Deploying Application Using CI/CD Tools on AWS

ABDULLAH, MOHAMMAD ZEESHAN, ANSARI ABDURRAHMAN,
IMRAN AKHTAR, SALMAN BAIG
Bachelor of Computer Engineering, Maulana Mukhtar Ahmad Nadvi
Technical Campus Malegaon, INDIA

Abstract: — These days, most businesses turn to cloud computing as their go-to answer. Due to this, DevOps methodologies in which developers work have been introduced. together with network engineers to guarantee their applications are deployed quickly and reliably. Continuous Integration and Delivery (CI/CD) pipelines, which automate the build process on specialized equipment, have been shown to yield a number of benefits, such as quicker release cycles, higher productivity, and early fault discovery. This work describes an automated pipeline from scratch, beginning with the detection of modifications in the source code of web applications, building new. This new version will be hosted using resources in the EC2, and the application will eventually be deployed in AWS. The solution uses Jenkins for Continuous Integration and adheres to DevOps best practices. We highlight two main issues that we encountered: lengthy wait times for builds and releases to be queued and finished.

Key-words: — DevOps, Amazon Web services, Continuous Integration, Continuous Deployment, Continuous delivery, Jenkins.

Received: March 19, 2024. Revised: August 29, 2024. Accepted: September 19, 2024. Published: October 17, 2024.

1. Introduction

In the IT sector, cloud computing has become one of the most pervasive ideas. Its success is attributed to its on-demand services rather than the deployment of a full infrastructure, which would entail extra expenses for things like hiring staff, paying for equipment, and so forth. 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. The cloud services are available in three different models [2]:

- Infrastructure as a service: the customer is provisioned with compute, storage and networking in order to manage operating systems (OS), middleware, runtime, data and applications.
- Platform as a service : in this model, the customer is additionally provisioned with OS, middleware and runtime for managing data and applications. This model is similar to the concepts of serverless computing
- Software as a service: compared to the previous models, SaaS delivers applications that are managed by a third-party provider directly to the customer.

With increasing competition in software market, organizations pay significant attention and allocate resources to develop and deliver high-quality software at much accelerated pace [3]. Some of the techniques targeted at assisting organisations in speeding up the development and delivery of software features without sacrificing quality include continuous integration (CI), continuous delivery (CDE), and continuous deployment (CD), referred to as continuous practices for the purposes of this study [4]. Whereas CDE and CD focus on the capacity to swiftly and reliably distribute value to consumers by introducing as much automation support as feasible, CI promotes integrating work-in-progress several times a day.[5], [6]. Continuous practices are expected to provide several benefits such as: (1) getting more and quick feedback from the software development process and customers; (2) having frequent and reliable releases, which lead to improved customer satisfaction and product quality; (3) through CD, the connection between development and operations teams is strengthened and manual tasks can be eliminated [7], [8]. A growing number of industrial cases indicate that the continuous practices are making inroad in software development industrial practices across various domains and sizes of organizations [7], [9], [10]. The main purpose of this project is to automate the development process of a project that includes the phrases of building the code, testing and deployment of the project which need more time when doing these tasks manually. Eight builds are involved which would take a huge time if done manually.

JetBrains TeamCity is a user-friendly and a powerful Continuous Integration and Continuous Deployment server that works out of the box [11]. It can run parallel builds simultaneously on different environments and platforms. Some of the main advantages of TeamCity are optimizing the code integration, reviewing on-the-fly test results reporting, using intelligent tests re-ordering, running code coverage, finding duplicates, customizing statistics on build duration, code quality, success rate and custom metrics. TeamCity contains an integrated builds antifactory repository.

Bamboo Atlassian is a CI tool which is suitable for multiple programming languages and other popular technologies like Docker, Amazon S3 and AWS Code Deploy. The user can choose from a big variety of tasks for both builds and deployment projects. Bamboo represents a CI server which can be connected with other Atlassian products and can be used to automate the release for a software application. According to [12], Bamboo uses the concept of a

plan with tasks and jobs in order to configure the actions in the workflow.

