Continuous Integration and Continuous Deployment, DevOps and MLOps

Jain, Rishabh
A20495530
College of Computing
Illinois Institute of Technology
Chicago, IL, USA, 60616
rjain35@hawk.iit.edu

Kumar, Rohit
A20501314
College of Computing
Illinois Institute of Technology
Chicago, IL, USA, 60616
rkumar23@hawk.iit.edu

***Abstract*— This paper provides an overview of integration the Machine Learning with CI/CD (Continuous Integration / Continuous Deployment) and DevOps (Development and Operations), this combination is further knowns as MLOps. As per the needs of advancing technologies MLOps is promoted over simple machine learning model pipeline because it provides on-demand scalability, flexible model deployment, and reuse of the trained models. Using the machine learning with CI/CD and DevOps is way more efficient as it is saving a huge amount of time which might not be saved if were still using the traditional pipeline method for the integration of machine learning models. On the other hand, MLOps provides the convenience of rapid deployment of machine learning models. Prior to the deployment users can choose the base ML models. Doing this will reduces the effort of installing dependencies and tool packages. Base models will work as a foundation while other services and functionality can be integrated on to that model for serving the desired purpose.**

Keywords—MLOps, DevOps, CI/CD.

I. Introduction

In modern times, machine learning (ML) models must be flexible, so they can handle the continuous changing input data. Once the model has been deployed to production, the performance of the model degrades over the stretch of time due to the high frequency of frequent changes. Therefore, monitoring of the model is essential as it alone should be responsible for the pipeline that can trigger, retrain and ensure models are working as desired. However, while monitoring the pipeline, we might come across critical challenges that might be difficult to solve. But, using MLOps can help us to overcome all these challenges.

DevOps can be defined as a set of principles and tools based on system engineering. Agile, on the other hand, is an iterative methodology that emphasizes teamwork, customer feedback, and small, frequent releases. DevOps and Agile are two pillars that overlap the traditional operational and developmental teams to create an environment that is always improving operations through a cross-functional team of developers and operators to assist in accomplishing business strategy. The DevOps strategic goal is to look into ways to improve service quality and features while also meeting consumer needs.

What is MLOps? It can be defined as the DevOps for machine learning. It provides the feasibility to the developers to collaborate and increase the pace at which AI models can be developed, deployed, scaled, monitored, and retrained.

The idea behind the paper presentation is to put together an investigation that gathers information on DevOps methodology for machine learning applications. By reviewing the research papers and interacting with experts in the field of CI/CD, we found that there was a knowledge gap among machine learning developers in building automation pipelines. Our main idea for this paper is not a methodology, but an investigation of approaches. This includes using existing advances, for example, containerization tools such as Docker, orchestration tools like Kubernetes, and MLOps platforms such as Amazon SageMaker, KubeFlow, and MLFlow that are demanding and easy to implement.

II. Literature Review

On reviewing the work of literature by Ioannis Karamitsos [1] we found studies which demonstrate to us how to integrate machine learning with DevOps and machine learning with operations (MLOPS). It is mentioned that the new service providers and individuals are using different tools and platforms to automate machine learning models. The authors described continuous integration for machine learning as a time-efficient process which allows users to provide a wide range of choices to achieve continuous improvement.

Furthermore, other authors of the relevant work agreed that the most critical challenges are data collection, data extraction and data cleaning. As a part of research and development, the authors create an application life cycle model based on MLOPs to optimize the manufacturing process.

The study shows how to apply DevOps CI/CD pipelines with machine learning applications. Several researchers including Karamitsos [1], Virmani [2] and Erich [3] have agreed that agile transformation is a key component to improve the efficiency of the companies. MLops can fill the gap between business users and development teams. These authors highlight the principles of DevOps and the guidelines to adopt continuous integration and continuous deployment which contributed as an increment in the development process and improve the quality of the result.

As per the current trends, we acknowledge that many applications are using machine learning techniques, but these techniques are not efficient to produce a great result. But as soon as we integrate these techniques with DevOps it shows a great efficiency factor in any field like healthcare safety etc. The primary goal of DevOps is to create cross-functional teams both operational and development tools work together.

