



TRUNG NGUYEN  
THE No.1 COFFEE!

# FINAL PROJECT REPORT

Prepared for :  
**TRUNG NGUYEN COFFEE CORP.**

Prepared by : Group 1

**Dang Vi Luan – 103802759**

**Nguyen Linh Dan – 103488557**

**Tran Bao Huy – 103505799**

**Vo Thanh Tam – 103487596**



## TABLE OF CONTENTS

<b>EXECUTIVE SUMMARY .....</b>	<b>1</b>
<b>1. CONSULTATION PHASE .....</b>	<b>2</b>
1.1 Project Background .....	2
1.2 Project Objectives and Problem Statement .....	2
<b>2. PRELIMINARY DESIGN CONCEPTS .....</b>	<b>3</b>
2.1 Design Phase .....	4
2.2 Development Phase .....	8
2.3 Expected Outcomes .....	9
<b>3. COMPATIBILITY OF DESIGN .....</b>	<b>10</b>
3.1 Specifications .....	10
3.2 Data Display .....	11
3.3 Code Explanation .....	13
3.4 Demonstration of the Solutions .....	17
<b>4. PROJECT OUTCOMES .....</b>	<b>19</b>
4.1 How the outcomes met the initial requirements .....	19
4.2 Advanced Features .....	20
4.3 Further Considerations .....	23
<b>CONCLUSION .....</b>	<b>23</b>
<b>APPENDIX .....</b>	<b>24</b>

## EXECUTIVE SUMMARY

Trung Nguyen Coffee Group, now referred to as the Client, is a renowned brand in the global coffee industry with a wide network of distributors across the globe. Trung Nguyen's coffee products have become an indispensable part of many coffee drinkers' lives. The Client is planning to use a web application to do e-commerce and receive online orders from customers. The development of new features for the website followed the traditional waterfall methodology.

This posed significant challenges of reduced efficiency, limited flexibility, and collaboration, especially in complicated change development. Additionally, the waterfall methodology delayed the testing procedures to the end of the projects, making it difficult to identify issues in time for remediation. More importantly, the conventional methodology typically extended the development life cycle, posing risks of increased time to market or operational disruption.

The scope of the project is categorized into functional and non-functional requirements. For functional requirements, our team would deliver a complete DevOps CI/CD pipeline that satisfies all of the Client's functionality requirements. For non-functional requirements, our team would ensure the pipeline's performance was optimized with scalability and minimal latency. Additionally, the final solution needs to achieve a high level of reliability, integrability, and observability.

By the completion of the project, the CI/CD pipeline has been successfully implemented with the current e-commerce website application of the Client. The efficiency, availability, and redundancy of the system are significantly improved with the new pipeline. However, to effectively utilize and maintain the pipeline, additional training sessions and documentation should be provided for the IT team members.

## 1. CONSULTATION PHASE

### 1.1 Project Background

Our client - Trung Nguyen Coffee Group, is entering a fast-paced e-commerce industry with a website application. In recent years, the growth of e-commerce businesses has been significant with many large companies establishing their e-commerce presence. With a shift in consumers' behavior from direct shopping to online shopping, a decent proportion of revenue is coming from e-commerce. Because of the dynamic nature of the industry, the website application is required to be continuously updated with new features and functionalities to provide customers with competent services.

The online website is projected to be a main sales contributor and an enhancement of customer engagement and brand loyalty. As Trung Nguyen is a newcomer in the e-commerce field, the development methodology used by their IT team was the traditional waterfall method. Requests for new features and functionalities were received and developed manually by the IT team. Testing procedures would only be conducted at the end of development. This made it difficult to identify and resolve functional issues on time. With extended development lifecycles, the requested changes or new features may not be deployed in time to facilitate the market's needs. In the e-commerce industry, the ability to instantly and continuously respond to users' demands or market trends is crucial to the success of a business. The website application is the representation of the client's flexibility and responsiveness to a forever-changing industry.

Additionally, for complicated features, the website had to experience serious downtime, disrupting the user experience. The website would need to be taken offline to deploy new features or perform maintenance. A significant issue in deployment can disrupt the business severely, directly damaging the revenue flow of the entire business. Moreover, with a large number of other e-commerce coffee businesses, customers would easily find new alternatives while Trung Nguyen's website was experiencing downtime.

Therefore, due to the growing demand for a website with high availability and agility, Trung Nguyen Coffee has decided to hire our project team to design and implement a secured CI/CD pipeline to mitigate the current problems and achieve their business goals.

### 1.2 Project Objectives and Problem Statement

#### **Project Objectives**

Our team has conducted onsite observation and evaluation of the current system at Trung Nguyen Coffee Group. After thorough analysis and exchangement with the Client's IT team and board of directors, our team has determined the following objectives for our CI/CD pipeline project:

- **Design the complete CI/CD pipeline**

Combine the team's technical expertise and detailed discussions with the client to produce a CI/CD pipeline design that satisfies all the client's requirements.

- **Prepare infrastructure for the CI/CD pipeline**

Set up and configure the required tools and infrastructure such as GitHub, Terraform, Docker Hub, SonarQube, ArgoCD, Kubernetes, and AWS services to fully support the CI/CD pipeline.

- **Implement the CI/CD pipeline**

Develop and implement the actual solution into the prepared infrastructure to ensure a smooth operation as intended in the planning.

- **Perform testing after implementation**

Conduct technical testing and user acceptance testing (UAT) to ensure the system works as intended and the user experience quality is satisfactory.

- **Prepare User Guide**

Create comprehensive user guides and training materials to help the students get familiar with the Agile application development methodology and technical aspects of the CI/CD pipeline.

- **Develop a new user interface for the e-commerce platform (minor objective)**

Design and develop a new e-commerce website for Trung Nguyen Coffee (UI only).

- **Integrate the new website with the CI/CD pipeline**

This is the most important objective of the project. The CI/CD pipeline must be integrated with the e-commerce platform and successfully perform automatic deployment, testing, etc.

### **Problem Statement**

Per discussion with the client, our team has established the following list of major problems with the current development process of the website application. The problems are summarized as follows:

**Inadaptability with changes:** With the expected increase in both quantity and complexity of new features and functionalities, the conventional waterfall approach limits the system's flexibility to develop and implement new changes to facilitate customers' demands and market trends.

**Delayed Feedback:** The testing procedures are only performed at the end of the development cycle. This makes it extremely difficult to debug or improve function quality.

**Extended Development Duration:** Due to the waterfall model's linear and sequential nature, the development time is prolonged as different developers or teams have to wait for the functional completion of the previous development to kick - start their processes.

**Increased Downtime:** For complicated features, complexity in deployment and integration may require the website application to be offline for a considerable amount of time, leading to business disruption.

**Security Risks:** The current system is heavily lacking the implementation of security measures to prevent unauthorized changes and access made to the system. Additionally, the waterfall methodology prolongs the integration of security measures till the end of the development, leading to potential security breaches.

## **2. PRELIMINARY DESIGN CONCEPTS**

In the initial project proposal, the requirements of our client – Trung Nguyen Coffee only covered the development of a secure CI/CD pipeline to support the business' e-commerce platform operations. The initial scope did not cover the development of the e-commerce platforms.

However, the project scope has broadened because the current e-commerce platform of Trung Nguyen was too outdated and might not be compatible with the CI/CD pipeline that would be developed in this project. Therefore, Trung Nguyen decided to broaden the project scope with the new e-commerce platform development. However, because the project timeline is limited, Trung Nguyen only requested the project team to support the user interface design and development. Other components related to the back end of the platform will be handled by another team.

In summary, the updated project scope of Trung Nguyen Coffee will cover the following points:

- Design and development of a secure CI/CD pipeline to support e-commerce operations.
- Design and development of the new e-commerce platform's user interface.

## 2.1 Design Phase

### Development methodology

Our project team will apply the Agile methodology to work on this project. This approach concentrates on iterative development and feedback, encourages close collaboration with stakeholders, and focuses on delivering incremental value throughout the project lifecycle. The project will be broken down into six 2-week sprints. Instead of following a sequential process, our team will work on the CI/CD pipeline and the user interface concurrently. This parallel approach is intended to help identify any conflicts between the CI/CD pipeline and the new user interface early on, in order to ensure that the project's objectives can be met within the overall 12-week timeline.

