Individuele opdracht 2e zittijd

DevOps project: Operations

PrB toegepaste informatica, HOGENT

Academiejaar 2021-2022

Samenvatting

In dit document vinden studenten die voor dit opleidingsonderdeel niet geslaagd waren in de eerste zittijd alle informatie over deelnemen aan de tweede zittijd, meer bepaald alle praktische informatie en de (individuele) opdracht.

Praktische informatie

Deelname aan de tweede zittijd

De evaluatie van dit opleidingsonderdeel is gebaseerd op twee onderdelen:

- 70 pct: opleveren product
- 30 pct: observatie van functioneren van de student

In de tweede zittijd kan je enkel het eerste onderdeel hernemen in de vorm van een **individuele opdracht**.

Voor het onderdeel “observatie van functioneren van de student” wordt de beoordeling uit de eerste zit overgenomen. Er is immers geen mogelijkheid om dit onderdeel tijdens het zomerreces te evalueren.

Als je voor het onderdeel “observatie” de vermelding “Afwezig” kreeg, zal je nu dus ook een afwezigheid krijgen, wat er in de praktijk op neer komt dat je niet kan deelnemen aan de tweede zittijd (OER, art. 36; DOER, art.6).

Individuele opdracht

Het gaat hier om een **individuele** opdracht, dus samenwerken is niet toegelaten. Je mag gebruik maken van de reeds bestaande broncode van je projectgroep, maar bouwt hier individueel op verder. Enkel jouw eigen bijdragen die je in de loop van de tweede zittijd gedaan hebt, worden meegerekend in de beoordeling.

Deliverables

De beoordeling van het resultaat van deze opdracht gebeurt op basis van volgende deliverables:

- Een individuele **Github-repository** (die je aanmaakt met Github Classroom via deze link: https://classroom.github.com/a/qKeiQFVM) met daarin:
  - Alle **broncode** voor het opzetten van de test- en productie-omgeving (zie verder)

  - Alle **procedurehandleidingen** die de lezer moeten in staat stellen om de omgevingen volledig te reproduceren, zonder verdere uitleg
- De omgevingen zelf, waarvan de werking a.h.v. een **demonstratie** aangetoond wordt

Deadline

De **deadline** voor de individuele opdracht is **maandag 22 agustus 2022 om 12:00** ('s middags, dus niet middernacht!).

De toestand van de Github-repository op dat moment geldt als je inzending. Aanpassingen na die datum zullen niet mogelijk zijn.

Als je geen Github-repo aangemaakt hebt, of als blijkt dat de inzending in het geheel niet voldoet aan de verwachtingen, dan volgt er geen demonstratie en krijg je de vermelding "Afwezig". Je wordt hiervan op de hoogte gebracht.

Studenten van wie de inzending ontvankelijk is om beoordeeld te worden, zullen een uitnodiging krijgen voor de verdediging van hun werk, tijdens het moment dat op het examenrooster ingepland is.

Opdracht

De opdracht is grotendeels gelijklopend met deze uit de eerste zittijd, met name het opzetten van een build-pipeline voor een .Net-applicatie.

In de hieronder opgesomde requirements duiden we telkens het belang aan volgends de MoSCOW methode:

- **(M)** Must have: al deze requirements moeten zonder uitzondering geïmplementeerd zijn om te kunnen slagen (voor het onderdeel "product", 70% van het totaal). Je toont dit aan tijdens de demonstratie
- **(S)** Should have: het implementeren van deze requirements zorgt voor een hogere score
- **(C)** Could have: idem
- **(W)** Won't have: deze requirement is expliciet *niet* gevraagd

Hou rekening met deze prioriteiten bij het uitwerken van de opdracht. Het heeft bv. geen zin om "Could Have's" te implementeren als de basisfunctionaliteit niet gerealiseerd wordt.

Als een bepaalde (niet-functionele) requirement niet gespecifieerd is (bv. te gebruiken Linux-distributie of cloud-platform), dan kan je zelf een keuze maken (bv. verder werken op resultaten uit de 1e zittijd).

Casus

Gebruik als casus deze voorbeeld-applicatie uit de cursus Enterprise C#: https://github.com/HOGENT-Web/csharp-ch-9-exercise-1/, meer bepaald de `solution` branch. De `main` branch bevat enkel de opgave en is onvolledig.

Maak een fork van deze applicatie onder je eigen Github-account zodat je kan simuleren wat er gebeurt als je een commit uitvoert en het build-proces in gang zet.

De webapplicatie kan gebouwd en opgestart worden door in directory `src/server` respectievelijk de commando's `dotnet build` en `dotnet run` uit te voeren.

Als het nuttig of nodig is om bestanden toe te voegen aan deze repository om deployment te vergemakkelijken (bv. Dockerfile, docker-compose.yml, Appsettings.*.json configuratie voor lokale/cloud-omgeving, enz.) dan is dat zeker toegelaten.

