International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I11P118

Volume 72 Issue 11, 165-171, November 2024

Original Article

End-to-End MLOps for Scalable Model Deployment: Engineering Best Practices for Efficient and Reliable Machine Learning Systems

Koushik Balaji Venkatesan

Independent Researcher, Seattle, WA, USA.

Corresponding Author : koushikbalaji.venkatesan@gmail.com

Received: 04 October 2024 Revised: 05 November 2024 Accepted: 24 November 2024 Published: 30 November 2024

Abstract *- Machine Learning Operations (MLOps) help integrate machine learning model development with production deployment using best practices from software engineering. The machine learning life cycle brings unique problems, and this paper outlines possible approaches to address and fix them. Key MLOps practices are reviewed, focusing on Continuous Integration and Continuous Deployment (CI/CD), automated testing, and adaptive scaling strategies. Techniques for deploying models based on latency and traffic demands are explored, including traffic routing and shadow deployments. Advanced strategies such as canary releases, A/B testing, automated monitoring and retraining are also discussed. The goal is for organizations to increase reliability, reduce downtime, create scalable, robust ML pipelines, and accelerate innovation by incorporating engineering best practices.*

Keywords *- CI/CD, Load balancing, Machine learning, MLOps, Shadow testing.*

1. Introduction

Machine learning is rapidly transforming and enabling industries to make meaningful predictions and gain data-driven insights for personalized recommendations, real-time fraud prevention, predictive maintenance, autonomous driving, and more. Training and deploying machine learning models at scale can come with unique problems, and MLOps is a tailored version of DevOps (Developer Operations) meant to tackle them. Continuous updates, quality control, and scaling are vital for real-world applications, and MLOps emphasizes automation and monitoring to achieve that.

Deploying ML models into production is a complex process beyond model development. Unlike traditional applications, ML models have a critical dependency on data, and a drift in data can cause unintended effects, making monitoring and updating crucial. Models need to be periodically retrained to maintain quality, adding complexity to the deployment pipeline.

Resource constraints can make it challenging for some types of models that require a lot of computing resources or applications that need real-time results for high traffic. The scope and objective of this article are to provide best practices for setting up scalable MLOps pipelines, focusing on incorporating engineering practices into model development, automated deployment, monitoring, and scaling.

2. Foundations of the MLOps Pipeline

2.1. Key Stages of MLOps

2.1.1. Data Management

Machine learning model performance relies on data, and data freshness is critical for model training and retraining. Data versioning helps keep track of various versions of data on which the model is trained. Consider tools such as DVC to maintain consistency. It is recommended to maintain robust preprocessing pipelines that can preprocess data into a format that the model needs, ensuring that the input data is clean, consistent, and free of anomalies. Look for features that improve model performance and explainability. Features that can lead to overfitting or data leakage should be avoided.

2.1.2. Model Training and Validation

Orchestrated training pipelines minimize manual work and allow rapid iteration. Consider tools such as MLFlow or SageMaker pipelines to automate end-to-end training workflows. Implement distributed training frameworks for large datasets and complex models to leverage multiple GPUs or TPUs. Validate models across a range of metrics, including accuracy, precision, recall, and latency, and run periodic stress/load tests to ensure the system behaves as expected at scale.

2.1.3. Deployment Automation

Automating deployments helps save engineers time and

avoid mistakes in the process. There can be challenges when models are expected to run in different environments, such as Windows, Linux, etc. Docker and Kubernetes are good solutions for deploying models consistently across different platforms. CI/CD pipelines help take all the heavy lifting between development and deployment from engineers. It helps manage different code versions, run automated integration and load tests, and deploy in different pre-production environments to analyze model performance and develop various deployment strategies. CI/CD pipeline tools like Jenkins or GitLab CI help streamline the model development life cycle.

2.1.4. Monitoring and Continuous Feedback

Once deployed in production, monitoring provides visibility into how a model performs in production. Monitoring for input data drift and ML output results is crucial to ensure models stay performant and fresh. Various performance metrics, such as latency, throughput, accuracy, drift, and infrastructure metrics, such as CPU/memory, need to be monitored. Alerts should be configured to change trends in any of these metrics. Also, consider adding custom metrics and developing real-time dashboards that help engineers gain insights into model performance. Continuous feedback from real-world performances of these models helps detect issues early, enhance user satisfaction, and improve model performance.

