International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 11 | November - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins

Miss. Kshitija Patil, Miss. Sayali Kapadnis, Mr. Ravindra Waghmare, Mr. Harshal Thakare, Prof. Mr. Rahul Raut

Department of Information Technology
Bachelor Engineering
Sandip Institute of Technology and Research Centre, Nashik

1. Introduction

Today, cloud computing is the preferred option for the majority of businesses. DevOps techniques have been developed as a result, requiring developers to collaborate closely with network engineers to ensure the quick and secure deployment of their applications. Many organizations adopt DevOps and Continuous Integration and Continuous Deployment (CI/CD) techniques to address the slow delivery of services to customers as in Traditional IT approaches, which better meets customer needs. Thanks to the high flexibility and scalability of the infrastructure in the Cloud, DevOps, which unifies developers and operations teams, can bridge the gap between them. Automation is a representation of the CI/CD central approach. This does away with the requirement for human checks, but it does not come without security risk.

2. Related Work / Literature Review

1] Author: Artur Cepuc, Robert Botez ,Ovidiu Craciun in the Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services published in the year 2020.

Cloud computing has evolved into one of the most ubiquitous concepts in the IT industry. Its success lies in on-demand services rather than deploying a whole infrastructure which would bring additional costs like purchasing and maintaining the equipment, paying the employees and so on. The National Institute of Standards and Technology (NIST) classifies whether or not a service is a cloud service according to the following characteristics[1] broad network access, rapid elasticity, measured service, on-demand self-service and resource pooling. There are three distinct models available for the cloud services[2].

- Infrastructure as a service (IaaS)
- Platform as a service (Paas)

- Software as a service (Saas)

Future Work : An automated CI/CD pipeline for deploying a Java-based web application on AWS is presented in this paper. While the DevOps best practices were followed, using Ansible in the CD process added something new. We were able to improve overall scalability and send commands directly to the Kubernetes cluster because of this. The results of the experiments demonstrated that the proposed solution is fast, scalable, and reliable, with no downtime. As a result, in just 37.6 seconds, any change to the application's source code is automatically detected and sets off a whole series of six different technologies. The system rolls back the most recent stable version whenever a Jenkins job fails to deploy a new version of the application.

2] Author: Sai Priya Sinde, Bhavika Thakkalapally, Meghamala Ramidi, SowmyaVeeramalla in Continuous Integration and Deployment Automation in AWS Cloud Infrastructure published in 11/06/2022.

The motivation behind this exploration was to research the way in which programmers expect to use GitHub actions to robotize work processes and the effect of activity reception on pull demands. They accumulated and broke down information from 3,190 dynamic GitHub repositories and found that small set of these repositories utilized GitHub actions, as well as 708 extraordinary predefined actions being utilized in work processes. They additionally assembled and dissected GitHub actions related issues, finding that most of the remarks were positive. By and large, the discoveries show that GitHub's actions were generally welcomed by designers.
This article examines the many methodologies and innovations that can be utilized to fabricate a fruitful CI/CD pipeline. They dealt with various apparatuses as well as the issues they confronted while carrying out ceaseless improvement processes in programming 4 advancement using DevOps. This paper presents the thought of on-request benefits, which alludes to the utilization of cloud assets on request and the capacity to scale assets as per request, as well as a survey of safety research in the field of cloud security[7].

Future Work: The approach we used to deploy an application is by using GitHub actions which minimize the number of tools used. Using GitHub actions as a CI/CD pipeline with the AWS EKS cluster, the Flow of deployment is continuous and errorless. The time taken for the deployment of an application with Jenkins as a CI/CD pipeline is 1 min 43 sec approximately. But the time taken for our approach to deploy an application is around 25 sec to 45 sec.

3] Author: S.A.I.B.S. Arachchi , Indika Perera in Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management.

From Department of Computer Science and Engineering, University of Moratuwa, Moratuwa, Sri Lanka in 30 July 2018

Agile processes with continuous integration and continuous delivery (CICD) have increased the efficiency of projects. In agile mode, new features are introduced to the system in each sprint delivery, and although it may be well developed, a delivery may fail due to performance issues. Due to the delivery timeline, a common solution in such situations is to switch to system scaling. But how much should the system scale? System scaling requires the current state of the system benchmark and the expected state of the system. Production benchmarking is a critical task because it interrupts a live system. The new version should go through a stress test to measure the expected health of the system. Traditional stress test methods are unable to identify production performance behavior due to simulated traffic patterns that deviate significantly from production. To overcome these issues, this approach extended the CICD pipeline to have three phases of automation called benchmark, load test, and scaling. It minimizes system disruption by using a test rig approach when benchmarking the system and using production traffic for stress testing that provides more accurate results. Once the benchmark and stress test phases are complete, system scaling can be evaluated[5]. Initially, the pipeline was developed using a Jenkins CI server, a Git repository, and a Nexus repository with Ansible automation. GoReplay is then used to duplicate traffic from the production environment to the test environment. Nagios monitoring is used to analyze system behavior at each stage, and the test bench result has shown that the scaling is able to handle the same application software change load, but does not significantly optimize the application response time. helps reduce application deployment risk by integrating this three-phase approach as an extended CICD automation feature. The research thus provides an effective way to manage CICD projects based on an agile approach[5].

