

Introduction to DevOps
Practical Assignment

Name: Thong P Dang

Email (associated with Moodle): Thong.P.Dang@student.lut.fi

(Do not input your student number here)

1. Describe how you improved the DevOps environment at company X regarding the following aspects.
 - a. Version control (explain how you solved the merge conflict)

Step 1: Regarding creating a merge request from the **feature_3** branch into main branch, at first, I created the merge request directly on Gitlab, and you could see there was an error notification about merge conflicts. To start solving the merge conflicts, I followed an instruction to check out, review, and resolve locally in my local environment as the below image.

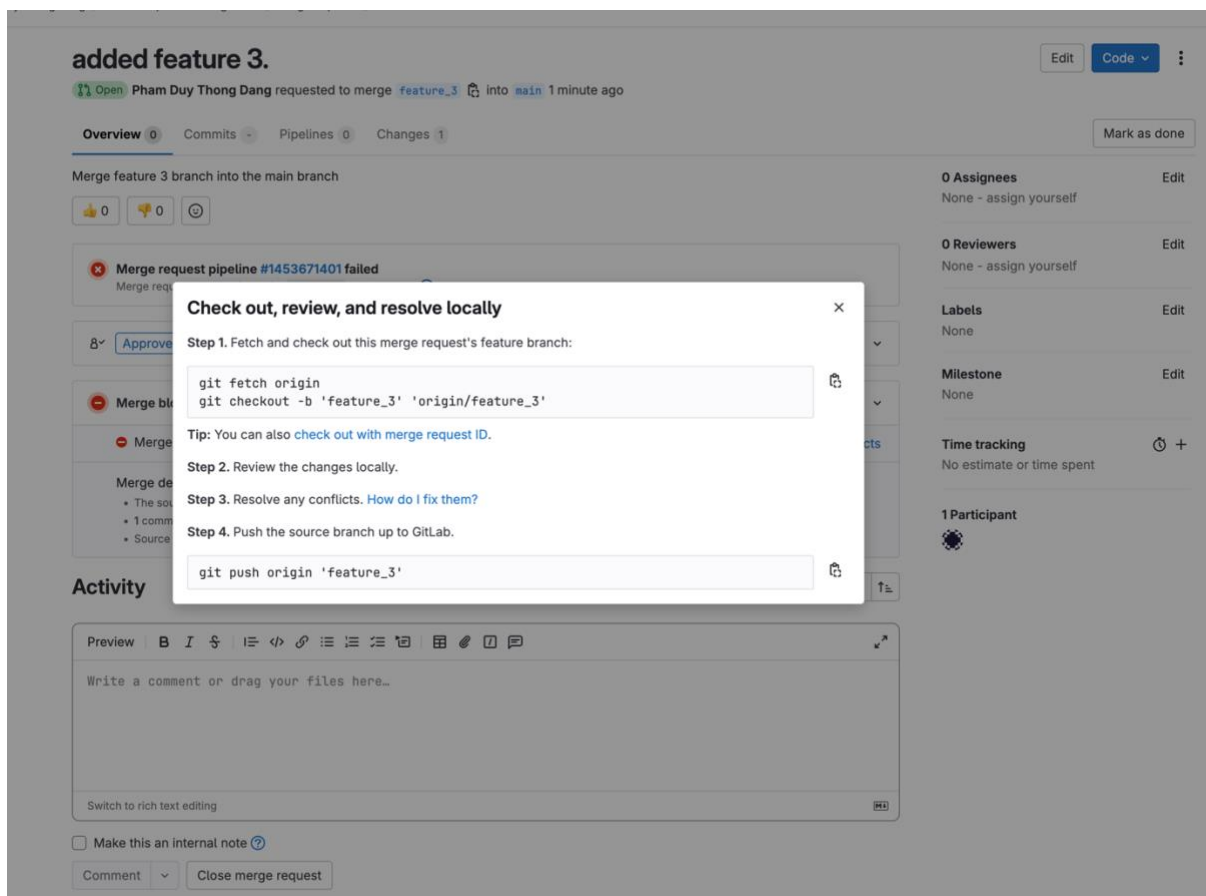


Figure 1: Instructions how to solve merge conflicts locally

Step 2: When fetching into my local, in the main branch, I ran the command **git merge feature_3**, as you could see that there were some solutions that I could choose to solve the merge conflicts as the below image. I click on **Accept Both Changes** solution to keep both the feature 4 in the **main** branch and the feature 3 in the **feature_3** branch.

