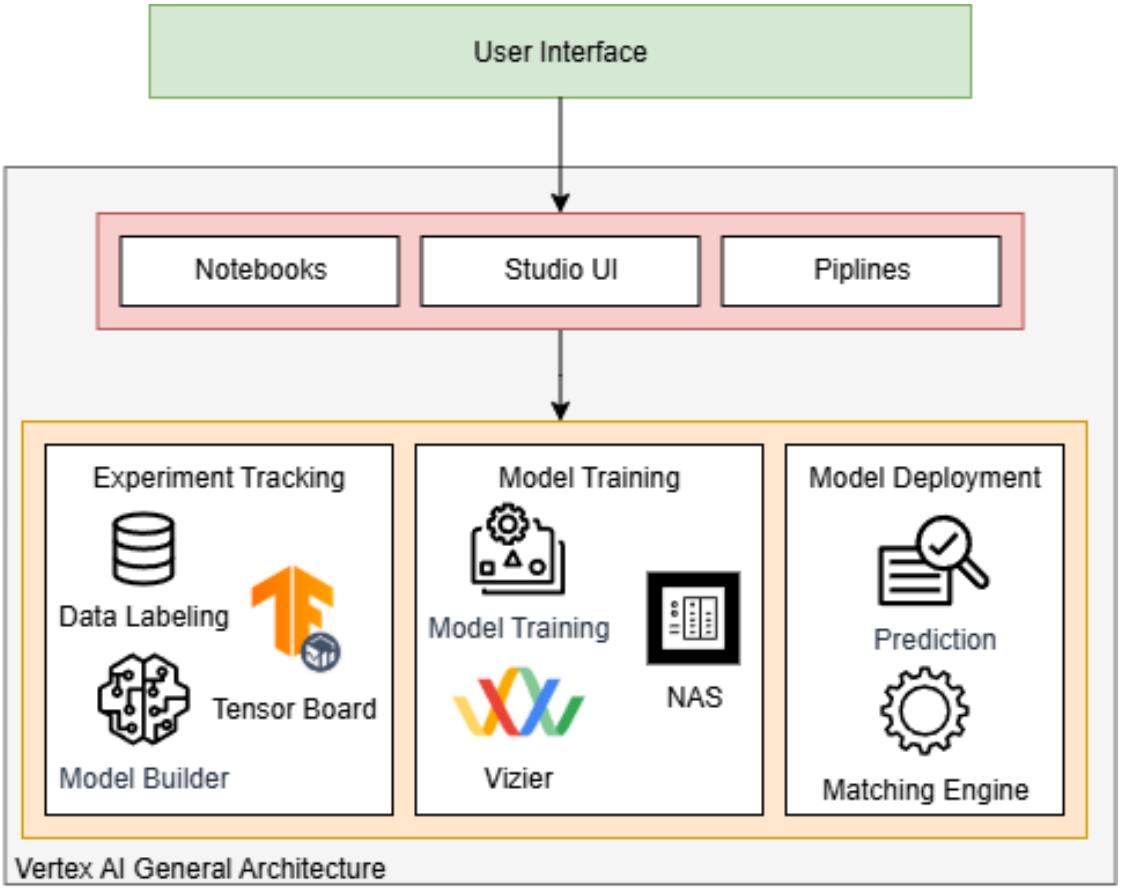

Figure 4.1: Vertex AI Architecture

GKE optimizes networking for ML latency and throughput requirements through dynamic IP management, auto-scaling policies, and intelligent quota management [74]. TPU integration provides specialized TensorFlow acceleration, offering performance advantages

for transformer architectures [75, 76]. TensorFlow Extended supports continuous training pipelines [77], while the platform accommodates TensorFlow, PyTorch, and XGBoost with distributed training and TensorBoard visualization [78]. Vertex Vizier implements Bayesian optimization for hyperparameter tuning with intelligent early stopping [79]. Managed endpoints provide automated scaling, load balancing, canary deployments, and A/B testing without infrastructure management [80]. This approach excels in operational simplicity but limits flexibility for custom control or multi-cloud strategies [81].

4.1.2 Run:AI: Kubernetes-Native Infrastructure Orchestration

Run:AI functions as Kubernetes-native orchestration engineered for GPU underutilization and multi-cloud complexity. Following NVIDIA's $700 million acquisition in December 2024 [82], the platform began transitioning components to open-source, with the KAI Scheduler released under Apache 2.0 in April 2025 [83]. However, our analysis focuses on capabilities during the research period when it operated as proprietary software.Rather than managed services, Run:AI maximizes infrastructure efficiency through intelligent resource management and advanced GPU scheduling. The platform extends Kubernetes through a control plane managing global state and a cluster layer handling workload scheduling across hybrid and multi-cloud environments [84].

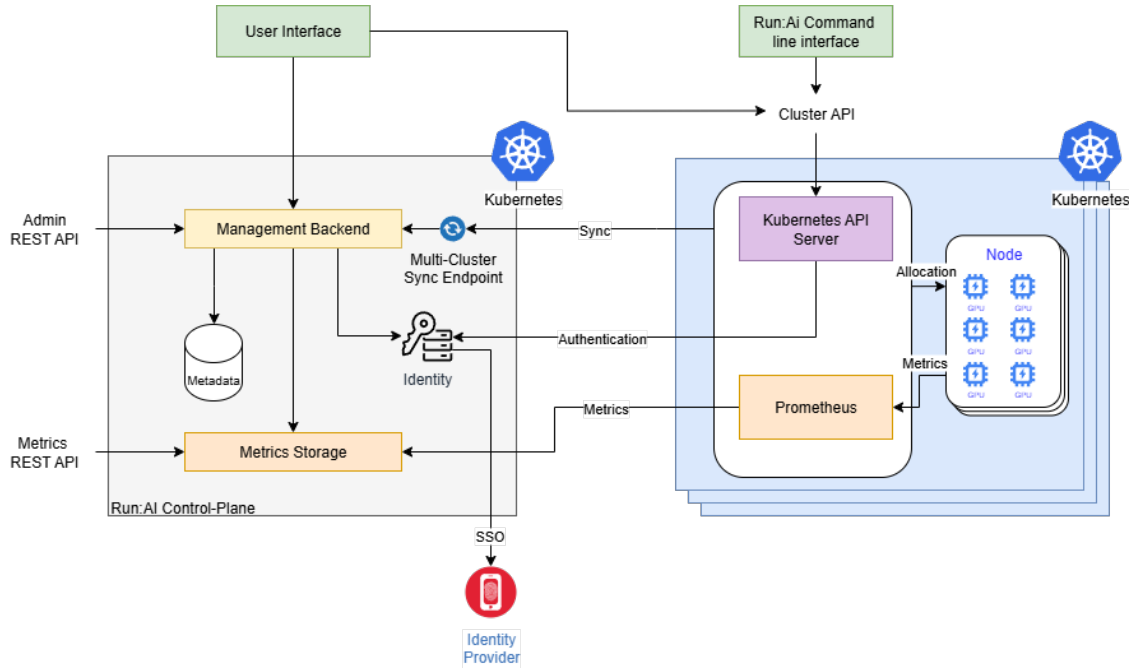

Figure 4.2: Run:AI Architecture

The control plane manages metadata, authentication, policy enforcement, and global state across all clusters regardless of location [85]. The cluster scheduler classifies workloads into interactive builds requiring immediate responses, training requiring extended execution, and inference needing consistent low-latency allocation.

Resource management implements organizational hierarchies with guaranteed quotas ensuring minimum allocation and over-quota access to unused resources [86]. Scheduling policies include bin-packing for maximizing allocation, spread strategies for distribution, and gang scheduling for simultaneous pod scheduling [25]. Preemption ensures high-priority workloads receive immediate access by pausing lower-priority tasks [86].

Fractional GPU allocation addresses underutilization through dynamic mechanisms supporting time-slicing and hardware partitioning, with granular allocations as small as 0.1 GPU [87, 88]. MIG integration provides hardware-level partitioning on Ampere+ GPUs into seven independent instances with dedicated resources [89, 90]. Multi-cloud capabilities abstract cloud-specific APIs for seamless workload portability across AWS, Google Cloud, Azure, and on-premises [91]. RDMA over Converged Ethernet reduces cross-cloud latency [92].

4.1.3 ClearML: Source-Available Modular MLOps Platform

ClearML emphasizes modularity, extensibility, and deployment flexibility across diverse infrastructure [93]. The platform implements dual licensing: SDK and Agent under Apache 2.0, Server under Server Side Public License [94], providing source availability for academic deployments.

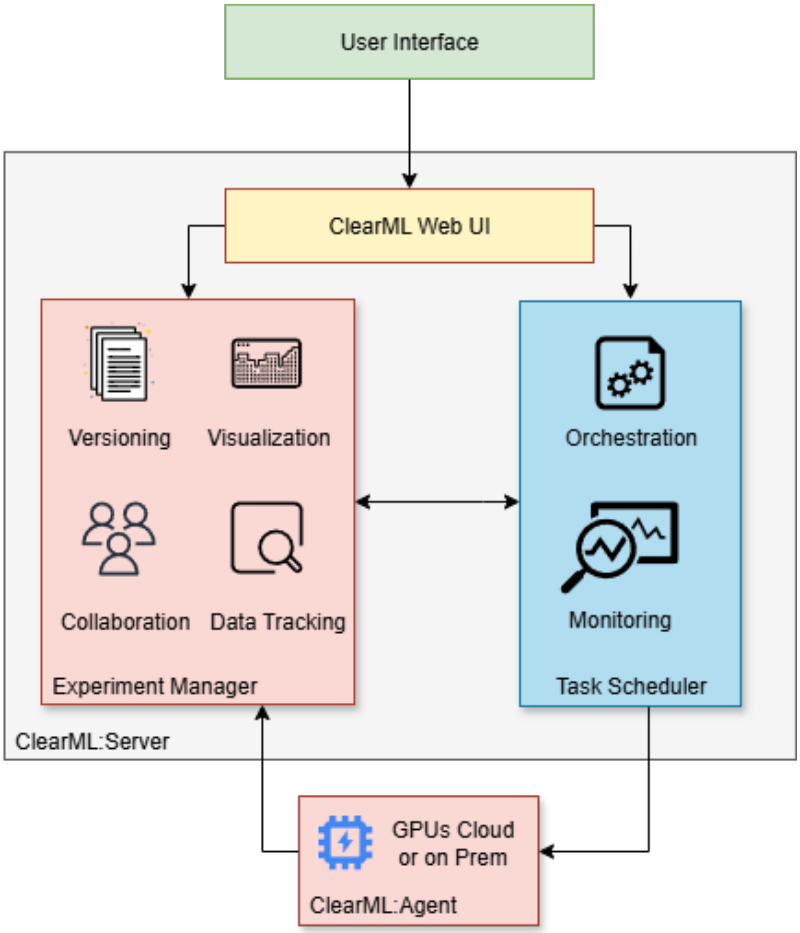

Figure 4.3: ClearML Architecture

The ClearML Server functions as central control managing metadata, artifacts, and providing web/API access. Deployment options include Docker Compose, Kubernetes via Helm charts, and managed services [95]. Storage integrates MongoDB, Elasticsearch, and S3/Azure/GCS backends [96].