The overall goal of DevOps is to improve the business value in the IT industry, it produces the best results in the agile world with the help of continuous integration and continuous deployment. DevOps overcome the hurdles between operations and development, and they collaborate with both machine learning and DevOps.

III. Continuos Integration and Continuous Deployment (CI/CD)

Continuous Integration and Continuous Deployment is an emerging areas of research and practice. It refers to developing, deploying and getting quick feedback from software and customer in a very rapid cycle. Continuous software engineering involves three phases: Business Strategy and Planning, Development and Operations. This strategy focuses on three software development activities: continuous integration, continuous delivery and continuous deployment.

A. *Continuos Integration*

Define Continuous integration (CI) is a programming philosophy and set of practices that encourage development teams to frequently implement small code changes and check them into version control repositories. Most modern applications require code to be developed on a variety of platforms and tools, so teams need a consistent mechanism for integrating and validating changes. CI establishes an automated way to build, package, and test your application. With CI, software companies can have shorter and more frequent release cycles, improve software quality, and improve team productivity. This practice includes automatic building and testing of software.

B. *Continuous Delivery*

Continuous Delivery (CDE) aims to ensure that the application is always in a production-ready state after successfully passing automated tests and quality checks. CDE employs a set of practices i.e., Continuous integration and deployment automation to deliver software automatically to a production-like environment. Therefore, this method has several advantages it reduces deployment risk, reduces costs, and collects user feedback more quickly.

C. *Continuos Deployment*

Continuous Deployment (CD) goes one step further and automatically and continuously deploys applications to production or customer environments. There is much debate in academia and industry about the definition and distinction between continuous deployment and continuous delivery. The difference between continuous deployment and continuous delivery is the production environment (that is, the actual customer). The goal of a continuous deployment practice is to deploy each change automatically and continuously to production.

D. *CI/CD tools and plugins*

This section presents the findings to answer, With the continuous delivery of software to end-users, the delivery pipeline is becoming more important, and the success of adopting continuous practices in enterprises heavily relies on deployment pipelines. Therefore, choosing the right tools and infrastructures to build such a pipeline can also reduce some of the challenges in adopting and implementing continuous integration, deployment, and delivery practices. We considered the deployment toolchains described in the literature and the tools for implementing the deployment pipeline. Because continuous delivery and deployment might be used interchangeably, we used the term deployment pipeline for the latest release engineering pipelines instead of continuous integration infrastructure and continuous delivery or deployment pipelines.

A deployment pipeline should include the stages of build and packaging to transfer source code from the code repository to the production environment. Automation is a critical practice in the deployment pipeline. however, sometimes manual tasks i.e quality assurance tasks are unavoidable in the pipeline. It is worth noting that there is no standard or single pipeline. Our literature reveals how different tools were integrated to implement toolchains to effectively adopt continuous practices. It should be noted that the tools reported such as GitHub, git, sonar cube, Jenkins and etc. are mostly existing open sources and commercial tools, which aim to form and implement a deployment pipeline.

However, the tools described in this section are intended to facilitate the implementation of ongoing practices. These tools can also be used as part of a deployment pipeline implementation if they are integrated and evaluated in the pipeline. As shown in Figure 1, we have split the deployment pipeline into seven phases. (I) Version control system, (II) Code management and analysis tools, (III) Build tool, (IV) Continuous integration server, (V) Test tool, (VI) Configuration and deployment, (VII) Continuous delivery or deployment server. Note that not all phases are mandatory and no major survey was found in the survey that implemented the pipeline containing all the phases shown in Figure 1. In the first phase, developers continually push code to the code repository. The most common version control systems used in deployment pipelines are Subversion and Git / GitHub. Only a few articles have been published for each. We used code management and analysis tools as part of our deployment pipeline to enhance the build process. The reported work-integrated SonarQube with the Jenkins CI server to collect and visualize metrics data such as test code coverage and coding standard violations for developers. The continuous integration server checks the code repository for changes and uses an automated build tool.

