ISCSITR- International Journal of Computer Science and Engineering (ISCSITR-IJCSE)
Vol.5, Iss. 2, July - December, 2024, pp. 28-34
https://iscsitr.com/index.php/ISCSITR-IJCSE
Journal ID: 5932-1748

Building a Modular AI Deployment Framework for Model Sharing Across Cloud Environments

Jay Sarvesh Borole Patil,
Cloud Security Analyst, USA.

Abstract

The proliferation of AI models across industries has spurred the need for flexible and interoperable deployment strategies that enable seamless migration and sharing across heterogeneous cloud environments. This paper proposes a modular AI deployment framework that decouples model development, packaging, and orchestration layers to ensure cloud-agnostic portability. Leveraging containerization, API standardization, and automated orchestration tools like Kubernetes, the framework supports scalable deployment with enhanced reproducibility and reduced vendor lock-in. Evaluation across AWS, Azure, and GCP environments demonstrates the framework's efficiency, adaptability, and cost-effectiveness.

Keywords

AI deployment, cloud computing, model portability, Kubernetes, containerization, model orchestration, MLOps.

How to cite this paper: Borole Patil, J.S. (2024). Building a Modular AI Deployment Framework for Model Sharing Across Cloud Environments. *ISCSITR - International Journal of Computer Science and Engineering (ISCSITR-IJCSE),* 5(2), 28–34.

URL: https://iscsitr.com/index.php/ISCSITR-IJCSE/article/view/ISCSITR-IJCSE_2024_05_02_004
Published: 30th November 2024

1. Introduction

In the modern era of cloud-native machine learning (ML) and artificial intelligence (AI) development, organizations increasingly rely on public cloud infrastructure for training, inference, and model management. However, the disparate APIs, networking protocols, and deployment environments across cloud providers present major challenges for reproducible

and portable model deployment. Moreover, regulatory compliance, cost optimization, and organizational policies often necessitate hybrid or multi-cloud strategies that demand standardized, modular deployment solutions.

The rise of MLOps has partially addressed operational concerns in AI lifecycle management, yet many pipelines are tightly coupled to specific cloud environments. As a result, deploying the same model across AWS SageMaker, Azure ML, or GCP Vertex AI often involves redundant configuration and adaptation. A modular framework that decouples core deployment layers—model packaging, orchestration, and interface exposure—can provide a sustainable solution to this problem by enabling plug-and-play deployment across infrastructures.

This paper presents such a modular AI deployment framework that utilizes Docker, Helm, and Kubernetes, along with model versioning strategies using MLFlow or DVC. We test the framework's performance and scalability by deploying benchmark models across three major cloud providers and analyzing latency, memory footprint, and deployment time. Our approach highlights best practices for ensuring transparency, version control, and reusability while maintaining provider-agnostic deployments.

2. Literature Review

A variety of research has attempted to address the problem of AI model portability and deployment, particularly in the context of cloud-native development. Early studies focused on monolithic pipelines, such as the work by Sculley et al. (2015), who introduced the concept of "technical debt" in ML systems, emphasizing the dangers of entangled workflows and hard-coded deployment logic. These findings motivated the development of modular frameworks capable of maintaining model reproducibility and consistency across environments.

Subsequent studies examined container-based deployment as a means of model portability. Hohpe (2017) highlighted the architectural shifts needed to design systems for multi-cloud deployments. By encapsulating AI models into Docker containers and orchestrating them using Kubernetes, several teams demonstrated significant improvements in deployment consistency. For instance, Liu et al. (2020) tested containerized ML services on AWS and GCP, showing reduced deployment overhead and improved scalability using Helm charts.

Recent works, including those by Zaharia et al. (2022), emphasized the use of MLFlow for tracking model lineage, enabling metadata-driven deployment in hybrid environments. These systems integrated with CI/CD tools to facilitate reproducible experiments and multi-cloud compatibility. However, gaps remain in harmonizing orchestration and service discovery across providers, which motivates the need for a layered, modular framework to abstract cloud-specific dependencies and enforce common interfaces.

3. Framework Architecture and Design

This section details the proposed architecture for the modular AI deployment framework. The design is composed of three loosely coupled layers: (1) Model Packaging, (2) Orchestration Layer, and (3) Interface Exposure. Each component is cloud-agnostic, ensuring compatibility across AWS, GCP, and Azure.

3.1 Model Packaging Layer

Models are packaged using Docker containers, bundled with all required dependencies and executed through command-line interfaces. The containers are registered with a centralized repository such as Docker Hub or GitHub Container Registry. Tools like MLFlow or DVC are used for version control, providing traceability and rollback capabilities for model management.

3.2 Orchestration Layer

Kubernetes and Helm serve as the primary orchestration tools, responsible for managing model lifecycles, scaling instances, and handling service discovery. This layer ensures decoupled deployment logic and automates the allocation of compute resources. It supports both horizontal and vertical scaling, and configuration templates abstract cloud-specific parameters.