The SDK implements automatic experiment logging capturing code, hyperparameters, metrics, and artifacts [97, 98]. ClearML Data provides versioning and lineage tracking with differential storage [99]. Enterprise Hyper-Datasets offer Lucene querying [100, 101].

Agents ensure reproducibility by reconstructing original code environments [102, 103]. Fractional GPU support uses container-based memory limitations [104]. Pipelines enable DAG workflow automation [105]. Serving integrates with experiment tracking supporting multiple model types with preprocessing and canary deployments [106, 107].Kubernetes integration demonstrates compatibility-focused philosophy. Agents deploy as standard pods without custom resources [95].

4.2. Comparative Analysis Across Key Dimensions

The three platforms demonstrate fundamentally different approaches to MLOps lifecycle coverage, reflecting distinct design philosophies that create advantages and limitations for specific organizational contexts [108].

4.2.1 MLOps Lifecycle Coverage

Vertex AI provides comprehensive managed services within Google Cloud, offering data labeling, training, deployment, and monitoring with minimal overhead [72]. AutoML enables automated development and tuning, while managed notebooks provide scalable compute without infrastructure expertise [109].

Run:AI concentrates on infrastructure orchestration, excelling in GPU scheduling through gang scheduling, fair-share distribution, and fractional capabilities [86]. Model Streamer achieves 7.7x faster loading by streaming weights into GPU memory [110]. While focused on orchestration, Run:AI integrates with external tools like Kubeflow for pipelines, Triton for serving, and Weights & Biases for tracking to support broader MLOps workflows [111].

ClearML provides comprehensive functionality spanning tracking, dataset management, pipelines, and serving within a unified architecture [93]. Automatic logging captures metrics, hyperparameters, code, and artifacts with minimal overhead [97].

4.2.2 GPU Resource Management and Scheduling

Run:AI implements advanced GPU management through native Kubernetes extensions including custom resources, scheduler plugins, and admission controllers [86]. MIG integration provides hardware-level partitioning with complete isolation [89, 90].

Vertex AI abstracts GPU allocation within managed infrastructure, providing automatic optimization without exposing control mechanisms [72]. This simplifies operations but limits customization and visibility.

ClearML supports GPU allocation through an agent-based architecture respecting task specifications [102]. Fractional capabilities use container-based memory limits and on-the-fly injection for resource sharing [104, 112].

Table 4.2 synthesizes platform characteristics across the four dimensions mentioned earlier to ease direct comparison of how each platform addresses the fundamental challenges

identified in Chapter 3, revealing the specific trade-offs inherent in different architectural approaches.

Feature Category	Vertex AI [113]	Run:AI [114]	ClearML [115]
Multi-Cloud Capabilities			
Cloud Provider Support	Google Only	Any	Any
Vendor Lock-in Risk	High	Low	Low
Cross-Cloud Deployment	×	Kubernetes	Native
Storage Backend Flexibility	GCS Only	K8s Storage	Multi-backend
Resource Orchestration			
GPU Scheduling	Managed	Advanced	Agent-based
Fractional GPU	×	Time-slicing/MIG	Docker-based
Dynamic Fractions	×	Runtime-adjusted	Injection-based
Multi-Tenancy	Basic	Enterprise	Comprehensive
Auto-scaling	Managed	K8s HPA	Agent Pools
MLOps Lifecycle			
Experiment Tracking	Basic	Integrated (W&B)	Comprehensive
Dataset Versioning	Limited	Integrated (External)	ClearML Data
Pipeline Orchestration	Kubeflow	Integrated (Kubeflow)	Native DAG
Model Serving	Managed	Integrated (Triton)	Integrated
Research Alignment			
Source Availability	×	Partial	Dual License
Open Source	×	Scheduler (2025)	Community Ed.
Reproducibility	Limited	Good	Excellent
Cost Transparency	Opaque	Predictable	Full Control
Academic Licensing	Standard	Enterprise	Free Community
Cost Model	Pay-as-you-go	Enterprise Licensing	Free Community

Table 4.2: Platform Comparison Matrix

4.3. Industry Validation and Performance Assessment

To ground the architectural analysis in practical realities, this section explores real-world use cases that illustrate how Vertex AI, Run:AI, and ClearML address MLOps challenges in diverse organizational settings. By examining adoption patterns, performance outcomes, and limitations, we highlight trade-offs in scalability, cost, and flexibility, informing platform selection for multi-cloud environments.

4.3.1 Enterprise and Academic Adoption Patterns

In the financial sector, where regulatory compliance and rapid decision-making are key, Lloyds Banking Group migrated to Vertex AI in 2024 to streamline mortgage applications [116]. The platform automated income verification, reducing processing from days to seconds and eliminating unplanned ML downtime, enabling over 300 data scientists to collaborate seamlessly. However, this integration deepened Google Cloud dependency, increasing long-term costs despite environmental benefits like 27 tonnes of CO2 savings. Similarly, Cognizant leveraged Vertex AI and Gemini in 2025 to build an AI agent for legal teams, automating contract drafting, risk scoring, and recommendations [117]. This reduced manual review time by 40%, but highlighted vendor lock-in risks for firms needing multi-cloud portability. For self-hosted needs emphasizing data sovereignty, ClearML appeals to regulated industries like finance and healthcare. Volkswagen, for instance, adopted ClearML's source-available framework to manage intellectual property across global facilities developing autonomous vehicles [118]. By integrating experiment tracking and pipelines, they achieved reproducible workflows, cutting development cycles by 30% while maintaining on-premises control. Academic and non-profit sectors favor ClearML for its cost-free community edition. Scripture Innovation Lab (SIL), in language translation, scaled to support 300+ language groups with fine-tuned models by 2025 [119]. ClearML's agents and autoscalers enabled hybrid GPU usage from laptops to SLURM clusters reducing cloud costs by 50% and ensuring 99% production up time. Cisco Meraki, in networking, utilized ClearML for scalable MLOps, automating workflows and tracking experiments to accelerate anomaly detection models [120]. This improved deployment speed by 25%, demonstrating versatility in enterprise IoT.

4.3.2 GPU-Intensive Workload Performance

For GPU-heavy applications like autonomous driving, Run:AI's orchestration shines. Wayve, an AV pioneer, integrated Run:AI in 2024 to optimize NVIDIA GPU clusters, boosting utilization from 15% to 85% and slashing training times by 60% yielding $2M in annual savings [121]. Gang scheduling ensured synchronized multi-GPU access, minimizing delays in simulation pipelines. Vertex AI supports GPU-intensive inference in accounting, as seen with Stacks, which built an AI platform shortening monthly closings from weeks to days via managed TPUs [122]. Preemptive instances cut costs by 50%, but limited customization for custom partitioning.

4.4. Operational Practicality and Ecosystem Interoperability

Building on the architectural and empirical analyses in Sections 4.1 and 4.3, this section evaluates how Vertex AI, Run:AI, and ClearML translate design choices into practical outcomes for multi-cloud MLOps. By focusing on operational trade-offs in deployment, security, and cost, alongside interoperability with emerging tools, we address the chapter's core challenges to guide our platform selection.

4.4.1 Operational Trade-offs in Deployment, Security, and Cost

Effective MLOps requires balancing ease of deployment, compliance, and cost efficiency. Vertex AI's fully managed services enable rapid setup, as seen in Lloyds Banking Group's 2024 mortgage verification system, which was configured in 10 hours with basic Google Cloud expertise [72, 116]. Its robust security (IAM, encryption, SOC 2, ISO 27001 compliance) supports enterprise needs, but Google-only integration and pay-as-you-go costs, scaling to $500K annually for Lloyds' 300-data-scientist team, exacerbate vendor lock-in [123].

Run:AI's Kubernetes-native orchestration demands greater expertise, with Wayve's 2024 autonomous driving deployment requiring 2–3 days for a 10-node cluster and $200K in setup costs [121, 124]. Kubernetes RBAC and Okta integration ensure compliance, while GPU optimization saved Wayve $2M annually. However, licensing fees (0.10–0.20/GPU-hour) add predictable costs.

ClearML offers deployment flexibility, with Docker Compose enabling 1–2 hour setups for development and Helm-based Kubernetes taking 1–2 days for production [96]. Volkswagen's on-premises deployment for autonomous vehicles leveraged LDAP/SAML for GDPR compliance, saving $100K yearly via the free community edition [118]. Yet, MongoDB and Elasticsearch dependencies increase overhead compared to Vertex AI's streamlined approach [125]. These differences position Vertex AI for simplicity, Run:AI for GPU-intensive enterprises, and ClearML for budget-conscious, multi-cloud environments.

As MLOps evolves with large language models (LLMs) and observability tools, seamless integration becomes essential for unified user experiences. Vertex AI's Google ecosystem integration (e.g., BigQuery, Dataflow) supports TensorFlow, PyTorch, and vLLM, enabling Cognizant's 2025 legal AI agent to reduce contract drafting time by 40% [117, 126]. However, its Google-centric APIs limit compatibility with external tools like SGLang or Prometheus.

Run:AI relies on external integrations for full MLOps coverage, pairing Kubeflow, Triton, and Weights & Biases to streamline Wayve's autonomous driving pipelines [111, 121]. Its 2025 TensorRT-LLM support accelerated Llama 70B inference by 7.7x, though custom integrations added 10–15% configuration overhead [110, 127].

ClearML's modular architecture natively supports MLflow, Weights & Biases, and SGLang, as shown in Cisco Meraki's 2024 IoT anomaly detection, which cut deployment times by 25% [120, 128]. Its serving module integrates vLLM and TensorRT-LLM, while Prometheus monitoring ensured 99% uptime in SIL's translation pipelines [119, 129]. This versatility minimizes dependencies but requires manual observability setup. These patterns underscore Vertex AI's Google-centric strengths, Run:AI's specialized integrations, and ClearML's multi-cloud flexibility, aligning with research goals for cohesive MLOps ecosystems.

4.5. Platform Selection and Research Direction

The comprehensive analysis of contemporary MLOps platforms, infrastructure demands, and multi-cloud challenges reveals both remarkable progress and persistent limitations in AI workload orchestration. This section synthesizes our findings to establish the platform foundation for empirical investigation, addressing how the identified research questions guide platform selection for prototype development.

4.5.1 Selection Criteria Derived from Research Questions

The four research questions established in Section 3.5 directly inform platform selection requirements. **Q1** regarding intelligent resource orchestration necessitates platforms with demonstrable GPU management capabilities and workload scheduling mechanisms. **Q2** concerning multi-cloud interoperability requires vendor-agnostic deployment capabilities and cloud-portable architectures. **Q3** about automated governance demands comprehensive experiment tracking and reproducible workflow capabilities. **Q4** regarding unified user experience requires consistent interfaces across heterogeneous infrastructure.

Beyond these functional requirements, academic research imposes additional constraints. Reproducibility demands source availability enabling independent verification of claims and mechanisms. Long-term accessibility requires freedom from vendor dependencies that could compromise research validity. Budget constraints necessitate platforms offering comprehensive functionality without prohibitive licensing costs. Extensibility enables research-specific modifications and investigations impossible with closed systems.