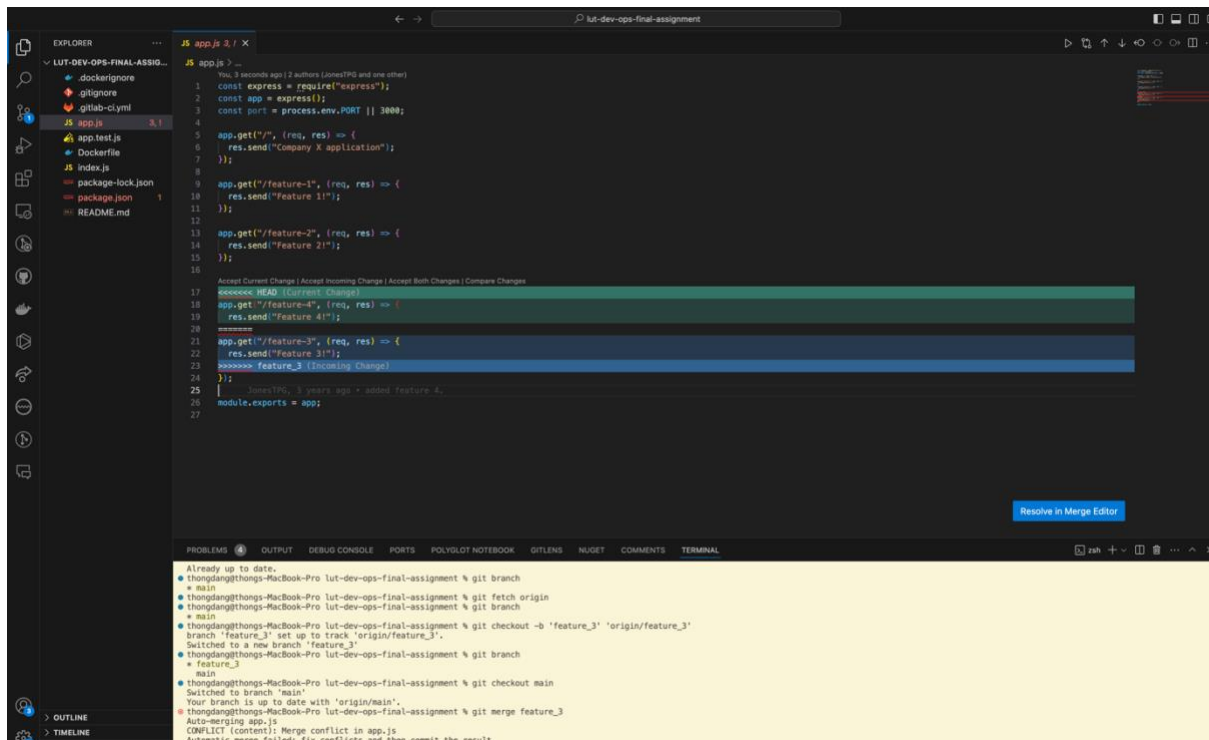
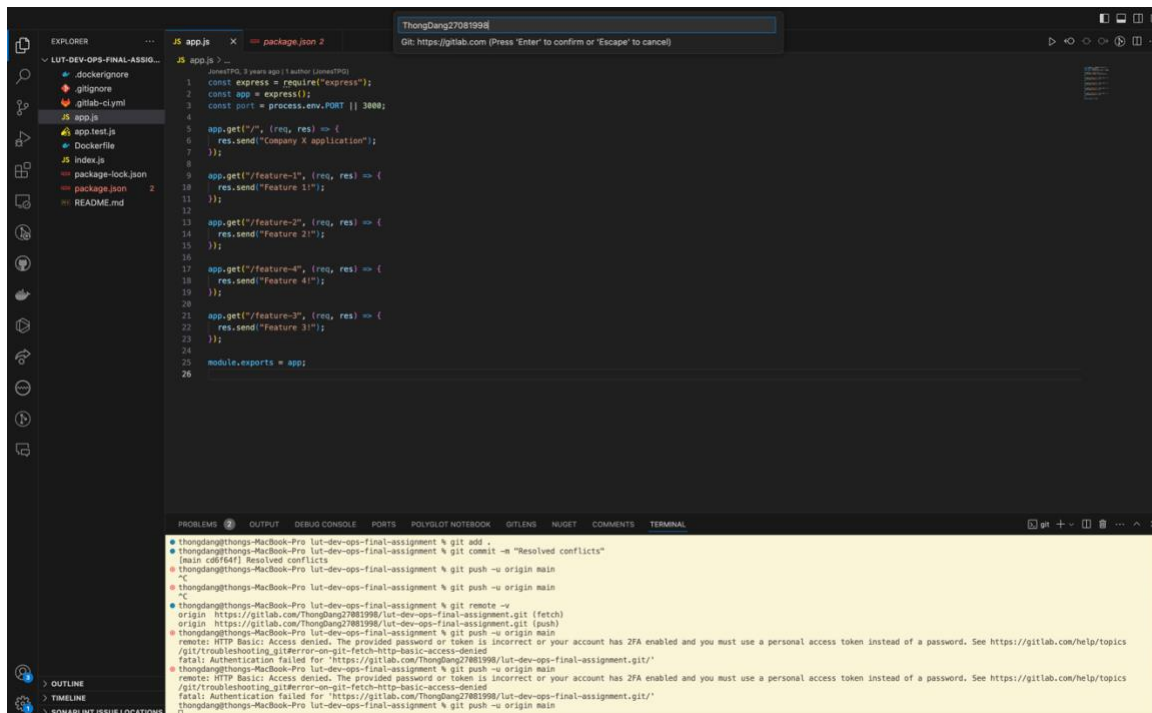


