ISSN 2278-3091

Volume 13, No.6, November - December 2024

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse021362024.pdf

https://doi.org/10.30534/ijatcse/2024/021362024

Software Development Pipeline Based on DevOps for Software Development Teams in Tertiary Institutions

Isma'il Aliyu[1], Umar Faruk Muhammad[2], Badamasi Imam Ya'u[3]

[1,2,3] Department of Computer Science, Abubakar Tafawa Balewa University Bauchi, Nigeria

{[1]ialiyu, [2]mufaruk.ug, [3]biyau}@atbu.edu.ng

Received Date : October 12, 2024   Accepted Date: November 16, 2024   Published Date: December 06, 2024

ABSTRACT

Software Development is a skillful task that has evolved over time with various tools, models and processes. DevOps is one of the recent methodologies that seek to strengthen collaboration, and automation culture during software development thereby improving efficiency. Despite adoption of DevOps by some teams/organizations as reported by various studies, many software development teams particularly those in tertiary institutions stick to traditional methodologies. Noncompliance with the principles that DevOps preaches usually results in chaos or confusion after initial deployment of application or deployment of new features. Guide on how to integrate available tools to fully realize what DevOps stands for, is a significant software engineering task. However, such guides are limited in the literature. In this work, a software development pipeline based on DevOps for improved efficiency of software development teams in tertiary institutions like Abubakar Tafawa Balewa University Bauchi (ATBU) is proposed. Cloud-based tools such as Git, Docker, Kubernete were leveraged upon for the implementation of the pipeline in line with the principles of DevOps. Procedures of the implementation are provided hoping this can serve as a guide to software development practitioners who are yet to embrace DevOps. Results indicate that the proposed model or pipeline outperformed the existing practice in the areas of collaboration, automated testing and quality checks, and scalability.

Key words: Software, DevOps, Software Development, Software Model.

1. INTRODUCTION

Unprecedented developments in computer related technologies have forced organizations to embrace automation or computerization of their business processes. The enabler of this great shift is software [1] which is a set of coded instructions that direct computer to perform a task.

Today, it is hard to come across an organization that do not depend on software for its internal operations. The mode of acquiring software by organizations are two. (i) outsourcing from external software development company. (ii) In-house development where ICT staff of the organization are mandated to develop the software. The latter is the practice in most Nigerian Universities including Abubakar Tafawa Balewa University (ATBU). Of course, there are institutions that outsource their software externally disregarding the threat which external outsourcing poses. While new technologies and software development methodologies have emerged in recent times, software development teams in Nigerian universities do not seems to align with such trend. One new innovation that seek to improve efficiency of software development teams is DevOps [2], [3].

DevOps is a set of practices and tools that integrate processes involve in development and deployment of software thereby increasing efficiency and facilitate speedy delivery of quality software (Atlassian, 2024). The emphasis of DevOps is on collaboration, cross-team communication and technology automation.

Several studies [4], [5], [6], [7] have reported the adoption of DevOps by organizations and their positive response regarding its usage. However, the focus of those studies has not been on educational institutions where there exist a unit saddled with responsibilities of developing or managing the institution's software and other ICT needs. Studies or even blogs that comprehensively presents how DevOps related tools and methods are used together, in bid to simplify DevOps adoption and usage do not exist in the literature. This perhaps could be the reason why some team of software developers have not yet embraced DevOps despite the fact that it perfectly fit well into their activities. In this work, a software development pipeline based on DevOps is proposed. The pipeline is more like a model that can be adopted by software development teams for better efficiency and productivity.

1.1 Motivation

There exist significant number of research papers encouraging the adoption of DevOps in organizations that are concerned with software development. However, there is little effort in detailing how various tools can be integrated or put together in order to achieve full realization of DevOps principles. This gap coincides with our desire to provide technical solution to the way in which software development team of our institution

(ATBU) develop and deploy software for university community use. The way in which the in-house software development team of ATBU effected changes in software used for UG registration, as a result of upward review of Undergraduate registration fees in 2021/2022 session was not carried out seamlessly as it created confusion among students as a result of errors that manifested. Upon enquiry on the development methodology the team uses. We noted the use of traditional development methods. For example, lack of collaboration and testing; whenever a developer (Senior or Junior) completed his assigned task, he or she submits the code on a flash drive for review, testing and deployment. This is a huge downfall to the modern collaboration in Software Development. Motivated by the above, we propose a development model based on global best practices that can be adopted not only by ATBU teams but other institutions in order to improve efficiency and productivity.

Following the changes to the students’ registration software in line with fees review, it was insinuated that the ATBU’s management resolved to engage the services of an external software company. If the move eventually succeeds, it is huge indictment signifying incompetence which will ultimately strip the in-house team of its roles. Whichever way the university management decided to go, this study is still relevant as there are many institutions that depend on their in-house team for their software needs. Even the ones that engage external software companies, they still return to their in-house teams in the end when the contract period elapse or breaks halfway.

Rather than engaging external software companies, we advocate that universities or tertiary institutions should empower their in-house teams, encourage them to adopt global trends and embrace new technological innovations. The beneficiary is the university because cost will be saved and the confidentiality of its data or records is guaranteed. This work is in line with this advocacy. Our contributions are summarized as follow follows:

- i) Software development pipeline based on DevOps that is suitable for adoption by teams saddled with responsibilities of developing and managing software in tertiary institutions is proposed and implemented.
- ii) Procedures of integrating software development tools to achieve full realization of DevOps principles are outlined.
- iii) The dangers of engaging private software companies to manage universities’ records are outlined and the need to empower in-house teams is strongly advocated.

This paper is structured as follows. Section 2 briefly outline the dangers of software outsourcing in university system. The basic concept of DevOps together with related tools are explained in section 3. Related works are presented in Section 4. The methodology is presented in section 5 where the existing development process and the propose pipeline are described. Section 6 presents how the proposed pipeline is implemented. The procedures for implementation were supported with appropriate screen shots in order to guide readers. Evaluation and results are presented in sections 7 and 8 respectively, while section 9 concludes the paper.