Below are the backlog items of 06 sprints:

No.	Item	Dependencies	Business Value (1– 10)	Sprint
<b>Sprint 1</b>				
F1	Gather the requirements for the secure CI/CD pipeline and the e-commerce platform's user interface.	-	9	Sprint 1
F2	Select the tools and technologies for the CI/CD pipeline.	-	8	Sprint 1
F3	Design the high-level architecture of the CI/CD pipeline, with the integration points with the e-commerce platform.	F2	10	Sprint 1
F4	Set up the development environment.	-	8	Sprint 1
<b>Sprint 2</b>				
F5	Implement the CI part of the CI/CD pipeline (automated testing and code quality checks).	-	10	Sprint 2
F6	Integrate the CI pipeline with the e-commerce platform codebase.	F5	10	Sprint 2
F7	Demonstrate the user interface mockups/prototypes to the clients.	-	9	Sprint 2



F8	Modify and finalize the user interface design based on clients' feedback.	F7	8	Sprint 2
<b>Sprint 3</b>				
F9	Implement the CD part of the CI/CD pipeline.	F5	10	Sprint 3
F10	Implement the basic layout and structure for the UI using HTML, CSS, and Javascript (1)	F8	8	Sprint 3
F11	Set up the monitoring tools for the CI/CD pipeline.	F9	8	Sprint 3
<b>Sprint 4</b>				
F12	Add the security checks, vulnerability scanning tools, and image scanning to the CI/CD pipeline.	F9	7	Sprint 4
F13	Implement the basic layout and structure for the UI using HTML, CSS, and Javascript. (2)	F10	8	Sprint 4
F14	Develop/add more UI components for the e-commerce platform, for example: <ul style="list-style-type: none"> <li>Navigation bar</li> <li>All web pages (homepage, menu, order page, registration page)</li> <li>Add product and brand images.</li> <li>Add content requested by the clients to the interface.</li> </ul>	F13	8	Sprint 4
F15	Implement data validation for all forms (order, register) on the e-commerce platform.	F14	7	Sprint 4
<b>Sprint 5</b>				
F16	Implement role-based access control (RBAC) and other security measures in the CI/CD pipeline.	F12	7	Sprint 5
F17	Finalize all e-commerce platform's web pages (with content and images added).	F15	9	Sprint 5
F18	Implement automated testing and deployment for the integration between the e-commerce platform and the CI/CD pipeline.	F16	10	Sprint 5
<b>Sprint 6</b>				
F19	Finish the e-commerce platform's UI and ensure consistency across all pages.	F18	8	Sprint 6
F20	Conduct usability testing and check the performance when integrating the CI/CD pipeline into the e-commerce platform.	F18	8	Sprint 6
F21	Prepare relevant documents, including: <ul style="list-style-type: none"> <li>The user manual</li> <li>Final project report</li> <li>The solution package</li> </ul>	-	8	Sprint 6

To deliver the project on time, the team will work on the CI/CD pipeline and user interface development in parallel. The team structure and roles will be outlined are follows:

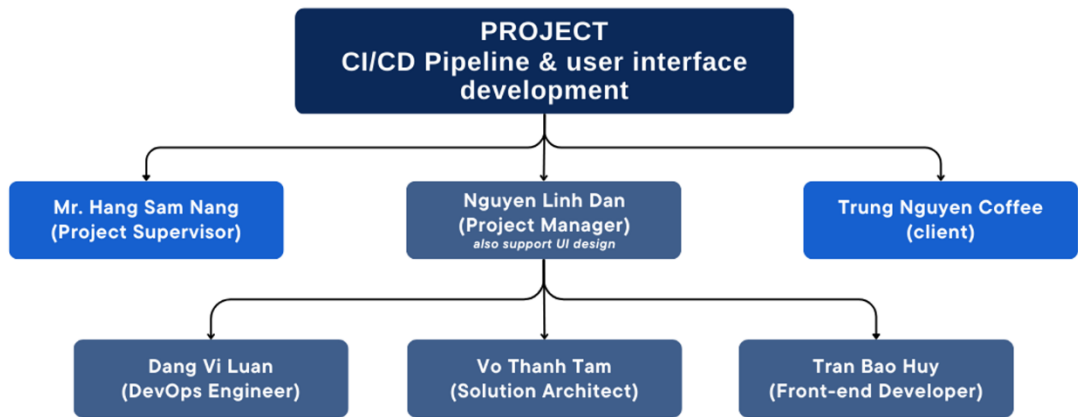


Figure 1: Project team structure

**Stage 1: Design the CI/CD pipeline architecture**

The planning for the CI/CD pipeline architecture involves a structured and strategic approach to ensure efficiency, security, and scalability. Utilizing GitOps as the DevOps methodology, we will leverage Git for version control and use GitHub Actions for the CI workflow due to its simplicity and integration capabilities. This workflow will enable automated build and testing processes, ensuring that every code commit is seamlessly containerized into an image and pushed onto Docker Hub. This method ensures consistent and traceable deployment artifacts.

For the CD workflow, we will use AWS EKS (Elastic Kubernetes Service) and ArgoCD for deployment and synchronization. AWS EKS provides a robust and scalable Kubernetes environment, while ArgoCD ensures continuous deployment by automating the synchronization of application state with the defined desired state. This setup allows for efficient and reliable deployment of containerized applications, maintaining consistency across different environments.

Monitoring and observability are crucial for maintaining the health and performance of the CI/CD pipeline. We will implement Prometheus and Grafana for this purpose. Prometheus will be used to collect and store time-series metrics, providing detailed insights into system performance. Grafana will visualize these metrics in real-time dashboards, enabling quick identification of issues and facilitating prompt incident response.

Integrating security into the CI/CD pipeline is a critical aspect of our design. We will implement DevSecOps practices to ensure robust security measures. For Static Application Security Testing (SAST), we will use SonarQube to analyze code for vulnerabilities and ensure code quality. Dynamic Application Security Testing (DAST) will be performed using OWASP ZAP to identify vulnerabilities in running applications. To manage sensitive credentials, GitHub Secrets will be used, ensuring secure access and storage of credentials throughout the CI/CD process.



## Stage 2: Design the e-commerce platform's user interface

Overall, the platform's user interface will include the following pages:

- Homepage
- Menu page
- Order page
- Registration page

Before developing the actual user interface, our team will first design mockups and prototypes using Figma. This will allow us to present the proposed UI designs to the clients for feedback and input. After gathering feedback from the clients, we will modify the prototypes accordingly. This approach ensures that the user interface is thoroughly planned and reviewed with the clients before development. It minimizes rework and helps us deliver a UI that meets the client's expectations and requirements.

Below is our mockup of the user interface:

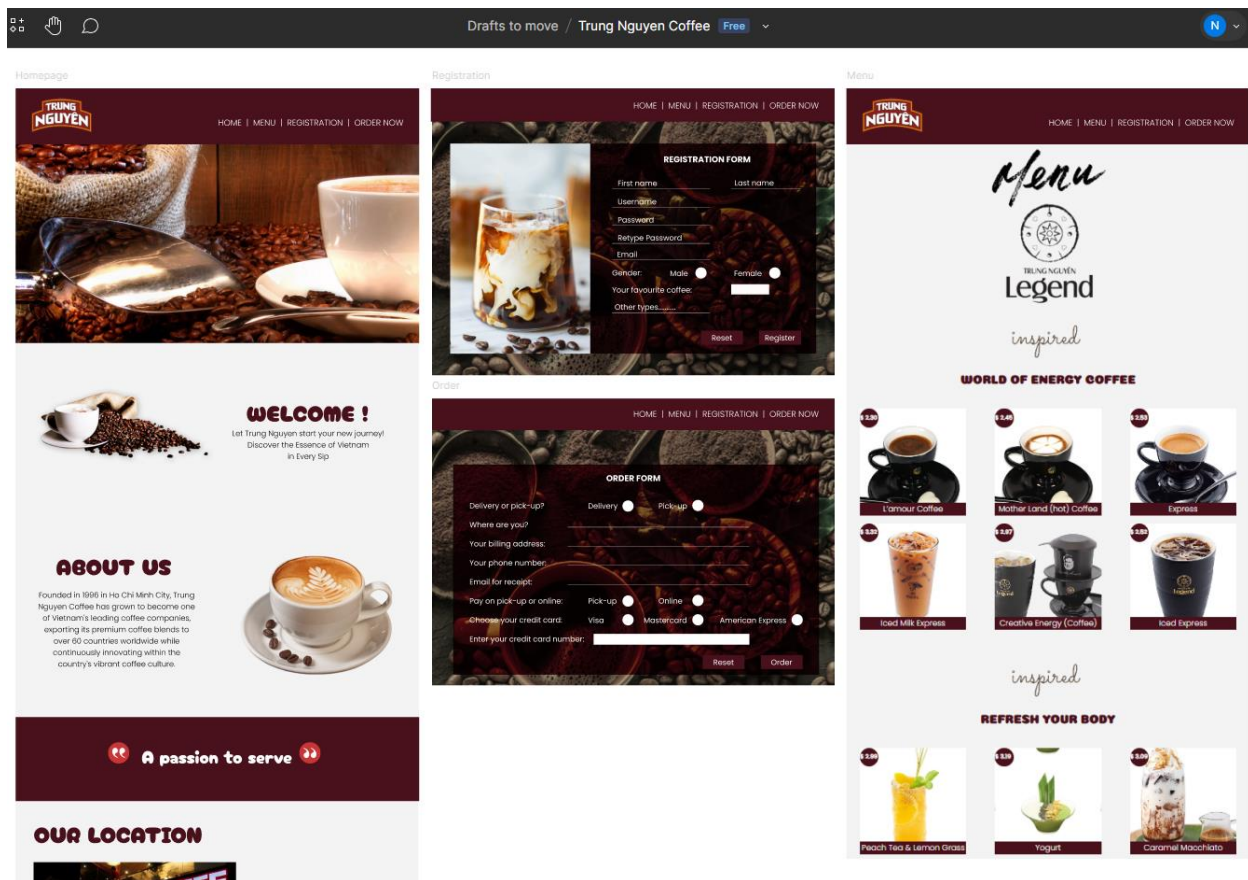


Figure 2: User interface mockups (Figma)

## 2.2 Development Phase

### Stage 1: Develop the CI/CD pipeline

Main PICs of this task:

- Vi Luan – DevOps Engineer
- Thanh Tam – Solution Architect

#### Action 1: Version Control Integration

The CI/CD development process begins with Version Control Integration, where version control systems are set up using Git. Repositories for the main application, associated services, and infrastructure code are created on platforms like GitHub, Terraform Registry, and DockerHub to ensure organized and systematic code management.

#### Action 2: CI/CD flow

In the Continuous Integration (CI) phase, automated build and testing processes are implemented. ArgoCD is used to trigger builds upon code commits or merges, ensuring continuous and efficient integration. This stage also incorporates unit tests, integration tests, and static code analysis tools to validate code quality and functionality.

Deployment involves integrating Docker for building and pushing container images. Dockerfiles are created for services, and the CI process is configured to automatically build and publish these Docker images. This ensures consistent and traceable deployment artifacts across different environments.

#### Action 3: DevSecOps and security measures integration

Finally, in the DevSecOps Integration phase, security measures are embedded into the CI/CD pipeline. Pre-commit hooks, AWS Secrets Manager, SonarQube, OWASP ZAP, and Docker image scanning tools are implemented to ensure robust security. Additionally, monitoring and alerting systems using Prometheus and Grafana are established to maintain system health and performance, providing timely incident response.

### Stage 2: Implement the platform's user interface

At this stage, the user interface design created in Figma has now been reviewed and approved by the clients. The next step for our team is to translate this approved design into functional code.

Main PIC of this task: Bao Huy - Front-end developer

Supporter: Linh Dan

#### Action 1: Develop HTML structure and layouts

The front-end developer will translate the final prototypes into semantic HTML layouts.

To avoid further modification, the HTML structure is recommended to be well-organized.

#### Action 2: Apply CSS styling

- The front-end developer will add custom CSS to style the HTML elements and create a visual appearance similar to the Figma design.

- The classes and IDs used in CSS code are recommended following a naming convention.

#### Action 3: Implement interactive features using Javascript.

- Add the mechanism to handle the form validation.
- Use Javascript to create a responsive navigation bar.

#### Action 4: Testing

- Validate and ensure that all components on the user interface are consistent and follow the initial design. Because our project team does not have any testers, the front-end developer will handle this action.

## **2.3 Expected Outcomes**

Below are the expected outcomes after proposing a secured CI/CD Pipeline for Trung Nguyen Coffee:

- Consistent and reliable builds: The team's proposed CI/CD Pipeline should guarantee the consistency of the built application regardless of the environment: Development, Staging, and Production. In case any unexpected failures occur in the newest version, it can quickly re-deploy the old version to deliver the most acceptable production for end-users.
- Automation testing: To enhance the productivity of the developer team, the CI/CD pipeline should incorporate automation testing such as unit testing, functional testing, etc. By integrating automated testing, bugs can be identified without manual work. Hence, all the issues will be addressed as soon as possible by the development team.
- Secured and compliance deployment: Integrating the DevSecOps pipeline (Git Secret, Sonarqube, etc.) ensures the best security practices in the final deployment, such as avoiding using credentials in the codebase, and guaranteeing clean code implementation.
- Faster delivery: The process of testing, building, and deploying is automated by the CI/CD Pipeline. Hence, it will save time and effort for the development team to update/release the new version/features of the application.
- High availability and reliability: The pipeline should ensure high availability in continuous operation, which means that it should minimize the downtime period of the application during the process of updating new features.
- Continuous improvement: The CI/CD Pipeline is implemented with monitor tools such as Prometheus and Grafana, allowing the development team to consistently keep track of the performance, identify the bottleneck, and optimize the speed, efficiency, and reliability.

### 3. COMPATIBILITY OF DESIGN

#### 3.1 Specifications

- **Secure CI/CD pipeline**

##### **Functional requirements**

###### A. DevOps CI/CD Pipeline

The DevOps CI/CD pipeline integrates and automates development, testing, and deployment to streamline software delivery. The codebase is managed with Git, using platforms like GitHub, Terraform Registry, and DockerHub for repositories. Automation, triggered by ArgoCD, handles build and testing upon code commits or merges. Functionality and quality are validated through unit tests, integration tests, and static code analysis, providing feedback to developers. Docker is used to build and push versioned container images, while Terraform handles cloud infrastructure provisioning and deploys the application to a Kubernetes cluster.

###### B. DevSecOps Integration

Security is embedded into the CI/CD pipeline with pre-commit hooks using git-secrets to prevent accidental commits of sensitive information and AWS Secrets Manager for credential control. SonarQube performs Static Application Security Testing (SAST), GitHub handles Source Composition Analysis (SCA), and OWASP ZAP identifies vulnerabilities in running applications. Docker images are scanned for vulnerabilities with tools like Clair. Prometheus collects and stores metrics, while Grafana creates dashboards for monitoring system health and performance, with alerts set up for timely incident response.

##### **Non-functional requirements**

The CI/CD pipeline must handle increasing workloads, support horizontal scaling, and optimize resource utilization to minimize delays and bottlenecks. High availability and fault tolerance are ensured through redundancy and failover mechanisms. Compatibility with various development tools and platforms is maintained through standard protocols and interfaces. Prometheus and Grafana provide observability with metrics collection and real-time dashboard visualization, while Prometheus and Telegram enable real-time alerting for prompt incident response.

- **Ecommerce platform's user interface**

##### **Functional requirements**

###### Homepage

- Display all required information sections (the slideshow, welcome, location, menu, and registration sections) clearly with the content and images provided by the clients.
- Provide the navigation bar correctly.

###### Menu page

- Display the product information, such as the images, product names, and prices correctly (with a maximum of 4 products per row).

- The product list is categorized into 3 sections: coffee, other drinks, and breakfast.

#### Registration page

- Display a form that allows new visitors to register an account on the platform.
- Form validation is added to handle missing information and information with incorrect format in the form.

#### Order page

- Display a form that can guide customers through a checkout flow, including filling in the shipping address, payment method, etc.
- Hide unnecessary sections; only display them when a specific option is selected. For example, only display the field to enter a card number when customers select to pay by card.

