On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models using MLOps

Satvik Garg[1], Pradyumn Pundir[1], Geetanjali Rathee[2], P.K. Gupta[1], Somya Garg[3], Saransh Ahlawat[3]
[1]*Jaypee University of Information Technology, Solan, India*
[2]*Netaji Subhas University of Technology, New Delhi, India*
[3]*Deloitte LLC, New York, NY, USA*
Email: {satvikgarg27, pundirpradyumn25, geetanjali.rathee123}@gmail.com, pkgupta@ieee.org, {somgarg, saahlawat}@deloitte.com

***Abstract*—In recent years, model deployment in machine learning is observed to be an interesting area of study. It can be seen as a process similar to the one established for traditional software development. Development and operations (DevOps) incorporating Continuous Integration and Continuous Delivery (CI/CD) have demonstrated to smooth out software advancement and speed up organizations. Nonetheless, employing CI/CD pipelines in an application that incorporates components of Machine Learning Operations (MLOps) has challenging issues, and pioneers in the field settle them with the utilization of exclusive tooling, frequently presented by cloud suppliers. This study gives a higher perspective on the machine learning lifecycle and the vital differences between DevOps and MLOps. We talk about tools and techniques to execute the CI/CD pipeline of machine learning frameworks in the MLOps approach. Subsequently, we deep dive into push and pull-based deployments in Github Operations (GitOps). Open exploration challenges are additionally distinguished and added that can direct future research.**

***Keywords*-MLOps, DevOps, GitOps, CI/CD, Deployment, Containerization, Orchestration, Kubernetes, Docker, Kubeflow**

I. Introduction

Artificial Intelligence (AI) is a field of study that has been on the ascent to develop applications fit for procuring information and providing models dependent on past knowledge [1]. For real-world applications, the data is so vast and made available in the production system only, which brings about the management of various tools, including libraries, and dependencies that can complicate model development [2]. Therefore, it has been valuable to incorporate the concepts of continuous training and continuous development (CI/CD) for automated deployment of AI models [3].

In real-world systems, machine learning (ML) models must be flexible according to changing input data. Once the model has been deployed to production, the performance of the model degrades due to frequent changes to the data [4]. Therefore, model monitoring is essential and responsible for adequately monitoring the pipeline that can trigger, retrain and ensure models are working as expected. However, tracking the performance of a model comes with several challenges throughout the machine learning lifecycle [5]. All of the these challenges can be solved using MLOps [6]. What is MLOps? In simple terms, MLOps is DevOps for machine learning. It enables developers to collaborate and increase the pace at which AI models can be developed, deployed, scaled, monitored, and retrained.

The idea behind presenting such a paper is to put together an investigation that gathers information on DevOps methodology for machine learning applications. On looking through research papers and interacting with experts in the field of CI/CD, we found a knowledge gap among machine learning developers in building automation pipelines [7] [8]. Our most significant commitment to this paper is not a methodology but an investigation of approaches. It covers using existing advances, for example, containerization tools such as docker [9], orchestration tools like Kubernetes [10], and MLOps platforms such as Amazon SageMaker [11], KubeFlow [12], and MLFlow [13] that are demanding and easy to implement. Subsequently, we provide a bird's eye view on the Github operations [14] related to model deployment to help guide new researchers.

The rest of the sections are organized as follows: Section 2 introduces the concepts of DevOps, including concepts of CI/CD, containerization, and orchestration. Section 3 involves MLOps levels and discusses the key differences between MLOps and DevOps. Section 4 provides information on Github operations for testing push and pull-based deployments. Section 5 explores open research challenges, and Section 6 concludes the paper.

II. Development and Operations

DevOps or development and operations can be termed as the practice used by an organization while developing software applications [3]. There are mainly two concepts in DevOps:

- Continuous Integration (CI): CI helps in time management and enables an organization to have short and frequent release cycles for improving software quality and increase overall team productivity.
- Continuous Deployment (CD): It helps automatically deploy software in production [15]. The main differ-

978-1-6654-3736-3/21/$31.00 ©2021 IEEE
DOI 10.1109/AIKE52691.2021.00010

ence between CI and CD is mainly about ensuring an application availability in production ready condition after performing quality checks.