2. SOFTWARE OUTSOURCING & UNIVERSITY SYSTEM

Outsourcing refers to engaging or contracting a third-party professional to execute a project or specific task [8]. It is one of the options through which a university can acquire software for its needs. In the context of university system, one advantage of software outsourcing is that individuals with technical expertise and vast experience can be involved. However, the disadvantages or rather threats associated with it cannot be overlooked.

The major threat is ***data breach***. University is an institution where academic records are very vital for its existence. At any time, University can be requested by other organizations to provide academic status or records of its graduates. Students’ data (semester results, transcripts etc) and other academic records are supposed to be kept on university’s dedicated server with high degree of confidentially. Leaving such data under the control of private company who is out to make profit will definitely have catastrophic effect on the university if anything goes wrong. Data breach or leak is a common phenomenon on cyber space. For instance, “*the social media giant Meta was fined $276 million over a data leak that affected 533 million Facebook users and resulted in their phone numbers, location, and other personal data being exposed”* [8]. Cases of data breach involving giant tech companies are reported by [9].

Another disadvantage is high cost. The cost involved in hiring or procuring an entity to render its services to university is usually higher than the amount to be spent on staff to do the job in house. Private software companies who are out to make profits usually charge exorbitantly. If fact they charge certain percentage per head (i.e per student) that uses the application. At the end of the day the money they walk away with is huge.

3. DEVOPS

The term DevOps a combination of words "**Dev**elopment" and “**Op**eration**s**” and is described as "a movement and a philosophy that emphasizes collaboration, communication, and automation in the software development process". In the traditional software development process, the development team is responsible for creating the software, while the operations team deploys and maintain it. This separation can lead to silos and conflicts between the two teams, as well as a lack of alignment and coordination in the overall process. DevOps aims to break down these silos and promote collaboration and communication between the development and operations teams. Thus, DevOps is concerned with improving collaboration and communication between the two traditionally distinct teams.

In order to provide comprehensive definition, [10] explored the components that are central to the definitions of DevOps reported in the literature. Their exploration covered 6 leading academic databases and came up with definition of DevOps as “*a development methodology aimed at bridging the gap between Development (Dev) and Operations (Ops), emphasizing communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilizing a set of development practices*”. For further discussions on the concepts and principles, we refer reader to [11], [12], [13], [14].

DevOps seeks to automate as many processes as possible, from the initial code development to the deployment and testing of the software. Automation is crucial for reducing errors, improving efficiency, and enabling rapid, iterative development.
The benefits of DevOps include improved collaboration and communication, increased efficiency and productivity, reduced risk and errors. Of course, implementation or adoption of DevOps is not without challenges. We refer reader to [15], [16] who reported challenges and mitigating strategies of using DevOps during software development. [6], [11], [12], [17] have also reported challenges associated with using or adopting DevOps. In addition to the benefits mentioned above, studies like [18], [19] reports DevOps has potential to promote or enhance software quality.

3.1 DevOps Tools

DevOps involves a lot of prominent tools that are used in different phases of development process. Some of these tools included:
Git: Git is a distributed Version Control System (VCS) that track changes in source code during software development. It allows multiple individuals to work on a project simultaneously, coordinating their work and managing code changes efficiently [20].
***Docker*:** Docker is a tool that allow deployment of applications in containers so that the applications can work in different environments efficiently without much configuration or management [21]. A container is a lightweight and standard unit of application and all its dependencies packaged together. Docker Engine is a core runtime that manages the lifecycle of containers. It handles the build process, and manages the storage and networking aspects of containers. Docker works well with tools such as VSCode, CircleCI, and Git [21]. According to [22] who analyses docker performance, costs of rebuilding cloud development platform can be reduced when traditional virtual machine is replaced with docker container. [23] equally evaluated performance of docker containers and virtual machines.
***Kubernetes*:** Kubernetes is a container orchestration platform or tool that is used to manage containerized applications. Kubernetes is a platform for running and managing containers from many container runtimes. Kubernetes supports numerous container runtimes including Docker [24]. Containers provides a way to bundle and run applications. Kubernetes is needed to manage the containers running applications in a production environment and ensure that there is no downtime [25]. [26] presents good discussions on both docker and Kubernetes. He referred to Kubernetes as "cluster manager for docker containers".
***Jenkins*:** Jenkins is the leading open-source automation server which allow developers to build, test and deploy their software. Jenkins helps automate part of software development that has to do with building, testing, and deploying thereby felicitating continuous integration (CI) and continuous Delivery (CD). It supports several version control tools including Git, AccuRev, CVS, Apache Subversion, Mercurial etc.

4. RELATED WORKS

This paper primarily focuses on two things; one, DevOps adoption by IT teams in organizations particularly tertiary. Two, using DevOps principles and associated tools to better software development process. The literature review is conducted in line with these two concerns. While thiswork is advocating for DevOps adoption, it equally takes a step further and present how DevOps related tools can be used together. The steps were clearly stated and supported with appropriate screen shots in order for practitioners to easily lay hands on. [4] conducted what they called "exploratory interview-based study" on the adoption of DevOps in 6 organizations in Netherlands, and observed that organizations are positive about their experience although some minor problems were encountered in the cause of the adoption. [7] also conduct exploratory study and present description of how DevOps is implemented in practice. They used web application and service development in 5 small and medium size companies as the context of their empirical investigation. They stated the type or nature of software applications those companies build, the build output, and the tools they use. However, their study did not state how the tools can be put together – this is very important for new practitioner who wants guide on implementing or adopting DevOps. [5] detail real scenarios of DevOps adoption. They proposed model (i.e., a workflow for DevOps adoption) and evaluated it on Brazilian Government institution. They provide evidence that collaboration is the core DevOps concern, contrasting with opinion in some quarters that automation and tooling can be enough to achieve DevOps. [27] studied the impact of DevOps implementation on teamwork quality. [28] conducted exploratory study to help practitioners and researchers to better understand the organizational structure and characteristics of teams adopting DevOps. The study involves 31 multinational software intensive companies. [29] present an approach of integrating standard-based security activities into the DevOps pipelines for Industrial Control Systems and highlight their automation potentials. [30] present a DevOps implementation framework for large scale agile financial organizations. [31], [32] proposed readiness models aimed at assisting practitioners in software industries to evaluate and improve their implementation of DevOps practices. [33] conducted evidence-based study aimed at exploring guidelines for sustainable DevOps implementation. [34] present a guide to implementing DevOps in small organizations.
In all the papers mentioned above, none focus on software teams that have to do with education sector. The work that is most similar to ours is that of [34]. However, the authors did not provide in details the steps of integrating the cloud tools to achieve the major principles of automation and collaboration.