#### **Non-functional requirements**

- User-friendly: The user interface should ensure a clean and responsive user interface that is suitable for various device sizes.
- Performance: Consider the size of the asset used on the interface (logos, images) to minimize the loading time.
- Easy to maintain: The code written for the user interface is well-structured, follows consistent naming conventions, and includes comments explaining the functionality of different parts of the code.

**Note:** *As the main purpose of this project is building a CI/CD pipeline and a new user interface for the Trung Nguyen e-commerce platform, other requirements related to the platform backend are out of scope.*

### **3.2 Data Display**

Trung Nguyen Coffee's website includes various pages, such as Home, Menu, Registration, and Order. Each section is designed to provide information about the company, its offerings, and ways for customers to engage. After packaging the source code of the website and deploying using the developed CI/CD pipeline, the website is successfully deployed and hosted on Amazon Cloud Services (AWS). All website's content is correctly displayed on the interface. Features we can observe after the deployment using the CI/CD pipeline are summarized as follows:

#### Homepage

- **Navigation Bar**: Includes links to Home, Menu, Registration, and Order pages.
- **Welcome Section**: A welcoming message encouraging visitors to start their journey with Trung Nguyen Coffee.
- **About Us**: Details about the founding of Trung Nguyen Coffee and its growth.
- **Quote**: A motivational quote emphasizing the company's passion for service.
- **Our Location**: Information on store locations, opening hours, and contact numbers.
- **Menu Highlights**: Overview of the menu items categorized into Energy Coffee, Fresh Drinks, and Healthy Breakfast.

- Membership and Online Ordering: Promotes membership registration and online ordering.

#### Menu page

- Navigation Bar: Links to Home, Menu, Registration, and Order pages.
- Full Menu Display: Detailed sections showcasing various items under categories such as World of Energy Coffee, Refresh Your Body, and Time for Breakfast.
- Food and Drinks: Specific items with names and prices, such as L'amour Coffee (\$2.30), Peach Tea & Lemon Grass (\$2.99), and Steak and Fried Eggs (\$3.99).

#### Order page

- Navigation Bar: Links to Home, Menu, Registration, and Order pages.
- Order Form: A form for customers to place orders, choose delivery or pickup, enter delivery and billing addresses, contact number, and payment method.
- Payment Options: Includes options for paying on pickup or online, and selecting credit card types like Visa, Mastercard, and American Express.

#### Registration page

- Navigation Bar: Links to Home, Menu, Registration, and Order pages.
- Registration form: The registration form allows users to input their personal information to register on the website. The form collects details such as name, username, password, email, postcode, gender, and favorite coffee.

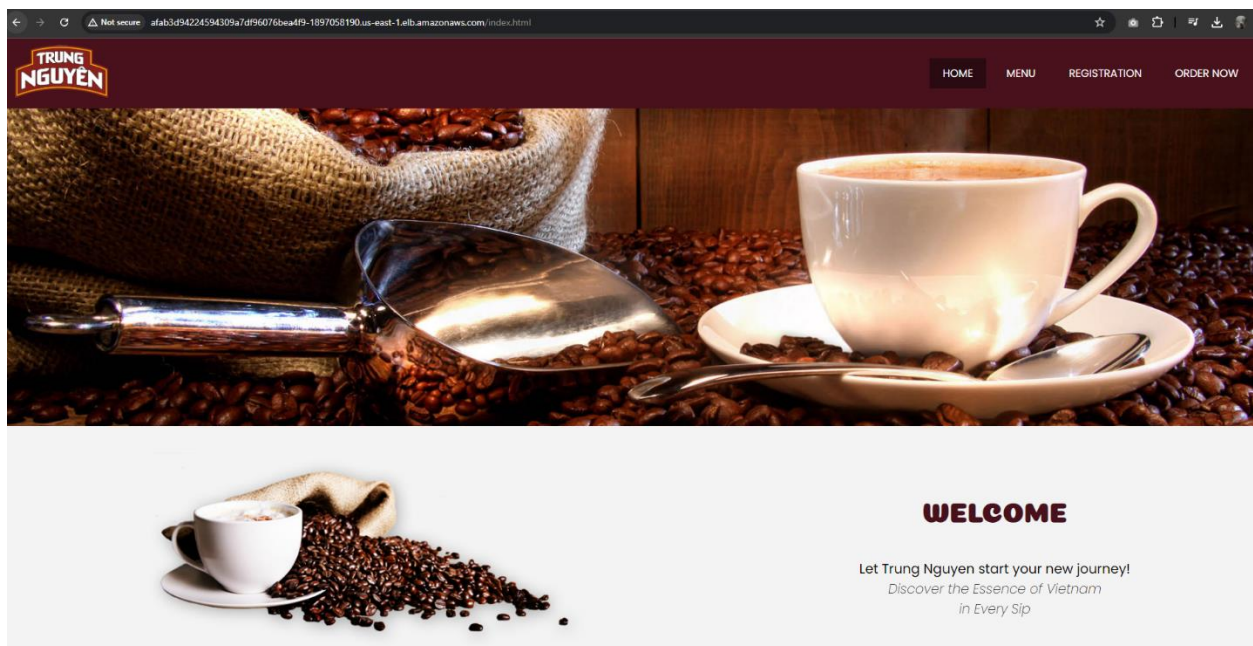


Figure 3: Homepage – the web is deployed and hosted on AWS



### 3.3 Code Explanation

#### Repository structure

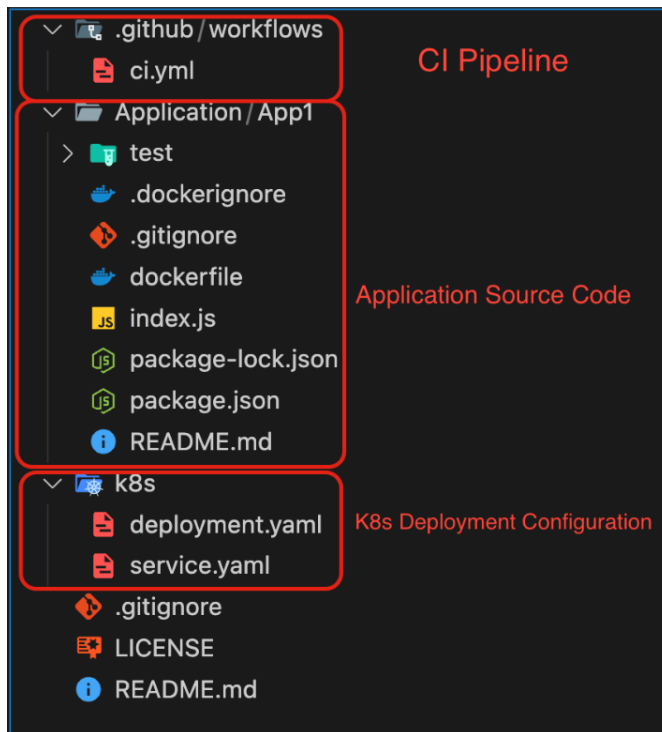


Figure 4: Repository structure

From *Figure 4* above, there are 03 folders that need to be noted:

- **.github** folder: The remote repository is Github. The folder `.github` was created for one reason: every time the code is pushed to the remote repository, Github will look at the file `.github/workflows/ci.yml` to trigger the CI Pipeline.
- **Application** folder: Developers will work at this folder for developing applications for Trung Nguyen Corporation Company.
- **K8s** folder: This folder stores the configuration files for deployment to K8s Cluster.