Jenkins is an open-source automation tool written in Java with different plugins developed for the purpose of continuous integration [13]. It is widely used for building and testing software projects continuously, facilitating the integration of changes in the project and obtaining a new build by developers. Jenkins integrates with a large number of testing and deployment technologies (such as Selenium, Kubernetes, etc.) offering an easy way to configure an environment of continuous integration or delivery (CI/CD) for almost any combination of programming languages or repository hosting services using pipelines. It can work in master-slave architecture and in this type of architecture a master node can execute jobs on multiple slaves running on different platforms. Although Jenkins does not eliminate the need of creation of scripts for individual steps, it brings a faster and more robust way to integrate the entire chain of tools for building, testing and deployment.

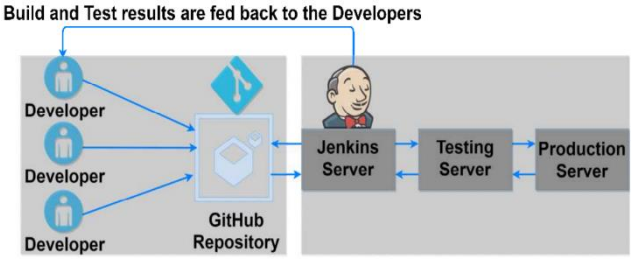

Fig. 1. *Jenkins Architecture [26]5*

An example of continuous integration with a Jenkins server is presented in Fig. 1. The developers commit changes to the source code to a source code management (SCM) tool such as GitHub. The changes are verified and validated (or not) by the Jenkins server and the builds with the test results are fed back to the developers.

TABLE I. *Comparison of Integration Tools*

	Team City	Bamboo	Jenkins
Open Source	No	No	Yes
Ease of Use	5/5	4/5	3/5
Built in Features	5/5	4/5	2/5
Integration	338	221	1447
Support	5/5	5/5	4/5
Cloud Support	Yes	Yes	Yes
Pricing	From $299	From $888	Free

2. Research Method

We employed one of the most popular research techniques in Evidence Based Software Engineering (EBSE), Systematic Literature Review (SLR).

A clear method for locating, assessing, and interpreting all relevant evidence for a given research question or topic is what systematic literature review (SLR) attempts to provide. There are three primary stages to this research method: developing a review protocol, carrying out a review, and reporting a review.

2.1 Search Strategy

In order to retrieve as many relevant studies as possible, we defined a search strategy [14], [15]. The elements of the search technique that were employed for this review are as follows:

1) Search Method

We used automatic search method to retrieve studies in six digital libraries (i.e., IEEE Xplore, ACM Digital Library, Springer Link, Science Direct, and Scopus) using the search terms introduced in Section III.B.2. We complemented the automatic search with snowballing technique [16].

2) Search Terms

We formulated our search terms based on guidelines provided in [14]. The resulting search terms were composed of the synonyms and related terms about ‘‘continuous’’ AND ‘‘software’’. After running a series of pilot searches and verifying the inclusion of the papers that we were aware of, we utilized the final search string as presented in the following. It should be noted that the search terms were used to match with paper titles, keywords, and abstracts in the digital libraries (except SpringerLink) during the automatic search.

3. Related Work

A recent qualitative study on DevOps in practice revealed that the concept is not well defined and that there are significant variations in how it is implemented [17]. Therefore, the desired and expected benefits of implementing DevOps vary just as much. [17] indicated the varying benefits organizations set out to achieve by initiating DevOps: reduced lead and release time, improved problem solving, feedback gathering and overall product quality, increased velocity, and increased focus on new features. The main benefits realized in these authors’ case study organizations were, in fact, higher deployment frequency, shorter lead times, improved automated testing, feedback gathering, and problem solving, fewer escalations (caused by friction between development and operations departments), more public facing services, and an increased velocity. However, these authors also reported that not all of these benefits were actually achieved by the organizations that implemented DevOps.

Furthermore, a 2017 SLR on DevOps [18] presents a set of 17 benefits that can potentially be achieved by implementing DevOps. These are all benefits found in literature up to 2017, however it does not necessarily mean that all these benefits are always achievable in practice.

