Stage RKITEK 2022

Realisatie

Bachelor Electronica / ICT
Cloud and Cybersecurity

Maxim Zeelmaekers

Academiejaar 2021 - 2022

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

LID VAN ASSOCIATIE KU LEUVEN

THOMAS MORE

INHOUDSTAFEL

1 VOORWOORD

In dit document vind u de project onderdelen dat ik gerealiseerd heb binnen mijn stage bij RKITEK. Mijn stageproject was een DevOps omgeving opzetten voor de Provincie Antwerpen. Op het moment gebruikt men veel software/Infrastructuur dat lokaal gehost & beheerd kan worden en hierin is alles manueel ingesteld. Dit zorgt voor bepaalde problemen, er word veel tijd verloren door het manueel uitrollen van zaken maar is er geen centrale plek om alles te beheren.

Mijn stage opdracht was helpen bij de overgang van de integratie naar de cloud.
De provincie Antwerpen koos Azure als cloud oplossing en hier zou men microservices gaan hosten in een kubernetes cluster. Men gebruikte ook nog subversion, een oud versie beheersysteem, hiervoor zou ik tijdens mijn stage ook een oplossing voor moeten vinden.

2 FASES

Ik heb mijn stage opgedeeld in drie fases, dit maakt het makkelijk om de vooruitgang te zien binnen mijn project. Maar ook de verschillende onderdelen op te volgen en welke technologieën ik gebruik.

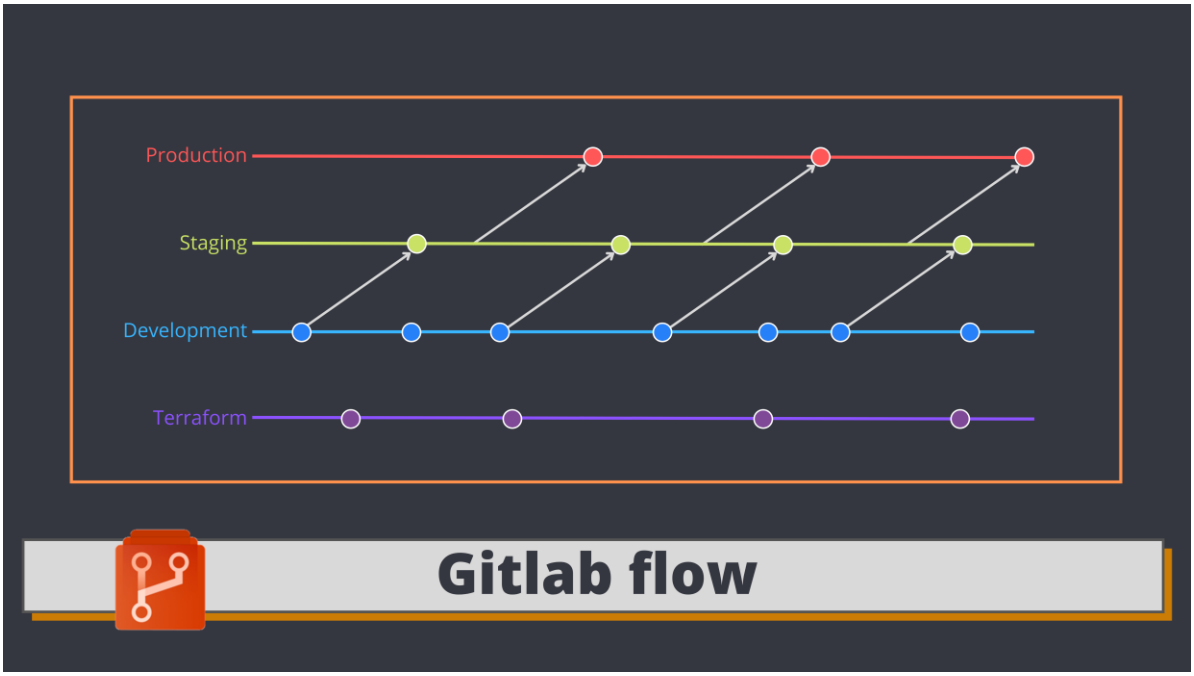
2.1 OnderzoekFase

Mijn eerste weken heb ik onderzoek gedaan naar wat DevOps en CI/CD inhoud, gezien ik Cloud and Cyber Security studeer heb ik weinig ervaring met het automatiseren van de development-cycle.