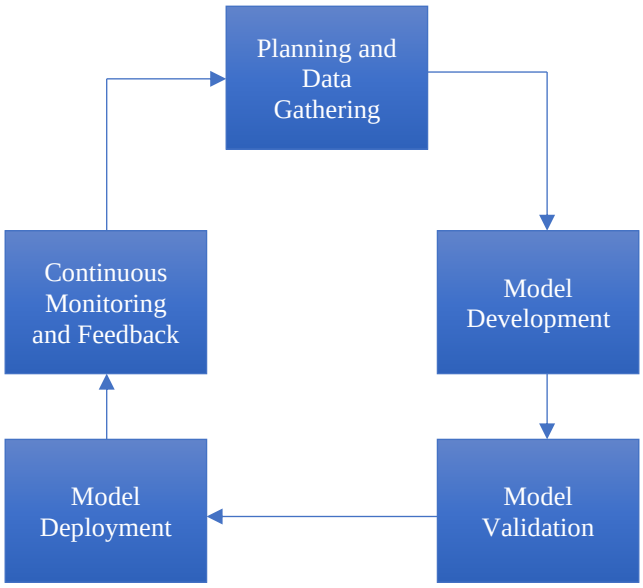

Fig. 1 Different stages of MLOps lifecycle

3. CI/CD Pipelines for Machine Learning

3.1. Continuous Integration for ML Models

3.1.1. Data Validation

Model training data needs to be consistent and clean. Hence, data validation checks and schema validation checks are quite important. Appropriate alarms should be raised if necessary checks were to fail.

3.1.2. Version Control and Dependency Management

Reproducibility and rollback capabilities are crucial for any software, and they become more important for critical services that handle high traffic with little to no room for failures. Tools such as Git help add version control to various changes made to the model over time. Solutions such as DVC (Data Version Control) help keep track of different versions of data to help with rollback and investigation.

3.1.3. Automated Testing

Like any good software, testing is paramount for ML models as well. It is good to incorporate various testing strategies for software development into model development and add tests specific to the model. Unit tests, especially for data transformation functions, help discover errors during development and ensure regression is not introduced for new changes. Integration tests for the end-to-end pipeline make sure individual components are working fine. Load tests help stress test the model and the deployment infrastructure to confirm that it is resilient during traffic surges. Model validation tests using thresholds for performance metrics help detect issues in newer model versions.

3.2. *Continuous Deployment (CD)*

3.2.1. Blue Green Deployment

Blue Green Deployment maintains two identical environments, one running the current production version of the model and the other running the new version that's about to be rolled out. Traffic is switched all at once from blue to green once the new model is performing as expected. This helps reduce downtime and enable quick rollbacks. This is a good choice for straightforward, low-risk updates and enabling quick rollbacks. It is also a good choice for straightforward, low-risk updates.

3.2.2. Canary Release

Canary deployment, like Blue-Green deployment, will have 2 identical environments running different versions of the model, but the newer model is only initially exposed to a small set of users. Depending on how that goes, traffic is slowly dialed up all the way to 100% for the new model. This helps limit the blast radius to a smaller subset of users, and rolling back to older models can also be done quickly.

3.2.3. Shadow Deployment

Shadow deployment is a process where a new potential model gets a copy of the same production traffic to process messages, and results are analyzed and compared against the production model.

3.2.4. A/B Testing and Traffic Dial-Up Control

A/B testing is a method to compare 2 different models by splitting traffic between both models and comparing their performances in live environments. There are various strategies for choosing the percentage of traffic to send to the new model version. One popular option in Meta is to roll out

new features to internal users first, then to a small group of external users, and then eventually to all users. This keeps risk to a minimum and allows for quick mitigation of issues. New features can also be rolled out to certain geographic locations before being made available nationwide or worldwide.

