DevOps, Continuous Integration and Continuous Deployment Methods for Software Deployment Automation

Mochamad Hanif Rifa'i Istifarulah[1*], **Rizka Tiaharyadini**[2]
[1]Program Studi Magister Ilmu Komputer, Fakultas Teknologi Informasi, Universitas Budi Luhur
[2]Jl. Raya Ciledug, Petukangan Utara, Kebayoran Lama, Jakarta Selatan 12260
Email: [1]istifarulah@gmail.com, [2]rizka.tiaharyadini@budiluhur.ac.id

Abstract - In the fast-paced landscape of software development, the need for efficient, reliable, and rapid deployment processes has become paramount. Manual deployment processes often lead to inefficiencies, errors, and delays, impacting the overall agility and reliability of software delivery. DevOps, as a cultural and collaborative approach, plays a central role in orchestrating the synergy between development and operations teams, fostering a shared responsibility for the entire software delivery lifecycle. Continuous Integration is a fundamental DevOps practice that involves regularly integrating code changes into a shared repository, triggering automated builds and tests. Continuous Deployment complements Continuous Integration by automating the release and deployment of validated code changes into production environments. The purpose of this research is to create a software deployment automation system to make it easier and reliable for organizations to deploy software. In conclusion, the results of this research show that by adopting DevOps, Continuous Integration, and Continuous Deployment, organizations can achieve enhanced collaboration, shortened release cycles, increased deployment frequency, consistent deployment, and improved overall software quality.

Keywords: *DevOps, Continuous Integration, Continuous Deployment, Software Development, CI/CD*

I. INTRODUCTION

PT EOA Teknologi Internasional with the branding name EOA Tech is an entity of the EOA Group whose business is mostly as a gold producer with the gold brand issued being EOA Gold. EOA itself is an abbreviation of Emas Optimasi Abadi. PT EOA Teknologi Internasional was founded in July 2020 as a software development company engaged in software development or Independent Software Vendor (ISV). PT EOA Teknologi Internasional has been heavily involved in helping clients solve problems with software creation solutions and providing services for hosting these applications on the company's servers. For cloud service providers, companies use services from Alibaba Cloud.

With the increasingly rapid development of software development and business demands that require feature releases and software improvements to be carried out quickly accompanied by increasing frequency, consistent and automated methods for software integration and deployment are needed to minimize human error. To support this, at the organizational level, the Product Development and Operation divisions at PT EOA Teknologi Internasional are required to work together and be able to coordinate and collaborate effectively. However, work bottlenecks often occur, communication problems and a lack of synergy between the Product Development and Operations divisions which cause feature releases and software improvements to be hampered and take longer. There are other problems during deployment, namely inconsistent deployment results between the development environment, staging environment and production environments, in the development environment all features work well and have been checked by Quality Assurance, but when deployed to the production environment the results of testing in the development environment which show all features are running well are often different when deployed in the production environment, there are several features that do not work well in the production environment, this causes inconsistencies in software deployment. Another problem that causes delays in software release or deployment is that making changes to deployed software will often require both testing and planning, as well as coordination between the different involved departments [1], the deployment process carried out by the Operations division is required to be processed by the relevant team manually, it is indeed a time consuming task to develop, assure quality of and it is not desirable to add even more costly overhead to this process [2]. Apart from that, there are cases where the software deployment process must be carried out at night so it will not cause disturbance of applications that are running and used by users during the day or during working hours. Because software deployment requires intervention from the Operations division, the Product Development division often had difficulties when deploying software at night due to the availability of the Operations division.

DevOps combines a number of approaches that bring together developers and operations staff to create software and services rapidly, reliably, and of higher quality [3]. Based on this background, the research objective is to apply DevOps, Continuous Integration and Continuous Deployment methods to automate software deployment, make it easier for companies to deploy software, avoid work bottlenecks between the Product Development and Operations divisions and minimize human error.

II. RESEARCH METHODOLOGY

2.1. Software Development Life Cycle (SDLC)

The systems development life cycle (SDLC) is a methodology for designing, building, developing and maintaining information and process systems [4]

By applying existing SDLC rules, you can provide an understanding of how an Information System (IS) can support business needs, design and build a system according to user needs [5]. The SDLC project is a directive project, namely organized and planned. SDLC project standards require release phases and in each phase there will be a build and deploy process. The deploy stage indicates that code development is complete. Then the code will be built and deployed in the Testing environment. If a bug appears in the Testing environment, the code must be fixed and then rebuilt and used in the Pre-production or Production environment.