Figure 2: Solve merge conflicts locally

Step 3: After solving the merge conflicts, I staged and committed the changes I just made and pushed back to update the **main** branch. I was requested to enter my Gitlab username and password. I filled out wrong credentials sometimes and could not push as you could see. Finally, after filling out correct credential information, I could push as you could see in the below 2 images.



```
thongdang@thongs-MacBook-Pro lut-dev-ops-final-assignment % git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 998 bytes | 998.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
To https://gitlab.com/ThongDang27081998/lut-dev-ops-final-assignment.git
a26ba33..cd6f64f main -> main
branch 'main' set up to track 'origin/main'.
```

Figure 3: Stage and commit solved conflicts, and push to the remote repository

Step 4: After updating new changes regarding fixing the merge requests, you could see the detailed information that I merged the **feature_3** branch into the **main** branch and solved the merge conflicts successfully. After that, I deleted the **feature_3** branch after merging successfully in the remote repository on Gitlab. I did not forget to delete that branch in my local environment also. You could see the below images.

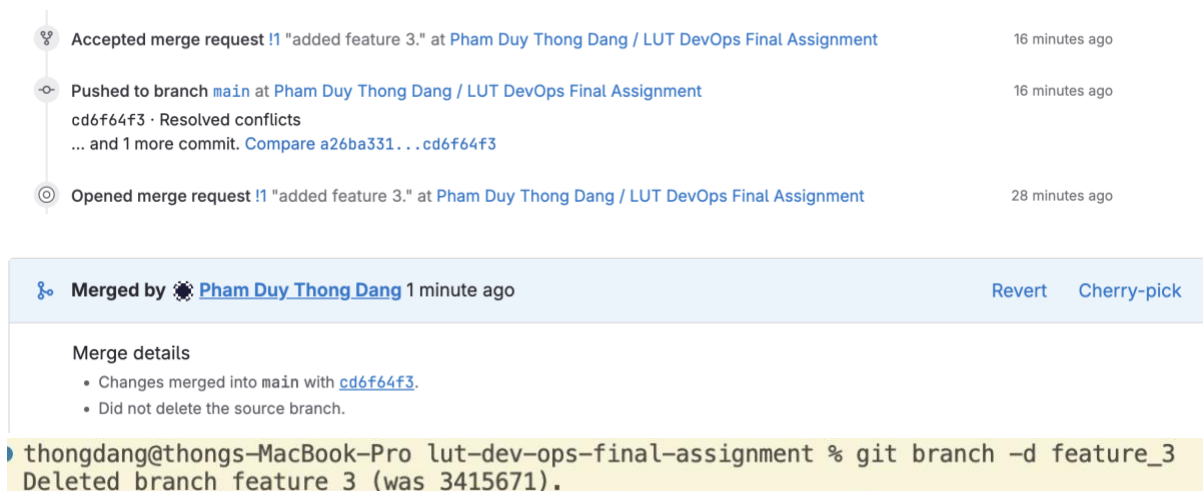
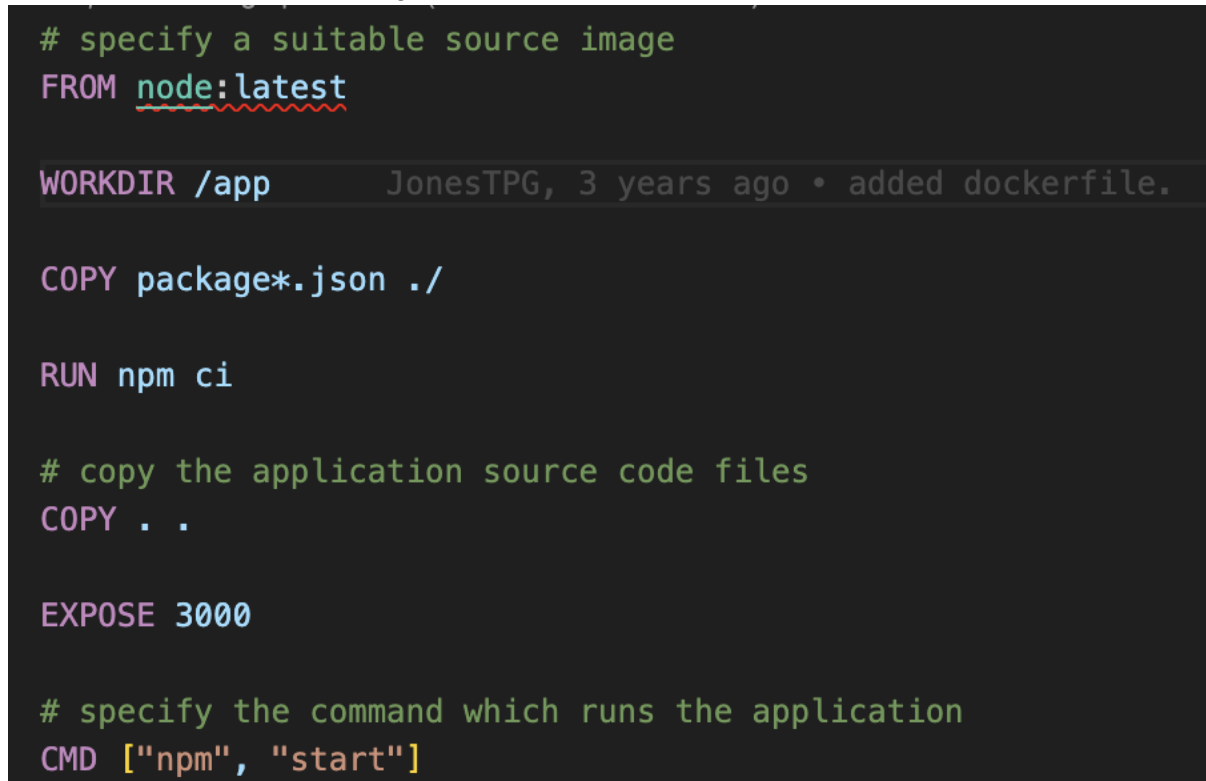


Figure 4: Notification about solving merge conflicts successfully and delete sub-branch in a local environment

b. Software containerization (how did you modify the Dockerfile?)