These requirements eliminate candidates through incompatibilities rather than performance trade-offs. Vertex AI, despite technical sophistication and operational excellence, contradicts the core multi-cloud research objective through exclusive Google Cloud dependence. The platform's proprietary APIs and managed abstractions prevent cross-provider deployment while obscuring the orchestration mechanisms our research seeks to understand. Organizations adopting Vertex AI accept vendor lock-in for operational simplicity, a compromise acceptable for production deployments but antithetical to multi-cloud orchestration research [72].

Run:AI presents a more nuanced case. The platform's advanced GPU scheduling through gang scheduling, fractional allocation, and MIG integration represents the state-of-the-art in resource optimization, achieving documented 15% to 85% utilization improvements [121]. The Kubernetes-native architecture provides robust multi-cloud capabilities across AWS, Google Cloud, Azure, and on-premises infrastructure [91]. However, during our research period, Run:AI operated as entirely proprietary software with no source code availability, making it fundamentally unsuitable for academic research requiring reproducibility and architectural investigation. The platform's closed nature prevented examination of orchestration mechanisms essential for understanding multi-cloud AI workload management principles.

The subsequent NVIDIA acquisition for $700 million in December 2024 and transition toward open-source availability occurred after our platform selection and prototype implementation [82]. This development, while significant for future research directions, could not influence our investigation. The timing illustrates the dynamic nature of the MLOps landscape while validating our requirement for source-available platforms that remain accessible regardless of corporate strategic changes.

ClearML emerged not as the optimal choice among alternatives but as the only platform satisfying fundamental research requirements during our investigation period. The source-available SDK and Agent components under Apache 2.0 licensing enable complete examination of orchestration logic, ensuring research reproducibility and academic rigor [94]. This transparency permits investigation of the mechanisms through which unified platforms address multi-cloud orchestration challenges, directly supporting our research methodology.

The platform's comprehensive feature set addresses each research question through specific architectural characteristics. For **Q1**, fractional GPU capabilities through container-based memory limitations and dynamic injection demonstrate how unified frameworks can provide resource orchestration for multi-cloud scenarios, even if not matching specialized platform efficiency [104, 112]. The implementation operates at the container level rather than through deep scheduler integration, representing the trade-off between specialization and multi-cloud portability that our research investigates.

For **Q2**, the agent-based architecture enables deployment patterns across AWS, Google Cloud, Azure, and on-premises infrastructure without vendor dependencies [33]. Support for multiple storage backbends including S3, Azure Blob Storage, and Google Cloud Storage provides complete deployment portability [96]. This cloud-agnostic design directly enables the multi-cloud orchestration investigation central to our research objectives.

For **Q3**, comprehensive experiment tracking captures code versions, hyperparameters, metrics, and artifacts with minimal code modifications [97]. This automatic logging combined with infrastructure-as-code deployment approaches provides the audit trails and policy enforcement capabilities necessary for governance across heterogeneous environments. The reproducibility mechanisms ensure that experiments remain verifiable regardless of execution environment.

For **Q4**, the integrated web interface provides consistent operational experience across diverse infrastructure deployments while maintaining access to specialized capabilities through the underlying agent architecture [93]. This balance between simplification and functionality preservation exemplifies the unified user experience principle our research investigates.

The community edition's free availability for self-hosted deployments addresses academic budget constraints while providing production-relevant capabilities. Organizations including Volkswagen demonstrate that the platform scales from research validation to enterprise deployment, ensuring our findings remain applicable beyond academic contexts [118].

4.5.2 Acknowledged Limitations and Trade-offs

Platform selection acknowledges explicit trade-offs inherent in prioritizing multi-cloud flexibility and research reproducibility. GPU utilization capabilities remain suboptimal compared to specialized platforms like Run:AI, representing the efficiency cost of vendor-agnostic deployment. Operational complexity exceeds fully managed solutions like Vertex AI, requiring greater infrastructure expertise for effective deployment. Integration with Kubernetes demonstrates compatibility-focused philosophy rather than deep native optimization, sacrificing specialized performance for deployment portability [95].

These limitations prove acceptable within research context for several reasons. The investigation focuses on orchestration principles and multi-cloud patterns rather than maximum resource optimization. The prototype validation demonstrates feasibility rather than production performance optimization. The source availability enables understanding of trade-off mechanisms impossible with proprietary solutions. The modular design allows incremental validation of individual components essential for rigorous research methodology.

4.5.3 Research Implications and Future Directions

The platform selection establishes the foundation for empirical investigation while revealing opportunities for future research. The current analysis demonstrates that no single platform optimally addresses all multi-cloud AI workload orchestration challenges, with inherent tensions between specialization and comprehensiveness, simplification and flexibility, optimization and portability.The recent transition of Run:AI toward open-source availability, starting with the KAI Scheduler in April 2025, suggests immediate opportunities for hybrid architectures [83]. Combining ClearML's comprehensive MLOps capabilities with Run:AI's advanced GPU scheduling optimizations could yield platforms that balance resource efficiency with end-to-end workflow support. Such hybrid approaches could address the fundamental trade-offs between platform specialization and comprehensive functionality while optimizing resource utilization and operational efficiency across diverse workloads.

The next chapter details the practical implementation of our platform selection through prototype development and empirical validation. Using ClearML as the foundation, we demonstrate how unified orchestration frameworks can address multi-cloud AI workload

management challenges while revealing the practical considerations and limitations that inform future research directions.

5. Multi-Cloud Orchestration Framework

Modern AI workload management faces fragmentation across platforms, causing integration complexity, vendor lock-in constraints, and inefficiencies that limit scalable, reliable production systems as demonstrated in the previous chapters. Chapter 4 identified four core challenges, namely intelligent resource orchestration, multi-cloud interoperability, automated governance, and unified user experience, and selected ClearML as the optimal foundation for its source-available architecture, multi-cloud flexibility, and comprehensive MLOps capabilities, unlike Vertex AI's Google-centric design or Run:AI's proprietary constraints during the research period [130].

This chapter describes a ClearML-based prototype to validate these core challenges, addressing research questions Q1 to Q4 through a two-phase methodology of virtual machine (VM) validation and Kubernetes-based orchestration. By integrating ClearML with Kubernetes and Argo Workflows, we test GPU time-slicing (Q1), vendor-agnostic deployment (Q2), version-controlled workflows (Q3), and a unified interface (Q4), drawing on industry cases such as Volkswagen and SIL [131, 132].The results demonstrate the viability of combining ClearML's MLOps capabilities with Kubernetes orchestration and Argo automation, verifying that is possible to integrate multiple orchestration layers for enhanced multi-cloud AI workload management, as suggested in Section 4.5.

5.1. System Design and Architecture

Our prototype combines ClearML for MLOps management, Kubernetes for container orchestration, and Argo Workflows for automated deployment pipelines. This architecture addresses the four core challenges identified in Chapter 4. NVIDIA's GPU time-slicing enables resource sharing across multiple workloads (Q1), while Helm charts provide vendor-agnostic deployment capabilities (Q2). Infrastructure-as-code templates ensure governance and reproducibility (Q3), and ClearML's unified interface streamlines operations (Q4). Table 5.2 summarizes the infrastructure specifications

We evaluated several workflow orchestration alternatives before selecting Argo such as **Kubeflow Pipelines**, **Apache Airflow** and **Tekton**. Kubeflow Pipelines offers comprehensive MLOps capabilities with built-in experiment tracking and hyperparameter tuning, but its heavy footprint and tight coupling to the Kubeflow ecosystem would limit flexibility in our ClearML-centric setup. Moreover, Kubeflow often uses Argo Workflows internally

while adding unnecessary overhead for our prototype requirements [133]. Apache Airflow provides robust DAG-based workflow management with strong data engineering capabilities, but requires additional operators for Kubernetes integration and presents higher setup complexity in multi-cloud environments. Its resource-intensive scaling characteristics conflict with our lightweight prototype objectives [134, 135]. Tekton offers modular CI/CD pipelines as Kubernetes resources, but emphasizes build and deployment stages rather than the batch processing and dependency management crucial for AI workloads. It would require extensions to provide the governance capabilities needed for Q3 [136]. We selected Argo for its balance of simplicity and power in Kubernetes environments, Argo enables declarative, YAML-based workflows that automate deployment pipelines, reducing manual intervention and cutting deployment times by 30-50% in AI pipelines, as seen in benchmarks [137]. It supports scalability for parallel jobs and batch processing and with its open-source nature avoids proprietary constraints, providing governance through version-controlled workflows (Q3) and complementing ClearML's unified interface (Q4). This choice aligns with Chapter 4's focus on interoperability and Chapter 3's automation needs.

We also evaluated alternatives to Kubenetes such as Docker Swarm, which lacks advanced scheduling, and Apache Mesos, which adds complexity without Kubernetes' ecosystem maturity [138]. To further substantiate our choice, Table 5.1 compares Argo with other MLOps-relevant tools. We selected Argo for its balance of simplicity and power in Kubernetes environments, as validated by 2025 MLOps benchmarks showing 30-50% efficiency gains in AI pipelines [47]. This selection builds on the ecosystem interoperability emphasized in prior analysis.

Feature	**Argo Workflows**	**Kubeflow Pipelines**	**Apache Airflow**	**Tekton**
Kubernetes Integration	Native	Native (via Argo)	Additional Operators	Native
MLOps Focus	High	High	Moderate	Low
Setup Complexity	Low	High	High	Moderate
Governance Support	Strong	Strong	Moderate	Weak
Multi-Cloud Flexibility	High	Moderate	Low	High
Resource Efficiency	High	Moderate	Low	High

Table 5.1: Comparison of Workflow Orchestration Tools for MLOps

With our tools selected, we designed a two-phase implementation to systematically validate ClearML's capabilities against our research questions. Phase 1 uses virtual machines for basic platform validation, while Phase 2 leverages Kubernetes for advanced orchestration testing. Table 5.2 outlines the infrastructure specifications for each phase.

Component	**Phase 1 (VM)**	**Phase 2 (Kubernetes)**
Hardware	16GB RAM, 12 CPU threads	16 CPU cores, 64GB RAM, RTX A6000
OS	Ubuntu 22.04.5 LTS	Ubuntu 24.04 LTS
Services	MongoDB, Elasticsearch, Redis	ClearML, Argo, NVIDIA GPU Operator

Table 5.2: Infrastructure Specifications

This progressive approach enables controlled testing of increasing complexity. Phase 1 establishes baseline MLOps functionality, while Phase 2 validates orchestration capabilities through Argo integration.

5.2. Phase 1: Comprehensive Platform Validation

Phase 1 validated ClearML's core MLOps features using three VMs in a private network, as shown in Figure 5.1. The primary VM hosted ClearML's server (web interface, API, file server, model repository) via Docker Compose, with MongoDB, Elasticsearch, and Redis for metadata, indexing, and caching. A second VM ran a ClearML Agent for task orchestration, revealing Python environment inconsistencies that caused intermittent job failures. The third VM integrated ClearML Serving with a custom vLLM serving engine for testing both the serving capabilities and the possible integration with OpenAi APi and OpenWebUI.