Agile methodologies are the standard practices for modern-day industries, Agile software development has a big focus on collaboration and the self-organizing team [6]. Topics that are the center of attention of Agile include its ability to reduce costs, increase speed and quality, and provide motivation to empower employees to support the development and success of a company. Scrum is the most popular of the Agile frameworks [7]. In the Scrum framework there are Sprints and teams that will work in them. After the Sprint is complete, the Demo is complete and the Definition of Done (DoD) is met according to the user story, then each team will integrate their code and build for use in a staging or test environment. Configuration management tools will be used to integrate the code. This is where the real problem arises. Fixing this issue will take a lot of time and cause the release time to be off schedule. Problems in the build and deploy phases are very common and occur frequently in Agile work environments. This also happens in application development at PT EOA Teknologi Internasional. DevOps is expected to be a solution to this.

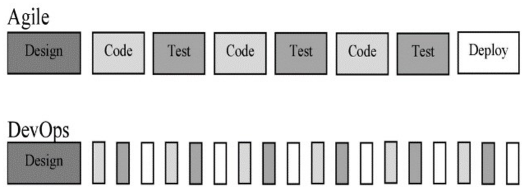


Fig 1. Differences in Design, Coding, Testing and Deploy Iteration in Agile and DevOps

2.2. DevOps

DevOps (development and operations) is a conceptual study of the development and delivery of software to infrastructure by taking a collaborative and integrative approach between developers (Dev) and software operations (Ops) [8]. DevOps is an organizational approach with the aim of creating collaboration, interaction and empathy across functions and divisions [9]. The adoption of DevOps is one example of a step to strengthen collaboration between IT teams, which is very much needed in software development and maintenance [10]

The DevOps method is proven to be able to reduce several development stages that existed in the old method [11]. DevOps is able to shorten the time between software development and operation without reducing the quality of the software itself [12]. DevOps reduces the gap between the development team, operations team and application users allowing to detect problems early. DevOps adoption can implement continuous development and more frequent application releases to users [13]. Not only that, DevOps also changes software engineering processes and practices to be faster which also increases the reliability, stability, resilience and security of the production environment [14].

2.3. Continuous Integration (CI)

[15] Continuous Integration is a method in software development, where code from each developer is combined and built periodically to detect errors as early as possible. In general, the stages in a CI system are as follows (triggered by push or commit commands):

1. Integrated
   This is the stage where each developer integrates the codes or results of their work.
2. Compiled
   Code is compiled into packages or executables
3. Tested
   Test executables manually or automatically
4. Archived
   Archive executable files, test results, and logs during the process.
5. Deployed
   Install or share build results.

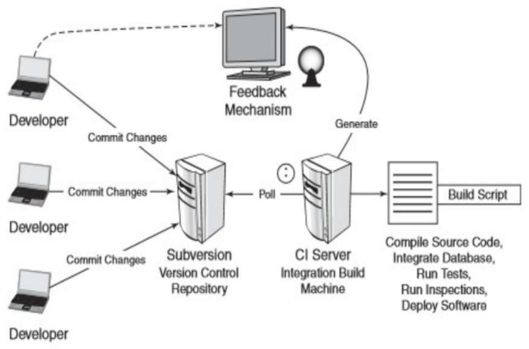

Feedback Mechanism
Developer
Commit Changes
Generate
Developer
Commit Changes
Poll
Build Script
Commit Changes
Subversion Version Control Repository
CI Server Integration Build Machine
Compile Source Code, Integrate Database, Run Tests, Run Inspections, Deploy Software
Developer

Fig 2. Continuous Integration Architecture System

2.4. Continuous Deployment (CD)

[16] Continuous Deployment (CD) is a software release process that uses automated testing to validate whether changes to the code base are correct and stable for immediate autonomous deployment to a production environment.

The difference between continuous deployment and continuous delivery can be confusing due to the nomenclature. Both are abbreviated as CD and have very similar responsibilities. Delivery is the beginning of deployment. In delivery, there is a final manual approval step before production release.