Based on the comment lines that I needed to fill out, firstly, I was requested to specify a suitable source image and filled out **FROM node:latest** to pull the latest node image from Docker Hub. In the second comment, I used the command **COPY . .** and this command copied all the files in the current local source main directory including files to run the application, test files, README file, etc. into the current local directory in the Docker image. Finally, I set up the command **CMD ["npm", "start"]** to allow the Docker image to run the application with the set up command **npm start** in package.json file of the application. You could see the modification as the below image.



```
# specify a suitable source image
FROM node:latest

WORKDIR /app

COPY package*.json ./

RUN npm ci

# copy the application source code files
COPY . .

EXPOSE 3000

# specify the command which runs the application
CMD ["npm", "start"]
```

Figure 5: Dockerfile modification

c. Cloud environment (how did you create and configure the cloud service?)

At first, on the Microsoft Azure platform, I created a new **Azure Web App for Containers** resource for using a container image for this final LUT assignment as the below images (the second image onwards). In this section, I needed to get the full image name and tag information of the assignment repository on Gitlab Container Registry to register on the Microsoft Azure platform. In addition, there were small configurations to be done on the Azure platform such as enabling the **Continuous Deployment** and **SCM Basic Auth Publishing Credentials** sections for WebHook configurations on Gitlab. Notably, I used the pipeline with the first two stages to publish a container image of the application to the GitLab Container Registry for configuring the **Azure Web App for Containers** resource on the Microsoft Azure platform as the below first image.

```

stages:
  - build_and_test
  - containerize
  # Is something missing here?

build_and_test:
  stage: build_and_test
  tags:
  - lutdevops2023
  # specify a suitable image where npm is available
  image: node:latest
  script:
  - npm install
  # test the application
  - npm run test
  retry:
    max: 2
    when: always

containerize:
  stage: containerize
  tags:
  - lutdevops2023
  image: docker:latest
  services:
  - docker:dind
  script:
  - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
  # build and tag the image (refer to the gitlab container registry for the correct tagname)
  - docker build -t registry.gitlab.com/thongdang27081998/lut-dev-ops-final-assignment .
  # push the image to the gitlab container registry
  - docker push registry.gitlab.com/thongdang27081998/lut-dev-ops-final-assignment
  retry:
    max: 2
    when: always

```

Figure 6: Pipeline with build_and_test and containerize stages to publish a container image of the application to the Gitlab Container Registry

Microsoft Azure

Search resources, services, and docs (G+/I)

[Home](#) > [Create a resource](#) >

Create Web App ...

Basics

Container

Networking

Monitor + secure

Tags

Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Azure for Students

Resource Group *

(New) final-lut-devops-assignment

[Create new](#)

Instance Details

Name

final-lut-devops-assignment

-fad7hab9d2fbcyh9.northeurope-01.azurewebsites.net

Unique default hostname (preview) on. [More about this update](#)

☒ Yes ☐ No

Publish *

☐ Code ☒ Container ☐ Static Web App

Operating System *

☒ Linux ☐ Windows

Region *

North Europe

Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Linux Plan (North Europe) *

(New) ASP-finallutdevopsassignment-b5bf

[Create new](#)

Pricing plan

Free F1 (Shared infrastructure)

[Explore pricing plans](#)

Figure 7: Create an Azure Web App for Containers resource on Microsoft Azure

Create Web App ...