Machine learning pipelines require components to be reusable, composable and potentially shareable across the pipeline. These components should ideally be containerized to decouple execution environment for custom code runtime and make the code reproducible between developers [16]. Docker [9] has been widely used for containerization. As shown in Figure 1, the docker platform provides the ability to package and run applications in a small isolated environment by using a client-server architecture, where the client communicate to docker daemon, for processes such as building, running, and distributing docker containers and performs essential functions. However, for large scale applications, thousands of containers need to be deployed which can be challenging to manage. Orchestration solves the problem of docker adoption by allowing developers to leverage the concepts of container automation, deployment, and networking [17]. In most of the cases, the docker containers are organized by kubernetes [10].

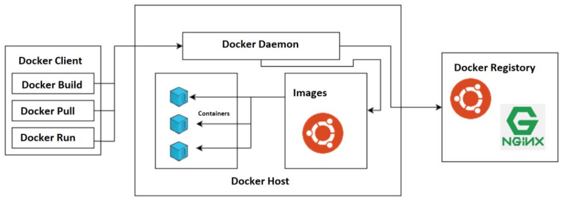

Figure 1. Docker Containerization.

III. Machine Learning Operations

The task of deploying AI models to production requires processes after the business model has been determined to ascertain the success rate. These processes provide ML models for production and can be accomplished manually or with the aid of automated workflows. As a result, three different levels of MLOps are defined namely, MLOps Level 0, MLOps Level 1, MLOps Level 2 [18].

A. MLOps Level 0

The fundamental level of maturity, or MLOps level 0, refers to simple workflows that are manually script-driven at every stage of the machine learning lifecycle. This level of MLOps suffers from sparse release iterations because it expects the generated model to not change very often. The concepts of CI/CD are not needed as there is no automation, resulting in lack of active performance monitoring. When models are repeatedly changed or trained, the manual-driven process may persist, but in practice, models are regularly changed or trained, demanding regular iterations.

B. MLOps Level 1

The steps of machine learning experiments are orchestrated at MLOps level 1, to automate the ML pipeline solely to undertake continuous model training. This enables it to provide model prediction services continually and automates retraining models in production using new data. As a result, model deployment setups and continuous model delivery are automated, allowing to leverage trained and validated models as a prediction service for making the online predictions. However, in order to test new ideas and quickly release new implementations of ML components, the need for CI/CD solutions to automate the creation, testing and deployment of ML pipelines is critical.

C. MLOps Level 2

Level 2 of MLOps includes automation of a CI/CD pipeline for more rapid and reliable updating of pipelines in production, which requires a more robust automated system. As shown in Figure 2, a total of six CI/CD processes has been incorporated that includes development and experimentation, where staged experiment phases can be carried out iteratively for training new algorithms and models [18]. It results in the output of pipeline steps' source code, subsequently pushed into the source repository. Next step involves continuous integration (CI), where the source code is built and undergoes various run tests, resulting in package executables and artifacts being deployed later. Finally, in continuous delivery (CD), the artifacts created in CI phases are deployed to the target environment, resulting in the implementation of updated AI models through the generated pipeline. This pipeline is automated in production and runs according to a schedule or a trigger. The generated trained models are then uploaded to the model registry, and this stage is also known as automatic triggering. It also includes CD, which serves prediction as a service. So, once a model prediction service has been installed, the statistics on model performance based on real-time data are collected. The result of this stage is a trigger to execute pipelines and start a new trial cycle.

D. MLOps vs DevOps

MLOps is DevOps to ML, but only to a certain extent, and there are key differences that needs to be addressed by the MLOps platforms [19].

- In a typical software application, if an error needs to be fixed, the task is only to edit the code, test, and deploy it. Nonetheless, model training in ML is an experimental field that requires various efforts on alternative data parameters, algorithms, and feature engineering techniques.
- Integration and unit testing are of central significance to guarantee the right execution of software ecosystems. In any case, the different testing procedures applied in

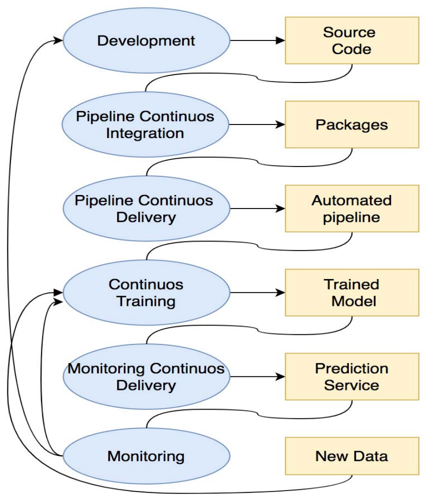

Figure 2. MLOPS Automated CI/CD Pipeline [18]

software systems are not usually implemented in ML-based systems.