In the delivery stage, developers will review and merge code changes which are then packaged into artifacts. This package is then moved to the production environment where it awaits approval to be opened for deployment. In the deployment phase, packages are opened and reviewed with an automated inspection system. If the check fails the package is rejected. When the checks pass, the package is automatically deployed to production.

Continuous Deployment can be a powerful tool for modern organizations. Deployment is the final step in the entire continuous pipeline consisting of integration, delivery and deployment. The true experience of continuous deployment is automation to the level where code is deployed to production, tested, and automatically reverted when there are errors, or accepted if there are no errors.

2.5. Docker Container

[17] A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application (code, runtime, system tools, system libraries and settings).

Containers separate software from its surroundings, for example the difference between development and staging, and are also a solution to conflicts between teams using different software on the same infrastructure.

2.6. Docker Swarm

Docker Swarm is an orchestration tool developed by Docker itself. Docker Swarm clusters allow one to add an unlimited number of nodes, and Docker allows one to run unlimited containers on nodes, this provides full-scale testing that is close to real conditions [18].

Docker Swarm provides clustering and an orchestration mechanism and thus can deploy several containers across different host machines. Docker Swarm also provides a fault tolerance mechanism not only by detecting failed containers on a host machine, but also redeploying the same container on another host machine [19].

2.7. Gitlab

Gitlab is a web-based Git, repository manager with a wiki, issue tracker and CI/CD pipeline. Gitlab uses an open source license developed by Gitlab Inc. In the release of Gitlab 10.0, Gitlab took a big step beyond just code management, but expanded into deployment and monitoring. Gitlab organizes and modifies people's permissions according to their roles and can provide issue tracking access without granting permissions to large pieces of code, which is great for teams and large companies with role-based contributions. Gitlab supports CI for free and is very useful for teams, besides that Gitlab also supports CI/CD automatically without human intervention [20]

III. RESULTS AND DISCUSSION

The software deployment automation system for the Product Development division and Operation division of PT EOA Teknologi Internasional consists of Gitlab as version control for developers to store source code, Continuous Integration (CI) Server and Continuous Deployment (CD) Server using the infrastructure provided by Gitlab, then Container Registry as a place to accommodate the Docker Image output from the Continuous Integration (CI) Server using services from Alibaba Cloud, and Elastic Compute Service (ECS) is the destination server for Continuous Deployment (CD). The server carries out software deployment after pulling the Docker Image from the Container Registry. Continuous Integration (CI) Server and Continuous Deployment (CD) Server when carrying out the integration or deployment process will provide output in the form of a running process log which can be seen on the account dashboard on the Gitlab site and provide notification via email if the Continuous Integration process or Continuous Deployment process has failed.

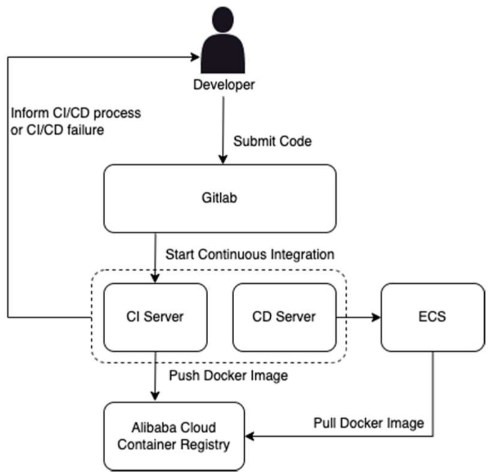

Developer
Inform CI/CD process or CI/CD failure
Submit Code
Gitlab
Start Continuous Integration
CI Server
CD Server
ECS
Push Docker Image
Alibaba Cloud Container Registry
Pull Docker Image

Fig 3. Software Deployment Automation System Design

Before the software deployment automation process can be carried out, the ECS for the software deployment destination must have Docker Engine and Docker Swarm installed, as well as the reverse proxy which in this study uses Traefik, must be configured and ready to use.

