Software Development Leveraging DevOps

Introduction

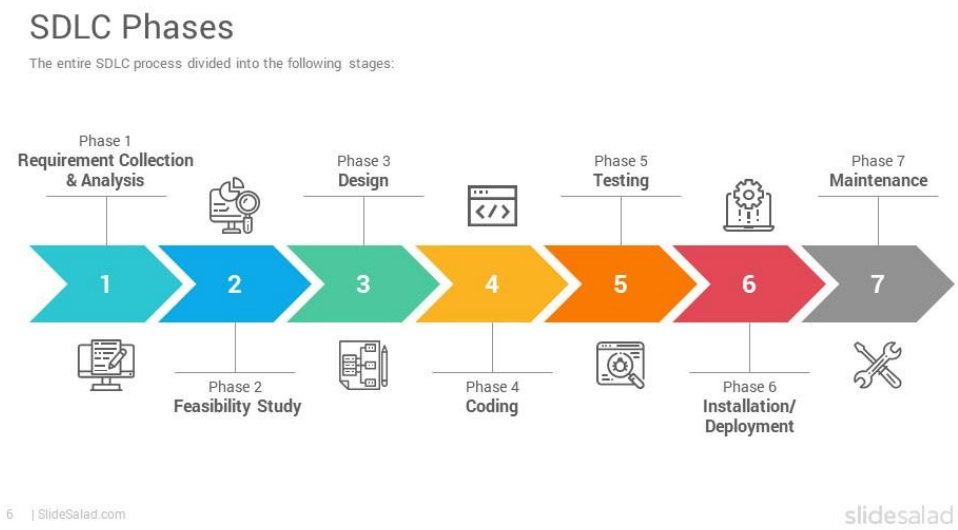
In this paper, I am trying to illustrate how the traditional Software Development Life Cycle (SDLC) which is highly structured with more focus on planning and less on customer value is becoming less adaptable to the growing velocity of changes needed in the market of software. The primary focus of this paper is to understand DevOps, the process to achieve a continuous iterative model, and how it is overcoming the limitations of SDLC by being more flexible to changes in less time.

Problem Definition and Discussion

It was during the 1960s when the rapid growth of the computer industry came to a point where a need for a framework to produce software was raised which grew into Software Development Life Cycle (SDLC).

SDLC consists of many stages which are analysis, design, coding, testing, deployment, and maintenance. [1]

In the traditional SDLC, these fore mentioned processes used to be sequential (Waterfall) i.e., every stage is separate and can only be reached if all the previous stages have been done. In other words, to bring a new feature to the software the team must go through every single stage in a sequential manner where one team has to wait until the other finishes its task.



SDCL Image [2]

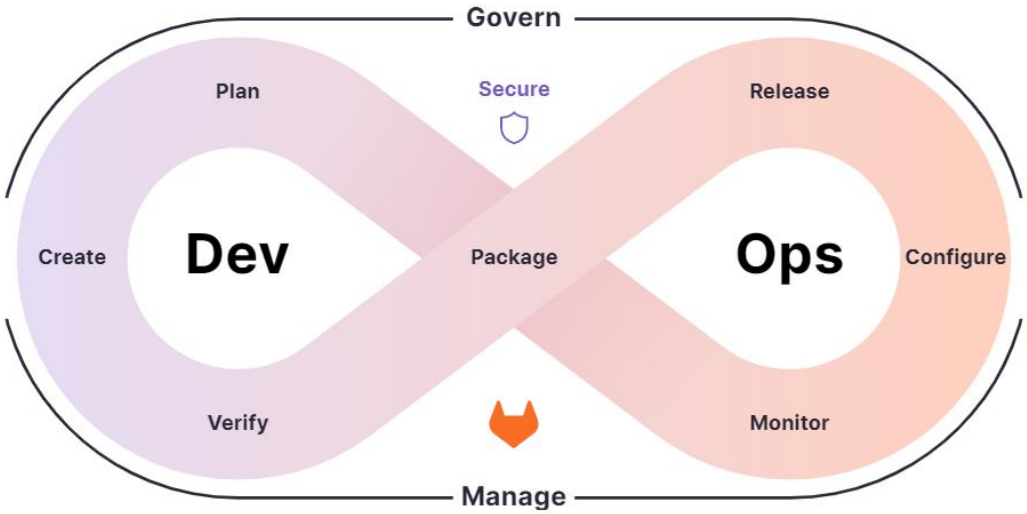
Limitations of the traditional model

1. Time-consuming: As discussed before the biggest drawback of traditional SDLC is the time taken to reach from the first to the last stage. Where every stage can only the started if all the previous stages are completed. This becomes more difficult if the product required frequent changes or the client is not sure about the requirements.
2. Less customer-focused: Customer feedback is only taken at the end stage. Rest all the staged do not involve any feedback from the client/customer.
3. Discrete teams: In a traditional SDLC each stage is taken care of by individual teams, the lack of a dynamic central organization structure leads to miscommunication and poor time management.