- When a model is put into production, it begins to make predictions based on the knowledge gained from real-time data. As business goals adjust, this knowledge is modified which leads to model degradation. Different types of monitoring, such as covariate-shift and pre-shift [20] needs to be implemented, unlike traditional software applications, which in most cases involves monitoring latency.

E. MLOps Platform

Recent years have seen a plethora of MLOps platforms to provide standards for deploying enterprise level AI applications [11]–[13]. These platforms are open-source projects that contains curated set of compatible tools and frameworks for machine learning to ease the process of automated development. Kubeflow [12] introduces the concept of the namespaces that enables multi-user support to simplify collaboration and access management. In addition, it includes a built-in Jupiter notebook environment in the clusters which makes it simple to perform jobs with dynamically scaled resources. Katib, which can be utilized for automated hyperparameter adjustment, is another crucial component in kubeflow, responsible for determining and visualizing the best configuration for the model before it goes into production.

Amazon SageMaker [11] allows connecting and loading data from sources such as Amazon S3 bucket, Amazon Athena, and Amazon Redshift. that leverages data manager, which gives the function of combining and merging original data characteristics within a few seconds. It is essential to make sure that the data and features are well balanced and defined. For this, the SageMaker Clarify can be used to verify each prediction. Finally, the Sage Maker debugger can remove them from the model to identify the source of errors.

In MLflow [13], the MLflow tracking helps to recover and query experimental data and results. Tools like ML projects and MLflow models allow implementing ML models in the environment. A model registration feature helps to store and manage models in a central repository. In addition, it has built-in integration with TensorFlow, PyTorch, Keras, python, Kubernetes, docker, etc. Many organizations such as Databricks, Toyota, Accenture, Facebook, and Microsoft are employing and contributing to MLflow, which easily provides community support for resolving bugs.

IV. GITHUB OPERATIONS

GitOps or Github Operations is considered as a continuous deployment method for cloud-native applications [21]. It can also be termed a delivery tool that uses a pull-based model, in contrast, employing a push-based model. The core idea of GitOps is to leverage the Git repository which contains the details of the infrastructure required in a production environment. It has an automated process to handle necessary steps required for deployment. For example, it can deploy a new application or update an existing application by simply updating the repository. It accesses a repository or an image registry, and users can give developers direct access to the environment. Tools such as ArgoCd, Flux and Fleet are developed to maintain the GitOps workflow.

GitOps supports the management of multiple pipelines by configuring different build pipelines to update the environment repository and helps to configure an operator or deployment pipeline to react to changes in a branch [22]. The deployment approach for GitOps can be implemented in two ways: push-based and pull-based deployments. The difference between these two deployment types is how the deployment environment matches the desired infrastructure. Pull-based approaches should be chosen whenever possible because they are deemed more secure and better practices for implementing GitOps.

V. OPEN RESEARCH CHALLENGES

Over time organizations have noticed the importance of MLOps in executing a constructive production pipeline. However, MLOps is still in its infancy, and most firms are still working on the best way to optimize it for their specific projects [23].

- One issue to be developed in the early stages is about monitoring the effectiveness of AI models and poor understanding of solution needs can affect solution performance. Changes in information are more frequent which can disrupt the entire model, even if the collected data is free of errors.

- Developers are still more inclined towards manually testing and deploying AI models in production as they are not vocal about automatically delivering each change.
- Understanding and porting code from one library to another is routinely non-trivial. The ascent of multiple libraries for AI development and the specialized basic knowledge of experts created challenges for coding. Libraries, for example, Tensorflow, Keras, and Pytorch have alternative backbones that allow a similar application to appear to be unique from each other.
- In most cases, the development and production teams are out of sync and communicate only at the end of solution design. Approval is required in case someone on the team changes a requirement that has to be reflected on the production server making the whole process slow.