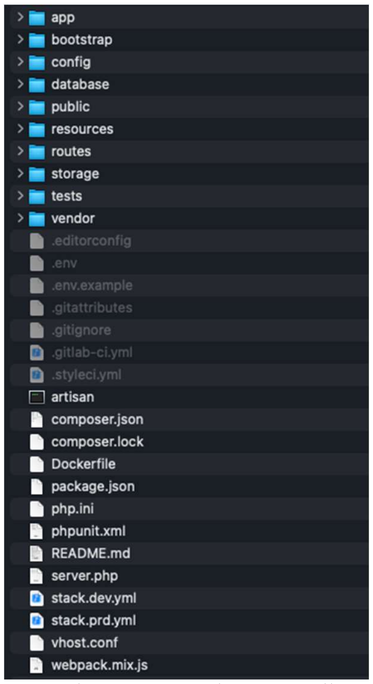
The entire Continuous Integration and Continuous Deployment process refers to scripts that have been prepared by the Operations division and have been submitted to the repository on Gitlab or version control. These scripts are in the form of .gitlab-ci.yml to manage the Continuous Integration and Continuous Deployment flow on the Gitlab CI/CD infrastructure. Dockerfile, stack.yml, and other supporting scripts required by the programming language in each application or repository.

The prototype of this software deployment automation system was built using the YAML (Yet Another Markup Language) programming language which is usually used to write configuration files. YAML is a popular programming language because it is human-readable and easy to understand. Meanwhile, the application used for software deployment automation testing uses the PHP programming language and the Laravel framework, because PHP and Laravel are popular programming languages and frameworks, that are widely used by developers.

The display and snippet of the script from the software deployment automation system are as follows:

1. Software Deployment Automation System Prototype File Structure

The following is the file structure of the software deployment automation system used to deploy software in the PHP programming language using the Laravel framework.

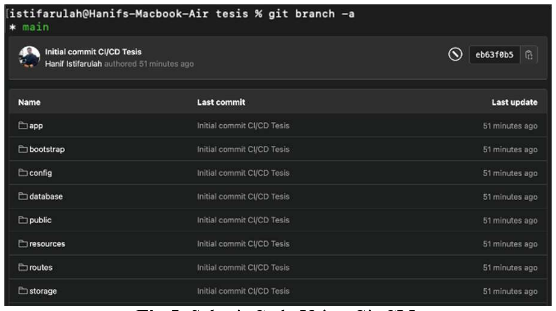


```
> app
> bootstrap
> config
> database
> public
> resources
> routes
> storage
> tests
> vendor
  .editorconfig
  .env
  .env.example
  .gitattributes
  .gitignore
  .gitlab-ci.yml
  .styleci.yml
  artisan
  composer.json
  composer.lock
  Dockerfile
  package.json
  php.ini
  phpunit.xml
  README.md
  server.php
  stack.dev.yml
  stack.prd.yml
  vhost.conf
  webpack.mix.js
```

Fig 4 Software Deployment Automation System File Structure in PHP Laravel

2. Submit Code using Git CLI (Command Line Interface)

In the following submit code, there are several commands that are executed on the macOS terminal, namely looking at what branches are available in the repository, then pushing it to the selected branch.



```
istifarulah@Hanifs-Macbook-Air tesis % git branch -a
* main
```

Initial commit CI/CD Tesis
Hanif Istifarulah authored 51 minutes ago
eb63f0b5

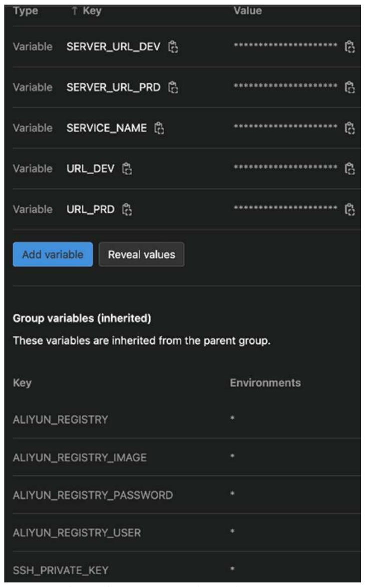
Name	Last commit	Last update
app	Initial commit CI/CD Tesis	51 minutes ago
bootstrap	Initial commit CI/CD Tesis	51 minutes ago
config	Initial commit CI/CD Tesis	51 minutes ago
database	Initial commit CI/CD Tesis	51 minutes ago
public	Initial commit CI/CD Tesis	51 minutes ago
resources	Initial commit CI/CD Tesis	51 minutes ago
routes	Initial commit CI/CD Tesis	51 minutes ago
storage	Initial commit CI/CD Tesis	51 minutes ago