5. METHODOLOGY

The methodology adopted is in line with the principles of DevOps. Before explaining the proposed pipeline, the existing development process is briefly explained.

5.1 The Existing Development Process

The software development process in most Nigerian universities including our case study (ATBU) is discovered to follow a traditional and linear approach with clearly defined phases and roles. The process begins with identification of software requirements which the university Management gives. The requirements are then translated into functional specifications by the software development team. Tasks are shared among both senior and junior developers who use their workstations to write codes. The most senior developer on the project then collates all the given tasks via a flash drive and rebuild, test and deploy the master application for manual exploratory test of the software functionality, after which the application is deployed to production. Figure 1 summaries the current development process.

5.2 Challenges of the Existing Development Process

The existing process is faced with myriads of challenges that have to do with collaboration, automation and scalability. These are stated below.

- *Lack of centralized version control*: Developers face challenges in coordinating their work. It is difficult to track changes in the project's code.
- *Limited code review and feedback mechanisms*: No robust code review process. This can result in delays in code reviews, lack of consistency in review feedback, and difficulties in ensuring code quality and adherence to coding standards.
- *Manual and error-prone build processes:* The absence of an automated build system like Jenkins. This makes it hard to reproduce builds reliably, leading to difficulties in troubleshooting and identifying issues.
- *Lack of automated testing:* Automated testing framework is not integrated. Testing are performed manually or inconsistently.
- *Manual scaling processes:* No auto-scaling capabilities provided by tools like Kubernetes, scaling applications manually becomes time-consuming and error-prone.

To address the above challenges, appropriate tools and methods have to be used.

5.3 The Proposed System

The proposed pipeline is shown in figure 2, and it addresses the challenges associated with the existing development method. The pipeline incorporates appropriate tools and practices, such as adopting Git and GitHub for version control and collaboration, integrating Jenkins for automated building and testing, and leveraging Kubernetes for auto-scaling and load balancing. The process begins with setting up a remote repository to house the application's code. The programmers involved in the project can then configure their machines and push the code they write to the remote repository.

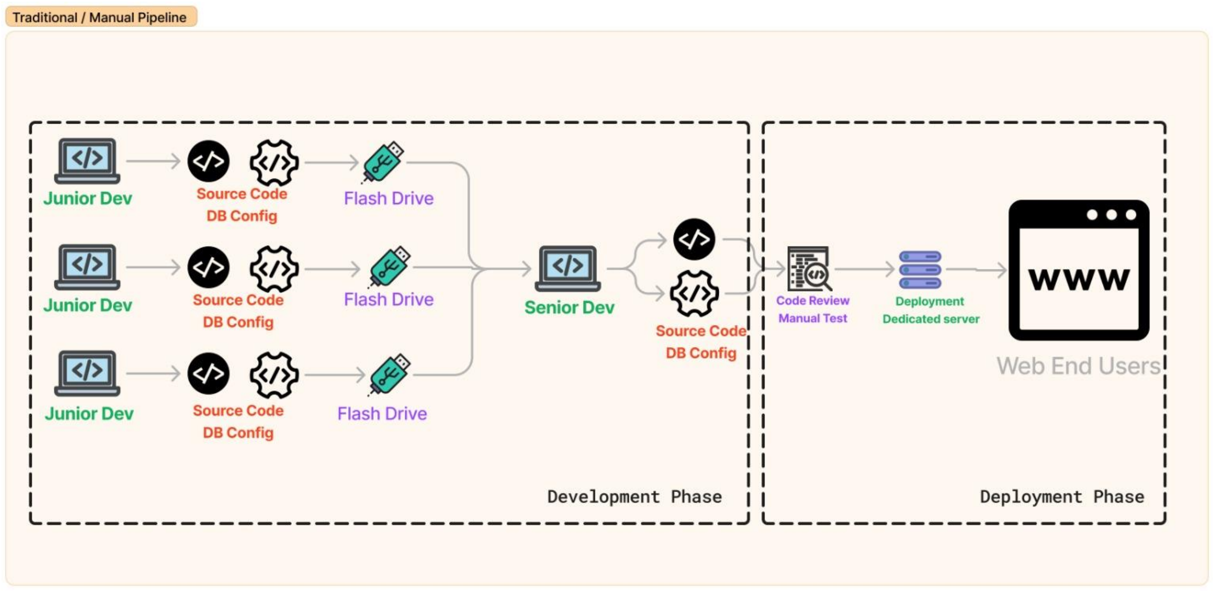
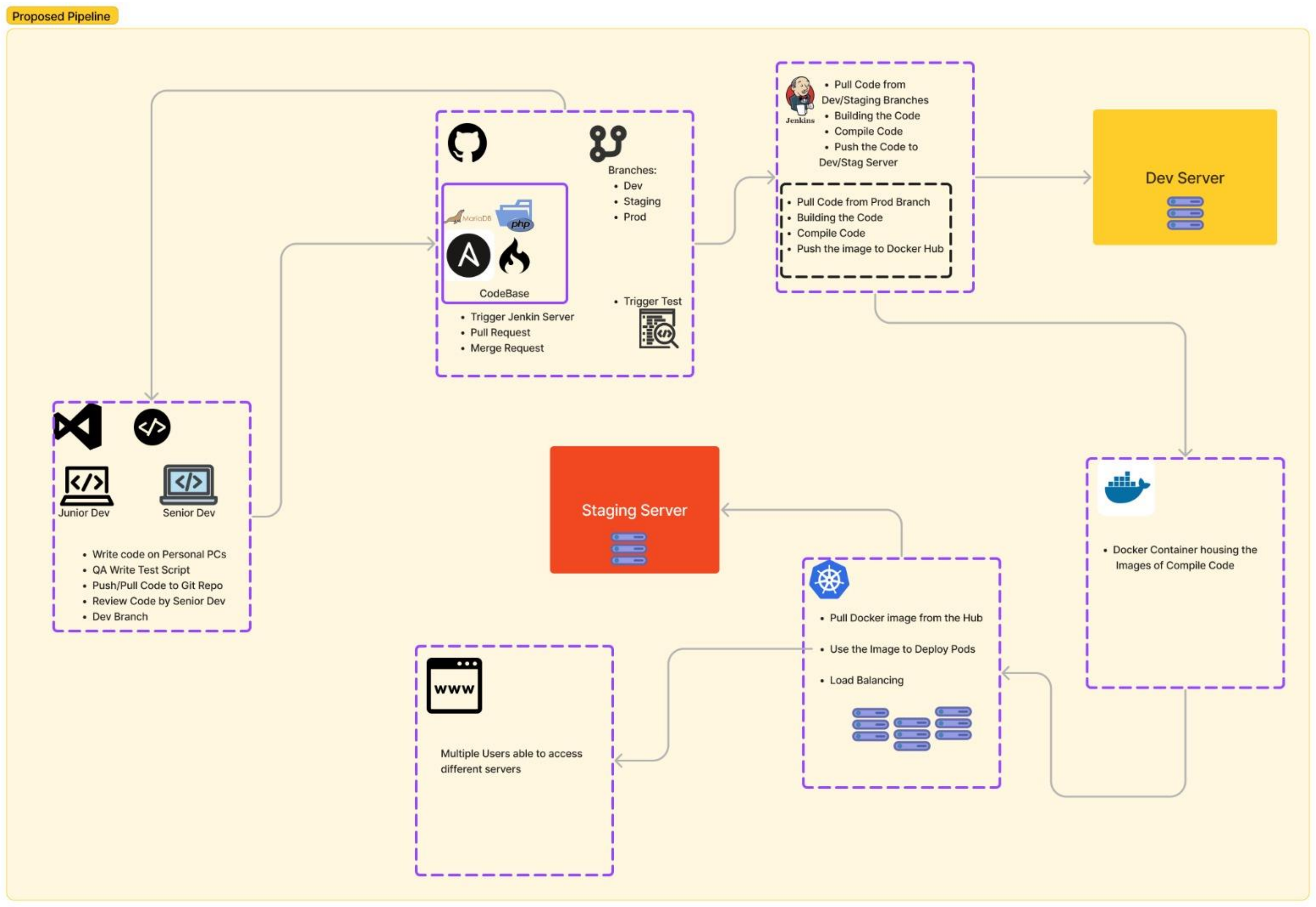