Basics Container Networking Monitor + secure Tags Review + create

Select your preferred source for container images. You can change these settings and other dependencies after creating the app. [Learn more](#)

Sidecar support (preview) ⓘ	<input type="radio"/> Enabled <input checked="" type="radio"/> Disabled
Image Source *	<input type="radio"/> Quickstart <input type="radio"/> Azure Container Registry <input checked="" type="radio"/> Docker Hub or other registries
Options	<input checked="" type="radio"/> Single Container <input type="radio"/> Docker Compose (Preview)
Docker hub options	
Access Type *	<input type="radio"/> Public <input checked="" type="radio"/> Private
Registry server URL *	<input type="text" value="https://registry.gitlab.com"/> ✓
Username *	<input type="text" value="ThongDang27081998"/> ✓
Password *	<input type="password" value="....."/> ✓
Image and tag *	<input type="text" value="thongdang27081998/lut-dev-ops-final-assignment"/> ✓
Startup Command ⓘ	<input type="text" value="Example: /bin/bash; -c; echo hello; sleep 10000"/>

Figure 8: Fill out the image information of the application from Gitlab Container Registry

Home >

Microsoft.Web-WebApp-Portal-1903b645-9392 | Overview

Deployment

Search x <

Delete Cancel Redeploy Download Refresh

- Overview
- Inputs
- Outputs
- Template

✓ Your deployment is complete

Deployment name: Microsoft.Web-WebApp-Portal-1903b645-9392

Subscription: [Azure for Students](#)

Resource group: [final-lut-devops-assignment](#)

Start time: 14/09/2024, 22:23:49

Correlation ID: 7eb1810a-5c09-43b0-a038-9cdcb55ca888

Deployment details

Next steps

Manage deployments for your app. Recommended

Protect your app with authentication. Recommended

Go to resource

Give feedback

Tell us about your experience with deployment

Figure 9: Create resource successfully

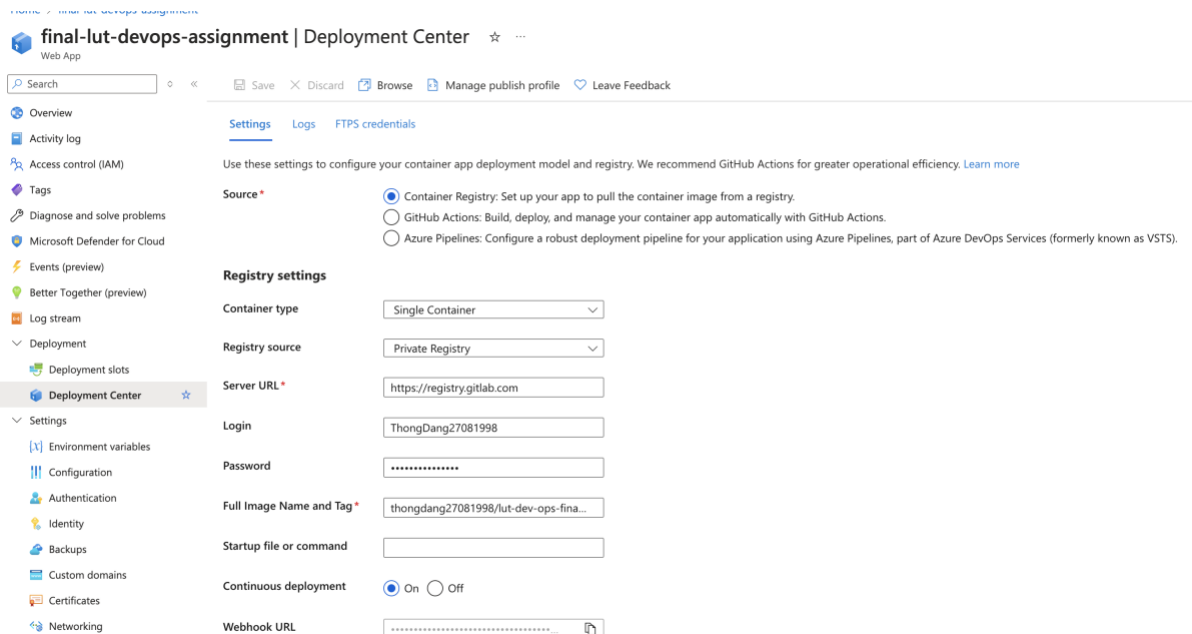


Figure 10: Detailed information about registry settings of resource on Azure

SCM Basic Auth Publishi... ☒ On ☐ Off

Figure 11: Enable SCM Basic Auth Publishing Credentials for WebHooks configurations on Gitlab

d. CI / CD pipeline (how did you modify the .gitlab-ci.yml?)