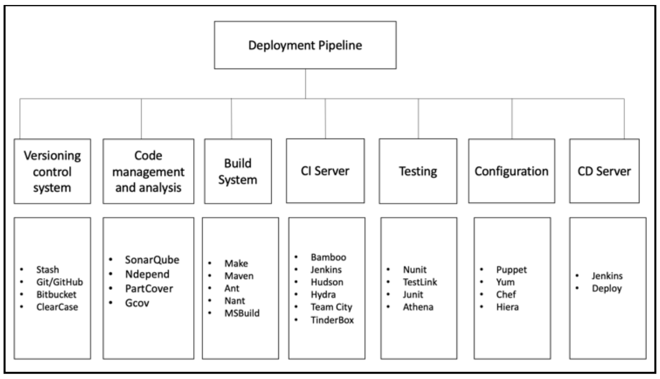

Figure 1: An overview of tools used for deployment pipeline.

IV. Developer and Operations (DevOps)

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity. This includes evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. A higher pace enables organizations to better serve their customers and compete more effectively in the market.

A. *DevOps Workflow*

There are total of eight stages in DevOps workflow, each of them is mentioned in detail below: -

1) Plan - This phase helps define business value and requirements. Some of widely used and known tools are Jira or Git. These tool helps to track down the known issues and perform project management.

2) Code - This phase includes software design and the software code creation. Tools known around the world include GitHub, GitLab, Bitbucket, or Stash.

3) Build - This phase manages software builds and versions, and uses automated tools to compile and package code for future product releases. Use a source code repository or a package repository. They also "package" the infrastructure required for product releases. Examples of tools are Ansible, Docker, Chef, Puppet, Gradle, Maven, or JFrogArtifactory.

4) Testing - During this phase, continuous testing (manual or automatic) is performed to ensure optimal code quality. Examples of tools include JUnit, Codeception, Selenium, Vagrant, TestNG and BlazeMeter.

5) Release - The release phase is a milestone in the DevOps pipeline. It's ready to deploy your build to production. At this point, each code change has passed a series of manual and automated tests.

6) Deployment - This phase can include tools that help you manage, coordinate, schedule, and automate the release of your product to production. Examples of tools are Puppet, Chef, Ansible, Jenkins, Kubernetes, OpenShift, OpenStack, Docker, or Jira.

7) Operation - In this phase, you manage the software in a production environment. Examples of tools are Ansible, Puppet, PowerShell, Chef, Salt, or Otter.

8) Monitoring - This phase identifies and collects information about issues with a particular software version in your production environment. Examples of tools include New Relic, Datadog, Grafana, Wireshark, Splunk, Nagios and Slack.

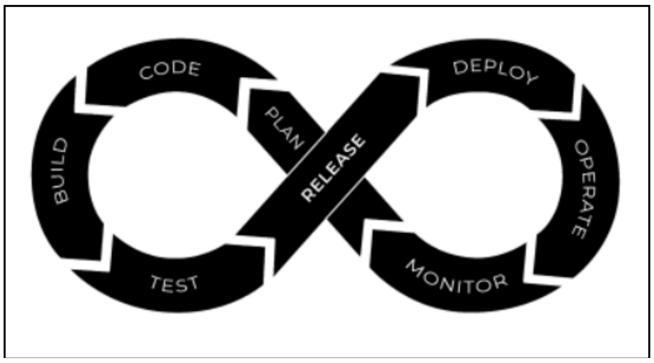

Figure 2: DevOps cycle [4]

B. *Relation between CI/CD and DevOps*

The goal of CI / CD and all DevOps is to make software processes faster and more robust. To make it easier to understand, DevOps as a culture provides a set of ideal practices for developing high-quality software. CI / CD, on the other hand, refers to all activities performed along the software delivery process from creation to final delivery.

C. *Benefits of DevOps*

1) Faster delivery time - DevOps utilizes automation to ensure a smooth flow of the SDLC. By promoting a collaborative culture, it offers the scope for quick and continuous feedback so that any glitches are fixed in time and the releases are done faster.

2) Greater customer experiences – DevOps can help organizations can improve their deployment frequency by 200x, recovery times by 24x, and lower change failure rates by 3x. By automating the delivery pipeline, it becomes possible to ensure the reliability and stability of an application after every new release.

3) Early defect Detection - The DevOps environment embraces a culture of sharing knowledge between teams. Automated continuous monitoring and testing of code helps improve overall build quality.

