CI-CD Deployment

Milan Ples - 20088120

01-01-2023

Contents

1 Plagiarism Statement

I confirm that the work contained in this report is my own, except where due reference is made in the text to the work of others. I have acknowledged all material, data or ideas from other sources, whether quoted verbatim or paraphrased, by the use of suitable references and footnotes. I have also acknowledged any assistance received in preparing this report.

2 Acknowledgements

I would like to thank my supervisor, Patrick Felicia for his guidance and support throughout this project. My team at Unum who helped me immensely during my internship in executing the project and providing endless support and help.

3 Abstract

In this project, I will be using automation tools and technologies to streamline the process of creating software. I will create a working prototype that showcases the capabilities of these tools in a real-world context, and will identify best practices for using these tools in software development projects. I will also identify any potential challenges or limitations in using these tools, and will develop strategies to overcome these challenges. Through this project, I hope to demonstrate the capabilities and benefits of automation tools and technologies in software development, and to create guidance and resources to help other developers adopt these tools in their own projects.

4 Introduction

Continuous Integration-Continuous Deployment (CI-CD) is a software development approach that aims to automate the process of building, testing, and deploying code changes. This approach is designed to improve the efficiency and quality of the software development process by allowing teams to deliver updates and new features faster, with fewer errors.

One way to implement a CI-CD workflow is through the use of tools such as git for version control, Github for code repository management, Jenkins for continuous integration, Docker for packaging code into containers, and AWS for deployment. In this project, these tools are used together to create a streamlined process for building, testing, and deploying code changes. This enables developers to focus on writing code and adding new features, while the CI-CD pipeline handles the rest.

4.1 Purpose, Intended Use and Audience

The purpose of this CI-CD project is to automate the process of building, testing, and deploying code changes, in order to improve the efficiency and quality of the software development process. It is intended to be used by software development teams who want to streamline their workflow and deliver updates and new features to their users faster, with fewer errors.

The intended audience for this project is software developers and devops professionals who are responsible for building, testing, and deploying code changes. By implementing a CI-CD workflow, these individuals can focus on writing code and adding new features, while the pipeline handles the rest of the process. This allows them to work more efficiently and effectively, and helps to ensure that code changes are thoroughly tested and deployed in a timely manner.

Overall, the goal of this CI-CD project is to make the software development process more efficient, reliable, and scalable, enabling teams to deliver high-quality software updates to their users more quickly and with fewer errors.

4.2 Rationale for the Project

As a software developer and Dev-Ops enthusiast, I am well aware of the benefits of automating processes in order to optimize the development process and decrease expenses. When I was first introduced to continuous integration and continuous deployment (CI-CD) during my internship, I was immediately drawn to the potential this methodology had to revolutionize how code modifications are created, tested, and deployed.

Utilizing tools such as git, Github, Jenkins, Docker, and AWS, a CI-CD pipeline can automate the steps of building, testing, and deploying code changes, decreasing the time and effort needed to bring new features and updates to production. This not only increases the efficiency of the development process, but also helps to guarantee the reliability and quality of the final product.

I have therefore made it a priority to learn about and understand the implementation of a CI-CD pipeline. I am confident that this approach has the potential to greatly benefit any organization and its users, and I am eager to see the positive impact it can have on the software development process. By embracing automation and adopting a CI-CD workflow, I believe it is possible to significantly enhance the speed and reliability of the development process, while also reducing costs and minimizing errors.

4.3 Risk Assessment

The risks associated with this project are as follows:

As I embark on this project, I am aware that there is a risk that I may not be able to deliver it as planned. This risk is especially relevant due to my lack experience with the technologies and tools being used, as I may encounter difficulties or delays in implementing the desired functionality. Additionally, if the project is overly complex or involves a large number of features, it may be difficult for me to manage and could result in delays or cuting out features.

I also need to consider the possibility that external factors such as third-party APIs or vendor support may not be reliable or may change unexpectedly, which could impact the project's timeline and success. To increase the chances of delivering the project successfully, I will need to carefully assess and manage these risks. This may involve seeking additional resources or expertise, breaking the project down into smaller tasks, or implementing contingency plans. By proactively addressing these risks, I can increase the likelihood of delivering the project as planned.

5 Methodology

Waterfall is a traditional linear method of software development that involves completing each phase of the development process before moving on to the next, while CICD involves continuously integrating and deploying code changes.

