CSD CA1 writeup

Craig Dillon
Technological University Dublin
x00205790@mytudublin.ie
MSc. DevOps
Continuous Software Delivery
CA1 Writeup

Contents

1 Introduction **3**
1.1 Assignment overview 3
1.2 Tooling 3

2 Method **3**
2.1 CI 3
2.2 CD 4

3 Branching/Deployment strategy **4**
3.1 Branching 4
3.2 Deployment 5
3.3 Other tasks 5

4 Improvements and shortcomings **5**
4.1 Improvements 5
4.2 Shortcomings 5

Abstract

This document will describe the reasoning, workflows, successes and shortcomings in undertaking the assignment set out for the CSD module. The repository can be found here: Craig Dillon CA1 submission.

Introduction

Assignment overview

The assignment is to successfully deploy an online web app that calculates blood pressure values and returns either an error message for invalid values or messages stating if the measurements are low, ideal, pre-high and high blood pressure. As part of the CI there is a requirement for unit testing, BDD testing, code analysis and vulnerabilty and dependency checks. For CD there needs to be a functioning release and deployment strategy, end-to-end testing, performance testing, security testing and telemetry if appropriate.

Tooling

The following tools were used in this project,

- Github
- Github Actions
- Dotnet
- Terraform
- Snyk
- Docker
- Azure Container Registry/Instances
- Puppeteer
- k6
- Renovatebot

Method

CI

A dotnet application was supplied for deployment with the purpose of being containerised. The CI section of the workflow is as follows;

- Build application and run unit tests
- Create and publish code coverage as a Github badge in the README.md fields
- Run security tests with Snyk which are published to the Snyk dashboard
- Dependency checks are handled by Renovatebot several times a day and creates merge requests to the staging branch when required

CD

For CD the application is built, containerised and deployed as follows;

- Dotnet build and publish
- Docker container built and pushed to the Azure Container Registry
- Terraform deploys a Container Group with the application container and a Caddy container for SSL termination
- Puppeteer tests user input with expected responses
- k6 tests load of 50 concurrent users
- Zap checks for vulnerabilities on the live webapp and opens issues in the Github repository when required

Branching/Deployment strategy

Branching

The branching strategy relies on running workflows in either the staging or production environments. There are 3 protected branches;

- staging
- main
- production

Where the main branch does not run workflows but is considered the true source of information. These branches can only be merged to and refuse direct pushes with the desired workflow being, merge to staging, then main and finally production. All protected branches require review. The workflow itself is an automated process that is triggered manually to ensure code reviews.

Deployment

The container built in the workflow is uploaded to the Azure Container Registry and tagged with the Github SHA variable which is then passed on to the Terraform job as an env var for later use. Terraform then performs several key tasks;

- Create storage account and share for Caddy certificates
- Create the container group (staging or production)
- Deploy both the application and Caddy containers

There are two workflows, staging and production and they are locked to the relevant environments in Github which prevents the workflows being run on the incorrect branch. Finally, once this is complete several other jobs are triggered to test user input and responses, performance load testing and final vulnerability scanning.

Other tasks

There are other workflow tasks that operate alongside the main workflows. One of these tasks is the initial setup of the environment, which is creating a storage account for the Terraform backend to store statefiles and the Azure Container Registry that will be later used to upload Docker containers. Another workflow is for housekeeping to clear old workflow runs that are no longer needed, removing visual clutter in the dashboard.

Improvements and shortcomings

Improvements

The main workflows exist as two separate workflow files. It would be more efficient to have this as a single workflow file, which then inherits the running environment and subsequent environment variables from the merging branch.

Shortcomings

The application has no extra functionality, error handling and unit testing code coverage is currently at 20%.