In the pipeline, most of the information was filled but there were still incomplete fields under comment tags to be done to make the pipeline comprehensive. To be precise, at first, for the comment tag “**is something missing here?**”, I needed to fill out the **deploy** stage tag to notice the pipeline to run the **deploy** stage. Secondly, the following comment tag “**specify a suitable image where npm is available**” in the **build_and_test** stage needed to be filled with **image: node:latest** to allow the stage to pull the latest suitable node version image to build and test the application completely inside the pipeline. Thirdly, for the following comment tag “**test the application**”, we needed to fill out the command **npm run test** as I configured in the **package.json** file to test the application. Fourthly, for the following comment tag “**build and tag the image**”, I used the command **docker build -t 'TAG_NAME'** to build the image for the application with the tag name from the given information on Gitlab Container Registry. Finally, for the eventual comment tag “**push the image to the gitlab container registry**”, I used the command **docker push 'REGISTRY_ADDRESS'** with the address information from the given information on Gitlab Container Registry. In addition, in the **build_and_test** and **containerize** stages, I also configured retry actions whenever the pipeline came to these stages and failed, it would retry in these stages with maximum of 2 times with any error issues. Notably, in the **deploy** stage, I needed to fill out the information about production environment I set up on Gitlab Environment section. Whenever, the pipeline came to this stage, I would come to this environment to trigger the WebHook I set up on Gitlab to allow an automation process to fetch an updated image from Gitlab Container Registry on Azure Platform for a continuous deployment process to Azure. To be precise, I needed to provide the name of the production environment that I set up on Gitlab and the given deployment URL link I was given after setting up **Azure Web App for Containers** resource successfully. In conclusion, with the

modifications, I could set up the pipeline to automate the entire process to build, test, containerize, and deploy the application to Azure platform. The final URL link for the application was shown below.

```

stages:
  - build_and_test
  - containerize
  # is something missing here?
  - deploy

build_and_test:
  stage: build_and_test
  tags:
    - lutdevops2023
  # specify a suitable image where npm is available
  image: node:latest
  script:
    - npm install
    # test the application
    - npm run test
    - echo "Build and test successfully"
  retry:
    max: 2
    when: always

containerize:
  stage: containerize
  tags:
    - lutdevops2023
  image: docker:latest
  services:
    - docker:dind
  script:
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
    # build and tag the image (refer to the gitlab container registry for the correct tagname)
    - docker build -t registry.gitlab.com/thongdang27081998/lut-dev-ops-final-assignment .
    # push the image to the gitlab container registry
    - docker push registry.gitlab.com/thongdang27081998/lut-dev-ops-final-assignment
    - echo "Containerize successfully"
  retry:
    max: 2
    when: always

deploy:
  stage: deploy
  tags:
    - lutdevops2023
  script:
    - echo "Deploy to Azure App Service"
  environment:
    name: production environment
    url: https://final-lut-devops-assignment-fad7hab9d2fbcyh9.northeurope-01.azurewebsites.net/

```

Figure 12: Completed modification for pipeline to automate CI process

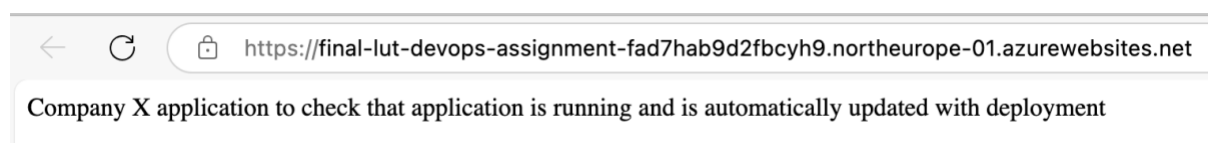


Figure 13: The final URL for the application

2. Provide a process model illustrating the created CI / CD pipeline. (You may use [UML](#), eg. An [activity diagram](#). You can use [this](#) tool to create the model).