5.1 Introduction:

Waterfall and CICD (Continuous Integration and Continuous Deployment) are two different development methodologies that are used to build software. While Waterfall is a traditional method that follows a linear, sequential process, CICD is a more modern approach that emphasizes rapid iteration and continuous delivery of software updates. In this article, we will compare the two approaches in terms of cost savings, speed, and workload.

5.2 Cost Savings:

One potential advantage of the Waterfall model is that it may be less expensive to implement, as it does not require the same level of infrastructure and automation as CICD. Waterfall also typically involves fewer resources, as teams do not need to constantly monitor and deploy code changes. However, the Waterfall model can be inflexible and may not be well-suited to projects with rapidly changing requirements or a high degree of uncertainty. As a result, teams may need to go back and redo work that has already been completed, leading to additional costs and delays.

On the other hand, CICD can offer significant cost savings by reducing the need for manual testing and deployment processes. By continuously integrating and deploying code changes, teams can quickly and easily make updates to their software and respond to changing business needs. This can also lead to reduced maintenance costs, as teams can more easily fix defects and make updates to the software as needed. However, CICD does require a significant investment in infrastructure and automation, as well as a well-defined process for testing and deploying code changes.

5.3 Speed:

The Waterfall model is typically slower than CICD, as each phase of the software development life cycle (SDLC) must be completed before moving on to the next phase. This can lead to delays and missed deadlines, particularly if there are issues or changes that require going back and redoing work that has already been completed.

In contrast, CICD allows for rapid iteration and continuous delivery of software updates, allowing teams to quickly respond to changing business needs or customer feedback. This can lead to faster time-to-market and the ability to deliver new features and functionality more quickly. However, it is important to ensure that code changes are thoroughly tested and do not break existing functionality, as this can lead to delays and additional work.

5.4 Workload:

The Waterfall model may require less workload upfront, as teams do not need to continuously integrate and deploy code changes. However, the inflexibility of the Waterfall model can lead to additional workload if changes or defects need to be addressed once a phase has been completed.

CICD, on the other hand, requires a more continuous workload, as teams must continuously integrate and deploy code changes. This can lead to a more complex and challenging workload, as teams must ensure that code changes do not break existing functionality and that the software is stable and reliable. However, the ability to rapidly iterate and respond to changing business needs can also lead to a more dynamic and engaging work environment.

5.5 Conclusion:

Both Waterfall and CICD have their own benefits and limitations, and the appropriate approach will depend on the specific needs and constraints of a project. Waterfall may be more suitable for projects with well-defined requirements and a low level of uncertainty, while CICD may be more appropriate for projects with rapidly changing requirements or a need for rapid iteration and innovation. Ultimately, the choice between the two approaches will depend on the specific cost, speed, and workload considerations of the project, as well as the resources and capabilities of the development team.

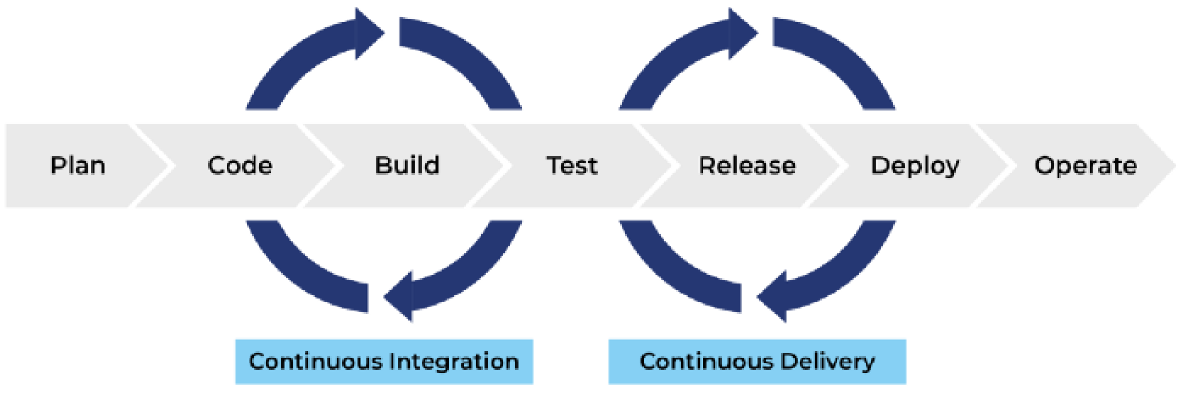

Figure 1: CICD Flow

6 Project Development cycle

I am developing this project with a focus on Agile principles, which prioritize flexibility and adaptability in the development process. By embracing an Agile approach, I aim to deliver value to stakeholders in a timely and efficient manner, while also allowing for rapid iteration and adaptation to changing requirements.