4. Reduced enhancement scope: There is less space for evolving or enhancing the product once the analysis and planning stage is completed everyone has to strictly follow it. [3]

Identified approaches to address the problem

The limitations of traditional SDLC can be addressed by an iterative (Agile) approach rather than a sequential one, where the development and the operation teams work hand in hand for a speedy delivery incorporating customer feedback. Hence the term DevOps.



DevOps Cycle [4]

DevOps aims for a much more automated SDLC where there is more collaboration between the development and operations teams. Improved scope for enhancements and more customer-focused

Advantages of DevOps:

1. Fast: leveraging some core DevOps practices such as continuous delivery and microservices to the changing requirement of customers or the market can be incorporated in lesser time
2. Improved delivery: the rate at which the team can deliver is much higher than traditional SDLC as it incorporates the potential of continuous integration and continuous delivery.
3. Scalable: DevOps utilizes cloud computing services to achieve much higher scalability.
4. Close collaboration: the increased coupling of teams reduces misinterpretation and miscommunication thereby improving the quality. [5]

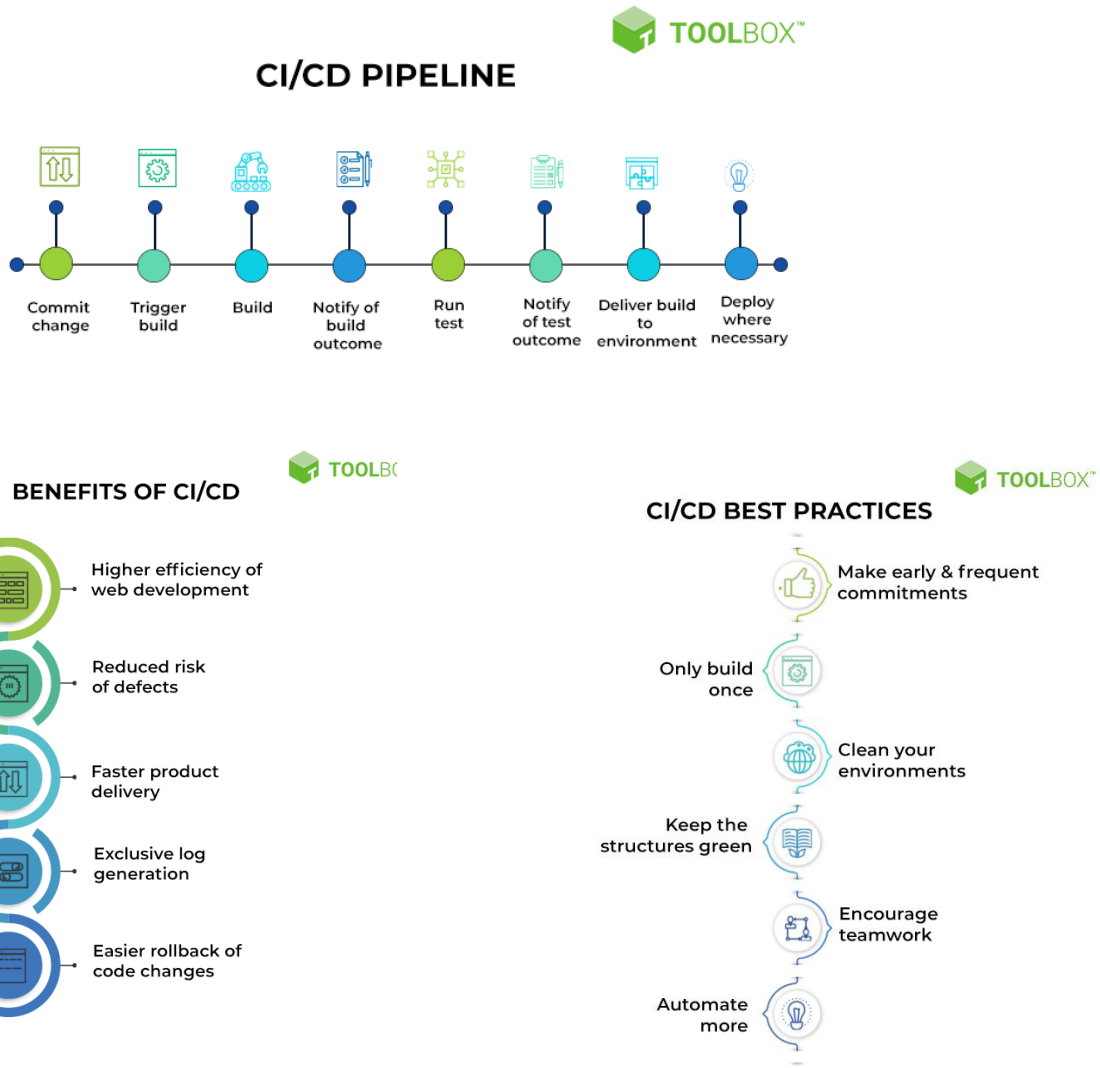
Some key practices in DevOps:

1. CI/CD: Continuous Integration and Continuous Delivery is a primary concept of DevOps where every change is continuously merged into the existing code and the changes are continuously tested and released.
2. Microservices: It is a process of designing an application by dividing them into numerous small services. They are developed separately and communicate using APIs
3. Monitoring: It is very important that there is continuous monitoring and logging as there is a higher rate of changes in the product. [5]

Solution Design

The solution being discussed in this paper is focused on a CI/CD pipeline using Azure DevOps. Let us have a close look at the same.

Continuous Integration and Continuous Delivery is a process that incorporates several tools like Azure, AWS, etc to achieve the key requirement of continuous integration of the changes made in the code and following a continuous deployment with automated build and test. Which keeps going in a repetitive manner with customer feedback involved.

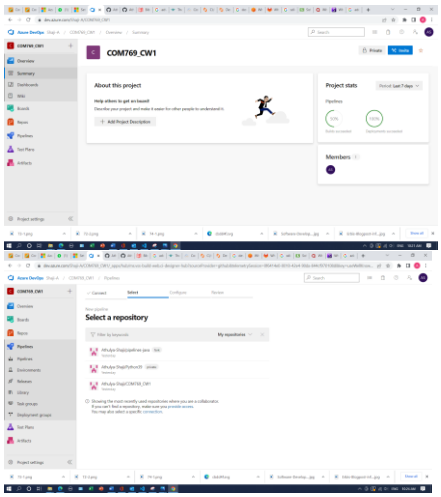


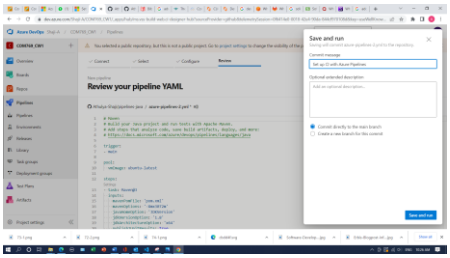
CI/CD Images [6]

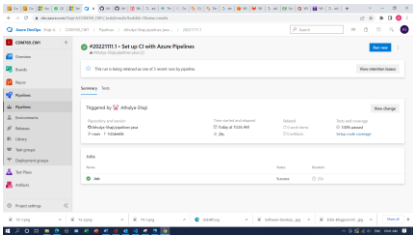
Let us discuss step by step process of creating a CI/CD pipeline for a java code from GitHub and using the Azure DevOps pipelines.

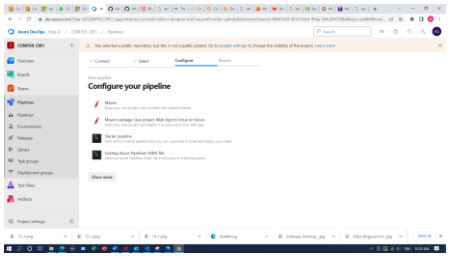
Log in to your Azure Account and navigate to Azure DevOps, under which select pipelines and click on create new pipelines (here COM769_CW1).

Following this select GitHub and the repository from the list. Select the Marvel pipeline template. Save and run and wait for the job to finish.