4] Author: Sriniketan Mysari, Vaibhav Bejgam in Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible.

From School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India in 27 April 2020

Automation is what every company needs to handle large projects where two or more people work together. CI/CD using Jenkins ansible is an optimized way to build and host any web applications. This document focuses on building and integrating with Jenkins with the pipeline methodology and deploying with Ansible and gives you its basic skeleton. Integrating with Jenkins is an easy task because it is open source and has many plug-in options[4]. Ansible is a configuration management tool that is also open source, and the configuration format is YAML, which can be easily readable by humans. Saving resources like time is a big challenge facing a developing company, so automating integration and deployment with Jenkins ansible saves a lot of time and also if there are frequent changes to the project, it can be easily updated. With Ansible, the shh approach is simple and can also run on a Jenkins node[8].

Future Work: Jenkins with a pipeline methodology for building and integration is one of the best automation processes because it is easily configurable, open source, user friendly, platform independent, flexible, saves a lot of time and also helps developers to build and test software. continuously. Deploying with ansible crowns is best in class because it is open source, efficient, powerful, easy to configure, automatic deployment and agentless. CI/CD using Jenkins ansible is an efficient and optimized way to build and host web applications because it accelerates development with better software quality.

5] Author: Mohammad Rizky Pratama, Dana Sulistiyo Kusumo in Implementation of Continuous Integration and Continuous Delivery (CI/CD) on Automatic Performance Testing.
From School of Computing, Telkom University, Bandung, Indonesia in 06 September 2021.

A performance test is one of the testing components in the development phase to ensure that the services created can meet the specified performance target workload and avoid performance bottlenecks. Nowadays, with the development of agile development processes, development processes can run faster and iteratively. Continuous Integration and Continuous Delivery (CI/CD) are methods used in agile development to automate and accelerate the following processes: creating, testing and validating services. This research aims to implement CI/CD into a performance test. In an existing test, the test still needs people to run the test. Our proposed solution for performance testing with CI/CD can be done automatically and reduce the human role in the test[6]. Implementation of CI/CD in performance test enables integration, automatic and periodic execution of test

processes. It can also respond quickly to changes in parameter values that may prevent the tester from setting new parameters in a new test scenario[8].

6] Author: Noor Fathima F., H.Y Vani in Building, Deploying and Validating a Home Location Register (HLR) using Jenkins under the Docker and Container Environment.
From Department of Information science and Engineering, JSS Science and Technology University, Mysuru, India in 07 October 2020.

This article introduces using Jenkins in a Docker and Container environment to build, deploy, and verify the Home Location Register (HLR). The proposed diagram shows how to use Jenkins, an open-source automation server in a Docker and Container environment rather than using a virtual machine. This paper explores the continuous integration of the Jenkins Pipeline to build, deploy, test, and validate HLR, breaking the entire project into smaller tasks. Each order fulfills sub-tasks of the project that is in process. The Home Location Register (HLR) is a database that contains administrative information that includes details about each participant as well as their last known location. The HLR contains information regarding the International Mobile Subscriber Identity (IMSI) and International Mobile Station Subscriber Directory Number (MSISDN) for each subscription[6]. The IMSI number is the primary key that uniquely identifies each entry in the HLR. A list of phone numbers for each subscription is stored in MSISDN. Other information stored in the HLR is about the services requested by the corresponding subscriber. HLR serves as the main component of the GSM architecture.

Future Work: Proposed idea “Building, deploying and validating HLR using Jenkins in a Docker and container environment” is very efficient as this project provides automated approach to complete the entire task by performing a pipeline and subsequently dividing the main task into smaller tasks and creating multiple tasks to perform these subtasks. Jenkins takes care of passing control from one job to another pipeline job based on whether the job passes or fails. His more efficient than docker and container technology it is used because it promotes isolation, flexibility, portability and is also scalable.

7] Author: Artur Cepuc, Robert Botez, Ovidiu Craciun, Iustin-Alexandru Ivanciu, Virgil Dobrota in Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins, Ansible and Kubernetes.

From Communications Department, Technical University of Cluj-Napoca, Cluj-Napoca, Romania in 22 January 2021.