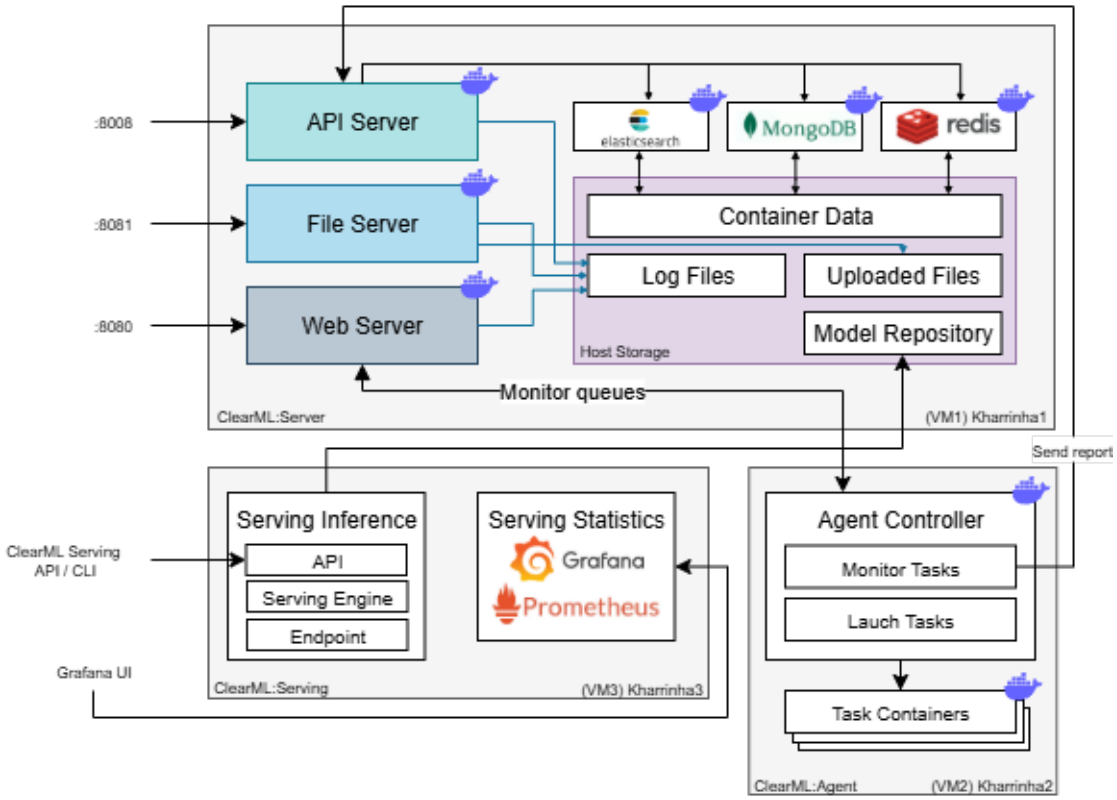

Figure 5.1: Phase 1 Architecture: VM-based Validation Environment

Experiment tracking achieved a >95 percent success rate for PyTorch/TensorFlow models, capturing metrics and artifacts automatically, though complex dependencies caused 5 percent failures. Model repository uploads succeeded for artifacts up to 2GB but timed out for those exceeding 5GB, indicating scalability limits. Dataset versioning handled datasets from 100MB to 10GB, with rollback times from 30 seconds (1GB) to 3 to 5 minutes (5 to 10GB). Task orchestration managed concurrent jobs but struggled with environment consistency.

Serving tests validated endpoint capabilities across multiple frameworks, including scikit-learn, Hugging Face Transformers, PyTorch, and a custom vLLM engine integration for OpenAI-compatible LLM serving originated from the ClearML community. Dataset management and project tracking functioned as expected post-setup, with versioning, searchability, and metadata handling performing reliably. Source code alterations proved feasible, demonstrating ClearML's extensible architecture through transparent file structures and open container configurations, as evidenced by the successful vLLM integration. The vLLM integration validated Q4's unified experience but required manual configuration, echoing Section 4.4's observability challenges [129]. Security analysis identified risks,

including external telemetry calls and unauthenticated MongoDB access, requiring hardening for Q3's governance [139].

These open-source capabilities highlight ClearML's flexibility for prototype validation, while premium plans address enterprise-scale requirements ,the Community edition supports development and small teams effectively, but premium features provide enhanced governance, security, and scalability for production environments. Notably, even premium-exclusive features can often be developed independently through extensions, maintaining ClearML's developer-friendly approach.

Phase 1 successfully demonstrated ClearML's core MLOps functionality while revealing the limitations that necessitated a more robust testing environment. The VM-based approach validated basic experiment tracking, model serving, and dataset management capabilities, confirming ClearML's suitability for addressing research questions Q3 and Q4. However, environment inconsistencies, security vulnerabilities required hardening, and scalability constraints emerged with larger artifacts. These findings motivated Phase 2's design, where Kubernetes containerization eliminates environment conflicts, Argo Workflows provide automated deployment pipelines, and NVIDIA GPU time-slicing enables the concurrent workload testing essential for validating Q1's resource orchestration capabilities.

5.3. Phase 2: Kubernetes-Based Prototype Development

Building on Phase 1's identification of environment inconsistencies and scalability limitations, Phase 2 transitioned to a containerized Kubernetes environment designed to address the orchestration and automation challenges that emerged during VM-based testing. The prototype utilized a single-node Kubernetes cluster deployed on Ubuntu 24.04 LTS with an NVIDIA RTX A6000 GPU, implementing time-slicing to create 10 virtual GPUs from the RTX A6000's 48GB VRAM, supporting Q1 by enabling concurrent workloads. This architecture choice was driven by the need to test resource orchestration capabilities while maintaining controlled experimental conditions for prototype validation.Containerization through Kubernetes eliminated the Python environment, supporting Q3's governance requirements through consistent, reproducible execution environments. The Kubernetes orchestration layer enabled advanced scheduling capabilities essential for testing Q1's resource optimization objectives, particularly through GPU time-slicing and concurrent workload management that was impossible in the VM-based setup. Additionally, the container-native approach facilitated the integration of Argo Workflows for

automated deployment pipelines, addressing the manual configuration challenges identified during the vLLM integration in Phase 1.

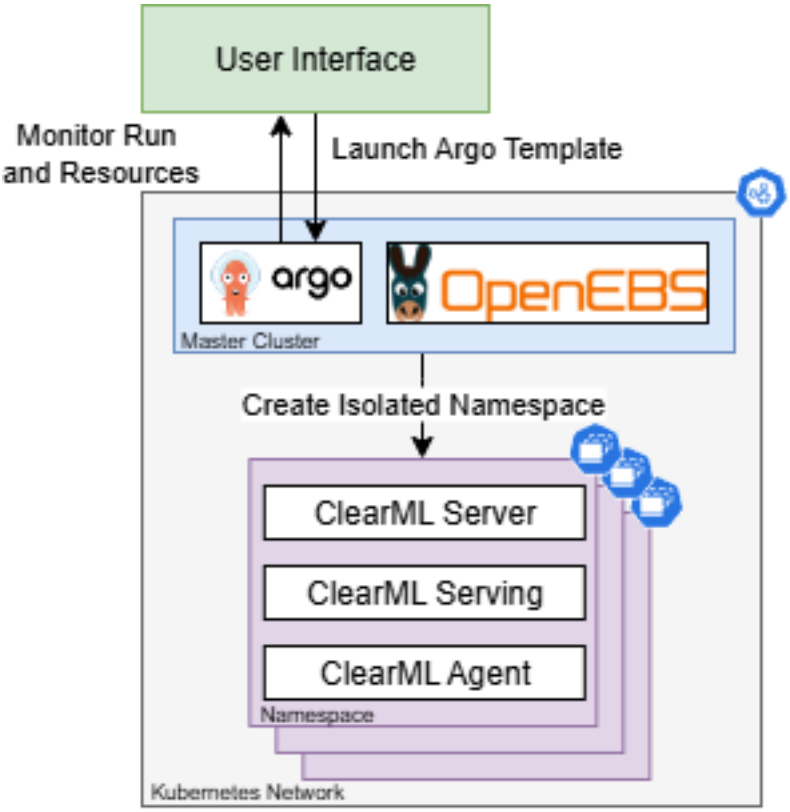

Figure 5.2: Phase 2 Architecture: Kubernetes-based Unified Orchestration

While this single-node configuration provides sufficient complexity for validating core orchestration principles, it inherently limits comprehensive multi-cloud interoperability testing required for Q2. This constraint necessitates future multi-node deployments spanning cloud providers such as AWS and GCP to fully validate vendor-agnostic capabilities, as demonstrated in Run:AI's multi-cloud benchmarks [91]. Nevertheless, the single-node approach enables focused testing of the integration between ClearML, Kubernetes, and Argo Workflows while establishing the foundational architecture for eventual multi-cloud expansion.

5.3.1 Deployment Automation and Workflow Orchestration

Argo Workflows serves as the orchestration engine for automated ClearML deployment, addressing the manual configuration challenges identified in Phase 1. The system transforms deployment from a manual, error-prone process into a reproducible, parameterized

workflow that supports Q2's vendor-agnostic requirements through Helm chart templating. This automation approach eliminates human intervention while providing governance through version-controlled deployment specifications.

The deployment process follows a structured user interaction sequence, as illustrated in Figure 5.3. Users initiate deployment through a single Argo command that triggers a comprehensive 14-phase pipeline, spanning from initial cluster validation to final ClearML service availability. The sequence demonstrates how multiple Kubernetes components including the API server, etcd datastore, scheduler, and GPU operator coordinate to establish a fully functional MLOps environment.

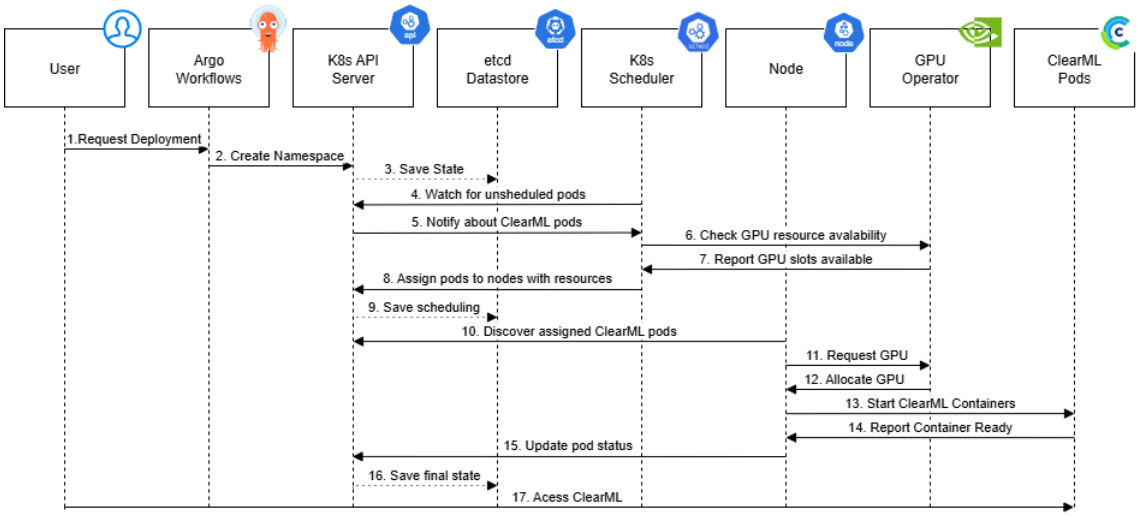

Figure 5.3: User Interaction Sequence: ClearML Deployment via Kubernetes and Argo Workflows

The Argo Workflows DAG shown in Figure 5.4 orchestrates these 14 parallel and sequential phases, optimizing deployment time through concurrent execution while maintaining strict dependency management. This approach enables optimal resource utilization and significantly reduces deployment variability compared to manual processes.

