Available *online* at : http://jitce.fti.unand.ac.id/
JITCE (Journal of Information Technology and Computer Engineering)
| ISSN (Online) 2599-1663 |

Research Paper

Continuous Integration and Continuous Deployment (CI/CD) for Web Applications on Cloud Infrastructures

Alde Alanda [1], H.A. Mooduto, [1] Rizka Hadelina [2]

[1] *Information Technology Department, Politeknik Negeri Padang, Kampus Limau Manis, Kota Padang 25163, Indonesia*
[2] *Computer Engineering Department, Universitas Andalas, Kampus Limau Manis, Kota Padang 25163, Indonesia*

ARTICLE INFORMATION

Received: July 22th, 2022
Revised: September 3rd , 2022
Available online: September 30th, 2022

KEYWORDS

Continous Integration, Continous Deployment, Automatic Deployment, AWS

CORRESPONDENCE

Phone: +6281267775707
E-mail: alde@pnp.ac.id

A B S T R A C T

At this time, the application development process has experienced much development in terms of tools and the programming language used. The application development process is required to be carried out in a fast process using various existing tools. The application development and delivery process can be done quickly using Continuous Integration (CI) and Continuous Delivery (CD). This study uses the CI/CD technique to develop real-time applications using various programming languages implemented on a cloud infrastructure using the AWS code pipeline, which focuses on automatic deployment. Application source code is stored on different media using GitHub and Amazon S3. The source code will be tested for automatic deployment using the AWS code pipeline. The results of this study show that all programming languages can be appropriately deployed with an average time of 60 seconds.

INTRODUCTION

Cloud computing is known for its flexibility and low costs (cost saving). Cloud computing technology is an effort to minimize the cost of procuring a fairly large information technology infrastructure. This technology accelerates the application production process without having to build infrastructure on a local network[1]. As cloud technology continues to evolve, the paradigm for developing and deploying applications in the cloud has changed not only because of scalability and reliability, but also because of cloud support for integration and delivery with minimal downtime[2].

Cloud technology support for application deployment by implementing the Continuous Integration and Continuous Deployment (CI/CD) concepts. CI/CD is a concept that functions to carry out the deployment process automatically which helps in checking the quality and function of each program that is made and helps in detecting bugs earlier, making it more effective, and efficient and saving more time in the process of making an application[3]. This can help the software industry, achieve high quality and productivity[4].

One of the most widely used cloud computing provider is Amazone Web Services (AWS), currently add up to ≈ 50% of the market share [5]. AWS provides wide range of cloud computing services that helps in development of a sophisticated application. Moreover, it has an outstanding performance in cloud computing because of the its excellent work in the area of security of data [6].

This study aims to evaluate the implementation of CI/CD concept, which focuses in automatic deployment of web application using the AWS code pipeline. The web application build by various programming language.

CI/CD

Contiuous Integration

Continuous Integration is a practical concept involved in the principles of application programming. CI states that all the code for an application should be stored in a common repository every time the developer checks the code into the repository [2][7]. CI is a regulator that combines many tools to achieve the objective of automating the software development process [8]. This phase is the core of CICD, in which the integration process between software development and operational processes will be carried

https://doi.org/10.25077/jitce.6.02.50-55.2022

out in this section. Each developer's commits must be detected, even if a bug exists. The tool commonly used is AWS CodePipeline.

AWS CodePipeline is a continuous delivery service that updates and provides automated management. It can release new features to customers in a short period. It reduces the cost and workforce of system maintenance, giving the brand more time to develop new projects. CodePipeline helps show the real-time status. The developer can check the details of any alert and retry the failed operation, the process is refined into each small part, and there is no need to start over when encountering errors[9].