3.3. Automated Retraining

Automated retraining of ML models is often overlooked but important for real-world systems. When set up correctly, it can dramatically reduce the scope for manual interventions and help keep the model fresh and reliable despite drifts in data. Pipelines need to be set up, such as a significant drop in performance of the existing model or a data distribution shift that automatically triggers retraining of the existing model. Automated training pipelines can manage model revision, data revision, testing, and deployment, allowing continuous updates.

4. Types of Models Based on Invocation Patterns

4.1. Batch Processing Models

Batch models are suitable when model outputs are expected periodically rather than in real-time. They will be set to run at different intervals, like twice a day, daily, weekly, etc., and usually, a process will be responsible for accumulating requests during that window and invoking the model. Suppose an online retailer wants to make weekly recommendations on products to buy for their customers; batch processing would be an ideal approach. Predictive maintenance processes run daily to predict machines likely to fail, which is another good example.

4*.2. Synchronous Models*

Synchronous, real-time models are suited for processes with minimal response latency. This also means that the caller usually waits for a response from the model before proceeding with the rest of their processes. Failure handling is important in such scenarios since failing fast is more desirable than taking more time than expected and succeeding. Fraud prevention for financial transactions is an example of synchronous processing where responses are expected in real-time, within a few seconds. Voice assistants are another great example of models that need to be synchronous since users would expect immediate feedback.

4.3. Asynchronous Models

Asynchronous models are ideal for situations where immediate response is not required and in cases where clients want a non-blocking mechanism. The process usually invokes the model with the given output and lets the model notify the invoker once the results are available. Requests are usually queued, and the model processes them in FIFO order.

Since they do not need to meet low latency requirements, they can run at lower priority, optimizing resources and costs. Content moderation on platforms like Facebook and TikTok usually uses async models to analyze images, videos, and text for potentially harmful content. E-commerce and financial institutions use async models to generate insights on customer behavior for targeted marketing campaigns.

5. Scaling and Load Management in Model Serving

5.1. Auto Scaling Strategies

5.1.1. Horizontal Scaling

Horizontal scaling is an approach in which instances are added to the production environment based on traffic demands, and users pay only for what they use.

5.1.2. Vertical Scaling

Vertical scaling is a strategy where more memory and/or processing power is added to individual instances to handle high-load requests.

5.1.3. Dynamic Scaling Policies

An auto-scaling policy to scale based on various metrics such as memory, CPU, and number of requests is recommended. This optimizes resource usage since it scales up and down as demand fluctuates, and most cloud providers charge only when a particular resource is being used.

5.2. Optimizing for Low Latency

5.2.1. Edge Computing

Edge computing is a mechanism where ML models are run on devices close to end users, like IoT devices, smartphones, or edge servers. This helps in reducing latency, data privacy, and cost efficiency since data does not have to be sent to a centralized cloud infrastructure. Edge computing is a popular option for autonomous vehicles that need to process enormous amounts of data quickly to make instant decisions.

5*.2.2. Caching and Pre-Computation*

Caching is another good strategy to store frequently used results without having to make redundant operations and improve response times. Various caching strategies dictate when to store and delete data from the cache. Batch processing is an example where model results are generated in batches and stored in a caching layer that is available to be consumed by clients. Frequently used features are cached in some cases instead of having to recalculate features for every input.

5.3. Concurrency Management and Load Balancing

5.3.1. Concurrency limits

Consider limiting the number of concurrent requests handled by individual instances so throttling can be avoided, and that performance does not degrade over time.

5.3.2. Load Balancing

Load balancing is critical for environments where ML models are used in high-traffic environments. This ensures efficient distribution of requests across multiple instances, thereby minimizing latency, improving performance, and

reducing single points of failure. There are various load balancing strategies, such as round-robin load balancing, where requests are distributed sequentially in a circular order among instances with similar processing power. Serverless load balancing is a popular option for highly fluctuating lightweight workloads. AWS Lambda and Google Cloud Functions are some services that offer serverless infrastructure.