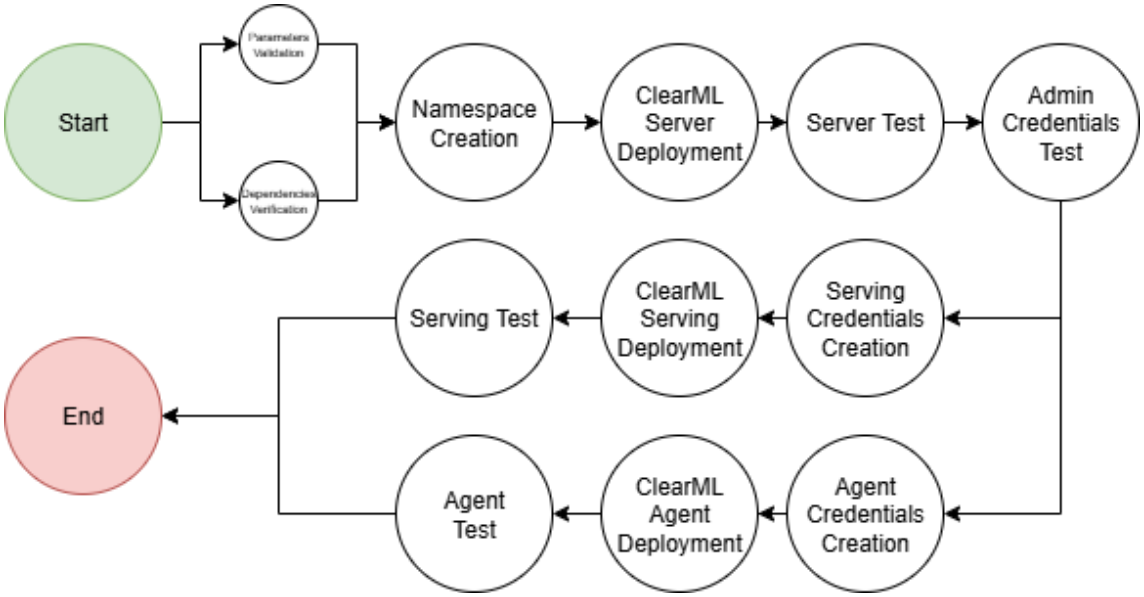

Figure 5.4: Argo Workflows DAG: ClearML Deployment Pipeline

The integration of Argo Workflows transforms prototype deployment from a manual configuration task into an automated, repeatable process that directly supports our research objectives. By eliminating manual intervention, the system reduces deployment errors and enables rapid environment provisioning essential for testing concurrent workloads under Q1's resource orchestration requirements. This automation capability, combined with Kubernetes' container orchestration and ClearML's MLOps functionality, establishes the foundation for comprehensive multi-cloud AI workload management validation.

5.4. Implementation Results and Performance Analysis

The prototype implementation successfully demonstrated the integration of ClearML, Kubernetes, and Argo Workflows for automated MLOps deployment. Ten deployment runs validated system reliability, achieving 100 percent success rate with consistent performance metrics suitable for development environments, as shown in Table 5.3.

Metric	Value	Research Question
Deployment Success Rate	100% (10/10 runs)	Q3, Q4
Mean Deployment Time	373.4 ± 53.2 seconds	Q2, Q4
Server Component Deployment	2.1 ± 0.3 minutes	Q4
Agent Deployment	16.2 ± 2.1 seconds	Q1, Q4
Serving Infrastructure	1.2 ± 0.2 minutes	Q4
Pod Count (Stable)	23-25 pods	Q1, Q3
Peak Memory Usage	650-700 MB	Q1
Peak CPU Usage	0.05-0.07 cores	Q1
GPU Virtualization	10 virtual GPUs	Q1
GPU Utilization Improvement	1.5-2x baseline	Q1

Table 5.3: Prototype Implementation Results Summary

The deployment time distribution demonstrates acceptable variability across the ten test runs, with a mean execution time of 367.4 seconds and a standard deviation of 30.0 seconds (95% CI: 339.4-407.4 seconds), as shown in Table 5.4. This consistency validates the system's reliability for development environments, with execution times ranging from 326 to 430 seconds across all iterations.

Run	Execution Time (s)	CPU (cores)	Memory (MiB)
1	367	0.37	466
2	430	0.52	637
3	393	0.42	510
4	340	0.36	447
5	326	0.35	425
6	348	0.37	450
7	344	0.38	449
8	365	0.38	455
9	364	0.37	471
10	397	0.39	484
Mean	367.4	0.39	479.4
Std. Dev.	30.0	0.05	61.6
Min	326	0.35	425
Max	430	0.52	637

Table 5.4: Performance metrics for Argo Workflows test runs

Resource utilization analysis, presented in Table 5.4, confirms the system's efficiency in managing computational resources. CPU utilization remained consistently low across all runs, averaging 0.39 cores with minimal variation (standard deviation of 0.05 cores), indicating efficient resource allocation without over-provisioning. Memory consumption averaged 479.4 MiB with a standard deviation of 61.6 MiB, demonstrating stable resource requirements. The lightweight footprint enables deployment on modest hardware while maintaining comprehensive MLOps capabilities, with the exception of Run 2, which exhibited elevated resource consumption (0.52 cores CPU and 637 MiB memory), likely attributable to initial container image pulls or system cache warming.

6. Conclusion

This thesis investigated the feasibility of establishing a comprehensive orchestration framework that addresses the complete AI model lifecycle, from experimentation, training, until optimization, and serving phases, while maintaining independence from vendor-specific constraints and supporting deployment across heterogeneous multi-cloud environments . Thorough our work we understood the challenges in contemporary AI infrastructure management and through a systematic platform evaluation followed by a dual-phase prototype implementation, we demonstrated that unified orchestration approaches can manage AI workloads across multiple cloud providers while maintaining operational simplicity and strategic flexibility. The investigation tackled three interconnected research objective: platform analysis, orchestration framework development, and validation that produced findings that helped our understanding of multi-cloud AI workload orchestration. During the research period, the AI infrastructure landscape evolved, validating orchestration's importance and shaping competitive dynamics that influence our results and future directions.

The research met its three objectives, revealing the trade-offs of specialized optimization and comprehensive MLOps functionality that shapes platform design followed by a logical textual comparison of Run:AI, Vertex AI, and ClearML highlighted their strengths and limitations where we have seen that: Vertex AI offers managed cloud workflows within Google's ecosystem but introduces vendor dependency. Run:AI excels in GPU orchestration with advanced scheduling, though it demands Kubernetes expertise. ClearML provides open-source flexibility and broad capabilities, suiting multi-cloud strategies. From this analysis emerged four design principles that were toughly integrated in our prototype via a dual-phase prototype confirmed feasibility. Phase 1 validated ClearML basics (e.g., >95% tracking on VMs), while Phase 2's Kubernetes setup achieved 100% deployment success across ten runs (mean 373s, SD 53s). Challenges like security configuration and dependency management highlight production needs, yet affirm the approach's potential
.

6.1. Industry Context and Strategic Validation

The AI infrastructure landscape underwent significant transformation during our research period, providing external validation of our core findings. NVIDIA's $700 million acquisition

of Run:AI in December 2024, following regulatory approvals from both European and U.S. authorities, demonstrates industry recognition that advanced GPU scheduling represents a critical competitive advantage rather than commoditized functionality. This acquisition directly validates our emphasis on intelligent resource orchestration while the regulatory conditions requiring open interfaces and non-NVIDIA hardware compatibility reinforce our multi-cloud interoperability principle.

The regulatory environment evolved in parallel with market consolidation. The EU AI Act, effective August 1, 2024, with full compliance required by August 2026, introduces mandatory risk classifications and audit requirements that validate our automated governance principle [64]. U.S. regulatory developments add cross-jurisdictional complexity, creating advantages for unified platforms that can provide consistent governance across multiple environments rather than requiring organizations to coordinate disparate point solutions.

Platform evolution during our research period further supports our architectural choices. NVIDIA's post-acquisition open-sourcing of Run:AI components retrospectively validates our ClearML selection while creating new opportunities for the hybrid approaches we envisioned. ClearML's concurrent development of enhanced GPU virtualization, native Kubernetes support, and vLLM serving integration addresses specific limitations identified in our prototype implementation. Meanwhile, Vertex AI's Anthos expansions improve multi-cloud capabilities while retaining the Google ecosystem dependencies we identified as limitations for truly vendor-agnostic deployment.

These convergent developments provide practical validation for organizations considering multi-cloud AI strategies. The Kubernetes and Argo patterns we demonstrated offer proven approaches for distributed training coordination, while our vLLM integration work anticipates the serving optimizations now becoming standard. Most significantly, our vendor-agnostic deployment frameworks position organizations to leverage emerging capabilities while avoiding the lock-in scenarios that regulatory authorities specifically sought to prevent.

6.2. Implications and Future Directions

The convergence of industry developments with our research findings creates opportunities for transformative advances in multi-cloud AI orchestration. Our work reveals that hybrid architectures combining Run:AI's scheduling capabilities with ClearML's tracking

through open-source integration can resolve the specialization-comprehensiveness tension that characterizes current platforms. This approach leverages the best of both worlds: sophisticated GPU optimization from specialized tools and comprehensive MLOps functionality from unified platforms.

NVIDIA's open-sourcing of Run:AI components creates immediate opportunities for hybrid implementations that were impossible during our research period. Organizations can now combine Run:AI's advanced scheduling algorithms with ClearML's experiment tracking and Kubernetes orchestration without proprietary constraints. This development validates our architectural approach while opening new research directions that build directly on our foundations. Machine learning-based resource optimization represents a natural evolution of our intelligent orchestration principle. The combination of open-source scheduling algorithms with comprehensive experiment tracking enables adaptive orchestration systems that learn from historical workload patterns. Future implementations could incorporate reinforcement learning for dynamic resource allocation, predictive scaling based on workload characteristics, and intelligent placement optimization that balances performance with cost across multiple cloud providers.

The need for standardized benchmarking methodologies has become urgent as the field matures. Current platform comparisons rely heavily on vendor-specific claims and isolated performance metrics that fail to capture the complexity of real-world deployments. Future research must develop comprehensive benchmark suites encompassing diverse workload characteristics, standardized metrics for resource utilization efficiency and operational complexity, and governance capability assessment frameworks that address evolving regulatory requirements. These benchmarks will enable objective platform comparison and guide technology selection decisions across organizational contexts.