In order to achieve these goals, I am leveraging a range of tools and techniques to automate and streamline my development process. These tools include Docker, Jenkins, and Ansible, which allow me to easily package, deploy, and manage my application across multiple environments. By using these tools, I can ensure that my application is reusable, scalable, and maintainable, and that I can easily and consistently deliver updates and improvements to stakeholders.

Furthermore, I am adopting a DevOps mindset, which emphasizes collaboration and integration between development and operations teams. This allows me to optimize my workflow and continuously deliver value to stakeholders by automating and streamlining the build, test, and deployment process. By leveraging the full capabilities of these tools, I am able to deliver high-quality, reliable software that meets the needs of stakeholders and users.

6.1 Project Design Objectives

In this project, I am separating the flow and design into two distinct sections: automation of infrastructure and configuration automation.

For the automation of infrastructure, I am using tools such as Terraform to define and provision my infrastructure resources in a predictable, version-controlled, and automated manner. These tools allow me to use code to define and manage my infrastructure resources, such as servers, networks, and storage, in a repeatable and consistent manner. By using these tools, I can leverage the benefits of automation and version control to manage my infrastructure resources in a more efficient and reliable way.

For configuration automation, I am using tools such as Ansible to automate the configuration of my resources, such as applications, services, and environments. These tools allow me to define and enforce configuration standards, install and configure software, and manage users and permissions in an automated manner. By using these tools, I can reduce the risk of errors and improve the efficiency of my workflow, while also ensuring that my resources are consistently configured according to my desired standards.

6.2 Automated Tests

As part of my development process, I am using Jenkins to automate the execution of my tests. Jenkins is an open-source automation server that helps developers automate parts of the development process, such as building, testing, and deploying code changes. It allows me to set up a series of tasks, known as “jobs,” that can be automatically triggered based on certain events or conditions.

In my project, I am using Jenkins to automatically run my tests whenever I commit code changes to my repository. This allows me to quickly and easily validate that my code is working as intended, and identify and fix any issues early on in the development process. By automating my tests, I can save time and reduce the risk of errors, ensuring that my code is of high quality and meets my desired standards.

However, it is important to note that while the use of Jenkins to automate my tests is a useful aspect of my development process, it is not the primary focus of my project. There may be other goals and objectives that are more central to my project, such as developing new features or improving the performance of the application. Nonetheless, the use of Jenkins to automate my tests helps me to ensure that my code is of high quality and meets my desired standards, which is an important factor in achieving my overall project goals.

7 Technologies Used

7.1 Git

Git: Git is a version control system that allows developers to track changes to their codebase and collaborate with other team members. It allows developers to save different versions of their code and switch between them as needed, making it easier to track and manage changes. Git also allows developers to work on the same codebase at the same time, without having to worry about overwriting each other's work. This enables teams to work more efficiently and effectively, and helps to ensure that code changes are thoroughly tested and deployed in a timely manner.

7.1.1 Project Use:

In this project, Git will be used to manage and track changes to the codebase as the project progresses.

7.2 Github

GitHub is a web-based platform that allows developers to host and review code, manage projects, and build software. It is built on top of Git, a version control system that enables developers to track changes to their code. By using GitHub, I can easily keep track of the history of my work and make sure that my code is organized and properly versioned. When changes are made to the codebase on GitHub, Jenkins will be notified automatically through Git, allowing me to streamline my workflow and stay up to date on the status of the project.

7.2.1 Project Use:

For this project, GitHub will be used to store the codebase and track changes through automatic versioning. It will also serve as a trigger point for the rest of the pipeline, allowing changes to the codebase to initiate the various tasks and activities in the development process.