Figure 1: Current Development and Operation Process



Proposed Pipeline
Junior Dev
Senior Dev
Write code on Personal PCs
QA Write Test Script
Push/Pull Code to Git Repo
Review Code by Senior Dev
Dev Branch
CodeBase
Branches:
Dev
Staging
Prod
Trigger Jenkin Server
Pull Request
Merge Request
Trigger Test
Pull Code from Dev/Staging Branches
Building the Code
Compile Code
Push the Code to Dev/Stag Server
Pull Code from Prod Branch
Building the Code
Compile Code
Push the image to Docker Hub
Dev Server
Docker Container housing the Images of Compile Code
Staging Server
Pull Docker image from the Hub
Use the Image to Deploy Pods
Load Balancing
www
Multiple Users able to access different servers

Figure 2: The proposed pipeline. In this figure, all the developers can work on the same project hosted on remote repository

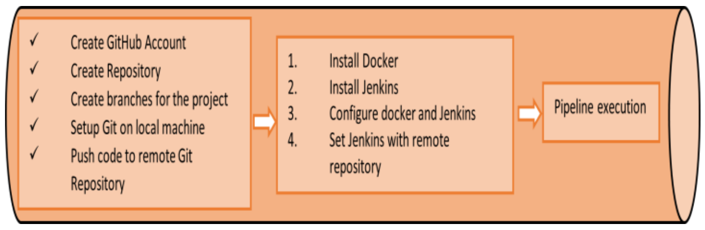
6. IMPLEMENTATION

In this section, procedures for implementation of the pipeline are presented with the hope this will serve as a useful guide to any developer who wishes to either or partially embrace DevOps tools and principles. The summary of the implementation steps are shown in figure3. It important to note that what is obtainable at the end of the process is the application's image that undergoes all the necessary quality test and checks. In this work, Git is used as version control system (VCS).

6.1 Developer's Check list

The following are the checklist required for implementation of the pipeline proposed in this work.

✓ Computer system
✓ Cloud based DevOps tools – used to achieve collaboration, and automation; Github Repository, Docker, Jenkins, Kubernetes
✓ Application development framework eg, CodeIgniter4.



✓ Create GitHub Account
✓ Create Repository
✓ Create branches for the project
✓ Setup Git on local machine
✓ Push code to remote Git Repository
1. Install Docker
2. Install Jenkins
3. Configure docker and Jenkins
4. Set Jenkins with remote repository
Pipeline execution

Figure 3: summary of the pipeline implementation process

In line with the principles of DevOps, the implementation is segmented with respect to the principles of collaboration, automation, and continuous integration.

6.2 Collaboration

The collaboration aspect of the pipeline involves using Version Control System to enable multiple developers work on the project. This is achieved using GitHub platform. The steps for the collaboration segments are as follows

***Step 1**: Create a GitHub account, create a repository that serves as the codebase of the project*. Once GitHub account is created, features that enable creation and manipulation of repositories can be accessed. In this work, repository name "**Project-atbu**" was created as shown in Figure 4. Of course, many branches can be created for a particular repository. Two branches are created for this work, these are *Development* and *Staging*. Subsequently, other branches can be created which may include bugfixes etc.