#### Using Git Secret to hide credentials

In the Continuous Integration Pipeline, there are some required credentials as listed below:

- Docker Username
- Docker Password
- Git Username
- Git Email

## CI/CD Pipeline Flow

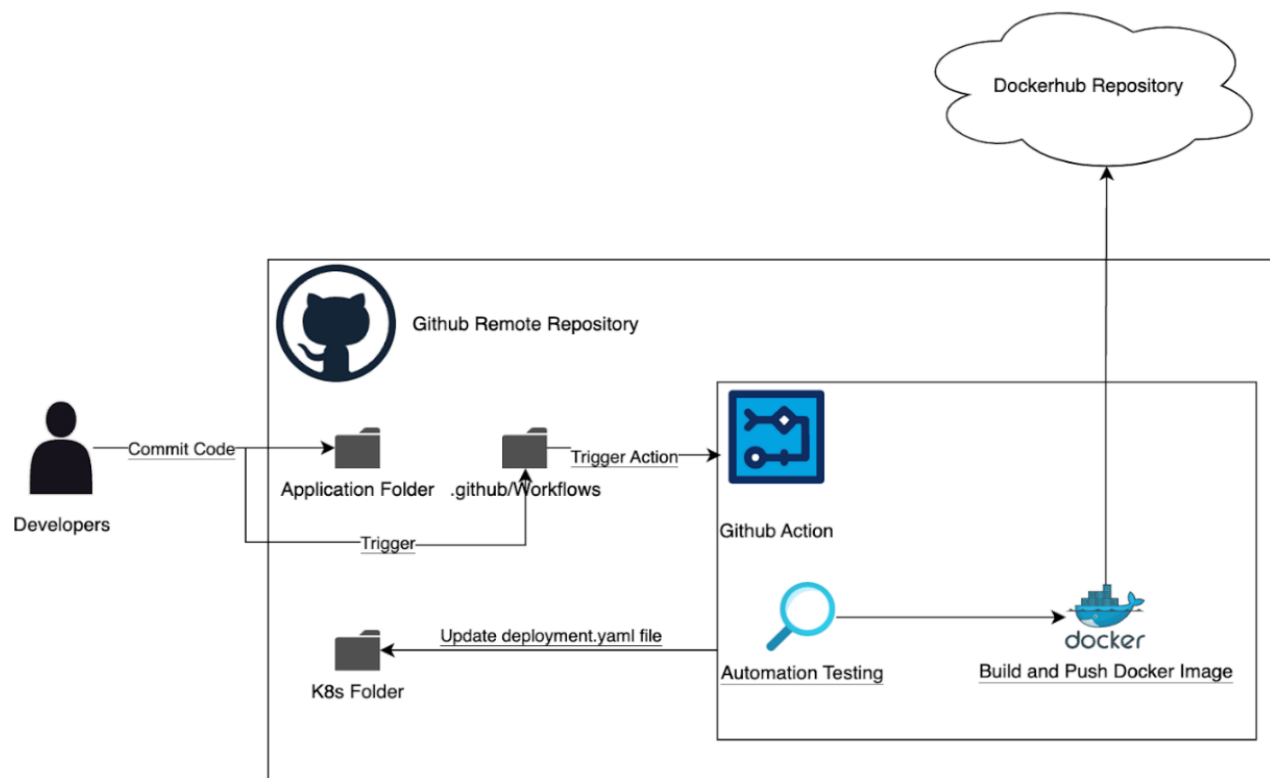


Figure 5: CI/CD pipeline flow

[1] The developers commit their code at folder *Application* and push the commit to the remote repository (Github)

[2] After pushing code to the remote repository, Github will trigger Github Action to process the Continuous Integration stage.

[3] Github Action will follow the configured structure in the *CI.yml* file, which is located in the folder *.github/Workflows*, and process the following jobs: automation testing, build docker image, and push the image to Docker Hub. Finally, Github Action will modify the file *deployment.yaml* located in the K8s Folder to prepare for the continuous deployment stage.

[4] ArgoCD will check the K8s Folder to see whether there is any modification in the codebase. If so, ArgoCD will follow the instruction configured in the folder K8s to deploy the application to K8s Cluster.

## Continuous Integration (CI) Pipeline with Github Action and Docker

This section will explain the codebase of the *CI.yml* file beginning with *Figure 6* below.

```
.github > workflows > ci.yml
1  name: CI Flow
2
3  on:
4    push:
5      branches:
6        - main
7    workflow_dispatch: {}
8
9  jobs:
10   test-image:
11     runs-on: ubuntu-latest
12     steps:
13       - uses: actions/checkout@v4
14
15       - name: Build and push test Docker image
16         working-directory: ./Application/App1/Docker
17         run: |
18           docker build . -t ${ secrets.DOCKER_USERNAME }}/ict30001-test:${ github.
19             sha }}
20           echo "${ secrets.DOCKER_PASSWORD }}" | docker login -u ${ secrets.
21             DOCKER_USERNAME }} --password-stdin
22           docker push ${ secrets.DOCKER_USERNAME }}/ict30001-test:${ github.sha }}
```

Figure 6.1: File CI.yml for continuous integration (1)

From lines 1 - 7 in *Figure 6.1*, any changes in branch 'main' will trigger the Github Action to process the CI Flow job. Next, the entire process from lines 9 - 20 is: Github Action will build the Docker Image based on the instruction in the Dockerfile, which is located in the folder Application/App1. Finally, Github Action will tag the built image with the latest commit code (84178e921, e.g.) and push the image to the Docker Hub Repository after successful authentication.

```
.github > workflows > ci.yml
9  jobs:
21
22   update-manifest:
23     runs-on: ubuntu-latest
24     needs: test-image
25
26     steps:
27       - name: Check out code
28         uses: actions/checkout@v2
29
30       - name: Update Image Tag Values
31         run: |
32           deployment_file="./k8s/deployment.yaml" # Update with the actual path to
33             your deployment.yaml
34           new_image_tag=${ github.sha }}
35
36           # Update the deployment.yaml file with the new image tag
37           sed -i "s|image: dixluwn/ict30001-test:.*|image: dixluwn/
38             ict30001-test:$new_image_tag|" "$deployment_file"
39
40       - name: Commit the changes made
41         run: |
42           git config --global user.name "${ secrets.GIT_USER_NAME }}"
43           git config --global user.email "${ secrets.GIT_USER_EMAIL }}"
44           git commit -am "Updating image tag in deployment.yaml"
45           git push
```

Figure 6.2: File CI.yml for continuous integration (2)

After pushing the built image to DockerHub, Github action will update the file deployment.yaml, which is located in the folder k8s, with the latest tag before the continuous deployment process with ArgoCD and K8s. Finally, Github Action will commit the new change and push it to the Github Repository. In other words, once developers push a new commit to the remote repository, another commit will be made by GitHub Action.

## Continuous Deployment with Kubernetes and ArgoCD

Now, the Continuous Deployment phase requires two configuration files deployment.yaml and service.yaml, which are both located in the folder k8s.

```
k8s > deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: app-server-deployment
5  spec:
6    replicas: 3
7    selector:
8      matchLabels:
9        app: app-server
10   template:
11     metadata:
12       name: app-server
13     labels:
14       app: app-server
15     spec:
16       containers:
17       - name: app-server
18         image: dixluwn/
19         ict30001-test:84178e921fc320ca806f9a5fea178630ea890a32
20         ports:
21         - containerPort: 80
```




Figure 7: deployment.yaml

In *Figure 7* above, our deployment is named 'app-server-deployment' and there will be 3 replicas (3 pods) to ensure the high availability. In this file, ArgoCD will look at the image tag, which can be changed in the future, on line 18 to launch pods for us.

```
k8s > service.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: app-server-service
5    labels:
6      app: app-server
7  spec:
8    type: LoadBalancer
9    selector:
10     app: app-server
11   ports:
12   - protocol: TCP
13     port: 80
14     targetPort: 80
15
```

Figure 8: service.yaml

To communicate the pods, which are configured in file deployment.yaml, we need to create a 'service' with type 'Load Balancer' as *Figure 8*. From now on, we will use the port 80 (or HTTP protocol) to access the application (Lines 11-14).

### 3.4 Demonstration of the Solutions

With this holistic CI/CD pipeline installed, the development process of Trung Nguyen Coffee can be ensured to be seamless and efficient. After making changes, the developers can commit their codes and the CI Workflow above will be executed and GitHub will build a new docker container to reflect such changes.

After that, the Kubernetes manifest files will be updated for both local and remote environments. The below diagram visualizes how environments can be changed based on Kubernetes manifest files.