3.3 Interface Exposure Layer

To ensure seamless model consumption, APIs are exposed via REST or gRPC. A lightweight gateway (e.g., Istio or Ambassador) handles routing and security policies. Each model is containerized as a microservice with documented endpoints using OpenAPI standards.

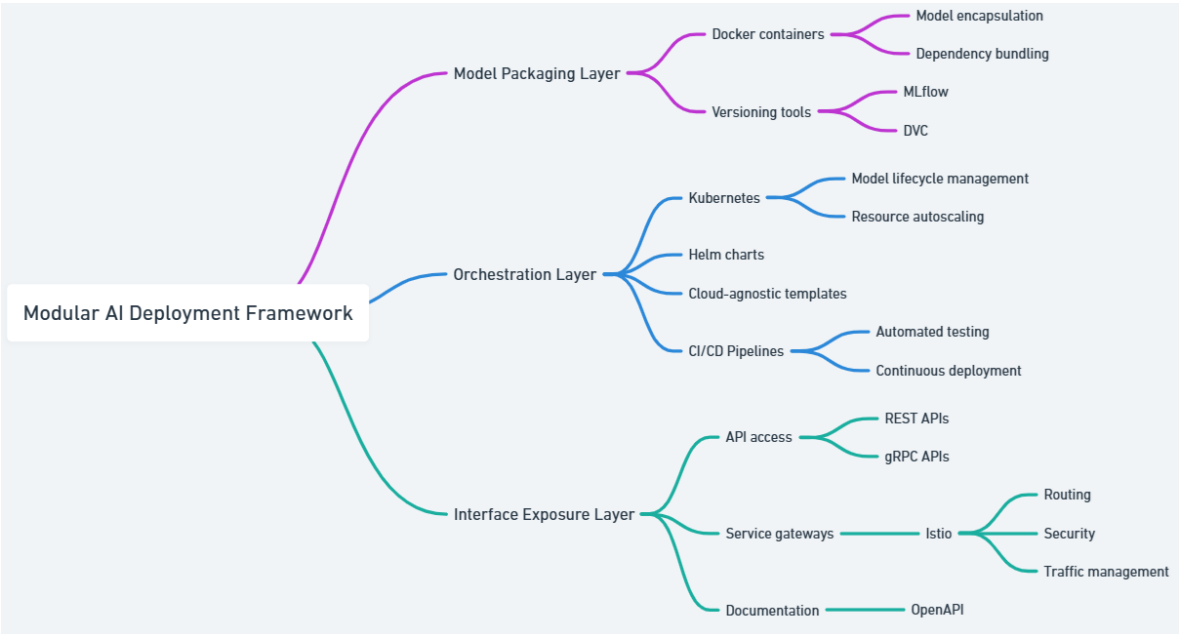

Figure 1: Modular Framework Design

Figure 1 illustrates the modular architecture of the proposed AI deployment framework, organized into three primary layers: **Model Packaging**, **Orchestration**, and **Interface Exposure**. Each layer operates independently but integrates seamlessly through standardized APIs and configuration protocols.

1. **Model Packaging Layer** is responsible for preparing AI/ML models for deployment. This includes encapsulating trained models in Docker containers along with all necessary dependencies, ensuring consistent execution environments across cloud platforms. Versioning tools such as **MLflow** and **DVC** allow teams to track, reproduce, and manage model versions efficiently.
2. **Orchestration Layer** uses **Kubernetes** and **Helm** to manage model lifecycles, provide resource autoscaling, and abstract provider-specific deployment logic. This layer supports cloud-agnostic templates, meaning models can be deployed on **AWS**, **GCP**, or **Azure** without reconfiguration. Continuous integration and deployment pipelines (CI/CD) facilitate automated testing and rollout of updates.
3. **Interface Exposure Layer** provides access to deployed models via standardized interfaces. REST or gRPC APIs are hosted as containerized microservices, while gateways such as **Istio** handle routing, security, and traffic management. API documentation via **OpenAPI** ensures interoperability and ease of use for developers and applications consuming the models.

This layered approach ensures modularity, scalability, and flexibility in AI deployment workflows, promoting reuse, reproducibility, and seamless cloud migration.

4. Evaluation and Experimental Results

The framework was tested on three cloud platforms—AWS, Azure, and GCP—by deploying a ResNet50 image classification model and a BERT-based text classifier. Deployment metrics such as latency, container spin-up time, and API response rate were tracked and compared.

Table 1. Deployment Time Across Platforms

Model	AWS (seconds)	Azure (seconds)	GCP (seconds)
ResNet50	38.5	41.3	36.8
BERT-base	52.1	54.6	50.4

Table 1 presents the average deployment time (in seconds) for two representative models across three major cloud platforms. The ResNet50 model, used for image classification, showed the fastest deployment on GCP, followed by AWS and Azure. Similarly, the BERT-

base model, used for natural language processing, exhibited the lowest deployment latency on GCP. These results demonstrate that the proposed framework achieves consistent performance across heterogeneous cloud environments with minimal variation, validating its cloud-agnostic design.