D. *DevOps on Cloud*

Earlier, when the cloud technologies were not famous in the market as they were less reliable, multinational companies used to set up their DevOps environment on-premises means they set up the whole workstations which include server racks, cooling units, power units etc. in their company campus only.

This strategy was not only costing them much but also taking a lot of space to set up the whole cluster for the production environment. Eventually, as the traffic keeps on increasing, they run out of space, and they need to buy more land to set up more clusters. Also, these all factors are overshadowed by one of the factors is the increase in the pollution as over the time electronic waste also increased and to get rid of this waste huge trash dumps were created inside the city. But, as soon as the rain hits the ground, all the material which was inside the batteries and other power equipment which was supporting the electronic item was mixed with the source of fresh water and was starting to damage the nearby freshwater resources.

As a solution to this problem, multinational companies decide to move their business to the cloud. This helps them to reduce the campus sizes, reducing the electronic waste as all the servers are now located on an isolated land which is not near to the human population and does not affect any freshwater resource.

As per today’s scenario, more than 90% of the companies are using virtualized hardware and do not own any single on-premises server. This strategy of hardware virtualization was very efficient as it was costing very less compared to the early days. Also, all the resources are readily available to use and scalable, which means the count of resources or their capacity of them can be increased or reduced at any point in time depending on the business need.

This strategy allow user to pay as per the concept knows as pay as per use. This means the cloud service providers does not charge the clients if they are not using the resources.

Some of the famous cloud service providers includes Amazon Web Service (AWS), Google Cloud Service (GCP), Microsoft Azure, Digital Ocean, Alibaba Cloud, Oracle cloud, IBM Cloud.

V. Machine Learning Operations (MLOps)

MLOps (Machine Learning Operations) represents a collection of techniques and tools for using ML models in production environments. Includes a combination of DevOps and machine learning. DevOps represents a set of practices whose primary purpose is to limit the time required to release a product, thereby reducing the gap between code improvement and activity. The two main components of DevOps are continuous integration (CI) and continuous delivery (CD). Continuous integration is a method by which a software development organization attempts to integrate code written by a development team on a regular basis. Therefore, they constantly test their code and make small improvements each time based on bugs and weaknesses revealed by the tests. Continuous delivery is the constant introduction of another variant of the product you are working on for testing, evaluation, and subsequent creation. With this training, product releases resulting from non-stop coupling with extensions and new highlights reach end customers much faster.

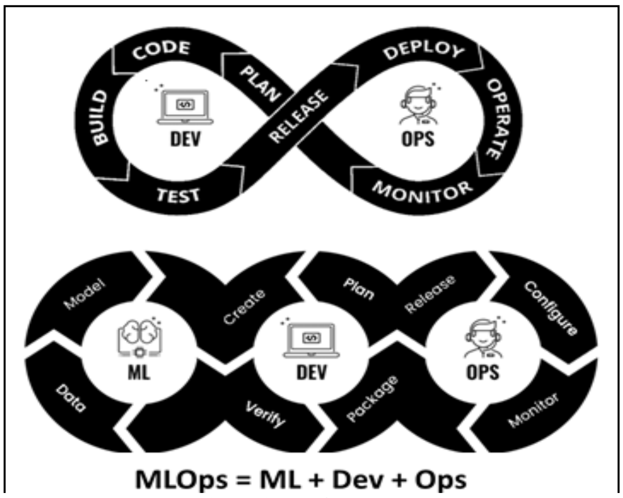

Figure 3: MLOps [5]

The mission of deploying AI fashions to manufacturing calls for techniques after the enterprise version has been decided to examine the fulfillment rate. These techniques offer ML fashions for manufacturing and may be done manually or with the useful resource of automatic workflows. As a result, 3 specific ranges of MLOps are described below.

A. *MLOps level 0*

The basic maturity level, or MLOps level 0, refers to a simple workflow that is manually scripted at each stage of the machine learning life cycle. This MLOps level suffers from repeated sparse releases, as the generated model is not expected to change very often. Since it is not automated, the concept of CI/CD is unnecessary and there is a lack of active performance monitoring. Repeated changes or training of the model can sustain a manually controlled process, but the model is changed or trained on a regular basis and requires regular iterations.