Next, a case study on DevOps implementation in an IT company in New-Zealand [19] lists the realized benefits and their relations. It reports two main categories of benefits, namely, increased development team engagement and improved customer experience. Furthermore, [20] presents from a practitioner’s perspective the potential benefits of architecting for CD. The author outlines five types of benefits that were noticed following the transfer of 22 software programs to CDs: quicker time to market, enhanced capacity to reliably create the ideal product, and increased output, improved product quality and improved customer satisfaction. we also found five papers that have reported reviews on different aspects of continuous software engineering - two studies have investigated continuous integration in the

literature [23], two papers have explored continuous delivery [10] and deployment [21], and one study has targeted rapid release [24] (See Table 2). We summarize the key aspects of these studies. Ståhl and Bosch [23] have presented a SLR on different attributes or characteristics of CI practice.

4. Implementation

4.1 Solution Architecture

An automated pipeline for CI/CD of a web application in AWS is presented in this paper. The suggested architecture, shown in Fig. 2, is made up of VPC.

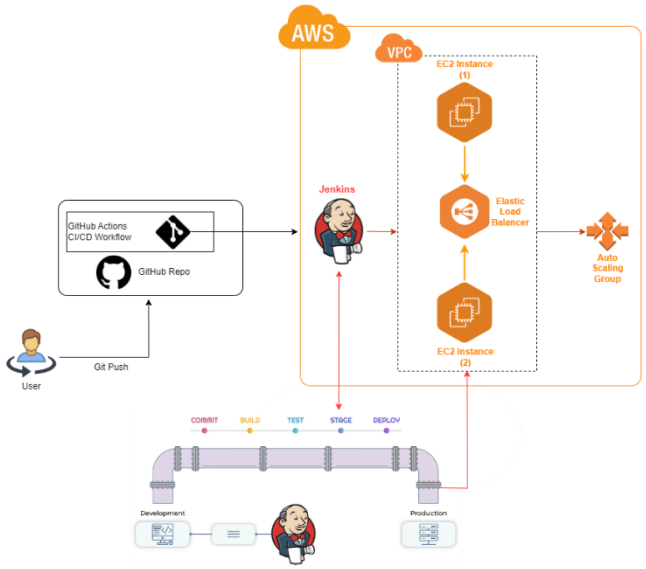

Fig. 2. *The proposed architecture*

The web application was developed using Java which helps describe the software project, its dependencies and other external components in a Project Object Model (POM). Once created, the source code of the application was loaded in GitHub.

A Jenkins server was set up and installed on an EC2 AWS instance in order to carry out continuous integration. Jenkins utilizes TCP port 8080 by default, thus you must allow incoming connections on this port. This was accomplished by establishing a new security group, which enables us to specify various security guidelines according on IP source address, protocol, and port range. Installing the GitHub. In final step in setting up Jenkins to guarantee that all three servers work together. The WAR file is used by Maven WAR Plugin for collecting all dependencies, classes and resources of the application. The new customized Docker image, called tomcat8, was pushed to DockerHub to be used from now on.

To monitor the complete CI/CD workflow was the last step: Dedicated processes for the web application on the Apache2 server, CloudWatch for the AWS resources, and the Email Jenkins notification extension plugin.

4.2 CI/CD Process

Which's implementation was covered in the section before this one. DevOps was the finest tool for implementing CI/CD concepts. procedures and equipment. Ansible was used to coordinate the CD process, whereas the Jenkins server is in charge of the CI process. Jenkins is used by most pipelines for both CI and CD, although. Jenkins on the other hand needs a separate SSH connection and specialized plugins for each target.

A change made to the project's source code in the GitHub repository starts the continuous integration task. This was accomplished by choosing Jenkins' Poll SCM configuration, which makes use of a Java plugin that is based on Timestamp. The GitHub repository in the local workspace is then replicated via the Jenkins process. Next, using, the code from the local workspace is compiled and packed.

5. Result

This section presents the results obtained in every step of the pipeline. GitHub resources, specifically a load balancer service, are used to host the web application on an AWS server. As seen in Fig. 3, the web application's initial version is thereby made available on the Internet.