7.3 Github Actions

GitHub Actions is a tool that allows developers to automate their workflow by setting up a series of tasks, known as “actions,” that can be triggered based on certain events or conditions. These actions can be used to build, test, and deploy code changes, as well as perform other tasks such as releasing software and publishing documentation. I am using GitHub Actions to automate the compilation of my Final Year Project pdf upon commit to GitHub. This allows me to easily and quickly generate a compiled version of my project every time I commit code changes, without having to manually perform the compilation process.

7.4 Jenkins

Jenkins is an open-source automation server that helps developers automate parts of the development process, such as building, testing, and deploying code changes. It allows developers to set up a series of tasks, known as "jobs," that can be automatically triggered based on certain events or conditions. For this project, Jenkins will be used to create a Docker image and upload it to Docker Hub once it detects changes on GitHub. This allows the project to be easily packaged and deployed in a containerized environment, making it easier to manage and scale. Jenkins is a powerful tool for automating development workflows and can be used in conjunction with other tools, such as GitHub Actions, to create a complete CI/CD pipeline.

7.4.1 Project Use:

In this project, Jenkins will be used to automate the building, testing, and deployment of code changes to different environments. It will be triggered by changes to the codebase on GitHub, and will execute a series of tasks to build, test, and deploy the code. One of the environments that Jenkins will be used to deploy to is AWS EKS (Elastic Kubernetes Service), a fully managed Kubernetes service that makes it easy to deploy and run containerized applications.

7.5 Docker

In this project, we are using Docker to containerize the application. Docker is a tool that makes it easier to create, deploy, and run applications in a containerized environment. A container is a lightweight, standalone, and executable package that includes everything an application needs to run, including code, libraries, dependencies, and runtime. By using Docker, we can easily package and deploy the application in a consistent manner across different environments. This allows us to manage and scale the application more easily, as well as ensure that it runs consistently regardless of the underlying infrastructure. Widely used saying: "It works on my machine" is no longer valid. Docker allows us to create a consistent environment for the application, so that it will run the same way on any machine.

7.5.1 Project Use:

In this project, we are using Docker to containerize the application.

7.6 AWS

AWS (Amazon Web Services): AWS is a cloud computing platform that provides a wide range of services, including computing, storage, and database management. It allows developers to build, test, and deploy applications at scale, without having to worry about the underlying infrastructure. AWS also offers a number of tools and services specifically designed for continuous integration and continuous deployment (CI-CD), such as CodePipeline and CodeBuild.

7.6.1 Project Use:

In this project, AWS will be used to host a variety of servers and services like Elastic Kubernetes Service (EKS). AWS will also be used to host other servers and services as needed, such as Jenkins, and maybe Nexus. Overall, AWS will be a key tool for hosting and deploying the various servers and services needed to support this project.

7.7 Kubernetes

Kubernetes: Kubernetes is an open-source container orchestration platform that allows developers to manage and deploy containerized applications at scale. It provides features such as automatic scheduling, self-healing, and horizontal scaling, which make it easier for developers to deploy and manage applications in a production environment.

7.7.1 Project Use:

In this project, Kubernetes will be used to manage and deploy the containerized application. It will be responsible for scheduling and deploying the containers, as well as managing their lifecycle and ensuring that they are running in a healthy state. Kubernetes will be a key tool for ensuring the reliability and scalability of the application in this project.

7.8 Terraform

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. It is an open-source tool that allows users to define infrastructure as code (IaC) and manage it using a command-line interface. With Terraform, users can create, update, and version infrastructure resources such as virtual machines, networking components, and storage solutions across multiple cloud providers (such as AWS, GCP, and Azure) and on-premises data centers.

Using Terraform, users can define infrastructure resources in a configuration file written in the HashiCorp Configuration Language (HCL). This configuration file is used to create and manage resources in a consistent, repeatable way. Users can use Terraform to create, update, and delete resources in a predictable manner, making it easier to manage infrastructure changes and roll back updates if necessary.

Terraform also includes features such as dependency management, which allows users to specify the order in which resources are created, and resource tracking, which allows users to keep track of resources that have been created and modified. Overall, Terraform is a powerful tool for managing infrastructure in a consistent and repeatable way, making it easier for organizations to build and maintain infrastructure on a variety of cloud and on-premises platforms.

7.8.1 Project Use:

In this project, Terraform will automate resource deployment across cloud providers, including AWS. It will define and manage the infrastructure, including servers, databases, and networking resources. Terraform will streamline resource deployment and management for the project.