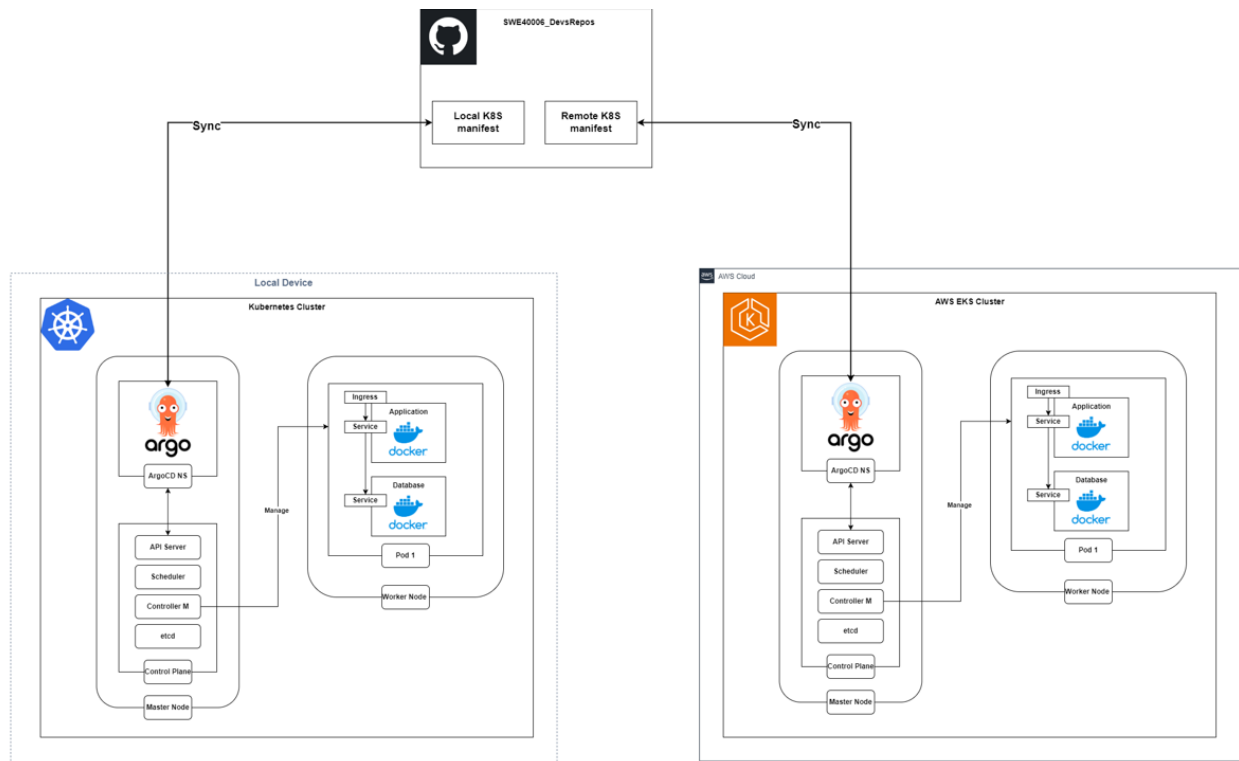


Figure 9: CD workflows

The AWS Elastic Kubernetes Services will be configured with 2 nodes always operating to ensure high availability, additionally our cluster can also scale to 3 nodes depending on the workload. The diagram below shows how AWS EKS works as the infrastructure for our application.

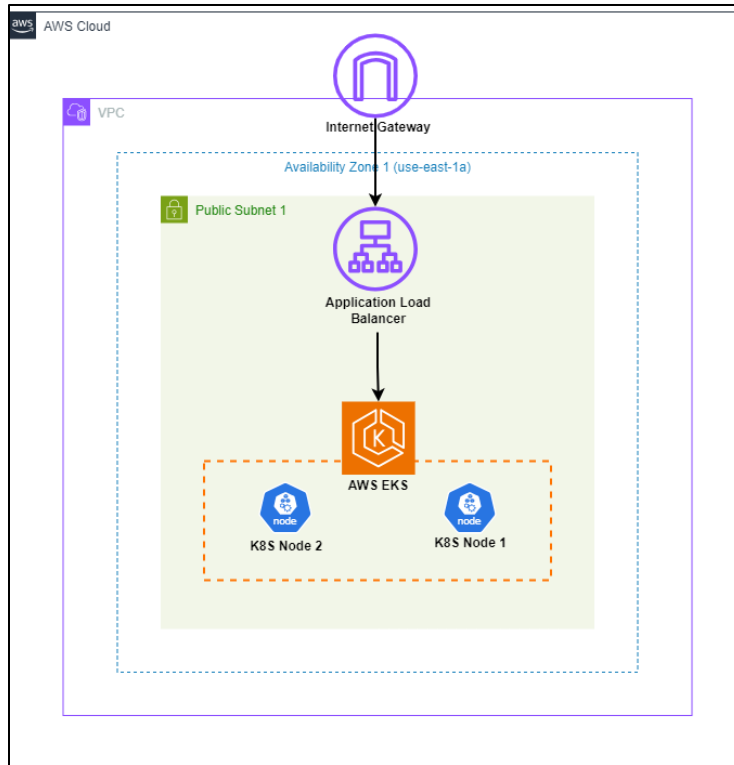


Figure 10: AWS EKS Infrastructure

With the infrastructure and operation services setup, the operation staff can create an application on ArgoCD UI to always track the state of the application's repository, changes will be made accordingly to the cluster should the developers commit any other codes.

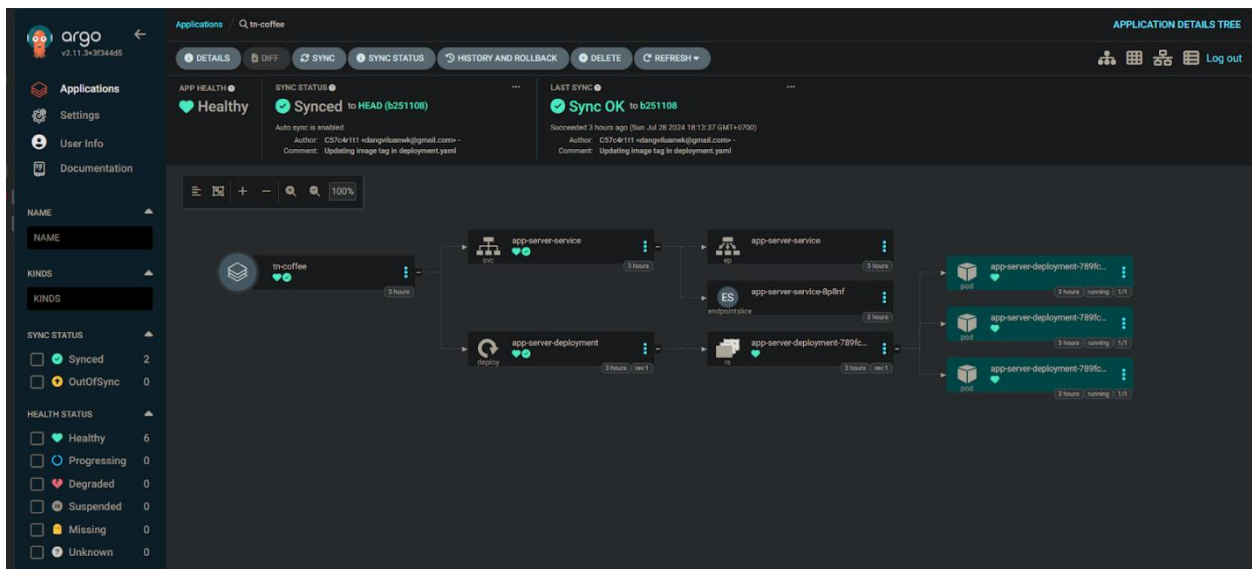


Figure 11: Create an application with ArgoCD



Monitoring features can also be accessed via the provided AWS Load Balancer's DNS.