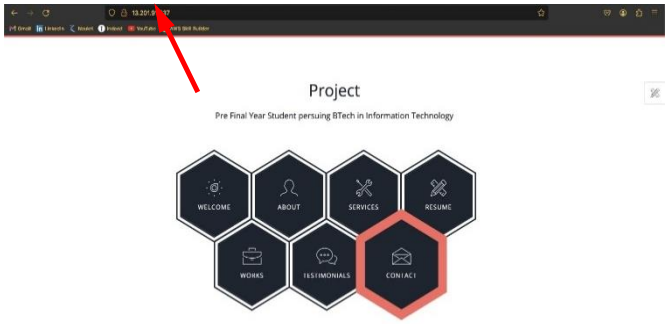

Fig. 3. *The first version of the web application*

The next step in the pipeline was to make a change in the application and push the new version to GitHub. This change was automatically detected by Jenkins which starts deploying resources for hosting the new application (see Fig. 4):

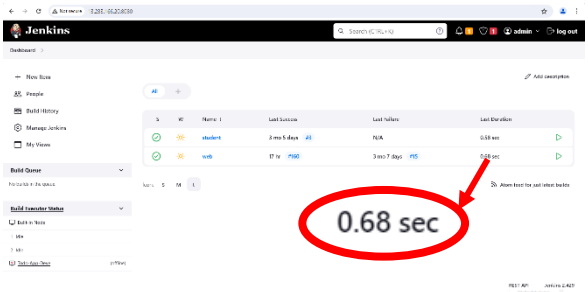

Fig. 4. *Jenkins Dashboard Node*

The CI portion of the pipeline will come to an end once this step is successfully completed. The CD portion is then implemented via a different Jenkins task, as seen in Fig. 4 shows that the deployment processes take 0.68 seconds and on AWS (EC2).

While the application is still accessible at the same IP address and port number as before, the changes are easily visible.

CloudWatch was utilized for the purpose of monitoring the application. The CPU utilization for each host running a pipeline process is shown in real time in Fig. 7.

Fig. 6. *The new version of the application*

Even when utilizing AWS instances with the fewest resources possible, all of the figures are less than 10%. The Jenkins server's increased CPU usage is correlated with the start of an automated deployment procedure. The deployment resource can be set up to restart the overloaded pods if the CPU usage beyond the pre-allocated limit.

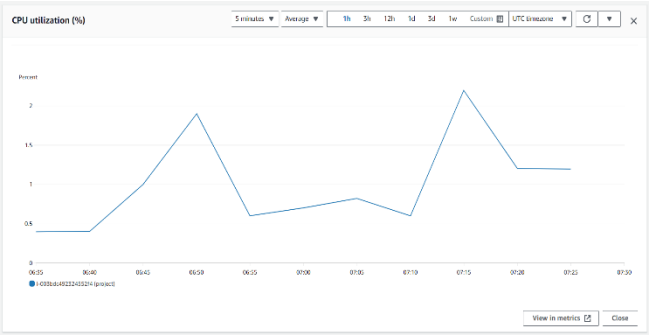

Fig. 7. *CPU utilization of the pipeline*

6. Conclusion

The automated CI/CD pipeline for AWS web application deployment using Java is presented in this paper. When using the existing system, developers must manually complete the building, testing, and deployment processes, which takes more time. Even when using scripts, users cannot pause a process while it is being executed. However, when using automation systems, developers can complete these tasks in a pipeline manner and receive reports on a regular basis. In addition, even if a phase contains errors, the system doesn't move forward.

The suggested solution is fast, easily scalable, dependable, and has a 0 second downtime, according to the experimental results. Therefore, any modifications made to the application's source code is automatically identified, it starts a whole series of events. When a Jenkins task fails during the deployment of a new version of the application, the most recent stable version of the application is rolled back by the system.

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] Phillips, M. Sens, A. de Jonge, and M. van Holsteijn, The IT Manager's Guide to Continuous Delivery: Delivering Business Value in Hours, XebiaLabs, Hilversum, The Netherlands, 2015

[4] J. Humble, and D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, 1st ed. Reading, MA, USA: Addison-Wesley, 2010.