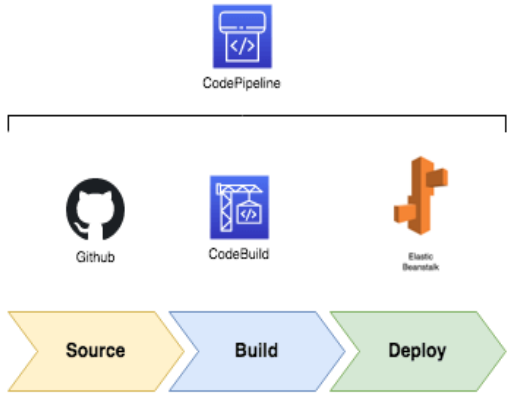


Figure 1. CodePipeline workflow [10]

The work process in CodePipeline has several stages: The first stage of the source will take the code from GitHub and send it as an artifact to the next stage. The second build stage will use AWS CodeBuild to build a Docker image from the code artifacts and save it to the repository. This stage will output the container name and URL from the image as an artifact to the next stage. Moreover, finally, the deployment phase will use AWS Elastic Beanstalk to deploy and update the clusters in the repository to use the new image URL for the container, as shown in Figure 1[10].

Continuous Deployment

In this phase, the code will be sent to the production server. The critical thing to note when sending code to a server is that the code can be used on all servers. The tool used is Elastic Beanstalk. AWS Elastic Beanstalk is one of the hyped services to host web applications on the elastic cloud. This service automates the setup, conFigureuration, and provisioning of other AWS services. It supports several platform conFigureurations for different web programming applications, including multiple versions of programming languages with the application server[11].

Figure 2 illustrates the workflow on Elastic Beanstalk. The source code to be run is uploaded, in the form of a bundle to Elastic Beanstalk. Elastic Beanstalk automatically provides the necessary environment for running the code.

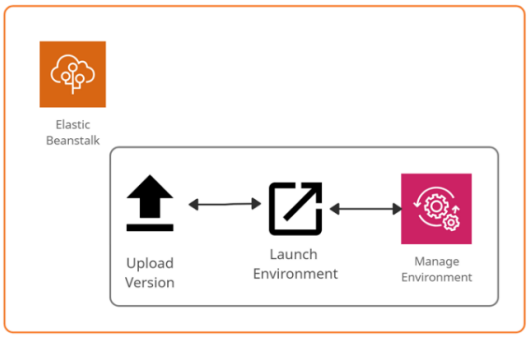


Figure 2. Elastic Beanstalk workflow

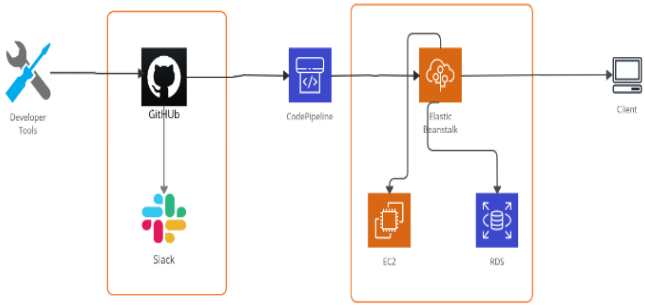
Continuous Monitoring

In this phase, information regarding the use of the software needs to be recorded to identify the proper application functionality. System errors, such as the security of the system itself, are things that need to be considered and need to be resolved. The tool commonly used is Elastic Beanstalk.

System Architecture

The design of the system architecture can be seen in Figure 1. The system consists of several components:

a) AWS CodePipeline: as an automation server to implement the CICD concept and carry out the build process that will be deployed automatically and displays logs of the deployment process.
b) AWS EC2: as the server on which to deploy the application code.
c) AWS RDS: to build databases
d) AWS Elastic Beanstalk: as the application deployment server
e) GitHub: as source location server for application code or repository management on the API to be deployed
f) Slack: as an application to monitor the update process. For each successful update, a notification will appear on the slack channel.



.

Figure.3 System Architecture

The specifications of the software used are:

a) Windows 10 OS for the operating system of the computer host
b) AWS Code Pipeline v 1.116.0
c) GitHub v1
d) AWS Elastic Beanstalk 2.0.0:
e) MySQL Workbench 8.0.26:

The hardware specifications of the system are shown in Table 1.