Nowadays, cloud computing has become a solution for most businesses. This has led to the introduction of DevOps techniques, in which developers work closely with network engineers to ensure rapid and reliable deployment of their applications. This document presents the entire automated pipeline, starting from detecting changes in the source code of a Java-based web application, creating new resources in a Kubernetes cluster to host this new version, and finally deploying a containerized application in AWS. The solution follows DevOps best practices and relies on Jenkins in the continuous integration phase. What's new is that we used Ansible for continuous deployment, increasing scalability and overall ease of use. The solution ensures zero downtime and proves itself quickly, despite combining six different technologies and requiring very few computing resources[3].

1. Motivation

- The motivation behind this project is to build a pipeline that drives software development through a path of building, testing, and deploying code, also known as CI/CD.
- By automating the process, the objective is to minimize human error and maintain a consistent process for how software is released.
- Tools that are included in the pipeline could include compiling code, unit tests, code analysis, security, and binaries creation.
- For containerized environments, this pipeline would also include packaging the code into a container image to be deployed across a public cloud.

2. Objective

- The originality is that we used Ansible for continuous deployment
  increasing scalability and overall ease of use.
- The solution ensures zero downtime and proves itself quickly, even when combined different technologies and requires very little computing resources.
- The solution follows DevOps best practices and relies on Jenkins Phase of continuous integration.
- This led to the introduction of DevOps techniques in which developers work closely with network engineers to ensure rapid and reliable deployment their applications.

3. Problem Statement

Nowadays, cloud computing has become the go to solution for most enterprises. This has led to the introduction of DevOps techniques in which developers work closely with network engineers in order to ensure fast and reliable deployment of their applications. In order to tackle the slow delivery of services to customers as in Traditional IT approaches, many organizations adopt DevOps and Continuous Integration and Continuous Deployment (CI/CD) techniques which better meet customer needs. The gap between developers and operations teams could be filled by DevOps, which combines them into one unit, thanks to the high flexibility and scalability of the infrastructure in the Cloud. The CI/CD central approach is represented by automation. This eliminates the need for human controls, but is not without security risks.