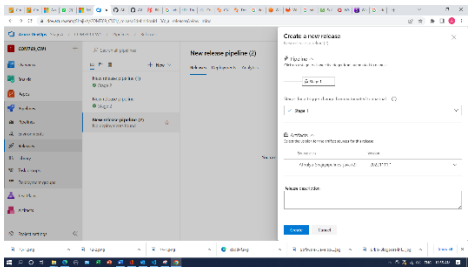


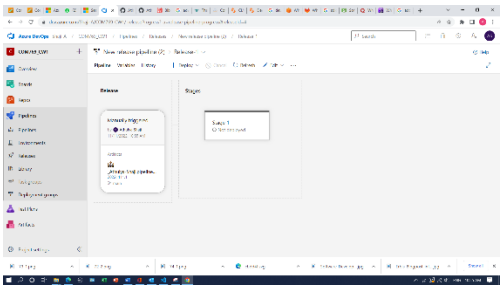


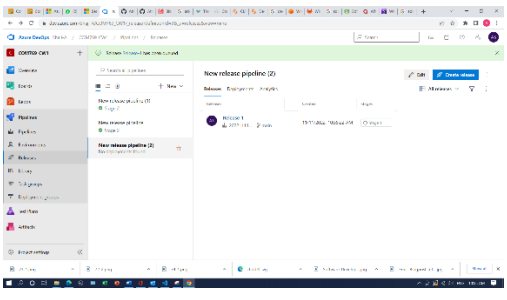


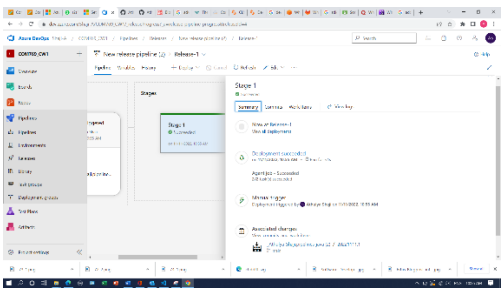
Following the creation of a pipeline build and test should be carried out which is not discussed here

To release the created pipeline, go to the release select the new release select the created pipeline and follow the wizard









Quantification of solution performance

Upon analyzing such a solution we can draw the following advantages:

1. Higher responding: the interaction of the code with the changing business requirement is very high compared to the traditional Software Development Life Cycle (SDLC).
2. Increased quality: the tools automated a lot of tasks making it easier to identify bugs.
3. Small SDLC cycles: the development and deployment cycle time span was reduced significantly making it easier to incorporate feedback
4. Monitoring and logging: the implementation of the CI/CD pipeline as above does strict monitoring and logging which is essential when it comes to the reliability of the application 24/7 which also has to undergo rapid changes.
5. Rollback: it is again connected to the last point made, as with the help of continuous monitoring and logging it is easier to go back to the previous state version in care of an error
6. Cost efficient: adding up all the fore mentioned advantages of CI/CD pipeline the cost of resources reduces compared to the traditional SDLC.[7]

Conclusion

CI/CD and DevOps as a whole can seem to be a really good fit for the current market of software development. But choosing the framework for software development highly depends on the nature of the application being developed.

Rapid changes are not always the priority, the monitoring required for such a system is high which could always not be a core business requirement. There is also a higher dependency on hardware requirements to incorporate changes sometimes.

As discussed in this paper a DevOps methodology can cater to a high range of modern software requirements in lesser time compared to traditional SDLC. But best to conclude that the selection of the development framework considering the key requirement of the customer/client has to be done smartly.

References

[1]Babeni, S. (2019, February 14). Continuous integration and continuous delivery (CI/CD): The modern DevOps-focused best practices. Ormuco; Ormuco Inc. https://ormuco.com/blog/continuous-integration-continuous-delivery-devops-best-practices
scalable

[4]de Kort, W. (2016). What Is DevOps? In DevOps on the Microsoft Stack (pp. 3–8). Apress.
scalable

[7]Longbottom, C. (2021, April 22). The pros and cons of CI/CD pipelines. SearchSoftwareQuality; TechTarget. https://www.techtarget.com/searchsoftwarequality/tip/The-pros-and-cons-of-CI-CD-pipelines
scalable

[2]Mohanan, R. (2022, April 1). What is CI/CD? Definition, process, benefits, and best practices for 2022 |. Spiceworks.com. https://www.spiceworks.com/tech/devops/articles/what-is-ci-cd/
scalable

[3]Software Development Life Cycle Models Google Slides Template. (n.d.). SlideSalad. Retrieved 11 November 2022, from https://www.slidesalad.com/product/software-development-life-cycle-models-google-slides-template/
scalable

[5]What are the limitations of traditional SDLC and how can they be good to businesses? (n.d.). Quora. Retrieved 11 November 2022, from https://www.quora.com/What-are-the-limitations-of-traditional-SDLC-and-how-can-they-be-good-to-businesses
scalable

[7](N.d.). Amazon.com. Retrieved 11 November 2022, from https://aws.amazon.com/devops/what-is-devops/
scalable