6. Monitoring, Logging, and Alerting in Production

6.1. Model Performance and Monitoring

6.1.1. Data and Prediction Drift Detection

Detecting drift in input data and model predictions is important to maintain long-term model performance. Statistical methods such as the Population Stability Index (PSI) can be used to identify changes in distribution. Additionally, automated monitoring tools can be configured to trigger alerts when alarming deviations are detected, ensuring timely intervention and model retraining.

6.1.2. Real-Time Metrics Collection

Exposing and tracking real-time metrics such as latency, accuracy, and error rates is important to maintain a model in production. Tools such as AWS CloudWatch and Prometheus can be used for metric visualization and analysis. Incorporating metrics such as throughput, resource utilization, and response times can help proactively identify bottlenecks and performance degradation.

6.2. Operational Metrics and Alarms

Infrastructure metrics such as CPU/Memory and GPU consumption should be tracked to understand usage patterns, optimize model deployments, and be cost-efficient. It is highly recommended that alert or alarm systems be set up to track important metrics and ensure they are within an expected range. This should also be integrated with an incident management system for manual intervention and quicker issue resolution.

7. Case Study: Synchronous Model Handling with Production and Shadow Deployments

7.1. Example Setup

Let us consider an e-commerce platform that relies on online models to recommend products, personalize search results, and block fraudulent transactions in real-time. Given the high traffic and low latency demands, the company uses synchronous models for recommendation and fraud detection while using asynchronous models to personalize website content based on user behavior.

To continuously improve model performance, they would need an automated mechanism to validate and deploy new models without affecting existing customers. A shadow deployment strategy sounds like a fitting approach for this use case.

7.2. Requirements and Challenges

7.2.1. Low Latency Requirements

The fraud detection models needed to respond within 50 milliseconds so that user experience is not affected and to keep bad actors from doing fraudulent things. Longer delays can affect checkout, possibly forcing customers to abandon their carts.

7.2.2. Frequent Model updates

Fraudulent activities are evolving as fraudsters find new opportunities to exploit. Fraud detection strategies must constantly adapt to changing patterns and require consistent model updates.

7.2.3. Model Validation in Real-Time

Since it is hard for them to reproduce production traffic in other environments, they need a safe way to validate new models at 100% production traffic without affecting user experience. Analysis should be performed on the predictions from the model under validation and compared with the existing production model.

7.2.4. Scalability Under High Traffic

Transactions can exceed 10,000 requests per second, especially during peak shopping seasons. Both the production and shadow models need to be supported by an infrastructure that can handle traffic surges.

7.2.5. Cost Constraints for the Infrastructure

The company would always need to maintain 2 different models at production traffic to keep up with changing patterns constantly. That would mean twice the storage and infrastructure cost. Deployment strategies and resource usage should be optimized for cost to meet cost targets. Resources should be scaled down during non-peak hours, and their infrastructure should not be overallocated with resources that are not being used.

7.3. Solution: Shadow Deployment and Real-Time Monitoring

7.3.1. Primary Production Model

This is the stable version of the model that has been serving production requests and has proved to respond with low latency under high traffic. This is being constantly monitored for performance drifts as well as errors.

7.3.2. Shadow Model

A newer version of the model that's deployed in a shadow environment receives 100% of production traffic. Results from this model will not be hooked to the production flow and will not affect end customers in any way. Engineers and scientists will monitor performance closely, evaluating it under real-world conditions.

7.3.3. Traffic Routing and Duplication

The load balancer that helps divert traffic to appropriate

instances hosting the production model is configured to duplicate all traffic and send it to the shadow environment.

7.3.4. Automated Metric Collection

Key metrics such as accuracy, latency, error rates, and resource usage metrics are all emitted and tracked across both models. Engineers can directly compare the performance of new vs. old models to make an informed decision on promoting newer versions of the model.

7.3.5. Real-Time Monitoring and Alerts

Dashboards are configured to monitor critical metrics for both models. Alarms are configured, looking at various metrics and notifying operations anytime there is an anomaly or a spike in error rates. Confidence thresholds are set on performance expectations for the new model, such as matching production latency with a 10% tolerance and maintaining similar accuracy.