Table 1.The hardware specifications

	Parameter	Specification
Host	OS version	: Windows 10, 64 bit
	Processor	: AMDa A9
	RAM	: 8 GB
	Hardisk	: 500 GB
server (AWS EC2)	Instance Type	: t2.micro
	Processor	: Intel(R) Xeon(R) CPU E5-2676 v3 @3,3GHz
	Number of vCPUs	: 1
	CPU credits/hour	: 6
	Memory	: 1 GiB
	Storage	: EBS
	Network Performance	: Low to Medium
server (AWS RDS)	Instance Type	: db.t2.micro
	Processor	: Intel(R) Xeon(R) CPU E5-2676 v3 @3,3GHz
	Number of vCPUs	: 1
	CPU credits/hour	: 6
	Memory	: 1 GiB
	Network Performance	: Low to Medium

As seen in Figure 3, to implement the system, some steps must be conducted:

- *Create web aplications:* A number of simple e-commerce website was built that displays electronic equipment. This website was built using various programming languages, namely native PHP, PHP Framework, HTML, Python and JS nodes.
- *Create repository on github:* GitHub workflow consists of a fork, clone, push, commit pull request, and merged. A fork is a step in copying a repository without affecting the source repository. Clone is a step in copying the source code and making changes to the existing code. Commit is a step in verifying the changes made to the code that has been made. Pull Request is a term that can be interpreted as a request to merge code. Usually, there will be a discussion to discuss the pull request that has been made. If accepted, the code will usually be merged.
- *Connecting GitHub with Slack:* Connecting GitHub with Slack is used so that notifications will automatically appear on Slack when the project development team succeeds in pushing code to GitHub.
- *Create Deployment Environments:* The Continuous Deployment pipeline requires a target environment that contains virtual servers or Amazon EC2 instances, where it will deploy code. The first step that must be prepared is the environment before creating a pipeline. To simplify setting up and conFigureuring an EC2 instance, first create a sample environment using AWS Elastic Beanstalk. Elastic Beanstalk makes it possible to easily host web applications without launching, conFigureuring, or operating virtual servers. Automatically provision and operate infrastructure and provision application stacks.
- *Setting up the AWS CodePipeline:* The pipeline conFigureuration process is carried out in three simple steps: source, build, and deploy. This setting involves the location of the data repository, i.e. GitHub, and the deployment environment that has been previously set.
- *Setting up the AWS Elastic Beanstalk:* It involves the process of connecting with the database and conFigureuration of email notifications. In this step, the notification conFigureuration functions to provide notification that every code is committed, and the code will automatically be redeployed. A notification will automatically appear in the email when the code is updated.

After all the settings are complete, the automatic deployment system will run according to the following steps:

1) The developer pushes the project to the GitHub repository. If the project is successfully pushed, a notification will appear on slack
2) The project on GitHub will be pulled to the CodePipeline server. In CodePipeline, each project creates a service to carry out the deployment process using the CICD concept
3) If the deployment process is successful, the API that has been deployed will be read and entered into the Elastic Beanstalk server. If the deployment process fails, then the process will stop.
4) The next step is to do a trial using Elastic Beanstalk. The trial process on the Elastic Beanstalk developer will automatically get a notification email if the process is successful or fails. If the production process is successful, the application can be accessed by users. The system will send messages or reports to the developer about notifications about the system, sell requests, or application health.
5) Process complete. The process will continue to repeat every time there is an update to the system.

RESULTS AND DISCUSSION

Github and Slack Integration

GitHub and Slack repository integration are needed to monitor the application development process, which is part of CI/CD. Integrating GitHub with Slack, a message will appear from GitHub indicating that Github and the workspace in Slack have been successfully connected, as shown in Figure 4.

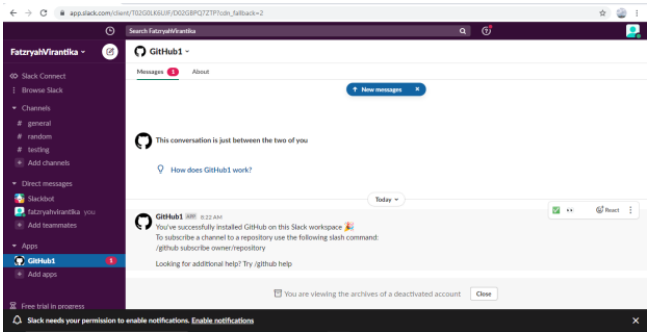


Figure 4. GitHub and Slack Integration

Every time a developer updates a GitHub repository already connected to a channel in the slack workspace, a message from

 https://doi.org/10.25077/jitce.6.02.50-55.2022

GitHub will automatically notify the user of the changes, as shown in Figure 5.