Fig 5, Submit Code Using Git CLI

Fig 6. Result of Git Push

3. CI/CD Variable in Gitlab

In the following Gitlab CI/CD variable, these are variables that will be used in the software deployment automation process. The contents of these variables cannot be seen by developers or can only be seen by

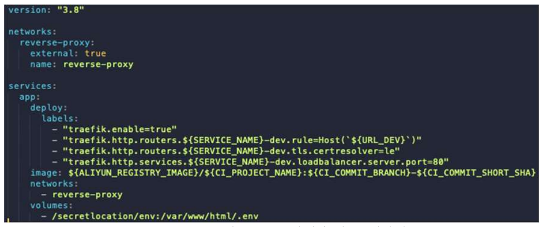


Type	Key	Value
Variable	SERVER_URL_DEV	********************
Variable	SERVER_URL_PRD	********************
Variable	SERVICE_NAME	********************
Variable	URL_DEV	********************
Variable	URL_PRD	********************

Group variables (inherited)
These variables are inherited from the parent group.

Key	Environments
ALIYUN_REGISTRY	*
ALIYUN_REGISTRY_IMAGE	*
ALIYUN_REGISTRY_PASSWORD	*
ALIYUN_REGISTRY_USER	*
SSH_PRIVATE_KEY	*

developers who have maintainer access or highest access to the repository. With this, system confidentiality is maintained even though many developers leave the repository due to resigning or moving to other projects.

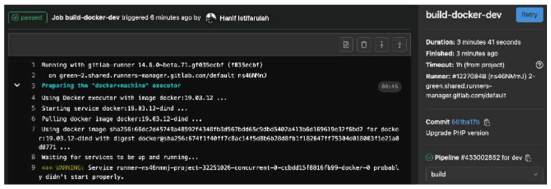


```yaml
version: "3.8"

networks:
  reverse-proxy:
    external: true
    name: reverse-proxy

services:
  app:
    deploy:
      labels:
        - "traefik.enable=true"
        - "traefik.http.routers.${SERVICE_NAME}-dev.rule=Host(`${URL_DEV}`)"
        - "traefik.http.routers.${SERVICE_NAME}-dev.tls.certresolver=le"
        - "traefik.http.services.${SERVICE_NAME}-dev.loadbalancer.server.port=80"
    image: ${ALIYUN_REGISTRY_IMAGE}/${CI_PROJECT_NAME}:${CI_COMMIT_BRANCH}-${CI_COMMIT_SHORT_SHA}
    networks:
      - reverse-proxy
    volumes:
      - /secretlocation/env:/var/www/html/.env
```

Fig 7. CI/CD Variable in Gitlab

4. Runner View of the Continuous Integration Process

The following is the runner display when the continuous integration process is running, Gitlab provides real time logs regarding the running process, and if a failure occurs, the relevant developer will automatically receive a notification email.



passed Job build-docker-dev triggered 6 minutes ago by Hanif Istifarulah

```
Running with gitlab-runner 14.6.0~beta.71.gf035ecbf (f835ecbf)
  on green-2.shared.runners-manager.gitlab.com/default ns46NMnJ
Preparing the "docker+machine" executor                                    00:46
Using Docker executor with image docker:19.03.12 ...
Starting service docker:19.03.12-dind ...
Pulling docker image docker:19.03.12-dind ...
Using docker image sha256:66dc2d45749a48592f4348fb3d567bdd65c9dbd5402a413b6d169619e32f6bd2 for docke
r:19.03.12-dind with digest docker@sha256:674f1f40ff7c8ac14f5d8b6b28d8fb1f182647ff75384c018083f1e21a8
d8771 ...
Waiting for services to be up and running...
*** WARNING: Service runner-ns46nmnj-project-32251026-concurrent-0-ccbdd15f8816fb99-docker-0 probabl
y didn't start properly.
```

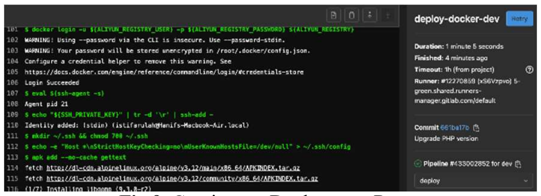
build-docker-dev

Duration: 3 minutes 41 seconds
Finished: 3 minutes ago
Timeout: 1h (from project)
Runner: #12270848 (ns46NMmJ) 2-green.shared.runners-manager.gitlab.com/default
Commit 661ba17b
Upgrade PHP version
Pipeline #433002852 for dev
build