7.3.6. Gradual Dial-up

Once shadow model results are validated and it looks good to be promoted as the production model, the company decided to dial up the traffic sent to the newer model gradually. They start with 10% of live production traffic sent to the shadow model, while the remaining 90% goes to the existing production model. Eventually, the shadow model gets all 100% of production traffic, and the production model is kept in reserve only for cases where a rollback is needed.

7.4. Results

7.4.1. Improved Model Validation and Real-Time Testing

Shadow deployments helped the company validate new models on live production traffic without risking customer experience. Engineers and scientists were able to validate and gain insights into the new model's performance in production conditions and look for issues such as drift, decay, or engineering bugs before a full launch.

7.4.2. Reduction in Rollback Incidents

Rollback incidents would have reduced considerably since there are much fewer surprises during a production deployment. Changes go through rigorous testing in lower environments and shadow testing with production traffic. The new shadow model is constantly monitored for output, trends, and metrics such as latency to ensure they are within the expected range.

7.4.3. Data-Driven Decision Making

Based on results from shadow deployments, engineers can make informed decisions about the type of instances to use, scaling needs, meeting requirements, etc., while science can gain insights on model performance, feature selection for newer models, hyperparameters, etc. Because of the gradual dial-up, stakeholders can identify and compare the affected population to the existing population to generate meaningful insights. Dial-up also helps us take a conservative step towards large rollouts, and the company can quickly dial down if they see unintended results.

7.5. Key Takeaways

7.5.1. Continuous Model Improvement

The company notices that developing and deploying features takes much less time. New fraud detection strategies or recommendation models are safely tested without risking customer experience. This helps them quickly leverage more performant models to keep up with the competition.