VI. Conclusion

As part of this exploratory research, we gathered information about the current MLOps, and the key motivators behind having operational processes that can facilitate clear and efficient workflow for delivering machine learning solutions. We inherited ideas from traditional software development, specifically the CI/CD techniques and presented use-cases for when those practices could be valuable in the machine learning development lifecycle. We proposed 3 different levels of MLOps that vary in efficiency and the amount of resources required to set them up, because the understanding is that not every ML solution would require similar amount of provisioning. The level 3 of MLOps is the most sophisticated solution proposed and we recommend using it in any enterprise machine learning solution. This level of granularity would achieve the most efficient and automated development workflow as it leverages the industry standard tools and techniques that have successfully delivered enterprise level software solutions in the past decade. In future work, we plan to produce a demo using a proof of concept applications to compare all three approaches for different size of applications.

References

[1] P. Langley and H. A. Simon, "Applications of machine learning and rule induction," *Communications of the ACM*, vol. 38, no. 11, pp. 54–64, 1995.

[2] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson, "Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions," *Information and Software Technology*, vol. 127, p. 106368, 2020.

[3] S. Garg and S. Garg, "Automated cloud infrastructure, continuous integration and continuous delivery using docker with robust container security," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019, pp. 467–470.

[4] Y. Wu, E. Dobriban, and S. Davidson, "Deltagrad: Rapid retraining of machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 355–10 366.

[5] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," *Advances in neural information processing systems*, vol. 28, pp. 2503–2511, 2015.

[6] S. Alla and S. K. Adari, "What is mlops?" in *Beginning MLOps with MLFlow*. Springer, 2021, pp. 79–124.

[7] Z. Wan, X. Xia, D. Lo, and G. C. Murphy, "How does machine learning change software development practices?" *IEEE Transactions on Software Engineering*, 2019.

[8] C. T. Wolf and D. Paine, "Sensemaking practices in the everyday work of ai/ml software engineering," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 86–92.

[9] "Empowering app development for developers — docker," https://www.docker.com/, (Accessed on 09/13/2021).

[10] G. Sayfan, *Mastering kubernetes*. Packt Publishing Ltd, 2017.

[11] "Amazon sagemaker – machine learning – amazon web services," https://aws.amazon.com/sagemaker/, (Accessed on 09/13/2021).

[12] "Introduction to kubeflow — kubeflow," https://www.kubeflow.org/docs/about/kubeflow/, (Accessed on 09/02/2021).

[13] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe *et al.*, "Accelerating the machine learning lifecycle with mlflow." *IEEE Data Eng. Bull.*, vol. 41, no. 4, pp. 39–45, 2018.

[14] "Gitops — gitops is continuous deployment for cloud native applications," https://www.gitops.tech/, (Accessed on 09/02/2021).

[15] C. Singh, N. S. Gaba, M. Kaur, and B. Kaur, "Comparison of different ci/cd tools integrated with cloud platform," in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2019, pp. 7–12.

[16] J. Turnbull, *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2014.

[17] J. Rufino, M. Alam, J. Ferreira, A. Rehman, and K. F. Tsang, "Orchestration of containerized microservices for iiot using docker," in *2017 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2017, pp. 1532–1536.

[18] "Mlops: Continuous delivery and automation pipelines in machine learning," https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning, (Accessed on 08/13/2021).

[19] I. Karamitsos, S. Albarhami, and C. Apostolopoulos, "Applying devops practices of continuous automation for machine learning," *Information*, vol. 11, no. 7, p. 363, 2020.

[20] "Detecting data drift with mlops — 10clouds," https://10clouds.com/blog/detecting-data-drift-mlops/, (Accessed on 09/7/2021).

[21] T. A. Limoncelli, "Gitops: a path to more self-service it," *Communications of the ACM*, vol. 61, no. 9, pp. 38–42, 2018.

[22] "Gitops — gitops is continuous deployment for cloud native applications," https://www.gitops.tech/, (Accessed on 09/15/2021).

[23] V. Lenarduzzi, F. Lomio, S. Moreschini, D. Taibi, and D. A. Tamburri, "Software quality for ai: Where we are now?" in *International Conference on Software Quality*. Springer, 2021, pp. 43–53.