B. *MLOps Level 1*

The machine learning experimental procedure has been tuned to MLOps Level 1 to automate the ML pipeline for continuous model training only. This allows you to continuously deploy model prediction services and use new data to automate model retraining in production. As a result, model deployment settings and continuous model deployment are automated, and trained and validated models can be used as predictive services to generate online forecasts. However, a CI/CD solution that automates the creation, testing, and deployment of ML pipelines is essential for testing new ideas and rapidly releasing new implementations of MLOps components.

C. *MLOps Level 2*

Level 2 of MLOps includes automation of the CI/CD pipeline, enabling faster and more reliable updates of the pipeline in production environments. This requires a more robust automated system. As shown in Figure 4, a total of six CI/CD processes, including development and experimentation, have been integrated to allow iterative step-by-step experimental phases to train new algorithms and models. As a result, the source code for the pipeline step is output and pushed to the source repository. The next steps include continuous integration (CI). In this case, the source code will be built, various execution tests will be performed, and the executable package files and artifacts will be deployed later. Finally, continuous delivery (CD) deploys the artifacts created in the CI phase to the target environment and implements an updated AI model through the creation pipelines. This pipeline is automated in a production environment and runs on a schedule or trigger. The generated trained model is uploaded to the model registry and this phase is also known as auto-trigger. It also includes a CD that acts as a predictive service. Therefore, when the Model Prediction Service is installed, model performance statistics are collected based on real-time data. The result of this phase triggers the pipeline to run and start a new test cycle.

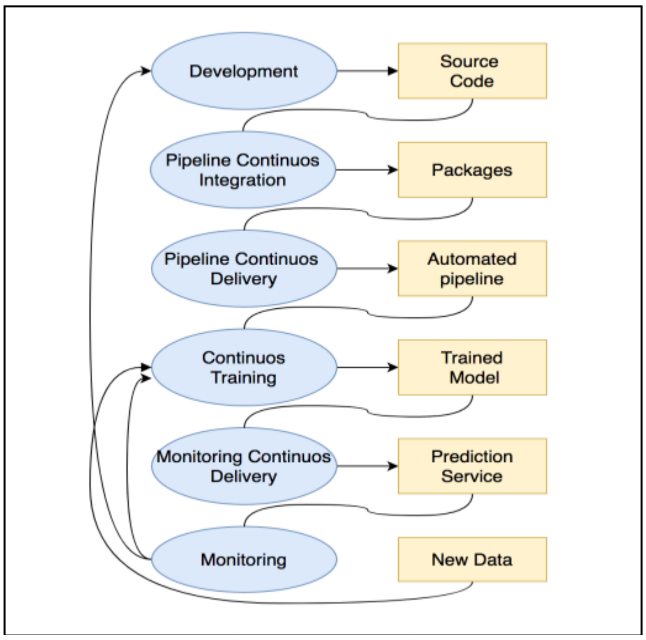

Figure 4: MLOPS Automated CI/CD Pipeline [6]

D. *MLOps Pipeline*

After the spread of DevOps and the practice of general "continuous software development", it became essential to apply the same principles of managing DevOps with machine learning models. In this way, these practices called MLOps (Machine Learning Operations) were born. MLOps seeks to automate the machine learning process using DevOps practices and approaches. The two most important DevOps principles are aimed at providing continuous integration (CI) and continuous delivery (DC)]. It looks easy, but it's not. This is because the machine learning model is not independent, it is part of a larger software system, and it consists of data as well as code. As the data is constantly changing, models are constantly being asked to retrain themselves with the new data that emerges. For this reason, MLOps is introducing continuous training (CT), a new practice aimed at automatically retraining models as needed, in addition to CI and CD. From the above, it's clear that MLOps is much more complex than DevOps and requires additional steps using data and models.

This is a software engineering approach that allows interoperable teams to build machine learning applications in new, smaller, and more secure versions based on code, data, and models, ensuring that they are replicated at any time in a short user-defined cycle. This approach includes collecting, selecting, and preparing data for use in model training, testing, and selecting the most efficient models after testing and experimenting, developing, and sending selected models to production.