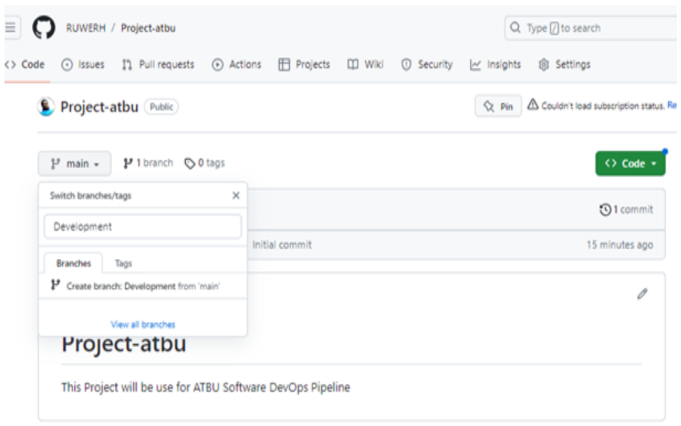


Figure 4: GitHub interface showing how branches can be created under Project-atbu repo

***Step 2**: Setup Git on local machine:* After creating the git repository, next is for the individuals involve in the project to set up git terminal and clone the remote git repository on their local machines. Download the latest version of git and install. Open the git terminal in order to access git account and subsequently use repositories under the account. Note that correct username and password of the git account must be supplied.

***Step 3**: Push code to the repository:* After setting up Git and GitHub next is for senior developer to push the initial codebase to the main branch**,** while other developers are to push the development branch. This should be done after they clone the project to their respective machines. The following steps explains further

- ✓ Open the terminal and navigate to the project folder
- ✓ Run the command "git clone https://github.com/RUWERH/Project-atbu.git" to clone the remote project to the local machine.
- ✓ Copy all the codebase to the new folder created
- ✓ Run the command "git add" to stage the entire codebase and be ready.
- ✓ Run the command "git status" to see the folders and file in the stage domain.
- ✓ Run the command "git commit –m "Initial code commit" to commit the files.
- ✓ Run the command "git push origin main/master" to push the entire codebase (or changes) to the repository on Github platform.

***Step 4**: Code review and merging pull request:* Whenever a developer commits or pushes codes to the development branch, it will be as a pull request, which the senior developer will go through to ensure the code does not conflict with the key components of the project before merging it to the main branch. This process is known as Code Review or Pull Request Review.

6.3 Automation

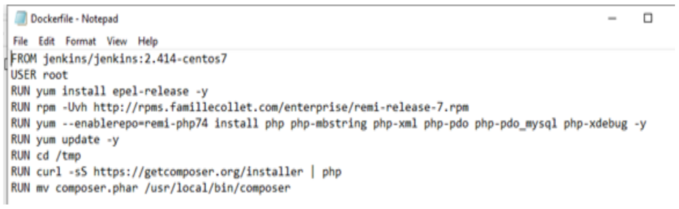
The essence of automation aspect of the pipeline is to subject the application's code to test and quality checks by automated tools available, before final deployment of the application for public consumption. This is achieved using docker and Jenkins. The senior developer is the one to proceed with this stage.

First, docker has to be installed. Its executable file for windows is available on the internet for download. After installing docker, open power shell and type the command "docker --version" to confirm the version installed.

Step 1:Setup Jenkins

A Docker file name "Dockerfile" is used to create the Jenkins Container. In the process, some Docker commands needs to be run on the Jenkins machines as part of the pipeline to deploy the application to staging Environment and Run a number of tests on it. Below are the steps

i. Create a text file "Dockerfile" in the project folder with the content shown in figure 5



```
FROM jenkins/jenkins:2.414-centos7
USER root
RUN yum install epel-release -y
RUN rpm -Uvh http://rpms.famillecollet.com/enterprise/remi-release-7.rpm
RUN yum --enablerepo=remi-php74 install php php-mbstring php-xml php-pdo php-pdo_mysql php-xdebug -y
RUN yum update -y
RUN cd /tmp
RUN curl -sS https://getcomposer.org/installer | php
RUN mv composer.phar /usr/local/bin/composer
```

Figure 5: Dockerfile containing the commands that install Jenkins

ii. **Build the application image**: Open PowerShell and navigate to the project folder where the Docker file is located and run the command "docker image build –it atbu-project" to build the application image.

iii. After Building the image, it can be used to create the Jenkins container using this command "docker run -d -p 49009:8080 -p 50000:50000 -v /var/run/docker.sock:/var/run/docker.sock -- name jenkins-atbu atbu-project"

iv. Check if the container is created and running using the command "docker ps", it also gives details about the container.

v. Open web browser of choice and navigate to the IP address of the machine with the port as provided while creating the container. It will display the initial page of Jenkins and request that an initial admin password should be inserted to proceed, the initial passwords can be gotten by checking the logs of the container using "docker logs containerID" command.

vi. After correct input of default admin password, another window or interface will pop up requesting to create a new admin user or proceed with the default one. Next is another window or interface that requests for installation of plugins. Click install suggested plugins and continue.

vii. If all the above steps are carried out correctly, the Jenkins dashboard will be displayed

Step 2: Install Docker on Jenkins Machine

i. Open PowerShell and run the command "*docker exec -it -u root jenkins-atbu bash*" to access the Jenkins container.

ii. Run "*yum install -y yum-utils*" to install yum utils on the container

iii. Run these two commands to install docker "*yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo* "
"*yum install -y docker-ce docker-ce-cli containerd.io*"

iv. Check and verify if it installed "*docker –version*"

v. Change the permission of the volume using "*chmod 666 /var/run/docker.sock*"

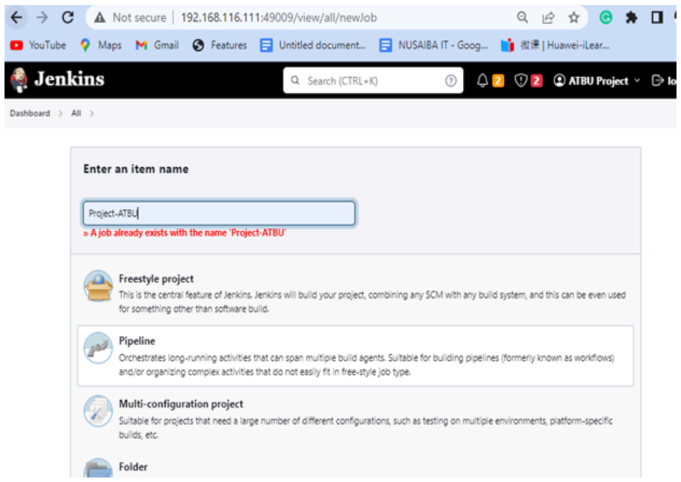
vi. Then Exit.