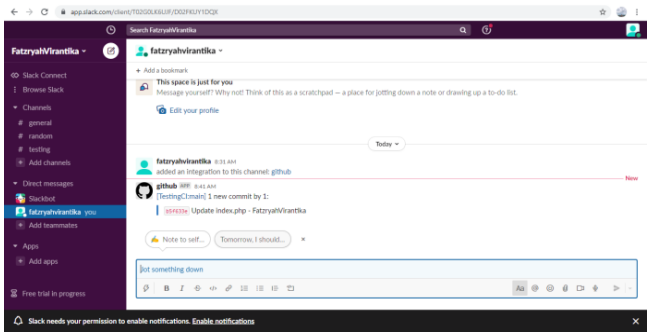


Figure 5 GitHub and Slack Channel Integration

Email notification

When the application is successfully produced, an email notification will appear from AWS Elastic Beanstalk indicating that the application has been successfully produced, updated, or provided application information whenever changes occur to the system, as shown in Figure 6.

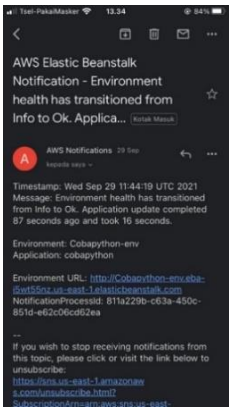


Figure 6. Email Notification

Automatic Deployment

Testing is carried out using several programming languages and a repository of application sources. The programming language used is native PHP, PHP Framework, HTM, Python, and JS nodes. Automatic deployment testing can be seen in Figure 7 – 9.

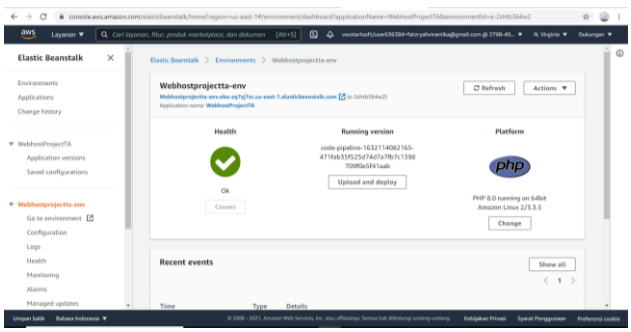


Figure 7. PHP Deployment

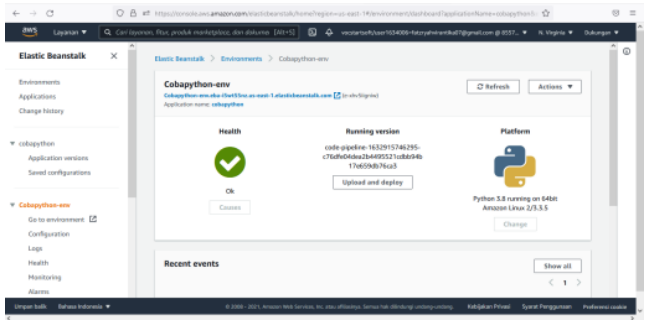


Figure 8. Python Deployment

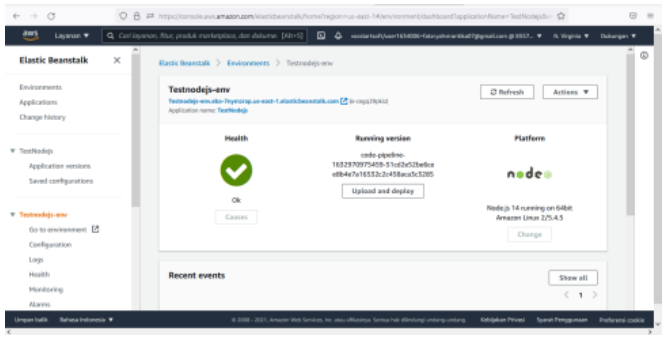


Figure 9. Node JS Deployment

From the implementation it is found that the website application has been successfully deployed and can be accessed by users and displayed. The webiste display is shown in the Figure 10

.