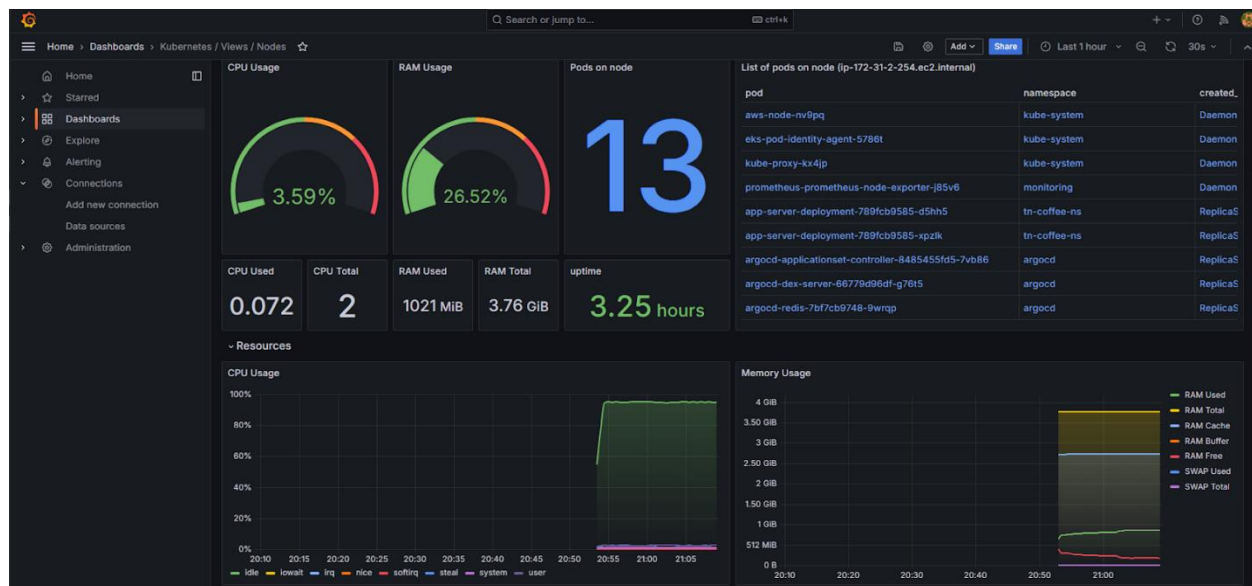


Figure 12: Grafana dashboard

## 4. PROJECT OUTCOMES

### 4.1 How the outcomes met the initial requirements

- **The CI/CD Pipeline solves the problem of inconsistent and insecure deployment**

As stated in the project brief, the current deployment stage requires a lot of manual work, which leads to delay in releasing/updating new features of the software. However, with the proposed CI/CD Pipeline, customers can quickly receive the new version of the application since the deployment process is automated and does not require any human involvement.

- **The CI/CD Pipeline solves the problem of lacking automated testing**

CI/CD Pipeline is integrated with automated testing. With automated testing, developers can quickly figure out bugs in the software as soon as possible and as much as possible.

- **The CI/CD Pipeline solves the problem of collaboration**

With version source control (VCS), developers have a place to store their code to improve the collaboration among the team. Additionally, developers can quickly see the deployed production with CI/CD Pipeline to be adaptive to any unexpected situation: any failures can be quickly detected and addressed. If errors cannot be fixed instantly, developers can solve the problem by reversing to the older version of the software.

- **The CI/CD Pipeline solves the problem of scalability**

Dealing with thousands of customers requires scalability in the infrastructure. Trung Nguyen Coffee Group can scale the system vertically by increasing the hardware resources. However, it will not be efficient since the company needs to spend its budget on vertical scaling. In the proposed CI/CD

pipeline, the team also integrated the feature infrastructure provision with Terraform and K8s. Hence, the system can scale up horizontally to deal with customers during peak periods and scale down properly to handle the usual number of customers.

- **The CI/CD Pipeline is integrated into the e-commerce platform and works correctly**

The CI/CD pipeline has been successfully integrated into the e-commerce platform and worked as expected. The pipeline can automate the build, test, and deployment processes.

## 4.2 Advanced Features

Our CI/CD pipeline adopts a specific method of DevOps called GitOps, which ensures that all code, infrastructure as code (IaC), and configuration files are managed using a single source of truth - Git. This approach not only simplifies the deployment process but also enhances the consistency and reliability of our infrastructure management.

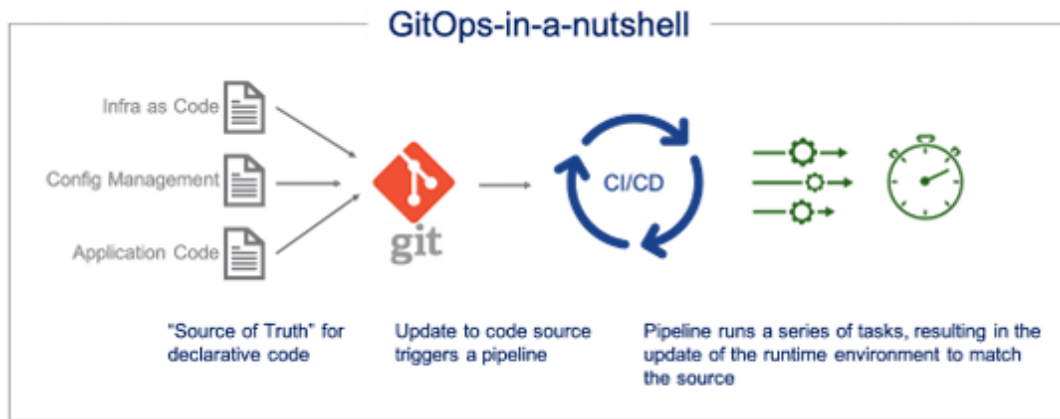


Figure 13: GitOps summary

By leveraging GitOps, any changes made in the Git repository are automatically reflected in the infrastructure, reducing the potential for human error and ensuring that the state of the infrastructure is always in sync with the codebase.

Additionally, our solution integrates robust monitoring capabilities using Prometheus and Grafana. These tools are essential for tracking the performance and health of the deployed product.

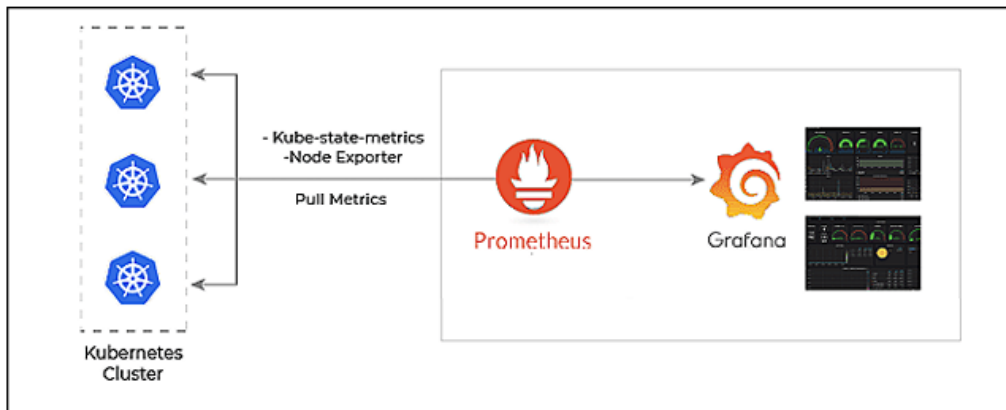


Figure 14: Prometheus and Grafana monitoring

Prometheus collects and stores metrics, while Grafana provides an intuitive interface for visualizing these metrics through customizable dashboards. This integration allows for real-time monitoring, proactive issue detection, and insightful analytics, helping maintain high availability and performance of the applications.

Security is also a crucial concern in our CI/CD pipeline, and we have integrated comprehensive security measures at each phase of the pipeline. These security measures include:

- **GitHub Secrets**

Secure management of sensitive information such as API keys and passwords.