Fig 8. Continuous Integration Runner

5. Runner View in Continuous Deployment Process

The following is a runner display when the continuous deployment process is running. Gitlab provides real time logs regarding the running process, and if a failure occurs, the relevant developer will automatically receive a notification email.



```
$ docker login -u ${ALIYUN_REGISTRY_USER} -p ${ALIYUN_REGISTRY_PASSWORD} ${ALIYUN_REGISTRY}
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
$ eval $(ssh-agent -s)
Agent pid 21
$ echo "${SSH_PRIVATE_KEY}" | tr -d '\r' | ssh-add -
Identity added: (stdin) (istifarulah@Hanifs-Macbook-Air.local)
$ mkdir ~/.ssh && chmod 700 ~/.ssh
$ echo -e "Host *\nStrictHostKeyChecking=no\nUserKnownHostsFile=/dev/null" > ~/.ssh/config
$ apk add --no-cache gettext
fetch http://dl-cdn.alpinelinux.org/alpine/v3.12/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.12/community/x86_64/APKINDEX.tar.gz
(1/7) Installing libgomp (9.3.0-r2)
```

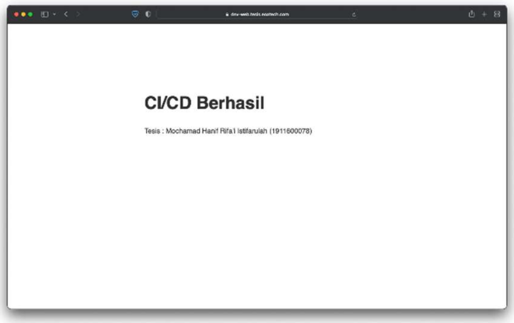
deploy-docker-dev

Duration: 1 minute 5 seconds
Finished: 4 minutes ago
Timeout: 1h (from project)
Runner: #12270859 (xS6Vzpvo) 5-green.shared.runners-manager.gitlab.com/default
Commit 661ba17b
Upgrade PHP version
Pipeline #433002852 for dev
deploy

Fig 9. Continuous Deployment Runner

6. Application Display After Successful Deployment Process

The following is the appearance of the application in a web browser, which is used for testing the software deployment automation system after successful software deployment. This application can be accessed by users using a web browser.



dev-web.tesis.eoatech.com

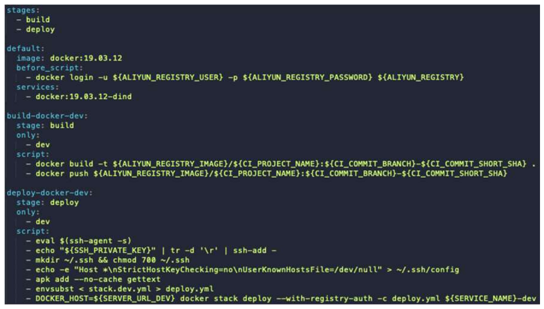
CI/CD Berhasil

Tesis : Mochamad Hanif Rifa1 Istifarulah (1911600078)

Fig 10. Application Display After the CI/CD Process is Successful

7. Snippet of .gitlab-ci.yml Script

The following is a snippet from the .gitlab-ci.yml script, to set up Continuous Integration and Continuous Deployment flows on Gitlab CI/CD infrastructure.