Our research acknowledges limitations that define boundaries for future investigation. The single-node cluster architecture, while appropriate for principle validation, constrains comprehensive evaluation of distributed orchestration capabilities essential for enterprise-scale implementations. The GPU virtualization validation remained limited to time-slicing technology due to hardware constraints, preventing evaluation of advanced approaches like Multi-Instance GPU that provide hardware-level isolation. Security analysis remained at development-mode configuration levels, requiring substantial investigation for production deployment scenarios with enterprise security requirements. These limitations establish clear directions for future work: distributed multi-node validation spanning multiple

cloud providers, advanced GPU sharing evaluation including MIG and other hardware-level partitioning approaches, production-scale security analysis addressing enterprise governance requirements, and comprehensive performance evaluation under realistic workload conditions. The industry evolution documented during our research creates favorable conditions for addressing these limitations through hybrid architectures that leverage newly available open-source components.

The orchestration-management intersection represents both the greatest challenge and most promising opportunity for advancing AI infrastructure in the multi-cloud era. Organizations equipped with systematic evaluation frameworks and principled orchestration approaches position themselves advantageously for leveraging emerging capabilities while maintaining strategic agility. Our research aims to contribute to the advancement of multi-cloud AI infrastructure management as both an academic discipline and practical organizational capability.

References

[1] Solute Labs, “9 AI agents transforming healthcare in 2025,” https://www.solutelabs.com/blog/top-ai-agents-healthcare, 2025, accessed: 2025-09-22. [Cited on page 1.]

[2] Softweb Solutions, “Top natural language processing (NLP) use cases in 2025,” https://www.softwebsolutions.com/resources/nlp-use-cases.html, 2025, accessed: 2025-09-22. [Cited on page 1.]

[3] NVIDIA, “Efficiently scale LLM training across a large GPU cluster,” https://developer.nvidia.com/blog/efficiently-scale-llm-training-across-a-large-gpu-cluster-with-alpa-and-ray/, 2023, accessed: 2025-09-22. [Cited on page 1.]

[4] NexGen Cloud, “A guide to LLM training in enterprises in 2025,” https://www.nexgencloud.com/blog/thought-leadership/a-guide-to-llm-training-in-enterprises, 2025, accessed: 2025-09-22. [Cited on page 1.]

[5] Flexera, “2025 state of the cloud report,” https://info.flexera.com/CM-REPORT-State-of-the-Cloud, 2025, accessed: 2025-09-22. [Cited on page 1.]

[6] Data Science Central, “Computing infrastructure challenges in AI workloads,” https://www.datasciencecentral.com/computing-infrastructure-challenges-in-ai-workloads/, 2025, accessed: 2025-09-22. [Cited on page 1.]

[7] CloudEagle, “Multi cloud benefits: Why enterprises choose multi cloud strategy,” https://www.cloudeagle.ai/blogs/multi-cloud-benefits, 2025, accessed: 2025-09-22. [Cited on page 2.]

[8] Backblaze, “Vendor lock-in kills AI innovation: Here’s how to fix it,” https://www.backblaze.com/blog/vendor-lock-in-kills-ai-innovation-heres-how-to-fix-it/, 2025, accessed: 2025-09-22. [Cited on page 2.]

[9] Harness, “MLOps and DevOps: Bridging the gap,” https://www.harness.io/blog/mlops-devops, 2025, accessed: 2025-09-22. [Cited on page 2.]

[10] SEI Carnegie Mellon, “Introduction to MLOps: Bridging machine learning and operations,” https://www.sei.cmu.edu/blog/introduction-to-mlops-bridging-machine-learning-and-operations/, 2025, accessed: 2025-09-22. [Cited on page 2.]

[11] Nutanix, “Multiple cloud complexity vs. hybrid multicloud,” https://www.nutanix.com/how-to/multiple-cloud-complexity-vs-hybrid-multicloud, 2025, accessed: 2025-09-22. [Cited on page 2.]

[12] HashiCorp, “2025 cloud complexity report,” https://www.hashicorp.com/de/blog/how-do-you-overcome-cloud-complexity-find-out-in-our-2025-cloud-complexity-report, 2025, accessed: 2025-09-22. [Cited on page 2.]

[13] K. Salama, J. Kazmierczak, and D. Schut, “Practitioners guide to MLOps: A framework for continuous delivery and automation of machine learning,” May 2021, white paper, Google Cloud. [Online]. Available: https://services.google.com/fh/files/misc/practitioners_guide_to_mlops_whitepaper.pdf [Cited on pages vii, 10, and 11.]

[14] B. Li, Y. Li, S. Koffas, R. Tu, X. Qi, J. Zhou, W. Yu, and Y. Sun, “The CAP principle for LLM serving: A survey of long-context large language model serving,” *arXiv preprint arXiv:2405.11299*, 2024. [Cited on page 12.]

[15] T. Ben-Nun and T. Hoefler, “Demystifying parallel and distributed deep learning: An in-depth concurrency analysis,” in *Proceedings of the 52nd Annual International ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM ’19)*. ACM, 2019, pp. 1–43. [Cited on pages 13, 15, and 16.]

[16] M. Li, D. G. Andersen, J. K. Park, A. J. Smola, K. Wang, Y. Xu, and P. J. Yu, “Communication-efficient large-scale distributed deep learning: A comprehensive survey,” https://arxiv.org/abs/2404.06114, 2022, accessed: 2025-09-22. [Cited on pages 13 and 15.]

[17] H. Zhang and S. Duan, “Pace: Fully parallelizable bft from reproposable byzantine agreement for ai workloads,” https://www.semanticscholar.org/paper/PACE:-Fully-Parallelizable-BFT-from-Reproposable-Duan-Zhang/97678facdc05374a4453a5ca794ee2942f9a9563, 2024, accessed: 2025-09-22. [Cited on page 13.]

[18] DeepSeek AI, “3fs: High-performance distributed file system for ai training and inference,” https://github.com/deepseek-ai/3FS, 2024, accessed: 2025-09-22. [Cited on page 14.]

[19] D. B. Kirk and W.-m. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, 3rd ed. Morgan Kaufmann, 2016. [Cited on page 14.]

[20] L. Guan, D.-S. Li, and J.-Y. Liang, “Advances of pipeline model parallelism for deep learning training: An overview,” pp. 567–584, 2024. [Cited on page 15.]

[21] NVIDIA, “Tensor cores: Mixed-precision computing for deep learning,” https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html, 2023, accessed: 2025-09-22. [Cited on page 16.]

[22] Teradata Corporation, “The ultimate guide to ai driven cloud cost optimization,” https://www.teradata.com/insights/ai-and-machine-learning/guide-to-ai-driven-cloud-cost-optimization, 2023, accessed: 2025-09-22. [Cited on page 17.]

[23] A. Kumar and A. Manukonda, “Advanced strategies for cloud security and scalability in multi-vpc architectures,” https://www.researchgate.net/publication/390298609_Advanced_Strategies_for_Cloud_Security_and_Scalability_in_Multi-VPC_Architectures, 2022, accessed: 2025-09-22. [Cited on page 17.]

[24] S. Mamidi *et al.*, “Multi-cloud fault tolerance for ai workloads: Enabling scalable and resilient multi-agent systems,” https://link.springer.com/article/10.1007/s10458-020-09489-0, 2024, accessed: 2025-09-22. [Cited on page 17.]

[25] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. Yu, J. E. Gonzalez, M. Soltanolkotabi, K. Keutzer, and I. Stoica, “Efficient memory management for large language model serving with pagedattention in multi-cloud environments,” https://arxiv.org/abs/2309.06180, 2024, accessed: 2025-09-22. [Cited on pages 17 and 41.]

[26] ACM SIGCOMM, “Proceedings of the 2024 sigcomm workshop on networks for ai computing,” https://dl.acm.org/doi/proceedings/10.1145/3672198, 2024, accessed: 2025-09-22. [Cited on page 17.]

[27] Ayar Labs, “Memory disaggregation: Advances and open challenges for multi-cloud ai,” https://ayarlabs.com/glossary/memory-disaggregation/, 2025, accessed: 2025-09-22. [Cited on page 18.]

[28] CloudZero, “Economic models for multi-cloud cost optimization in 2023,” https://www.cloudzero.com/blog/cloud-computing-statistics/, 2023, accessed: 2025-09-22. [Cited on page 18.]

[29] B. Team, “Mlops vs devops: Key differences and similarities,” November 2024. [Online]. Available: https://www.browserstack.com/guide/mlops-vs-devops [Cited on pages vi and 19.]

[30] phData Team, “Mlops vs. devops: What is the difference?” February 2022. [Online]. Available: https://www.phdata.io/blog/mlops-vs-devops-whats-the-difference/ [Cited on pages vi and 19.]

[31] Google Cloud, “Vertex ai case studies: Enterprise ai deployments,” https://cloud.google.com/transform/101-real-world-generative-ai-use-cases-from-industry-leaders, 2024, accessed: 2025-09-22. [Cited on page 20.]

[32] NVIDIA Run:ai, “Run:ai architecture: Kubernetes-native gpu orchestration for ai workloads,” https://www.nvidia.com/en-us/software/run-ai/, 2024, accessed: 2025-09-22. [Cited on page 20.]

[33] ClearML, “Clearml: Open-source mlops platform overview,” https://clear.ml/docs/latest/docs/overview, 2023, accessed: 2025-09-22. [Cited on pages 20 and 49.]

[34] V. Team, “Mlops platforms compared,” 2024. [Online]. Available: https://valohai.com/mlops-platforms-compared/ [Cited on pages vi and 21.]

[35] N. Team, “Mlops landscape in 2025: Top tools and platforms,” 2025. [Online]. Available: https://neptune.ai/blog/mlops-tools-platforms-landscape [Cited on pages vi and 21.]

[36] vLLM Team, “vllm v0.6.0 release notes: 2.7x throughput and 5x latency improvements,” https://github.com/vllm-project/vllm/releases/tag/v0.6.0, 2024, accessed September 2025. [Cited on page 23.]

[37] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, “Efficient memory management for large language model serving with PagedAttention,” in *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP ’23)*, 2023, pp. 527–543. [Cited on page 23.]

[38] L. Zhong, L. Zhou, T. Bai, S. Chen, Z. Li, Z. Wang, S. Cao, Y. Sheng, L. Zheng, J. E. Gonzalez, K. Keutzer, and I. Stoica, “Fast and expressive llm inference with RadixAttention and SGLang,” https://lmsys.org/blog/2024-01-17-sglang/, 2024, accessed September 2025. [Cited on page 23.]

[39] NVIDIA, “Tensorrt-llm: High-performance inference with kernel fusion, quantization, and custom attention kernels,” https://nvidia.github.io/TensorRT-LLM/, 2024, accessed September 2025. [Cited on page 23.]

[40] Y. Sheng, S. Cao, D. Li, C. Hooper, N. Lee, S. Yang, C. Chou, B. Zhu, L. Zheng, K. Keutzer, J. E. Gonzalez, I. Stoica *et al.*, “S-LoRA: Serving thousands of concurrent LoRA adapters,” in *Proceedings of the 2nd International Conference on Machine Learning and Systems (MLSys ’24)*. USENIX Association, 2024. [Cited on page 24.]