E. MLOps Pipeline Components

1) Data Ingestion – Is the process of importing, amassing and storing records for fast use or saved in a database for similarly use. It offers with extracting records from numerous records sources(Example: database, records warehouse, records lake) and consuming the specified records for version training.

2) Data Preparation - It is also referred as "data preprocessing," this process is for transforming the raw data so that analysts and data scientists can run machine learning algorithms to generate insights and make predictions.

3) Model Training - A machine learning model is a file trained to recognize a particular type of pattern. It trains the model on a dataset and provides the model with algorithms that can be used to think and learn about that data.

4) Model Deployment - Deploying is a way to integrate a machine learning model into an existing production environment to make actionable business decisions based on data.

5) Operationalization - It's primarily related to the organization's managers and other parts that need to do to bring these machine learning models into production. As mentioned earlier, from a management perspective, operationalization can be thought of as MLOps.

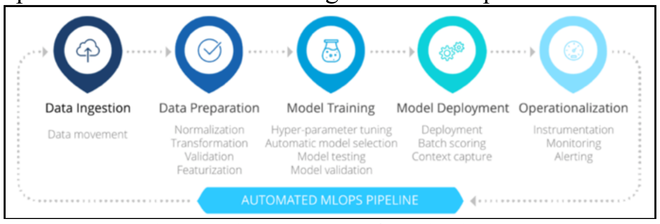

Figure 5: Automated MLOps Pipeline [7]

F. MLOps Testing

Traditional software is built by writing fixed rules against well-defined static assumptions about the world around them. Relatively simple to test each rule (unit test) or group (integration test) when everything is already defined in advance.

By definition of machine learning is establishing dynamic rules based on ever-changing data, which makes testing much more difficult. It's like trying to hit a moving target when it comes to machine learning testing. The behavior of the system is determined by the dynamic properties of the data as well as by the many configuration options of the model.

1) Data Testing

1.1) Data is just as important (if not more) than code in an ML effort. Data validation tests for training and inference inputs should be similar to unit tests for your code in that they specify and validate your assumptions about the inputs. All nulls, statistical outliers in a feature, and feature interactions should be checked.

1.2) For example, if your input is expected to be random English, one way to check is to calculate that "the" is the most common word, since this is a known assumption. If any other word appears more often than "the", it could mean that your data has an unexpected skew, or maybe some words are unintentionally Spanish. Another example of validating your assumptions might be that you make sure your input has an equal split between male and female data if that's a valid assumption for your instance.

2) Model Testing

2.1) Once the data assumptions have been tested, we can move on to testing the models and their training.

2.2) You can check the effect of each hyperparameter during training. Search grids or more complex search approaches can reveal reliability challenges while improving predictive performance. When possible, use separate test sets from training and validation sets.

2.3) As you deploy your model, test the relationship between your offline metrics and the model's actual real-world impact. The correlation between offline accuracy and your site's actual click-through rate can be measured in a small-scale A/B test.

3) Infrastructure Testing

3.1) By training two or more models in parallel with the same data and measuring the difference between the metrics, you can test the reproducibility of your training. Also test things like the ability to resume training from a crash checkpoint midway through, predictably. Remember to set up integrated smoke testing for the entire pipeline, from initial data validation to model deployment. These types of checks should be performed regularly and when deploying a new model version.

3.2) Reverting to a stable model can also be important under unexpected circumstances or due to human error. You should constantly test your recovery infrastructure, as it is your last line of defense when all other tests fail.

VI. Methodology

In this section we will be further discussing about the Model Deployment on GCP using Kubernetes and CI/CD Pipeline. Below are the detailed steps we followed to complete our objective.

1) Our Source code is in the GitHub. So, any commit we make the same changes are required to be in the running container. Therefore, any changes we make in GitHub we need to trigger the cloud build so it can update the existing image in container registry with the new changes. Further these changes will be adopted by the container.

2) Cloud build can import source code from variety of repository and cloud storage. For example GitHub and Bitbucket.

2.1) When updated source code reaches to the cloud build, it will forward that data as an image to the container registry. Also, these new changes will be reflected in the newly created container.

2.2) Parallelly cloud build will also trigger the Kubernetes manifest (deployment.yaml) file further used to create a pod. Each pod provide unique service, which are defined in the (service.yaml) file.