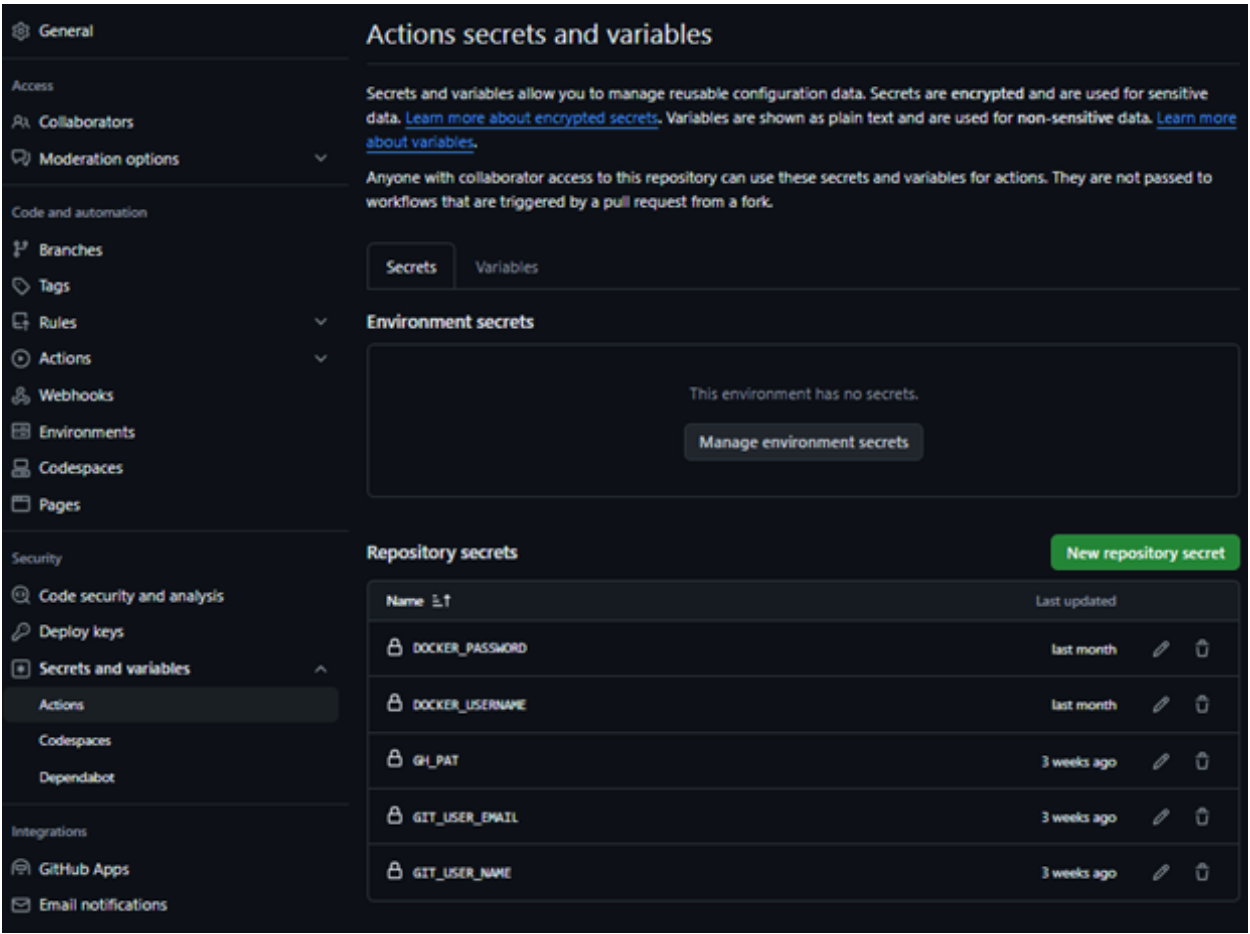


Figure 15: GitHub Secrets

- **Static Application Security Testing (SAST)**

Using SonarQube to analyze source code for vulnerabilities like SQL injection and cross-site scripting (XSS).

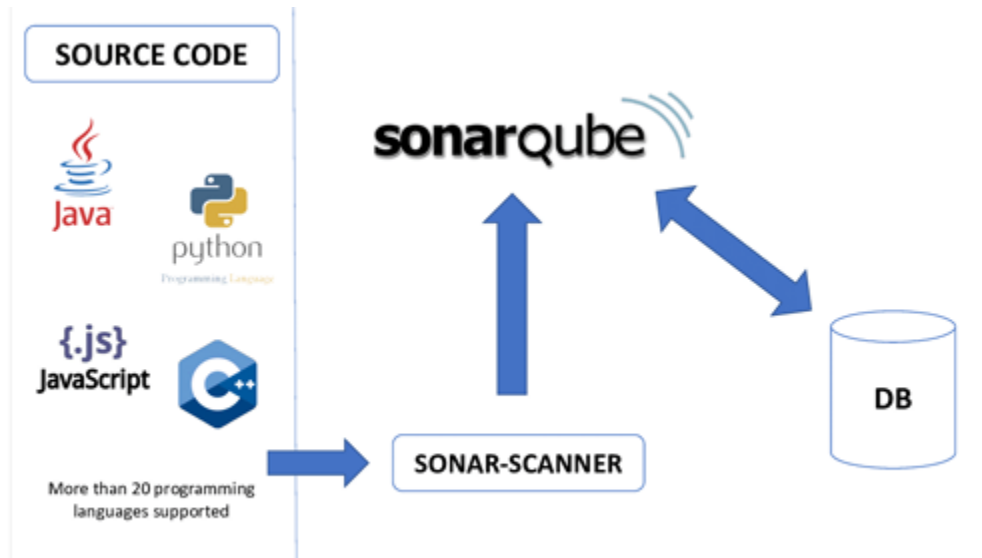


Figure 16: Sonarqube for SAST

- **Dynamic Application Security Testing (DAST)**

Using OWASP ZAP to simulate real-world attacks and identify security flaws in running applications.

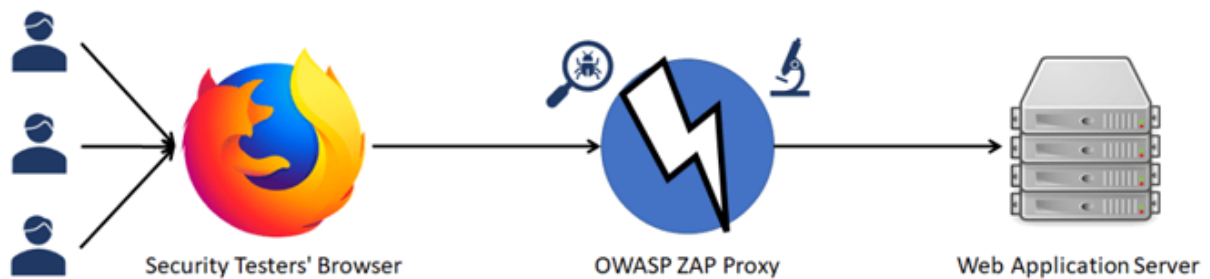


Figure 17: OWASP ZAP for DAST

### 4.3 Further Considerations

Currently, our product uses GitHub Actions as the CI engine, which provides seamless integration with our GitHub repositories and automates various stages of the CI/CD pipeline. However, we are considering integrating other CI engines such as Jenkins or Harness to enhance flexibility and scalability.

At present, the continuous deployment (CD) pipeline requires manual interaction to provision AWS Elastic Kubernetes Service (EKS) clusters. To further streamline our deployment process and improve scalability, we are exploring the automation of EKS provisioning. Automating this aspect would significantly reduce the time and effort required to deploy new environments, allowing for faster scaling and more consistent deployments.

We are also considering integrating Clair to enhance the security and compliance of our infrastructure as code (IaC). Clair is a powerful tool that scans container images and other IaC components for vulnerabilities, ensuring that any potential security issues are identified and addressed before deployment. Currently, our solution does not require additional infrastructure to be provisioned, therefore, Clair is not needed at the current stage.

By addressing these considerations, we aim to enhance the robustness and efficiency of our CI/CD pipeline, ensuring a seamless Development and Delivery process for Trung Nguyen Coffee.

## CONCLUSION

The project has successfully achieved its aim of delivering a secured CI/CD pipeline to the Client - Trung Nguyen Coffee Group. The CI/CD pipeline has satisfactorily accomplished all of the Client's functional and non-functional requirements, leading to significant improvements in both efficiency and security for the website application.

The implementation of continuous development, testing, and deployment has improved the website's flexibility and adaptability to new changes, facilitating customers' demands and market trends. Additionally, automated security features have enhanced the level of resilience against potential cybersecurity threats, ensuring the availability and reliability of the client's e-commerce platform.

More importantly, the CI/CD pipeline has transformed the Client's general approach to technological change in a more agile and active manner. However, the current CI/CD pipeline may also need to be updated and improved to serve a greater volume of traffic and more complicated functional requests following the business expansion of Trung Nguyen Coffee Group.

## APPENDIX

Solution Package: [Link](#)

*(includes the CI/CD pipeline source code and E-commerce website source code)*