**CI/CD PIPLINE USING K8**

|  |  |
| --- | --- |
| Project Mentors | 1) Animesh Kumar |
| Project Heads | 1) Chinmayi C. Ramakrishna  2) K. Keerthana  3) Neha G. Pai  4) Annette Elsa Manoj |
| Mentees | 1)  2)  3) |

**Aim**

We are trying to create a CI/CD pipeline using Kubernetes engine.

**Methodology:**

* Creating the Application files
* Installing Jenkins, Ansible and Docker
* Configuring Jenkins User to connect to the Cluster
* Creating the Jenkins Pipeline Job
* Configuring Jenkins Credentials for GitHub and Docker Hub
* Creating the JenkinsFile
* Testing our CD pipeline

**Timeline**

|  |  |  |  |
| --- | --- | --- | --- |
| **Checkpoint No.** | **Deadline** | **Goal/Objective** | **Comments**  **(if any)** |
| 1 | \*31st August, 2020 | Installation and creating the application files |  |
| 2 | \*30th November, 2020 | Creating the JenkinsFile |  |
| 3 | \*End of January | Testing the pipeline |  |
|  |  |  |  |

**Literature Review**

This section should consist of a brief description of related works along with a citation of the source (paper or article) you have referred to. Include at least 3 different sources. Avoid references of unreliable sources like Wikipedia.

An example has been shown below. The project is titled ‘Design and Development of an Electric Scooter’. Edit this out once changes are made.

**Introduction**

A CI/CD Pipeline implementation, or Continuous Integration/Continuous Deployment, is the backbone of the modern DevOps environment. It bridges the gap between development and operations teams by automating the building, testing, and deployment of applications. DevOps is a software development approach which involves continuous development, continuous testing, continuous integration, continuous deployment, and continuous monitoring of the software throughout its development lifecycle. When you have a team of developers, each of whom is responsible for a separate feature, you need to integrate the different features before you’re ready for a release. By integrating so frequently, your team can surface errors earlier. And when those are caught, the amount of backtracking needed to find the cause is also much reduced. Therefore, your team can resolve the integration errors much faster.This is the process adopted by all the top companies to develop high-quality software and shorter development lifecycles, resulting in greater customer satisfaction, something that every company wants.

**References**

[1] A. F. Nogueira, J. C.B. Ribeiro, M. A. Zenha-Rela and A. Craske, "Improving La Redoute's CI/CD Pipeline and DevOps Processes by Applying Machine Learning Techniques," 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC), Coimbra, 2018, pp. 282-286, doi: 10.1109/QUATIC.2018.00050.

The complexity inherent to software development and maintenance - not only in technical terms, but also from a human perspective - entails challenges that can be addressed as learning problems. Machine learning techniques may be employed as tools to gain insight about strategies that can lead to the improvement of the quality of software processes and products. Defect proneness prediction, in particular, may be identified as an active research field. As stated by DevOps guidelines, the possibility of obtaining quick feedback allows teams to operate in an agile mode in which communication, decision taking and problem solving are expeditious, allowing companies to boost business value. This paper describes ongoing research for applying machine learning techniques to improve the quality of processes and products inside the DevOps pipeline of the La Redoute's IT department.

[2] A. Steffens, H. Lichter and J. S. Döring, "Designing a Next-Generation Continuous Software Delivery System: Concepts and Architecture," 2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE), Gothenburg, Sweden, 2018, pp. 1-7.

Continuous Integration and Continuous Delivery are established practices in modern agile software development. The DevOps movement adapted these practices and places the deployment pipeline at its heart as one of the main requirements to automate the software development process and to deliver and operate software in a more robust way with higher quality. Over the time a lot of systems and tools have been developed to implement the deployment pipeline and to support continuous delivery. But software development is complex, its process even more and due to the individual organization of software vendors no real all-in-one solution for CD exists.

[3] S. A. I. B. S. Arachchi and I. Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management," 2018 Moratuwa Engineering Research Conference (MERCon), Moratuwa, 2018, pp. 156-161, doi: 10.1109/MERCon.2018.8421965.

Agile practices with Continuous Integration and Continuous Delivery (CICD) pipeline approach has increased the efficiency of projects. In agile, new features are introduced to the system in each sprint delivery, and although it may be well developed, the delivery failures are possible due to performance issues. By considering delivery timeline, moving for system scaling is a common solution in such situations. System scale requires current system benchmark status and expected system status. Benchmarking the production is a critical task, as it interrupts the live system. The new version should go through a load test to measure expected system status.