[5] M. Fowler, Continuous Integration, accessed on Oct. 21, 2015. [Online]. Available: http://martinfowler.com/articles/continuousIntegration.html

[6] B. Fitzgerald and K.-J. Stol, ''Continuous software engineering: A roadmap and agenda,'' J. Syst. Softw., vol. 123, pp. 176–189, Jan. 2017.

[7] M. LeppÃď nen et al., ''The highways and country roads to continuous deployment,'' IEEE Softw., vol. 32, no. 2, pp. 64–72, Mar. 2015.

[8] L. Chen, ''Continuous delivery: Huge benefits, but challenges too,'' IEEE Softw., vol. 32, no. 2, pp. 50–54, Mar. 2015.

[9] A. A. U. Rahman, E. Helms, L. Williams, and C. Parnin, ''Synthesizing continuous deployment practices used in software development,'' in Proc. Agile Conf. (AGILE), Aug. 2015, pp. 1–10.

[10] H. H. Olsson, H. Alahyari, and J. Bosch, ''Climbing the 'stairway to heaven': A mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software,'' in Proc. 38th Euromicro Conf. Softw. Eng. Adv. Appl., Sep. 2012, pp. 392–399.

[11] "Continuous integration with TeamCity", TeamCity 2020, [Online], Available: https://www.jetbrains.com/help/teamcity/continuousintegration-with-teamcity.html#ContinuousIntegrationwithTeamCityBasicTeamCityconcepts.

[12] R. Varga, "Changing Dashboard build system to Bamboo", CERN Summer Student Program, No. CERN-STUDENTS-Note-2013-135, 2013, [Online], Available: https://cds.cern.ch/record/1596224.

[13] 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), Chandigarh, 2015, pp. 1-6, doi: 10.1109/RAECS.2015.7453279.

[14] B. Kitchenham and S. Charters, ''Guidelines for performing systematic literature reviews in software engineering,'' Keele Univ. and Univ. Durham, U.K., Tech. Rep. Ver. 2.3., 2007.

[15] H. Zhang, M. A. Babar, and P. Tell, ''Identifying relevant studies in software engineering,'' Inf. Softw. Technol., vol. 53, no. 6, pp. 625–637, 2011.

[16] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham, ''Using mapping studies in software engineering,'' in Proc. 20th Annu. Meeting Psychol. Programm. Interest Group (PPIG), 2008, pp. 195–204.

[17] Erich, F., C. Amrit, M. Daneva (2017) A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process, 29(6): p. e1885.

[18] Ghantous, G.B., A. Gill (2017) DevOps: Concepts, Practices, Tools, Benefits and Challenges. In PACIS 2017 Proceedings. 96.

[19] Senapathi, M., J. Buchan, H. Osman (2018). DevOps Capabilities, Practices, and Challenges: Insights from a Case Study. in Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018. ACM.

[20] Chen, L.P. (2015b) Towards Architecting for Continuous Delivery. 12th Working IEEE/IFIP Conference on Software Architecture, ed. L. Bass, P. Lago, and P. Kruchten. 131-134.

[21] P. Rodríguez et al., ''Continuous deployment of software intensive products and services: A systematic mapping study,'' J. Syst. Softw., vol. 123, pp. 263–291, Jan. 2017.

[22] E. Laukkanen, J. Itkonen, and C. Lassenius, ''Problems, causes and solutions when adopting continuous delivery—A systematic literature review,'' Inf. Softw. Technol., vol. 82, pp. 55–79, Feb. 2017.

[23] D. Ståhl and J. Bosch, ''Modeling continuous integration practice differences in industry software development,'' J. Syst. Softw., vol. 87, pp. 48–59, Jan. 2014.

[24] M. V. Mäntylä, B. Adams, F. Khomh, E. Engström, and K. Petersen, ''On rapid releases and software testing: A case study and a semisystematic literature review,'' Empirical Softw. Eng., vol. 20, no. 5, pp. 1384–1425, 2015.

[25] Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications Using Tools Cepuc, R Botez, O Craciun… - 2020 19th RoEduNet …, 2020 - ieeexplore.ieee.org

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.