Step 3: Setup Jenkins with Remote Repository

Below are the steps to set up Jenkins Pipeline with the remote repository:

i. On the Jenkins landing page, navigate to the "New item" button and click.

ii. Type the name of the Pipeline you want to create and select the pipeline tab option as shown in figure 6 below



Not secure | 192.168.116.111:49009/view/all/newJob

Dashboard > All >

Enter an item name

Project-ATBU

» A job already exists with the name 'Project-ATBU'

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

Figure 6: Jenkin dashboard

iii. Configuration window will open after clicking Pipeline. Click General tab by the left side of the window. Type the Project Description. Select or tick the check box for "GitHub Project". Type the url of git repository housing the project codes. In our case, the url of our git repository is "https://github.com/RUWERH/Project-atbu/". Under **Build Triggers**, select or tick the checkbox for "GitHub hook trigger for GITScm polling" – this will trigger the pipeline whenever a new pull request is merged.

iv. On the same configuration window, click Pipeline tab by the left side of the window. Select "Pipeline script from SCM" which will look for Jenkins pipeline script on the Source Code Management tool in use. Select Git and provide the Repository URL. **NOTE**: if the repository is private, it credentials need to be provided using the Jenkins Credentials feature for the secret. Type the name of the branch that will be built in the pipeline, and provide the name of the script file residing in the repository.

v. Click Apply and save. The Pipeline is ready to build the project

Step 4: Creating Pipeline script and Test Scripts

Jenkins pipeline script is constructed for this project which has 10 stages namely:

i. Build: This stage builds the project and installs the necessary plugins and dependencies to run the application some of which include: composer, php artisan, database configurations etc.

ii. Unit Test: This stage performs a unit test on the application written by the developers of the Quality Assurance Engineer, in this case, an artisan is used to run the test.

iii. Code Coverage: This stage runs a coverage test on the application since the application is running PHP, PHPUnit supports code coverage tests so it's used to achieve that.

iv. Static Code Analysis Larastan: In this stage Perform a code analysis on the application using Larastan, before this will work Larastan needs to be installed.

v. Static Code Analysis phpcs: This Stage also performs code analysis on the application using phpcs, before this will work PHP CodeSniffer needs to be installed.

vi. Docker Build: This stage runs the Docker command to build the projected image using Dockerfile included in the Repository and name it "album-project".

vii. Docker Push: This stage also runs the Docker command to push the build image to the Docker hub using the specified username and password provided and send IT.

viii. Deploy to Staging: This stage Deploys the containerized application to the staging environment where it can be accessed and tested.

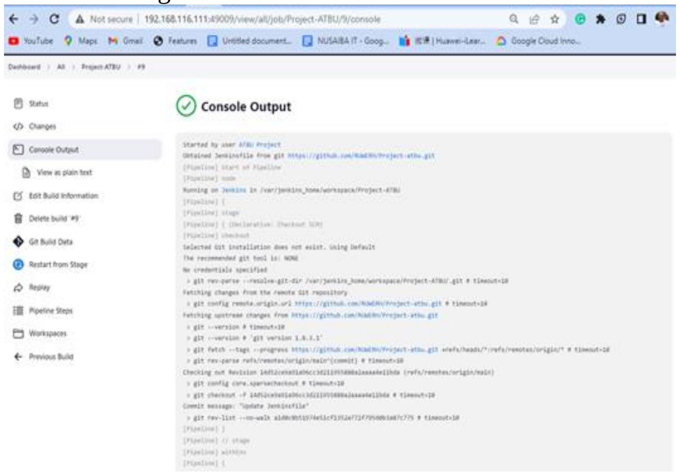
ix. Acceptance test CURL: This stage performs an acceptance test base on the test cases provided in "acceptance_test.sh", it will change the permission and run the script on the container.

x. Acceptance Test Code Exception: This stage also performs an acceptance test with code exception, before it can be achieved a PHP co-deception needs to be installed on the project, and also "RegisterTest.php" test file needs to be created, after it's completed then it will stop the container.

The successful integration of Git, Jenkins, and Docker resulted in a seamless CI/CD pipeline. Developers could confidently push their code changes to the Git repository, knowing that automated builds, tests, and deployments would be triggered. The pipeline ensured that the application's code was continuously integrated, tested, and delivered to the staging or production environment with minimal manual intervention.

Step 5: Pipeline Execution

When the senior developer finishes code review and merged the code to the main branch. The Pipeline is triggered and all the stages are carried out, if one failed the rest wouldn't proceed and hence it will be unsuccessful. To see the entire pipeline execution process, navigate to Console Output page as shown in Figure 7


Console Output

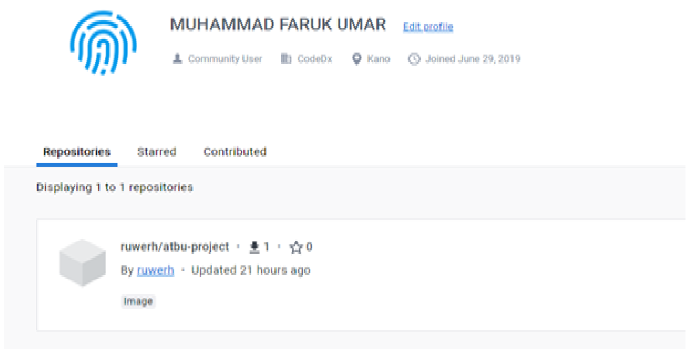
```
Started by user ATBU Project
Obtained Jenkinsfile from git https://github.com/[illegible]/Project-atbu.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Project-ATBU
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/jenkins_home/workspace/Project-ATBU/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/[illegible]/Project-atbu.git # timeout=10
Fetching upstream changes from https://github.com/[illegible]/Project-atbu.git
 > git --version # timeout=10
 > git --version # 'git version 1.8.3.1'
 > git fetch --tags --progress https://github.com/[illegible]/Project-atbu.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision [illegible] (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f [illegible] # timeout=10
Commit message: "Update Jenkinsfile"
 > git rev-list --no-walk [illegible] # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
```

Figure 7: console output of the pipeline execution

The summary of the process is as follows.