3) Again, cloud build will be called by the Kubernetes manifest file. In the next step a new container is deployed in Kubernetes cluster where deployment.yaml and service.yaml will be forwarded by the cloud build and image for the container will be called from the container repository.

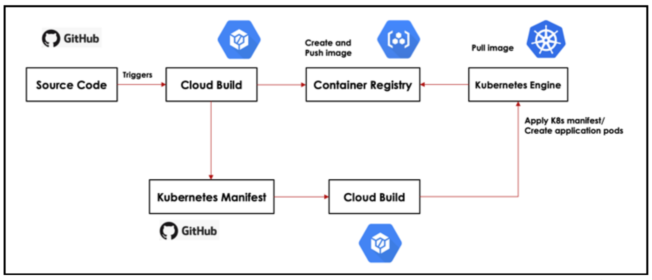

Figure 6: Model Deployment on GCP using Kubernetes and CI/CD Pipeline

VII. Conclusion

Research on continuous practice, especially continuous deployment, is gaining increasing interest and attention from software engineering researchers and practitioners, following a steady upward trend. In addition, we promote the implementation of continuous practice in the following ways. B. Reduce builds and test time in CI, increase visibility and awareness of build and test results in CI, support semi-automatic continuous testing, and detect violations, bugs, and errors in CI Address security and deployment pipeline scalability issues. Improves the reliability and reliability of the deployment process.

As part of this exploratory research, we gathered information on current MLOps and the key motivations behind operational processes that can facilitate clear and efficient workflows for deploying machine learning solutions. Incorporating ideas from traditional software development, especially CI / CD methods, we have introduced use cases where these methods may be useful in the machine learning development life cycles. Proposed three different levels Understanding that not all ML solutions require similar deployment work, the efficiency of MLOps and the number of resources required to set it up will differently. MLOps Layer 3 is the most mature solution proposed and is recommended for use in enterprise machine learning solutions. This level of granularity leverages industry-standard tools and techniques that have provided enterprise-level software solutions over the last decade, resulting in the most efficient and automated development workflow. In future work, we will create a demo of the proof-of-concept application to compare all three approaches for applications of various sizes.

Finally, MLOps is the maximum green manner to combine system studying fashions into production. Every year, extra corporations rent those methods, and an extra take a look is carried out in this area. MLOps, on the other hand, might also additionally have a one-of-a-kind use. An absolutely mature MLOps device with non-stop education can cause extra green and sensible ML fashions, further to their use in production. Choosing appropriate equipment for every operation is likewise a consistent issue. Although there are numerous papers and articles for the one-of-a-kind equipment it isn't clean to observe the recommendations and contain them withinside the maximum green manner. Sometimes we must pick between flexibility and robustness with the respective professionals and cons. Finally, tracking is a level that needs to be one of the most important factors of interest. Monitoring the country of the complete device, the usage of sustainability, robustness, fairness, and explainability is from our factor of view the important thing for mature, automated, strong, and green MLOps systems. For this reason, it's important to broaden fashions and strategies which allow this type of tracking inclusive of explainable systems studying fashions. AutoML is probably a sports changer in adult and performance hunting. For this reason, additional complete and meaningful research is needed to use AutoML with MLOps.

References

[1] Applying DevOps Practices of Continuous Automation for Machine Learning Link [Accessed: 13 July 2020]

[2] Understanding DevOps & bridging the gap from continuous integration to continuous delivery Link [Accessed: 22 May 2015]

[3] A qualitative study of DevOps usage in practice Link [Accessed: 28 June 2017]

[4] The DevOps Cycle Link

[5] A Comprehensive Guide on MLOps for Machine Learning Engineering Link [Accessed: 21 March 2022]

[6] On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models using MLOps Link [Accessed: 7 February 2022]

[7] On Our Mind: MLOps Link [Accessed: 17 May 2021]

[8] MLOps -- Definitions, Tools and Challenges Link [Accessed: 1 January 2022]

[9] Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices Link [Accessed: 5 March 2017]

[10] Continuous integration and continuous delivery explained Link

[11] MLOps: Continuous delivery and automation pipelines in machine learning Link