4. Proposed Approach /Methodology

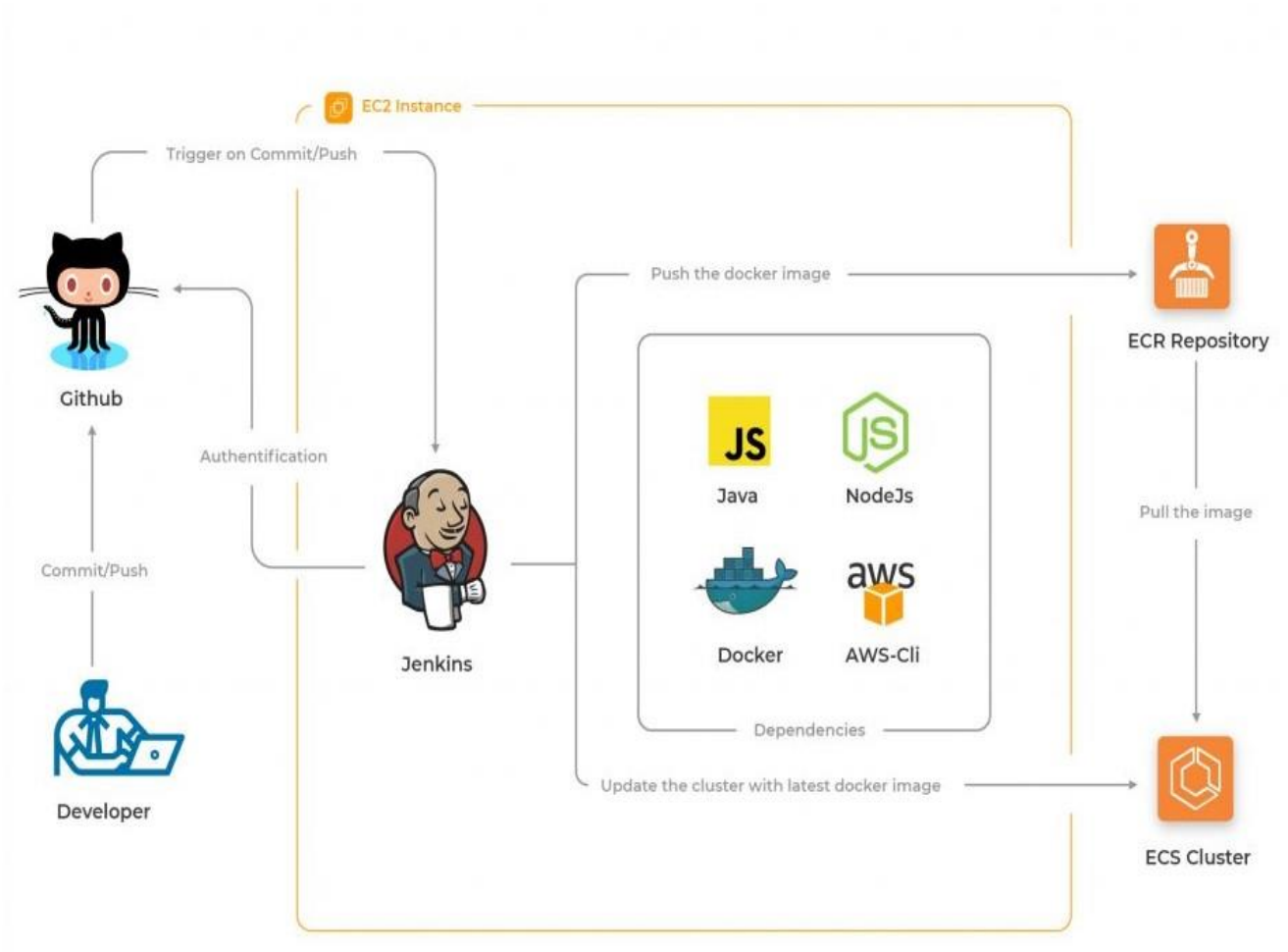

Fig 1.1 Architecture Diagram

This is how our architecture will look like after setting up the CI CD Pipeline with Docker.

5. Hardware Requirement

- **Processor:** 1 gigahertz (GHz) or faster processor
- **RAM:** 4 gigabytes (GB) for 32-bit or 8 GB for 64-bit
- **Hard disk space:** 512 GB for 32-bit OS or 1 TB for 64-bit OS

6. Software Requirement

- **GitHub:** At a high level, GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. To understand exactly what GitHub is, you need to know two connected principles:
  1. Version Control
  2. Git

- **Jenkins:** Jenkins is a tool that is used for automation, and it is an open-source server that allows all the developers to build, test and deploy software. It works or runs on java as it is written in java. By using Jenkins, we can make a continuous integration of projects(jobs) or end-to-endpoint automation.

- **Ansible:** Ansible is a simple but powerful configuration management and orchestration tool. Ansible is mainly used for the automation of cross-platform computer support tasks. It is fundamentally intended for IT professionals, who use it for configuration management, cloud provisioning, application deployment, intra-service orchestration, updates on workstations and servers, and nearly for anything a systems administrator does on a day-to-day basis.

- **Amazon Web Services:** Amazon Web Services (AWS) is a cloud service from Amazon, which provides services in the form of building blocks, these building blocks can be used to create and deploy any type of application in the cloud.

- **Docker:** Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications

7. Project Plan

Weeks	Work Plan
1-3	Requirement Analysis of software's and other hardware's for project.
2-10	An overview of the technologies needed for our project.
11-18	System Design and Planning Architecture
19-26	Implementation of Planned Architecture
27-30	Feedback Protocol Development and Reassessment
30-40	Check overall progress of work and revised strategies if necessary. Documentation and paper/patent filing.

8. References

REFERENCES

[1] "The NIST Definition of Cloud Computing", Computer Security Resource Center, NIST 2020, [Online], Available: https://csrc.nist.gov/publications/detail/sp/800-145/final.

[2] S. Watts, M. Raza, "SaaS vs PaaS vs IaaS: What's The Difference & How to Choose", BMC Software, 2019, [Online], Available: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-thedifference-and-how-to-choose/.

[3] Artur Cepuc, Robert Botez, Ovidiu Craciun, Iustin-Alexandru Ivanciu and Virgil Dobrota, "Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins, Ansible and Kubernetes", 19th RoEduNet Conference: Networking in Education and Research, 2020.

[4] N. Seth and R. Khare, "ACI (automated Continuous Integration) using Jenkins: Key for successful embedded Software development", *2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS)*, pp. 1-6, 2015.

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

[6] Rajdeep Dua ,A Reddy Raja,Dharmesh Kakadia,"Virtualization vs Container- ization to support PaaS", 2014 IEEE International Conference on Cloud Engineering.

[7] Sai Priya Sinde, Bhavika Thakkalapally, Meghamala Ramidi, Sowmya Veeramalla

"Continuous Integration and Deployment Automation in AWS Cloud Infrastructure" 30 June 2022. [online] Available: https://www.semanticscholar.org/paper/Continuous-Integration-and-Deployment-Automation-in-Sinde-Thakkalapally/8c984945d423208724342c175d609797000a349b

[8] Sriniketan Mysari; Vaibhav Bejgam "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible" 27 April 2020. [online] Available: https://ieeexplore.ieee.org/document/9077670