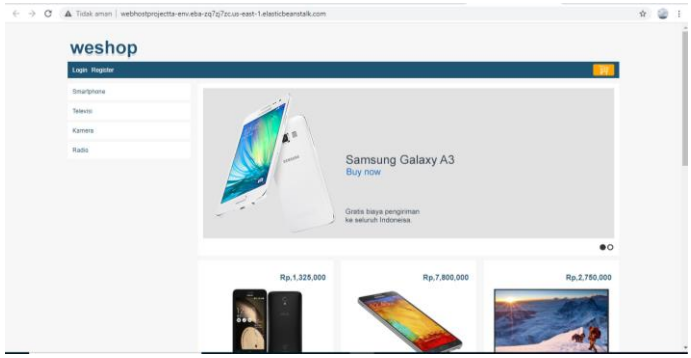


Figure 10. Web Application

Based on the test, it was found that the average deployment time for these five source codes was 60 seconds. The fastest deployment time is on the PHP framework, and the longest is on PHP native and HTML, as shown in Figure 11.

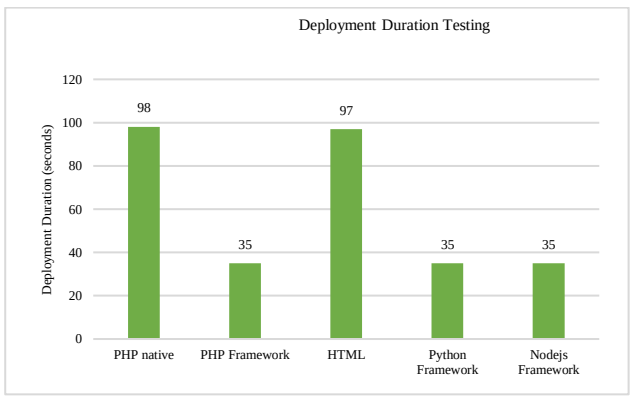


Figure 11. Deployment duration

Tests were also conducted based on the application source code repository type using Github and Amazon S3. The source code that uses Amazon S3 has a total duration of 99 seconds, while using Github has a total duration of 104 seconds, as shown in Figure 12.

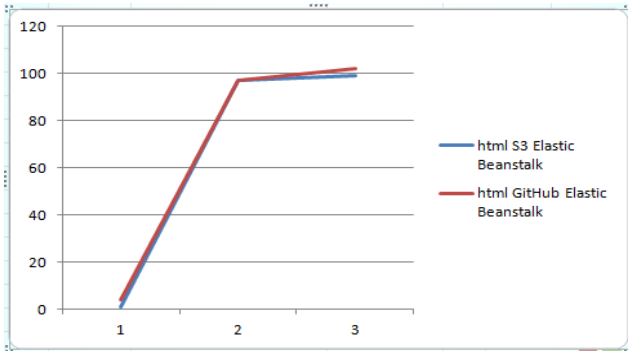


Figure. 12 Deployment duration of S3 vs Github

Testing the deployment method using Elastic Beanstalk and Cloudformation, the test results can be seen in Figure 13. Using Cloudformation requires a total deployment duration of 43 seconds and 45 seconds for Elastic Beanstalk.

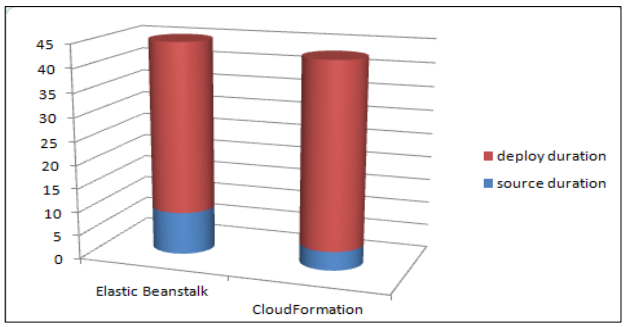


Figure. 13 Elastic Beanstalk vs CloudFormation Deployment duration

The test results using Elastic Beanstalk and Cloudformation show several differences in CI/CD implementation. This difference can be seen in Table 2.