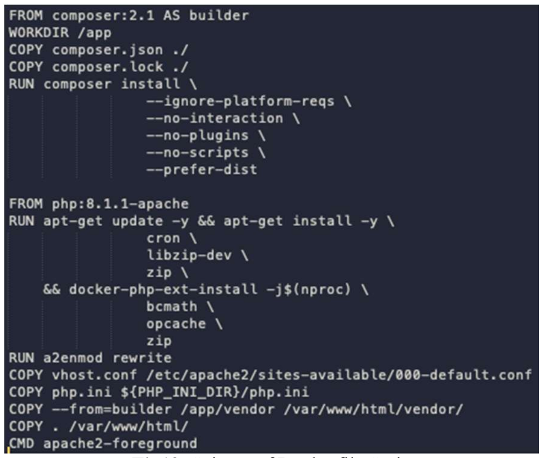
```yaml
stages:
  - build
  - deploy

default:
  image: docker:19.03.12
  before_script:
    - docker login -u ${ALIYUN_REGISTRY_USER} -p ${ALIYUN_REGISTRY_PASSWORD} ${ALIYUN_REGISTRY}
  services:
    - docker:19.03.12-dind

build-docker-dev:
  stage: build
  only:
    - dev
  script:
    - docker build -t ${ALIYUN_REGISTRY_IMAGE}/${CI_PROJECT_NAME}:${CI_COMMIT_BRANCH}-${CI_COMMIT_SHORT_SHA} .
    - docker push ${ALIYUN_REGISTRY_IMAGE}/${CI_PROJECT_NAME}:${CI_COMMIT_BRANCH}-${CI_COMMIT_SHORT_SHA}

deploy-docker-dev:
  stage: deploy
  only:
    - dev
  script:
    - eval $(ssh-agent -s)
    - echo "${SSH_PRIVATE_KEY}" | tr -d '\r' | ssh-add -
    - mkdir ~/.ssh && chmod 700 ~/.ssh
    - echo -e "Host *\nStrictHostKeyChecking=no\nUserKnownHostsFile=/dev/null" > ~/.ssh/config
    - apk add --no-cache gettext
    - envsubst < stack.dev.yml > deploy.yml
    - DOCKER_HOST=${SERVER_URL_DEV} docker stack deploy --with-registry-auth -c deploy.yml ${SERVICE_NAME}-dev
```

e-ISSN: 2614-8404
p-ISSN:2776-3234

Fig 11 Snippet of .gitlab-ci.yml Script

8. Snippet of Dockerfile Script

The following is a snippet from the Dockerfile script, as a basic for building the output in the form of a Docker Image.



```
FROM composer:2.1 AS builder
WORKDIR /app
COPY composer.json ./
COPY composer.lock ./
RUN composer install \
                --ignore-platform-reqs \
                --no-interaction \
                --no-plugins \
                --no-scripts \
                --prefer-dist

FROM php:8.1.1-apache
RUN apt-get update -y && apt-get install -y \
                cron \
                libzip-dev \
                zip \
    && docker-php-ext-install -j$(nproc) \
                bcmath \
                opcache \
                zip
RUN a2enmod rewrite
COPY vhost.conf /etc/apache2/sites-available/000-default.conf
COPY php.ini ${PHP_INI_DIR}/php.ini
COPY --from=builder /app/vendor /var/www/html/vendor/
COPY . /var/www/html/
CMD apache2-foreground
```

Fig12. Snippet of Dockerfile Script

9. Snippet of stack.yml Script

The following is a snippet from the stack.yml script, as a basic for configuring a reverse proxy so that applications that have been deployed can be accessed by users. This script also arranges for copying environment files which contain secrets related to database access or other secrets for the application which is deployed. This script also regulates which Docker Image must be pulled for use in the software deployment process.

Figure 13 Snippet of stack.yml Script

IV. CONCLUSION

Software deployment that has been carried out using a software deployment automation system makes the results of software deployment in the development environment, staging environment and production environment consistent.

Because the software deployment process has been automated, there is no longer a work bottleneck in the Operations division because the software deployment process no longer requires human intervention after the software deployment automation system is configured at the start of creating the repository/project.

By building a software deployment automation system, human errors caused by the software deployment process being carried out manually can be minimized.

With the software deployment automation system, the software deployment process can be carried out at any time by the Product Development division, without having to ask the Operations division to stand by and intervene. This leads to shortened release cycle, increased deployment frequency thus improve software quality.

Suggestions from the author for the development of this system in the future:

1. In the future, this software deployment automation system can be developed using Kubernetes container orchestration with auto scaling.
2. Implement system monitoring using Grafana, Prometheus and other supporting applications.
3. This software deployment automation system can be developed to create a High Availability System.
4. There needs to be good cooperation between the Product Development division and the Operations division regarding the use of DevOps, Continuous Integration and Continuous Deployment methods.

REFERENCES

[1] Bibi, S., Katsaros, D., & Bozanis, P. (2012). Business Application Acquisition: On-Premise or SaaS-Based Solutions? *IEEE Software*, *29*, 86–93. https://doi.org/10.1109/MS.2011.119

[2] Touma, Y. (2019). *An investigation of Automating Software Deployment Using Continuous Delivery Tools : A cost-benefit study in the case of multiple system instances.* https://api.semanticscholar.org/CorpusID:196201845

[3] Battina, D. S. (2021). The Challenges and Mitigation Strategies of Using DevOps during Software Development. *International Journal of Creative Research Thoughts (IJCRT)*, *9*(1), 4760–4765.