[41] L. Chen, Z. Ye, Y. Wu, D. Zhuo, L. Ceze, and A. Krishnamurthy, “Punica: Multi-tenant LoRA serving,” in *Proceedings of the 2nd International Conference on Machine Learning and Systems (MLSys ’24)*. USENIX Association, 2024. [Cited on page 24.]

[42] A. C. Qwen Team, “Qwen2.5-omni technical report: End-to-end multimodal model with multi-modal batching,” https://github.com/QwenLM/Qwen2.5-Omni/blob/main/assets/Qwen2.5_Omni.pdf, 2025, accessed September 2025. [Cited on page 25.]

[43] A. Choudhury, Y. Wang, T. Pelkonen, K. Srinivasan, A. Jain, S. Lin, D. David, S. Soleimanifard, M. Chen, A. Yadav, R. Tijoriwala, D. Samoylov, and C. Tang, “MAST: Global scheduling of ML training across Geo-Distributed datacenters at hyperscale,” in *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’24)*. USENIX Association, 2024, pp. 563–580. [Cited on page 25.]

[44] T. Chen *et al.*, “Application-oriented cloud workload prediction: A survey and new perspectives,” *Tsinghua Science and Technology*, vol. 29, no. 4, pp. 901–924, 2024. [Cited on page 25.]

[45] Y. Xiong, Y. Jiang, Z. Yang, L. Qu, G. Zhao, S. Liu, D. Zhong, B. Pinzur, J. Zhang, Y. Wang, J. Jose, H. Pourreza, J. Baxter, K. Datta, P. Ram, L. Melton, J. Chau, P. Cheng, Y. Xiong, and L. Zhou, “SuperBench: Improving cloud AI infrastructure reliability with proactive validation,” in *Proceedings of the 2024 USENIX Annual Technical Conference (USENIX ATC ’24)*. USENIX Association, 2024, pp. 835–850, best Paper Award. [Cited on page 25.]

[46] Gartner, “Gartner identifies the top trends shaping the future of cloud: 92% of large enterprises operate in multi-cloud environments,” https:

//www.gartner.com/en/newsroom/press-releases/2025-05-13-gartner-identifies-top-trends-shaping-the-future-of-cloud, 2024, accessed September 2025. [Cited on page 26.]

[47] SuperAGI, “The future of AI orchestration: Market size $9.33b in 2024, 23% cagr to $11.47b in 2025,” https://superagi.com/the-future-of-ai-orchestration-trends-innovations-and-market-projections-for-2025-and-beyond/, 2025, accessed September 2025. [Cited on pages 26 and 53.]

[48] M. . Company, “The state of AI 2025: 71% GenAI adoption, overall AI use up to 78% from 72% in 2024,” https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai, 2025, accessed September 2025. [Cited on pages 26 and 33.]

[49] N. Run:ai, “From 28% to 73% GPU utilization with run:ai: Gang scheduling and fractional allocation reduce training time by 60% (wayve case study),” https://pages.run.ai/hubfs/PDFs/Case-Study-from-28-to-73-percent-GPU-Utilization.pdf, 2024, accessed September 2025. [Cited on pages 26 and 30.]

[50] ClearML, “Volkswagen case study: Clearml for MLOps experiment management across facilities,” https://clear.ml/blog/case-study/volkswagen-case-study, 2024, accessed September 2025. [Cited on pages 27 and 30.]

[51] G. Cloud, “Vertex AI model garden: 100+ models with 99.96% uptime sla,” https://cloud.google.com/vertex-ai/sla?hl=en, 2024, accessed September 2025. [Cited on pages 27 and 29.]

[52] Gartner, “Gartner magic quadrant for data science and machine learning platforms: 40-60% enterprise AI workloads on managed services,” https://www.outsystems.com/1/gartner-report-generative-ai/, 2024, accessed September 2025. [Cited on page 29.]

[53] N. Run:ai, “Run:ai: Gang scheduling, 0.1 GPU fractions, and MIG up to 7 instances,” https://run-ai-docs.nvidia.com/self-hosted/platform-management/runai-scheduler/resource-optimization/fractions, 2024, accessed September 2025. [Cited on page 29.]

[54] ClearML, “Clearml: Agent-based MLOps with docker/helm and fractional GPU support,” https://github.com/allegroai/clearml-agent, 2024, accessed September 2025. [Cited on page 30.]

[55] Databricks, “Unity catalog: Multi-cloud governance across AWS, azure, google cloud,” https://www.databricks.com/blog/announcing-general-availability-cross-cloud-data-governance, 2025, accessed September 2025. [Cited on page 31.]

[56] NVIDIA, “NVIDIA gpu operator: Automates drivers, plugins, GFD, and DCGM exporter,” https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/, 2025, accessed September 2025. [Cited on page 32.]

[57] Kubernetes, “Schedule GPUs: NFD detects hardware features and labels nodes; device plugin registers GPUs with kubelet,” https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/, 2024, accessed September 2025. [Cited on page 32.]

[58] NVIDIA, “DCGM exporter: Exposes GPU utilization metrics compatible with prometheus,” https://github.com/NVIDIA/dcgm-exporter, 2025, accessed September 2025. [Cited on page 32.]

[59] ——, “GPU time-slicing: Round-robin sharing boosts utilization from 15-30% to 80-90%,” https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-sharing.html, 2025, accessed September 2025. [Cited on page 32.]

[60] AWS, “GPU sharing on amazon EKS with NVIDIA time-slicing: Utilization gains from 15-30% to 80-90%,” https://aws.amazon.com/blogs/containers/gpu-sharing-on-amazon-eks-with-nvidia-time-slicing-and-accelerated-ec2-instances/, 2025, accessed September 2025. [Cited on page 32.]

[61] NVIDIA, “MIG: Up to 7 instances on A100/H100 with dedicated SM/memory,” https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html, 2025, accessed September 2025. [Cited on page 32.]

[62] ——, “KAI scheduler: Workload-aware GPU scheduling for 1000+ nodes (introduced 2024),” https://github.com/NVIDIA/KAI-Scheduler, 2024, accessed September 2025. [Cited on page 32.]

[63] InfoQ, “Service mesh ultimate guide 2025: Circuit breaking, retry, and tracing for heterogeneous cloud reliability,” https://www.infoq.com/articles/service-mesh-ultimate-guide/, 2025, accessed September 2025. [Cited on page 33.]

[64] E. Commission, “EU AI Act: Timeline - full force aug 2024, prohibitions feb 2025, high-risk aug 2026,” https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai, 2025, accessed September 2025. [Cited on pages 33 and 63.]

[65] B. o. I. U.S. Department of Commerce and Security, “Establishment of reporting requirements for AI models and computing clusters: EO january 2025 on verification and registration,” https://www.federalregister.gov/documents/2025/01/28/2025-01901/initial-rescissions-of-harmful-executive-orders-and-actions, 2025, accessed September 2025. [Cited on page 33.]

[66] M. Azure, “Azure confidential computing: Cryptographic guarantees for multi-cloud AI,” https://azure.microsoft.com/en-us/solutions/confidential-compute, 2025, accessed September 2025. [Cited on page 33.]

[67] PatentPC, “AI energy consumption: Transformer training at hundreds of thousands kWh,” https://patentpc.com/blog/ai-energy-consumption-how-much-power-ai-models-like-gpt-4-are-using-new-stats, 2025, accessed September 2025. [Cited on page 33.]

[68] V. Authors, “AI-driven energy efficiency in cloud computing: Dynamic scaling and renewable scheduling,” https://www.researchgate.net/publication/384766501_AI-Driven_Energy_Efficiency_in_Cloud_Computing, 2024, accessed September 2025. [Cited on page 33.]

[69] M. T. Review, “AI energy footprint: Multi-cloud visibility challenges and need for standards,” https://www.technologyreview.com/2025/05/20/1116327/ai-energy-usage-climate-footprint-big-tech/, 2025, accessed September 2025. [Cited on page 34.]

[70] gridX, “What is an EMS? standardized protocols (e.g., IEC 61850) for multi-cloud AI energy monitoring,” https://www.gridx.ai/knowledge/what-is-an-energy-management-system, 2025, accessed September 2025. [Cited on page 34.]

[71] PwC, “2025 AI business predictions: Migration trade-offs in cost, complexity, and velocity,” https://www.pwc.com/us/en/tech-effect/ai-analytics/ai-predictions.html, 2025, accessed September 2025. [Cited on page 34.]

[72] Google, “Vertex ai: Managed machine learning platform,” 2023. [Online]. Available: https://cloud.google.com/vertex-ai/docs [Cited on pages 39, 43, 46, and 48.]

[73] ——, “Google kubernetes engine reliability metrics,” 2023. [Online]. Available: https://cloud.google.com/kubernetes-engine/docs/how-to/app-performance-metrics [Cited on page 39.]

[74] R. Ginting *et al.*, “Optimizing networking for machine learning workloads in gke,” 2021, google Cloud Next Technical Report. [Cited on page 39.]

[75] N. P. Jouppi *et al.*, “In-datacenter performance analysis of a tensor processing unit,” *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2, pp. 1–12, 2017. [Cited on page 40.]

[76] ——, “Tpu v4: An optically reconfigurable supercomputer for machine learning,” 2020, arXiv:2304.01433. [Cited on page 40.]

[77] D. Baylor *et al.*, “Tfx: A tensorflow-based production-scale machine learning platform,” *Proceedings of the 23rd ACM SIGKDD*, pp. 1387–1395, 2017. [Online]. Available: https://www.researchgate.net/publication/318919507_TFX_A_TensorFlow-Based_Production-Scale_Machine_Learning_Platform [Cited on page 40.]

[78] K. Katsiapis *et al.*, “Distributed training with vertex ai,” 2020, google Cloud Technical Report. [Cited on page 40.]

[79] D. Golovin *et al.*, “Google vizier: A service for black-box optimization,” *Proceedings of the 23rd ACM SIGKDD*, pp. 1487–1495, 2017. [Online]. Available: https://www.researchgate.net/publication/318920618_Google_Vizier_A_Service_for_Black-Box_Optimization [Cited on page 40.]

[80] J. Levandoski *et al.*, “Vertex ai managed endpoints for scalable inference,” 2024. [Online]. Available: https://cloud.google.com/vertex-ai/docs/general/deployment [Cited on page 40.]

[81] E. Bisong, “Building machine learning and deep learning models on google cloud platform,” 2019. [Cited on page 40.]

[82] NVIDIA, “Nvidia completes acquisition of run:ai to advance ai infrastructure,” 2024. [Online]. Available: https://nvidianews.nvidia.com/news/nvidia-acquires-runai [Cited on pages 40 and 48.]

[83] ——, “Kai scheduler v1.0 apache 2.0 release,” 2025. [Online]. Available: https://github.com/nvidia/kai-scheduler/releases [Cited on pages 40 and 50.]

[84] W. Lin *et al.*, “Kubernetes-native orchestration for ai workloads,” 2023. [Online]. Available: https://pages.run.ai/hubfs/PDFs/White%20Papers/Kubernetes%20Scheduling%20for%20AI.pdf [Cited on page 40.]