- ✓ Clone the project and install all the necessary dependencies. Perform unit test.
- ✓ Execute the tests to ascertain code or branch coverage.
- ✓ Perform static code analysis.
- ✓ Build image of the application using Docker file in the repository.
- ✓ Push the image to docker hub.
- ✓ Deploy the build application to a staging environment. Execute the test script provided and stop the container.

To verify, navigate to the Docker hub. The application's image push by the pipeline will be available as shown in Figure 8. The application's image can be deployed to a Kubernetes cluster having multiple nodes that will ensure the scalability of the application.


MUHAMMAD FARUK UMAR Edit profile

Community User · CodeDx · Kano · Joined June 29, 2019

Repositories Starred Contributed

Displaying 1 to 1 repositories

ruwerh/atbu-project · 1 · 0

By ruwerh · Updated 21 hours ago

Image

Figure 8: showing the image of the application pushed by the pipeline

7. EVALUATION

To evaluate the proposed pipeline, software is developed based on the steps outlined in implementation section. This will give room for comparison with the existing development model based on some metrics: collaboration, development speed, automation, code quality.

For the sake of evaluation, any software with significant number of functional requirements can be developed. In our case we development student registration system using CodeIgniter4, php framework development of web application. The project code was stored on github and each member of the team clone the project to his/her local machine and proceed with coding the tasks assigned to him. Eventually the application's image was obtained after passing through the stages outlined.

8. RESULTS & DISCUSSION

In order to ascertain the superiority or otherwise of the pipeline proposed in this work, 4 metrics shown in table 1 were used.

Table 1: Remarks on the metrics used to access the performance of the proposed pipeline against the existing process

Metrics	Remarks
Collaboration	Ability of many individuals to work simultaneously on the same project, and track their performances.
Automation	Continuous integration of code and other components, automatically build application upon code pull request, little or minimal human intervention in during deployment.
Scalability & Resource utilization	Efficient management of resources, ensuring optimal performance during peak loads and reducing resource wastage during low loads.
Quality & stability	Quality checks such as testing, bug fixing to ensure stability of the application.

The summary of results indicating the performance of the proposed model against the existing process is shown in table 2 below.

Table2: Summary of the results showing the proposed pipeline outperformed the existing development model.

Metrics	Existing Development model	Proposed Model
Collaboration	Poor	Excellent
Automation	Poor	Excellent
Scalability	Poor	Very Good
Quality	Fair	Very Good

In all the metrics used, it is clear that the proposed pipeline is exceptionally better compared with the existing model. This represents significant shift for better software development practices.

8.1 Comparison with Pre-DevOps Practices

It is actually difficult if not impossible to justify the metrics stated in table 1 with respect to software development where DevOps is not incorporated. Overall, the DevOps pipeline proposed in this work demonstrated significant improvement over the existing model. With respect to the software development method proposed in this work, the following are handy benefits that can be derived:

- ✓ Individual efforts in building the application codes can be measured.
- ✓ No need of manual collation of codes from the programmers involved in the project.
- ✓ No need of manual testing.
- ✓ Little human intervention during deployment.

9. CONCLUSION

In this work, software development model based on DevOps is presented. This is in line with our collective resolve to offer solution for better software development process to software development teams in Nigerian tertiary institutions. Cloud based tools such as git, docker were used for the implementation. The steps to integrate these tools together are described, and it was shown how docker image of the application can be obtained as the end product of the pipeline. The overall aim is to realize full DevOps utilization and adoption. The results obtained shows the proposed pipeline outperformed the existing development model signifying that the pipeline's adoption will significantly improve collaboration, efficiency and productivity of software development teams in producing quality software.
Having proposed the pipeline, next is to conduct an empirical and exploratory study on DevOps adoption in some selected Nigerian Tertiary institutions in order to access their implementation strategies and challenges if any. Perhaps this requires conducting interviews across the institutions that might be selected for the study. We leave that as future work.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable input and suggestions.
This work is supported by Tertiary Education Trust Fund (tetfund) through Institution Based Research (IBR) year 2023 intervention for Abubakar Tafawa Balewa University. We remain grateful to tetfund for its various supports to Nigerian tertiary institutions. Indeed, the initiative is making positive impact.