[4] Alhamidi. (2017). Membangun Sistem Aplikasi untuk Seleksi Calon Mahasiswa Undangan pada Tingkat Sekolah Menengah Atas. *Jurnal J-Click; Vol 3 No 2 (2016): J-Click*. http://ejurnal.jayanusa.ac.id/index.php/J-Click/article/view/26

[5] Syamsiyah, N., & Sesunan, M. F. (2018). Penerapan Metode System Life Cycle Development Dan Project Management Body of Knowledge Pada Pengembangan Sistem. *Ikraith-Informatika*, *2*(2).

[6] Trivedi, D. (2021). Agile Methodologies. *International Journal of Computer Science & Communication*, *12*(2), 91–100.

[7] Naik, N., & Jenkins, P. (2019). Relax, it's a game: Utilising gamification in learning agile scrum software development. *IEEE Conference on Computatonal Intelligence and Games, CIG*, *2019-Augus*. https://doi.org/10.1109/CIG.2019.8848104

[8] Jha, P., & Khan, R. (2018). A Review Paper on DevOps: Beginning and More To Know. *International Journal of Computer Applications*, *180*(48), 16–20. https://doi.org/10.5120/ijca2018917253

[9] Dyck, A., Penners, R., & Lichter, H. (2015). *Towards Definitions for Release Engineering and DevOps*. https://doi.org/10.1109/RELENG.2015.10

[10] CaTechnology. (2013). *TechInsights Report : What Smart Businesses Know About DevOps*. *September*, 300.

[11] Taryana, A., Fadli, A., & Nurshiami, S. R. (2020). Merancang Perangkat Lunak Sistem Penjaminan Mutu Internal (SPMI) Perguruan Tinggi yang Memiliki Daya Adaptasi Terhadap Perubahan Kebutuhan Pengguna secara Cepat dan Sering. *Jurnal Al-Azhar Indonesia Seri Sains Dan Teknologi*, *5*(3), 121. https://doi.org/10.36722/sst.v5i3.372

[12] Erich, F., Amrit, C., & Daneva, M. (2017). A Qualitative Study of DevOps Usage in Practice. *Journal of Software: Evolution and Process*, *00*. https://doi.org/10.1002/smr.1885

[13] Erich, F., Amrit, C., & Daneva, M. (2014). Report: DevOps Literature Review. *Https://Www.Researchgate.Net/Publication/267330992_Report_DevOps_Literature_Review*, *October*, 1–27. https://doi.org/10.13140/2.1.5125.1201

[14] Mohamed, S. I. (2015). DevOps Shifting Software Engineering Strategy Value Based Perspective. *IOSR Journal of Computer Engineering Ver. IV*, *17*(2), 2278–2661. https://doi.org/10.9790/0661-17245157

[15] Priera, J. M., & Ganefi, R. T. (2017). Automatic Deployment System Dengan Menggunakan Metode Continuous Integration Di Kakatu. *Jurnal Ilmiah Komputer Dan Informatika*.

[16] Pittet, S. (2021). *Continuous Deployment*. Atlassian. https://www.atlassian.com/continuous-delivery/continuous-deployment

[17] Docker. (2018). *What is a Container?* https://www.docker.com/resources/what-container

[18] Shichkina, Y. A., Kupriyanov, M. S., & Moldachev, S. O. (2018). Application of Docker Swarm cluster for testing programs, developed for system of devices within paradigm of Internet of things. *Journal of Physics: Conference Series*, *1015*, 32129. https://doi.org/10.1088/1742-6596/1015/3/032129

[19] Ismail, B. I., Goortani, E. M., Karim, M. B. A., Tat, W. M., Setapa, S., Luke, J. Y., & Hoe, O. H. (2015). Evaluation of Docker as Edge computing platform. *2015 IEEE Conference on Open Systems (ICOS)*, 130–135. https://doi.org/10.1109/ICOS.2015.7377291

[20] Peham, T. (2017). *GitLab vs GitHub: What are the key differences? The Ultimate Guide*. https://usersnap.com/blog/gitlab-github/