Table.2 Elastic Beanstalk vs CloudFormation

Elastic Beanstalk	CloudFormation
With Elastic Beanstalk, users can select the desired programming language platform.	In CloudFormation, users can use templates to create, update, and delete a stack as a single unit as often as needed and do not have to manage resources individually.
With Elastic Beanstalk, we can conFigure databases, notifications, instances, software, and others.	When using AWS CloudFormation, the user is working with templates and stacks. The user creates a template environment that will be used
With Elastic Beanstalk, we can see and control the running application and environment status directly	Creating an EC2 instance using CloudFormation requires permission to create an instance. Users will need similar permissions to stop instances when deleting stacks with instances. Users use AWS Identity and Access Management (IAM) to manage permissions

CONCLUSIONS

Automatic deployment was successfully implemented by implementing the CI/CD concept. The application of CICD to application development can speed up the application development process from development to production. Implementations that are made can support applications that can be implemented on various programming language platforms.

Using Amazon S3 as a source code repository has the advantage of faster deployment than using an external repository like Github. Deployment using Cloudformation is more efficient than using Elastic Beanstalk but from an ease-of-use perspective Elastic Beanstalk is superior.

REFERENCES

[1] A. Alanda and D. Satria, "Implementasi Cloud Based Video Conference System Menggunakan Amazon Web Service," *JITCE (Journal Inf. Technol. Comput. Eng.*, vol. 5, no. 02, 2021, doi: 10.25077/jitce.5.02.75-80.2021.

[2] C. Singh, N. S. Gaba, M. Kaur, and B. Kaur, "Comparison of different CI/CD Tools integrated with cloud platform," 2019, doi: 10.1109/CONFLUENCE.2019.8776985.

[3] J. Mahboob and J. Coffman, "A Kubernetes CI/CD Pipeline with Asylo as a Trusted Execution Environment Abstraction Framework," 2021, doi: 10.1109/CCWC51732.2021.9376148.

[4] N. Rathod and A. Surve, "Test orchestration a framework for Continuous Integration and Continuous deployment," 2015, doi: 10.1109/PERVASIVE.2015.7087120.

[5] F. Palumbo, G. Aceto, A. Botta, D. Ciuonzo, V. Persico, and A. Pescapé, "Characterization and analysis of cloud-to-user latency: The case of Azure and AWS," *Comput. Networks*, vol. 184, p. 107693, 2021, doi: 10.1016/j.comnet.2020.107693.

[6] A. Kaur, G. Raj, S. Yadav, and T. Choudhury, "Performance Evaluation of AWS and IBM Cloud Platforms for Security Mechanism," *Proc. Int. Conf. Comput. Tech. Electron. Mech. Syst. CTEMS 2018*, pp. 516–520, 2018, doi: 10.1109/CTEMS.2018.8769215.

[7] D. Ståhl, T. Mårtensson, and J. Bosch, "The continuity of continuous integration: Correlations and consequences," *J. Syst. Softw.*, vol. 127, 2017, doi: 10.1016/j.jss.2017.02.003.

[8] S. Ferdian, T. Kandaga, A. Widjaja, H. Toba, R. Joshua, and J. Narabel, "Continuous Integration and Continuous

 https://doi.org/10.25077/jitce.6.02.50-55.2022

Delivery Platform Development of Software Engineering and Software Project Management in Higher Education," *J. Tek. Inform. dan Sist. Inf.*, vol. 7, no. 1, 2021, doi: 10.28932/jutisi.v7i1.3254.

[9] Y. Pan, "Lululemon Provides Better Costumer Services through Digital Ecosystem," *Highlights Business, Econ. Manag.*, vol. 1, pp. 127–130, 2022, doi: 10.54097/hbem.v1i.2332.

[10] P. Barus, "No Title," *Building CI/CD Pipeline using AWS CodePipeline, AWS CodeBuild, Amazon ECR, Amazon ECS with AWS CDK*, 2020. https://dev.to/petrabarus/.

[11] N. Neelima, B. Basaveswar Rao, K. Gangadhara Rao, and K. Chandan, "An experimental evaluation of running cost analysis for web application on cloud using queueing model," *Int. J. Eng. Adv. Technol.*, vol. 8, no. 3, pp. 629–634, 2019.