REFERENCES

[1] A. Fuggetta, "Open source Software - An Evaluation," *Journal of Systems and Software*, vol. 66, no. 1, pp. 77–90, 2003
[2] P. Debois, "DevOps: A Software Revolution in the Making," *Journal of Information Technology Management*, vol. 24, no. 8, pp. 3–39, 2011
[3] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Softw*, vol. 33, no. 3, pp. 94–100, 2016
[4] F. M. Erich, C. Amrit, and M. Daneva, "A Qualitative Study of DevOps Usage in Practice," *Journal of Software: Evolution and Process*, vol. 29, no. 6, 2017
[5] L. W. Pinheiro, G. Pinto, and R. Bonifacio, "Adopting DevOps in the Real World: A Theory, a Model, and a Case Study," *Journal of System and Software*, vol. 157, 2019.
[6] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, and T. Männistö, "DevOps Adoption Benefits and Challenges in Practice: A Case Study," in *In proceedings of 17th International Conference, PROFES, Trondheim*, Norway, 2016, pp. 590–597.
[7] L. E. Lwakatare *et al.*, "DevOps in Practice: A Multiple Case study of Five Companies," *Journal of Information and Software Technology*, vol. 114, pp. 217–230, 2019, doi: https://doi.org/10.1016/j.infsof.2019.06.010.
[8] F. Yolcu, "Pros and Cons of Hiring Software Development Company," https://www.datrick.com/pros-and-cons-of-hiring-a-soft ware-development-company/ Accessed on 05/02/2024.
[9] M. Hill and D. SwinHoe, "15 Biggest Data breaches of the 21st Century," https://www.csoonline.com/article/534628/the-biggest-d ata-breaches-of-the-21st-century.html. Accessed on 05/02/2024.
[10] R. Jabbari, N. Bin Ali, K. Petersen, and B. Tanveer, "What is DevOps? A Systematic Mapping Study on Definitions and Practices," in *In proceeding of the scientific Workshop of XP2016*, 2016, pp. 1–11.
[11] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of DevOps Concepts and Challenges," *ACM Computing Surveys (CSUR) 52(6), 1-35* , vol. 52, no. 6, pp. 1–35, 2019.
[12] G. Bou Ghantous and A. Gill, "DevOps: Concepts, Practices, Tools, Benefits and Challenges," in *In proceedings of Pacific Asia Conference on Information Systems (PACIS),* 2017.

[13] M. Gall and F. Pigni, "Taking DevOps Mainstream: A Critical Review and Conceptual Framework," *European Journal of Information Systems*, vol. 31, no. 5, pp. 548–567, 2022.

[14] R. W. Macarthy and J. M. Bass, "An Empirical Taxonomy of DevOps in Practice," in *46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2020, pp. 221–228.

[15] D. B. Sindhu, "The challenges and Mitigation Strategies of Using DevOps during Software Development," *International Journal of Creative Research Thoughts (IJCRT)*, 2021.

[16] M. T. Hossain, M. Sarker, G. Uddin, and A. Iqbal, "A Mixed Method Study of DevOps Challenges," *Journal of Information and Software Technology*, vol. 161, 2023.

[17] M. Shoaib Khan, A. K. Khan, F. Khan, M. Adnan Khan, and T. K. Whangbo, "Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review," *IEEE Access*, pp. 14339–14349, 2022.

[18] P. Perera, R. Silva, and I. Perera, "Improve Software Quality through practicing DevOps," in *In Proceedings of 17th International Conference on Advances in ICT for Emerging Regions (ICTer)*, 2017, pp. 1–6.

[19] A. Mishra and Z. Otaiwi, "DevOps and software quality: A Systematic Mapping," *Comput Sci Rev*, vol. 38, 2020.

[20] Git, "Learn About the version control system, Git, and how it works with GitHub," https://docs.github.com/en/get-started/using-git/about-git Retrieved on 13-02-2024 .

[21] Docker, "Docker: Accelerated Container Application Development," https://www.docker.com. Retrieved 13-02-2024.

[22] R. B. Bashari, H. John Batti, and M. Ahmadi, "An Introduction to Docker and Analysis of its Performance," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 3, pp. 228–235, 2017.

[23] A. M. Potdar, D. G. Narayan, S. Kengond, and M. Moin Mulla, "Performance Evaluation of Docker Container and Virtual Machine," in *3rd International Conference on Computing and Network Communications*, 2020, pp. 1419–1428.

[24] J. Campbell, "Kubernetes Vs Docker," https://atlassian.com/microservices/microservices-architecture/Kubernetes-vs-docker Accessed on 13-02-2024.

[25] Eshwari H M, Rekha B S, and G. N. Srinivasan, "Hybrid Cloud Technologies: Dockers, Containers and Kubernetes," *International Research Journal of Engineering and Technology (IRJET)* , vol. 7, no. 6, pp. 7628–7634, 2020

[26] D. Berstein, "Containers and cloud: from lxc to docker to kubernetes," *IEEE cloud computing*, vol. 1, no. 3, pp. 81–84, 2014.D. Berstein, "Containers and cloud: from lxc to docker to kubernetes," *IEEE cloud computing*, vol. 1, no. 3, pp. 81–84, 2014.

[27] A. Hermawan and L. M. Parningoton, "The Effect of DevOps Implementation on Teamwork Quality in Software Development," *Journal of Information Systems Engineering and Business Intelligence*, vol. 7, no. 1, p. 84, 2021.

[28] D. Lopez-Fernandez, J. Diaz, J. Garcia, J. Perez, and A. Gonzalez-Prieto, "DevOps team Structure: Characterization and Implication," *IEEE Transactions on Software Engineering*, vol. 48, no. 10, pp. 3716–3736, 2021.

[29] F. Moyon, R. Soares, M. Pinto-Albuquerque, D. Mendez, and K. Beckers, "Integration of Security standards in DevOps Pipelines: An Industrial case study," in *In Proceedings of 21st International Conference, PROFES*, Turin, Italy, 2020, pp. 434–452.

[30] A. D. Nagarajan and S. J. Overbeek, "A DevOps Implementation Framework for Large Agile-based Financial Organizations," in *OTM Confederated International Conferences on the Move to meaningful Internet Systems*, 2018, pp. 172–2018.

[31] S. Rafi, W. Yu, M. A. Akbar, S. Mahmood, A. Alsanad, and A. Gumael, "Readiness Model for DevOps Implementation in Software Organizations," *Jornal of Software: Evolution and Process*, vol. 33, no. 3, 2020.

[32] N. M. Noorani, A. T. Zamani, M. Alenezi, M. Shameem, and P. Singh, "Factor Prioritization for Effectively Implementing DevOps in Software Development Organizations: A SWOT-AHP Approach," *Axiom MDP*, vol. 11, p. 498, 2022.

[33] M. Zohaib, A. Alsanad, and A. Alhogail, "Prioritizing DevOps Implementation Guidelines for Sustainable Software Projects," *IEEE Access*, vol. 12, pp. 71109–71130, 2024.

[34] M. Munoz and M. N. Rodriguez, "A Guidance to Implement or Reinforce a DevOps Approach in Organizations: A Case study." *Journal of Software: Evolution and Process*, vol. 36, no. 3, p. e2342, 2024