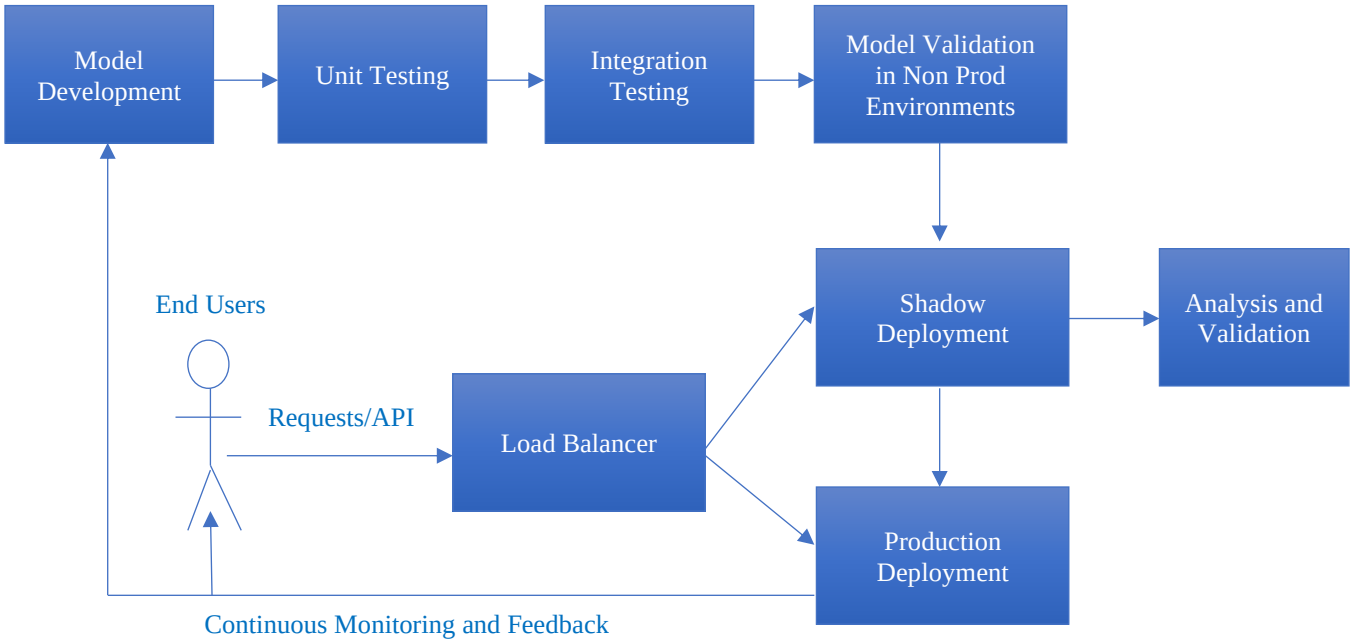

Fig. 2 Architectural overview from the case study

7.5.2. Operational Efficiency and Scalability

Thanks to effective load-balancing strategies to distribute traffic, infrastructure usage is optimized for peak times. Most cloud providers, like AWS, Azure, etc., provide options to pay only for what you use, so the company benefits from using resources appropriately based on varying traffic conditions.

7.6. Lessons Learned

7.6.1. Establish Clear Success Metrics

It is important to identify specific thresholds for metrics such as latency, accuracy, and resource usage to track the existing production model's performance and to decide when a shadow model is ready for production.

7.6.2. Use Automated Monitoring and Alerts

Real-time dashboards are crucial for operations and engineers to monitor performance and detect performance issues early. It is important to configure automated alerts so teams are notified immediately of issues.

7.6.3. Gradual Deployment

Shadow results showed that the new model's performance was as expected. The company has gradually deployed the new model into different geographic locations to help mitigate risk. This also helped them gain insights into model performance differences between customers using the new and existing models. These effective deployment and validation strategies allowed the company to achieve low latency and high reliability in fraud detection and recommendation services. They maintain an agile development model, constantly improving their results without sacrificing customer experience.

8. Conclusion

8.1. Summary of Key Points

This paper depicts some effective engineering practices that allow safe and efficient scaling of machine learning models. It focuses on the CI/CD pipeline, testing automation, efficient deployment strategies, and adaptive scaling techniques. MLOps best practices will make an organization's model development life cycle easier. This enables them to deploy low-latency models serving high traffic with much-reduced deployment times, improving their reliability and customer experience. Techniques that include shadow deployment, gradual dial-up, traffic routing, and load balancing enable teams to validate model updates against production traffic safely. With a well-engineered MLOps framework, teams can speed up their model development and release cycle to reduce the risk of failures and ensure that the models remain aligned with business goals.

8.2. Future Trends in MLOps

As the adoption of machine learning continues to expand within organizations, MLOps will continue to advance and improve to keep pace with demands around automation, scalability, robustness, and data privacy.

8.2.1. Edge and IoT Applications

MLOps practices are evolving to support distributed learning across a network of edge devices. A major challenge in this respect is model optimization, so they run on resource-constrained devices efficiently. As chips evolve and specialized chips become common, MLOps practices will evolve to use newer hardware capabilities better. There will also be a great focus on green MLOps, ensuring that systems are energy efficient.

8.2.2. Federated Learning

Federated learning is a machine learning technique that enables the training of models on distributed datasets without the need for centralizing data. Since this does not require an exchange of data from a client to some global service, federated learning is helpful in reduced data transfers, privacy preservation, and continuous learning. There are challenges related to federated learning in terms of communication efficiency and the management of fairness and bias that it eventually needs to overcome.

8.2.3. AIOps and AutoML Integration

AIOps, or AI for Operations, and AutoML, or Automated Machine Learning, are two major and fast-evolving areas in machine learning. AIOps is, in simple words, adopting Artificial Intelligence and Machine Learning to enhance and automate operations, enhancing their efficiency, performance, and general operations. For instance, models may find anomalies in logs or metrics and trigger an alert or act accordingly, as set. AutoML aims to automate the process of applying machine learning to real-world problems by making ML more accessible to non-ML experts and increasing ML experts' productivity. Automated feature engineering and hyperparameter optimization are some good examples.

8.2.4. Ethical AI and Model Interpretability

Monitoring for fairness and bias and explaining model outputs clearly will also be some of the core components of MLOps in the future. An efficient MLOps framework can handle data privacy and help with the secure transfer of data so that there is no scope for mishandling.