Build server

Gebruik Vagant om een lokale VirtualBox-VM op te zetten als build server:

- **(M)** Gebruik Jenkins als CI/CD-tool. Verder zijn alle nodige tools en libraries geïnstalleerd zijn om de applicatie te bouwen (build) en uit te rollen (deploy).
  - **(M)** De installatie is volledig geautomatiseerd (met Bash of Ansible). `vagrant up` geeft een werkende Jenkins-server
  - **(C)** Configuratie is bij voorkeur geautomatiseerd, of **(M)** beschreven ahv een gedetailleerde procedurehandleiding
- Doelstelling is om twee build pipelines op te zetten:
  1. **(M)** Build en deploy naar lokale appserver
  2. **(S)** Build en deploy naar appserver op cloudplatform
- De build pipeline(s) bevat(ten) volgende fasen:
  1. **(C)** Statische analyse/linting
  2. **(M)** Build
  3. **(W)** In deze opstelling is **geen** teststap voorzien
  4. **(M)** Deployment naar applicatieserver

Na lanceren van de build pipeline is de laatste revisie van de applicatie beschikbaar voor gebruikers op de applicatieserver.

Applicatieserver

Voorzie twee instanties van de applicatieserver:

- **(M)** Een lokale VirtualBox-VM, opgezet met Vagrant
  - De installatie is volledig geautomatiseerd (met een Bash-script of Ansible): na `vagrant up` is de server zonder verdere manuele handelingen klaar om de applicatie te hosten.
  - De applicatie is gebaseerd op .NET 5.0. Met de runtime heb je voldoende om de applicatie te lanceren (je hoeft dus niet de SDK te installeren).
- **(S)** Een applicatieserver gehost op een cloud-platform (naar keuze)
  - Het opzetten van de VM is beschreven in een gedetailleerde procedurehandleiding
  - Na initialiseren van de VM is de verdere installatie en configuratie geautomatiseerd met hetzelfde script of playbook als de lokale instantie.

Ter info, een video over het hosten van een ASP.NET Core applicatie op Azure: https://www.youtube.com/watch?v=UpWeffxf790.

Verder hou je rekening met volgende requirements:

- **(M)** De applicatie draait binnen een container
- **(S)** De database draait in een aparte container
- **(M)** De volledige installatie en configuratie van de applicatieserver is geautomatiseerd. Na uitvoeren van deze stap is de server dus klaar om de applicatie te hosten.
- **(M)** Via port-forwarding wordt er voor gezorgd dat de webapplicatie zichtbaar is door in de webbrowser `https://IP_ADRES/` of `https://HOSTNAAM/` in te tikken (dus zonder een poortnummer toe te moeten voegen).

- **(C)** De toestand van de applicatieserver is op te volgen via een monitoring-systeem, meer bepaald:
  - Host-systeem: CPU, Geheugen, Disk I/O, Disk usage
  - Overzicht containers: CPU, geheugen, ...
  - Webserver-statistieken: requests, errors, ...
  - Database-server statistieken: queries, errors, ...

Documentatie

Zoals je in de beschrijving van de opdracht al merkte, wordt er ook belang gehecht aan goede documentatie. Voor deze opdracht verwachten we specifiek:

- **Procedurehandleidingen:** deze laten iemand toe om de gehele gerealiseerde opstelling (build- en applicatieserver(s)) volledig te reproduceren en te gebruiken.
- Aanvullende **technische handleidingen**: andere documentatie, bv. resultaat van onderzoeken welk cloudplatform het meest geschikt is voor deze casus, samenvattingen van opgezochte informatie, cheat sheet van vaak gebruikte commando's, enz.

Beoordeling

De toegekende score hangt af van in hoeverre je de requirements gerealiseerd hebt:

- **Totaal onvoldoende (0/70)**:
  - Er is een Github-repository aangemaakt, maar daar zijn geen commits op gebeurd, of de aanwezige code is totaal onvoldoende om de opdracht te realiseren
- **Onvoldoende (1-34/70)**:
  - De Must-have **(M)** requirements zijn slechts gedeeltelijk gerealiseerd.
  - De student heeft de correcte werking van de build pipelines of applicatieservers niet kunnen aantonen tijdens de demonstratie
  - De documentatie is onvolledig
- **Voldoende (35-49/70)**:
  - Alle must-haves **(M)** zijn gerealiseerd
  - Should-haves **(S)** of could-haves **(C)** zijn niet of gedeeltelijk gerealiseerd
  - De student heeft de correcte werking van de build pipeline(s) of applicatieserver(s) aangetoond tijdens de demonstratie en aan de hand van de aanwezige code. Voorbeeldscenario: (1)De applicatie draait en is zichtbaar in een webbrowser; (2) Er wordt een wijziging aangebracht in de broncode van de applicatie (bv. achtergrondkleur veranderen); (3) De build pipeline wordt opgestart; (4) Na afloop zien we in de webbrowser dat de wijzigingen doorgevoerd zijn.
  - De documentatie is volledig en overzichtelijk
- **Goed (50-59/70)**:
  - Alle must-haves **(M)** en should-haves **(S)** zijn gerealiseerd
  - Could-haves **(C)** zijn niet of gedeeltelijk gerealiseerd
  - De student heeft de correcte werking van de build pipeline(s) of applicatieserver(s) aangetoond tijdens de demonstratie en aan de hand van de aanwezige code
  - Tijdens de demonstratie toont de student aan inzicht te hebben in de materie
  - De documentatie is volledig en overzichtelijk
- **Zeer goed (>= 60/70)**:
  - Ook alle could-have **(C)** requirements zijn geïmplementeerd

- De student heeft de correcte werking van de build pipelines of applicatieservers aangetoond tijdens de demonstratie en aan de hand van de aanwezige code
- Tijdens de demonstratie toont de student aan een diep inzicht te hebben in de materie
- De documentatie is uitmuntend in volledigheid, overzichtelijkheid en detail