Het DevOps projectplan

Projectoverzicht

Voor deze opdracht ben ik bezig geweest met het implementeren van een DevOps-pipeline voor mijn C# ASP.NET-applicatie. De broncode en pipeline zijn toegankelijk via GitHub. https://github.com/NalanyW/library-app

Gebouwde componenten:

Volledige CI/CD-pipeline

Ik heb een complete DevOps-pipeline opgezet met de volgende werkende stages:

1. Build stage
2. Test stage
3. Deploy stage

Deze stages zijn verspreid over meerdere workflows. De pipeline zorgt voor geautomatiseerde builds, tests en deployments bij elke code-wijziging. De builds en tests worden bij elke push geactiveerd, zo kun je goed zien of wat je gebouwd hebt, niet alleen lokaal, maar ook op andere machines bouwt en werkt. De deployments worden alleen uitgevoerd wanneer er gepushed wordt naar de main branch. Hierdoor voorkomen we dat er foutieve code wordt gereleased, aangezien het dan bijvoorbeeld bij de pull-request al duidelijk is geworden dat er iets mis is.

Docker Container

Ik heb Docker Desktop gebruikt voor lokale ontwikkeling en heb een Docker-image gecreëerd voor mijn applicatie. De docker image wordt gemaakt aan de hand van een dockerfile die ik multiplatform heb ingericht, zodat deze dan makkelijk gebouwd en aangezet kan worden op systemen met verschillende architectuur. De release stage van mijn pipeline upload de gemaakte docker image naar een docker hub registry. De build wordt dus constant gereleased zodat onze nieuw gemaakte build automatisch en snel bij onze eindgebruikers in handen komt.

Uitgebreide tests

Ik heb de volgende testen geïmplementeerd:

- 12 werkende unit tests voor code validatie
- 3 werkende infrastructuur tests met GOSS-testing (dgoss) voor validatie van onze Docker-containers

- Deze tests worden ook geserveerd als healthcheck endpoint

Deze tests worden automatisch uitgevoerd in de Test stage van mijn pipeline

Code kwaliteitscontrole

Ik heb een Code Quality Check geïntegreerd in mijn pipeline, met behulp van codeql-analysis. Dit helpt bij het handhaven van code standaarden en het vroegtijdig identificeren van potentiële problemen.

Documentatie en zichtbaarheid

Ik heb Pipeline badges toegevoegd aan de README van mijn GitHub-project. Deze badges geven real-time inzicht in de status van mijn builds, tests en deployments. Helaas is het me niet gelukt om een Monitoring dashboard werkend te krijgen.

Voordelen van mijn aanpak:

1.Door de geautomatiseerde CI/CD-pipeline kunnen we snel en betrouwbaar nieuwe features uitrollen.
2. De uitgebreide testen en code quality checks helpen bugs vroeg in het ontwikkelproces te identificeren.
3.Door het gebruik van Docker garanderen we consistentie tussen ontwikkel-, test- en productieomgevingen.
4. De toegankelijke broncode, pipeline en status badges zorgen voor volledige transparantie in het ontwikkelproces.

Conclusie

Voor deze opdracht heb ik een DevOps-omgeving opgezet voor mijn C# ASP.NET-applicatie. Deze setup stelt mij in staat om efficiënter te ontwikkelen, testen en deployen, wat uiteindelijk leidt tot een hogere kwaliteit van mijn software.