8.3. Closing Remarks

MLOps will become fundamental in ensuring ML models are more accessible, reliable, and useful. While the importance of ML models is increasing daily, they should be updated regularly based on variable customer trends. Building a concrete foundation in MLOps will help teams innovate and launch newer models safely. Automated solutions for testing, training, and validations, coupled with advanced risk mitigation and deployment techniques, position organizations to accelerate growth without compromising customer experience.

Conflicts of Interest

The author declares (s) that there is no conflict of interest regarding the publication of this paper.

References

[1] David Sculley et al., "Hidden Technical Debt in Machine Learning Systems," *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems*, vol. 2, pp. 2503-2511, 2015. [Google Scholar] [Publisher Link]

[2] Dev Kumar Chaudhary, Sandeep Srivastava, and Vikas Kumar, "A Review on Hidden Debts in Machine Learning Systems," *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*, Bangalore, India, pp. 619-624, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[3] Dominik Kreuzberger, Niklas Kuhl, and Sebastian Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," *IEEE Access*, vol. 11, pp. 31866-31879, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Satvik Garg et al., "On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models Using MLOps," *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, Laguna Hills, CA, USA, pp. 25-28, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Matteo Testi et al., "MLOps: A Taxonomy and a Methodology," *IEEE Access*, vol. 10, pp. 63606-63618, 2022, [CrossRef] [Google Scholar] [Publisher Link]

[6] Georgios Symeonidis et al., "MLOps - Definitions, Tools and Challenges," *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, pp. 453-460, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Meenu Mary John, Helena Holmstrom Olsson, and Jan Bosch, "Towards MLOps: A Framework and Maturity Model," *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Palermo, Italy, pp. 1-8, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Antonio M. Burgueno-Romero et al., "Towards an Open-Source MLOps Architecture," *IEEE Software*, vol. 42, no. 1, pp. 59-64, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[9] Yue Zho, Yue Yu, and Bo Ding, "Towards MLOps: A Case Study of ML Pipeline Platform," *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, Beijing, China, pp. 494-500, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] AWS, Serverless Computing - AWS Lambda, Run Code without Thinking about Servers or Clusters, Amazon Web Services, 2024. [Online]. Available: https://aws.amazon.com/pm/lambda/?gclid=EAIaIQobChMIsdOKmJOOigMV3KlmAh2A_S7bEAAYAiAAEgIQwvD_BwE&trk=5cc83e4b-8a6e-4976-92ff-7a6198f2fe76&sc_channel=ps&ef_id=EAIaIQobChMIsdOKmJOOigMV3KlmAh2A_S7bEAAYAiAAEgIQwvD_BwE:G:s&s_kwcid=AL!4422!3!651612776783!e!!g!!amazon%20web%20services%20lambda!19828229697!143940519541

[11] AWS, Amazon SageMaker Pipelines, Purpose-Built Service for Machine Learning Workflows, Amazon Web Services, 2024. [Online]. Available: https://aws.amazon.com/sagemaker/pipelines/

[12] DVC By Iterative, Data Version Control - and Much More - For the GenAI Era Free and Open Source, Forever. [Online]. Available: https://dvc.org/

[13] MLflow, ML and GenAI Made Simple, Build Better Models and Generative AI Apps on a Unified, End-to-End, Open Source MLOps Platform, 2024. [Online]. Available: https://mlflow.org/

[14] Docker, Develop Faster, Run Anywhere, Build With the #1 Most-Used Developer Tool, 2024. [Online]. Available: https://www.docker.com/

[15] Kubernetes, Kubernetes, Also Known As K8s, is an Open Source System For Automating Deployment, Scaling, and Management of Containerized Applications, 2024. [Online]. Available: https://kubernetes.io/

[16] AWS, Amazon CloudWatch Documentation, Amazon CloudWatch Provides a Reliable, Scalable, and Flexible Monitoring Solution That You Can Start Using Within Minutes. You No Longer Need to Set Up, Manage, and Scale Your Own Monitoring Systems and Infrastructure, 2024. [Online]. Available: https://docs.aws.amazon.com/cloudwatch/

[17] Prometheus, Prometheus is an Open-Source Systems Monitoring and Alerting Toolkit Originally Built at SoundCloud, 2024. [Online]. Available: https://prometheus.io/docs/introduction/overview/