7.9 Ansible

In this project, we are also using Ansible to automate tool configuration and installation on remote devices. Ansible is a configuration management and automation tool that allows developers to easily provision, configure, and manage remote infrastructure. It works by using simple, human-readable configuration files called "playbooks" to define the desired state of the infrastructure and the tasks needed to achieve it.

7.9.1 Project Use:

In our project, we are using Ansible to automate the configuration and installation of the application on remote devices. This allows us to easily and consistently deploy the application to multiple devices without the need to manually perform the configuration and installation process on each device. Using Ansible helps us to save time and reduce the risk of errors by automating these tasks. It also allows us to easily update and manage the configuration of the application on the remote devices.

8 Other Tools used

8.1 Pandoc:

Pandoc is a tool that I am using in this project to help me convert files from one format to another. It allows me to easily convert files between different markup languages, such as Markdown, HTML, and LaTeX. This is particularly useful for me because it allows me to preserve the formatting and layout of my documents, even when I need to convert them to a different format.

8.2 LaTeX:

LaTeX is a typesetting system that I am using in this project to help me create high-quality documents with a consistent layout and formatting. It is particularly useful for documents that contain a lot of mathematical notation or other complex formatting. By using LaTeX, I can ensure that my documents look professional and are easy to read, even when they contain complex formatting.

8.3 Mermaid Diagrams:

A mermaid diagram, also known as a flow diagram, is a tool that visualizes the flow of data or steps within a system.

9 Expected Outcomes:

The primary goal of this project is to demonstrate the capabilities of modern automation tools and technologies, and to show how these tools can be used to streamline the development process and improve the efficiency and effectiveness of software development projects. To achieve this goal, the project will leverage a variety of automation tools and technologies, including Git and GitHub, Jenkins, Kubernetes, and Terraform, among others.

The expected outcomes of this project will include:

- The successful demonstration of the capabilities of automation tools and technologies in streamlining the development process and improving the efficiency and effectiveness of software development projects.
- The creation of a working prototype that showcases the capabilities of these tools and technologies in a real-world context.
- The identification of best practices for using automation tools and technologies in software development projects, and the creation of guidance and resources to help other developers adopt these tools in their own projects.
- The identification of any potential challenges or limitations in using automation tools and technologies, and the development of strategies to overcome these challenges.
- The identification of potential future developments or improvements for the project, including the exploration of new tools and technologies as they become available, and the integration of existing tools and technologies in new and innovative ways. Overall, the expected outcomes of this project will be to provide a clear and compelling demonstration of the capabilities and benefits of automation tools and technologies in software development, and to provide guidance and resources to help other developers adopt these tools in their own projects.

10 Conclusion

In this project, I am utilizing a variety of automation tools and technologies to optimize the development workflow and ensure the timely and high-quality delivery of the project. These tools include version control systems, build and test automation tools, deployment automation tools, and infrastructure management tools. These tools will be integrated into the overall workflow to automate various stages of the development process, such as version control, building and testing, deployment, and infrastructure management.

This project is part of a larger initiative to demonstrate the capabilities of modern automation tools and technologies. Its primary goal is to showcase how these tools can be used to streamline the development process and improve the efficiency and effectiveness of software development projects. To achieve this goal, I will be using a variety of technologies and tools, including Git and GitHub, Jenkins, Kubernetes, and Terraform, among others.

One of the main challenges being addressed in this project is the need to deploy and manage applications in a consistent and reliable manner across a range of different environments. To overcome this challenge, I am leveraging the automation capabilities of these tools to automate the deployment and management of the application and its dependencies. This will help to ensure that the application can be deployed and run smoothly in a variety of different environments, including on-premises, in the cloud, or in a hybrid environment.

Looking ahead, I plan to continue refining and improving the automation tools and technologies being used in this project. This will involve exploring new tools and technologies as they become available, as well as integrating existing tools and technologies in new and innovative ways. By staying at the forefront of automation technology, I hope to continue to drive the development of more efficient and effective software development processes, and to help organizations around the world realize the full potential of automation in their own projects.

11 References

- Oakland, J. (1987). The Economics of Non-excludability. The Journal of Economic Perspectives, 1(1), 19-32. doi:10.1257/jep.1.1.19