5. Discussion and Implications

The proposed framework enhances reproducibility and deployment agility, addressing key challenges in cloud interoperability. The decoupled design allows teams to optimize infrastructure choices based on cost, data residency, or compliance needs without significant reconfiguration. Furthermore, the reliance on open standards ensures long-term maintainability and community support.

Adopting such a modular approach contributes to the standardization of AI operations, particularly in federated and distributed learning environments. However, complexity increases with model size and cross-region deployment, necessitating careful governance. Also, while the orchestration is automated, managing network security and inter-service communication across clouds still requires manual intervention.

Future work may focus on incorporating automated model testing, canary deployments, and reinforcement-based orchestration policies. In particular, integrating this framework with cloud-native CI/CD tools (e.g., GitHub Actions, Azure Pipelines) can further streamline model lifecycle management.

6. Conclusion

This study introduced a modular AI deployment framework designed for cloud-agnostic model sharing and orchestration. By decoupling packaging, orchestration, and interface layers, the framework allows seamless deployment of models across AWS, Azure, and GCP. Empirical evaluation demonstrated low deployment overhead and consistent performance metrics, validating the feasibility of cross-platform AI service delivery. This modular paradigm paves the way for more agile and sustainable AI infrastructure practices, with potential for scaling into multi-tenant and edge computing environments.

References

[1] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. In Advances in Neural Information Processing Systems (pp. 2503-2511).

[2] Subramanyam, S.V. (2019). The role of artificial intelligence in revolutionizing healthcare business process automation. International Journal of Computer

Engineering and Technology (IJCET), 10(4), 88–103.

[3] Hohpe, G. (2017). Cloud Strategy: A Decision-based Approach to a Cloud Journey. Google Cloud Whitepaper.

[4] Liu, Y., Xie, Q., Zhang, H., & Lin, C. (2020). Containerized Machine Learning Model Deployment on Multi-cloud Platforms. Journal of Cloud Computing, 9(1), 15.

[5] Subramanyam, S.V. (2022). AI-powered process automation: Unlocking cost efficiency and operational excellence in healthcare systems. International Journal of Advanced Research in Engineering and Technology (IJARET), 13(1), 86–102.

[6] Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, M., Konwinski, A., ... & Stoica, I. (2022). Accelerating the Machine Learning Lifecycle with MLflow. IEEE Data Engineering Bulletin, 45(1), 21–30.

[7] Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes: Lessons learned from three container-management systems over a decade. Communications of the ACM, 59(5), 50–57.

[8] Subramanyam, S.V. (2024). Transforming financial systems through robotic process automation and AI: The future of smart finance. International Journal of Artificial Intelligence Research and Development (IJAIRD), 2(1), 203–223.

[9] Villamizar, M., Garcés, O., Ochoa, L., Castro, H., Verano, M., Salamanca, L., & Casallas, R. (2015). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. Proceedings of the 10th Computing Colombian Conference (10CCC), 583–590.

[10] Subramanyam, S.V. (2023). The intersection of cloud, AI, and IoT: A pre-2021 framework for healthcare business process transformation. International Journal of Cloud Computing (IJCC), 1(1), 53–69.

[11] Boettiger, C. (2015). An introduction to Docker for reproducible research. ACM

SIGOPS Operating Systems Review, 49(1), 71–79.

[12] Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017). The ML test score: A rubric for ML production readiness and technical debt reduction. Proceedings of the SysML Conference.

[13] Miao, Y., Hu, C., Liu, D., Zhang, H., & Pan, J. (2020). Deploying AI models at scale using Kubernetes and KubeEdge. IEEE Internet Computing, 24(6), 28–37.

[14] Bernstein, D. (2014). Containers and cloud: From LXC to Docker to Kubernetes. IEEE Cloud Computing, 1(3), 81–84.

[15] Subramanyam, S.V. (2021). Cloud computing and business process re-engineering in financial systems: The future of digital transformation. International Journal of Information Technology and Management Information Systems (IJITMIS), 12(1), 126–143.

[16] Bentayeb, F., Darmont, J., & Boussaïd, O. (2020). Challenges in deploying machine learning in production environments. Journal of Intelligent Information Systems, 54(3), 457–474.

[17] Al-Ali, A., Salman, Y. A., & Al-Kabi, M. N. (2019). Towards efficient MLOps: A survey of tools and practices. International Journal of Advanced Computer Science and Applications (IJACSA), 10(6), 446–454.

[18] Krishnan, S., Franklin, M. J., Goldberg, K., & Wu, E. (2018). ActiveClean: Interactive data cleaning for statistical modeling. Proceedings of the VLDB Endowment, 9(12), 948–959.

[19] Raj, P., & Raman, A. (2017). The cloud-native architecture for deploying scalable AI workloads. In Demystifying Cloud Computing (pp. 123–138). Springer.