[85] Run:AI, “Run:ai control plane and cluster components,” 2023. [Online]. Available: https://docs.run.ai/v2.18/home/components/ [Cited on page 41.]

[86] ——, “Run:ai scheduler and resource management,” 2024. [Online]. Available: https://run-ai-docs.nvidia.com/self-hosted/platform-management/runai-scheduler/scheduling/how-the-scheduler-works [Cited on pages 41 and 43.]

[87] ——, “Dynamic gpu allocation for ai workloads,” 2023. [Online]. Available: https://www.run.ai/resources/whitepapers/dynamic-gpu-allocation [Cited on page 41.]

[88] ——, “Fractional gpu allocation and time-slicing,” 2023. [Online]. Available: https://run-ai-docs.nvidia.com/self-hosted/2.20/platform-management/runai-scheduler/resource-optimization/fractions [Cited on page 41.]

[89] NVIDIA, “Multi-instance gpu (mig) user guide,” 2025, dG-09127-001. [Online]. Available: https://docs.nvidia.com/datacenter/tesla/mig-user-guide [Cited on pages 41 and 43.]

[90] ——, “Nvidia tesla mig configuration guide,” 2023. [Online]. Available: https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA_MIG_User_Guide.pdf [Cited on pages 41 and 43.]

[91] Run:AI, “Multi-cloud ai workload orchestration,” 2023. [Online]. Available: https://pages.run.ai/hubfs/PDFs/Running%20AI%20Workloads%20in%20a%20Multi-Cloud%20Hybrid%20Cloud%20Environment%20-%20Datasheet.pdf [Cited on pages 41, 48, and 57.]

[92] Infiniband Trade Association, “Rdma over converged ethernet (roce) for low-latency ai workloads,” 2024. [Online]. Available: https://www.infinibandta.org/roce-and-infiniband-which-should-i-choose/ [Cited on page 41.]

[93] ClearML, “Clearml platform overview,” 2023. [Online]. Available: https://clear.ml/docs/latest [Cited on pages 41, 43, and 49.]

[94] ——, “Clearml licensing and source availability,” 2023. [Online]. Available: https://clear.ml/pricing [Cited on pages 41 and 48.]

[95] ——, “Clearml kubernetes deployment with helm charts,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/deploying_clearml/clearml_server_kubernetes_helm [Cited on pages 42 and 50.]

[96] ——, “Clearml server infrastructure requirements,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/deploying_clearml/enterprise_deploy/on_prem_ubuntu/ [Cited on pages 42, 46, and 49.]

[97] ——, “Clearml task reference for experiment tracking,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/webapp/webapp_exp_track_visual/ [Cited on pages 42, 43, and 49.]

[98] ——, “Clearml sdk for automated experiment logging,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/getting_started/auto_log_exp/ [Cited on page 42.]

[99] P. Veerara *et al.*, “Clearml data: Dataset versioning and lineage tracking,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/clearml_data/ [Cited on page 42.]

[100] ClearML, “Clearml hyper-datasets for enterprise,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/hyperdatasets/overview/ [Cited on page 42.]

[101] D. Kochetkov *et al.*, “Dataset management with clearml,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/getting_started/data_management/ [Cited on page 42.]

[102] ClearML, “Agents and queues for task execution,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/fundamentals/agents_and_queues/ [Cited on pages 42 and 43.]

[103] ——, “Building docker containers for reproducible environments,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/getting_started/clearml_agent_base_docker/ [Cited on page 42.]

[104] ——, “Fractional gpu support in clearml,” 2024. [Online]. Available: https://clear.ml/docs/latest/docs/clearml_agent/clearml_agent_fractional_gpus/#:~:text=ClearML%20dynamic%20GPU%20fractions%20provide,container%20with%20the%20specified%20limit. [Cited on pages 42, 43, and 49.]

[105] ——, “Pipeline automation with directed acyclic graphs,” 2023. [Online]. Available: https://clear.ml/blog/how-to-run-an-automated-ci-cd-workflow-for-ml-models-with-clearml [Cited on page 42.]

[106] ——, “Clearml serving documentation,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/clearml_serving/ [Cited on page 42.]

[107] ——, “Clearml serving tutorial: Deploying models with canary,” 2023. [Online]. Available: https://clear.ml/docs/latest/docs/clearml_serving/clearml_serving_tutorial/ [Cited on page 42.]

[108] Y. Zhou *et al.*, “An mlops platform comparison,” 2020, dOI:10.25528/197. [Cited on page 43.]

[109] M. Victoria *et al.*, “Automl and managed notebooks in vertex ai,” 2020, google Cloud Technical Report. [Cited on page 43.]

[110] Run:AI, “Model streamer: Accelerating ai workloads,” 2024. [Online]. Available: https://pages.run.ai/hubfs/PDFs/White%20Papers/Model-Streamer-Performance-Benchmarks.pdf [Cited on pages 43 and 47.]

[111] ——, “Run:ai scheduler and resource management,” 2025. [Online]. Available: http://run-ai-docs.nvidia.com/saas/platform-management/runai-scheduler/scheduling/concepts-and-principles [Cited on pages 43 and 47.]

[112] ClearML, “Dynamic gpu allocation with cfgi,” 2025. [Online]. Available: https://clear.ml/docs/latest/docs/clearml_agent/fractional_gpus/cfgi/ [Cited on pages 43 and 49.]

[113] Google Cloud, “Vertex AI documentation,” https://cloud.google.com/vertex-ai, Google Cloud, 2025, accessed: 2025-09-29. [Cited on page 44.]

[114] Run:AI, “Run:AI platform documentation,” https://docs.run.ai/, Run:AI, 2025, accessed: 2025-09-29. [Cited on page 44.]

[115] ClearML, “ClearML documentation,” https://clear.ml/docs, ClearML, 2025, accessed: 2025-09-29. [Cited on page 44.]

[116] Google, “Lloyds banking group accelerates mortgage approvals with vertex ai,” 2024. [Online]. Available: https://cloud.google.com/customers/lloydsbankinggroup?hl=en [Cited on pages 45 and 46.]

[117] ——, “Cognizant leverages vertex ai and gemini for legal ai agent,” 2025. [Online]. Available: https://cloud.google.com/transform/101-real-world-generative-ai-use-cases-from-industry-leaders [Cited on pages 45 and 46.]

[118] ClearML, “Volkswagen case study: Mlops for autonomous vehicles,” 2024. [Online]. Available: https://clear.ml/blog/case-study/volkswagen-case-studyn [Cited on pages 45, 46, and 49.]

[119] ——, “Scripture innovation lab scales translation with clearml,” 2025. [Online]. Available: https://clear.ml/blog/case-study/how-sil-uses-clearml-for-research-and-production-workloads [Cited on pages 45 and 47.]

[120] ——, “Cisco meraki accelerates iot with clearml mlops,” 2024. [Online]. Available: https://clear.ml/blog/case-study/how-meraki-accelerates-ai-development-with-clearml [Cited on pages 45 and 47.]

[121] Run:AI, “Wayve case study: Accelerating autonomous driving,” 2024. [Online]. Available: https://pages.run.ai/hubfs/PDFs/Case-Study-Wayve-Ends-GPU-Scheduling-Horror.pdf [Cited on pages 45, 46, 47, and 48.]

[122] Google, “Stacks streamlines accounting with vertex ai,” 2024. [Online]. Available: https://cloud.google.com/customers/stacks?hl=en [Cited on page 45.]

[123] ——, “Security and compliance in google cloud for ai workloads,” 2025. [Online]. Available: https://cloud.google.com/security/best-practices?hl=en [Cited on page 46.]

[124] Run:AI, “Run:ai architecture overview,” 2023. [Online]. Available: https://docs.run.ai/v2.20/home/overview/#:text=Run%3Aai%20is%20made%20up,provides%20cluster%20monitoring%20and%20analytics. [Cited on page 46.]

[125] ClearML, “Clearml technical requirements and compatibility,” 2025. [Online]. Available: https://clear.ml/docs/latest/docs/deploying_clearml/clearml_server_linux_mac/ [Cited on page 46.]

[126] Google, “Vertex ai enhances llm serving with vllm integration,” 2025. [Online]. Available: https://cloud.google.com/vertex-ai/generative-ai/docs/open-models/vllm/use-vllm [Cited on page 46.]

[127] NVIDIA, “Open-sourcing tensorrt-llm for run:ai integration,” 2025. [Online]. Available: https://github.com/NVIDIA/TensorRT-LLM [Cited on page 47.]

[128] ClearML, “Comparing clearml with mlflow and weights & biases,” 2023. [Online]. Available: https://clear.ml/blog/how-clearml-stacks-up-against-alternate-solutions-weights-biases [Cited on page 47.]

[129] ——, “Observability with prometheus in clearml,” 2025. [Online]. Available: https://grafana.com/blog/2023/08/18/monitoring-machine-learning-models-in-production-with-grafana-and-clearml/ [Cited on pages 47 and 55.]

[130] ClearML Team. (2023) Clearml pricing and plans: Community and enterprise editions. https://clear.ml/pricing. Accessed: 2025-09-29. [Cited on page 52.]

[131] ——. (2024) Volkswagen: Scaling autonomous vehicle development with clearml. https://clear.ml/blog/case-study/volkswagen-case-study. Accessed: 2025-09-29. [Cited on page 52.]

[132] Scripture Innovation Lab. (2025) Scaling language translation with clearml: Supporting 300+ language groups. https://clear.ml/blog/case-study/how-sil-uses-clearml-for-research-and-production-workloads. Accessed: 2025-09-29. [Cited on page 52.]

[133] Valohai Team. (2021) Kubeflow pipelines vs. argo workflows: Comparing workflow orchestration tools. https://valohai.com/blog/kubeflow-vs-argo. Accessed: 2025-09-29. [Cited on page 53.]

[134] J. Doe. (2023) Apache airflow vs. argo workflows: Choosing the right orchestration tool for data pipelines. https://medium.com/@vivekpemawat/understanding-the-differences-between-apache-airflow-and-argo-workflows-3def70b12f40. Accessed: 2025-09-29. [Cited on page 53.]

[135] Pipekit Team. (2022) Airflow vs. argo: Which workflow orchestrator is right for you? https://pipekit.io/blog/airflow-vs-argo-workflows. Accessed: 2025-09-29. [Cited on page 53.]

[136] mkdev Team. (2024) Tekton vs. argo workflows: Comparing kubernetes-native ci/cd and workflow orchestration. https://mkdev.me/posts/is-tekton-still-alive-comparing-tekton-pipelines-with-argo-workflows-argocd-and-jenkins. Accessed: 2025-09-29. [Cited on page 53.]

[137] SuperAGI Team. (2025) Argo workflows. https://superagi.com/. Accessed: 2025-09-29. [Cited on page 53.]

[138] B. Burns, *Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services*. O'Reilly Media, 2016, accessed: 2025-09-29. [Cited on page 53.]

[139] ClearML, "Clearml security and compliance features," 2023. [Online]. Available: https://clear.ml/blog/new-features-from-clearml-strengthen-security-and-vector-image-search [Cited on page 56.]