2.1.1 GIT

Als eerste zocht ik een locatie voor alles op te kunnen slagen, gezien men subversion gebruikte dacht ik dat een overstap naar een GIT Service een goed idee is, GIT bied veel betere versie beheer functies aan en men kan zowel offline als online werken.

Als GIT structuur zou ik de Gitlab flow implementeren gezien deze gebruiksvriendelijk is en voor een duidelijke structuur zorgt dat makkelijk overdraagbaar is van subversion.



Een belangerijk punt was de integratie mogelijkheid van Azure DevOps Pipelines as dit gebruikt ging worden om de code van de GIT service naar Azure uit te rollen. De keuze was moeilijk maar al snel had ik drie mogelijke services die gebruikt konden worden:

2.1.1.1 BitBucket

Bitbucket is een GIT service gemaakt door Atlassian. Atlassian heeft een heel CI/CD verhaal als ecosysteem en ook Jira integratie, een groot pluspunt gezien men Jira gebruikt als ticketing systeem.

2.1.1.2 GitHub

Github momenteel in handen van Mircosoft is de meest gebruikte GIT service momenteel, hoewel dit een zeer handig platform is en Enterprise opties heeft was deze optie duur en niet de beste oplossing voor deze doelcase.

2.1.1.3 Azure DevOps Repos

Azure Devops Repos is de GIT service ontworpen door Mircosoft, het heeft een directe integratie met Azure Devops Pipelines maar is op lange termijn niet de goede optie gezien Mircosoft wil focussen op het verder ontwikkelen van GitHub

Conclusie:
Als GIT service ben ik gegaan voor Bitbucket, het belangrijkste argument is de integratie met Jira, gezien de provincie Antwerpen het volledige ticketing systeem doet via Jira. Dit is een enorm pluspunt naar efficiëntie toe en was dit een duidelijke keuze. Tickets kunnen geautomatiseerd worden wanneer een pull request aanvaard word bijvoorbeeld een status update van in progress naar done. Deze ideën heb ik gepiteched voor het deel van het integratie team and men gaf groen licht om hiermee aan de slag te gaan.

2.1.2 Microservices & Kubernetes

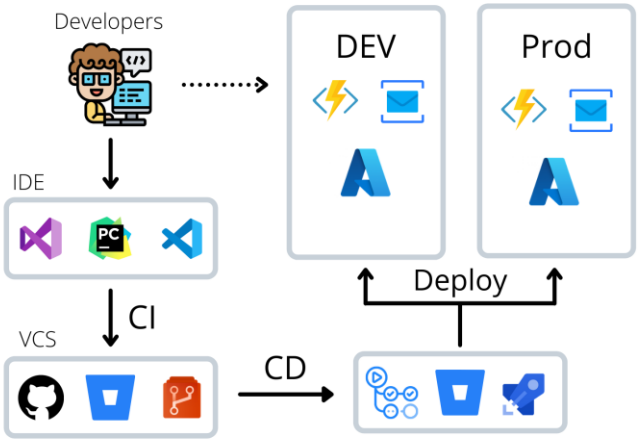
Nadat ik groenlicht kreeg voor het de gekozen GIT service ben ik me gaan focussen op de microservices, hier was ik niet goed vertrouwd mee en een opfrisser van hoe containers werkte en deze in een kubernetes cluster gezet moeten worden. Hiervoor heb ik de Desktopapplicatie van Docker gedownload en een test applicatie "helloservice" gedownload dat geconditioneerd kon worden. Deze heb ik in een lokale kubernetes kluster gezet wanneer dit vlekkeloos werkte probeerde ik deze in een eigen AKS (Azure Kubernetes Service) cluster uit te rollen.

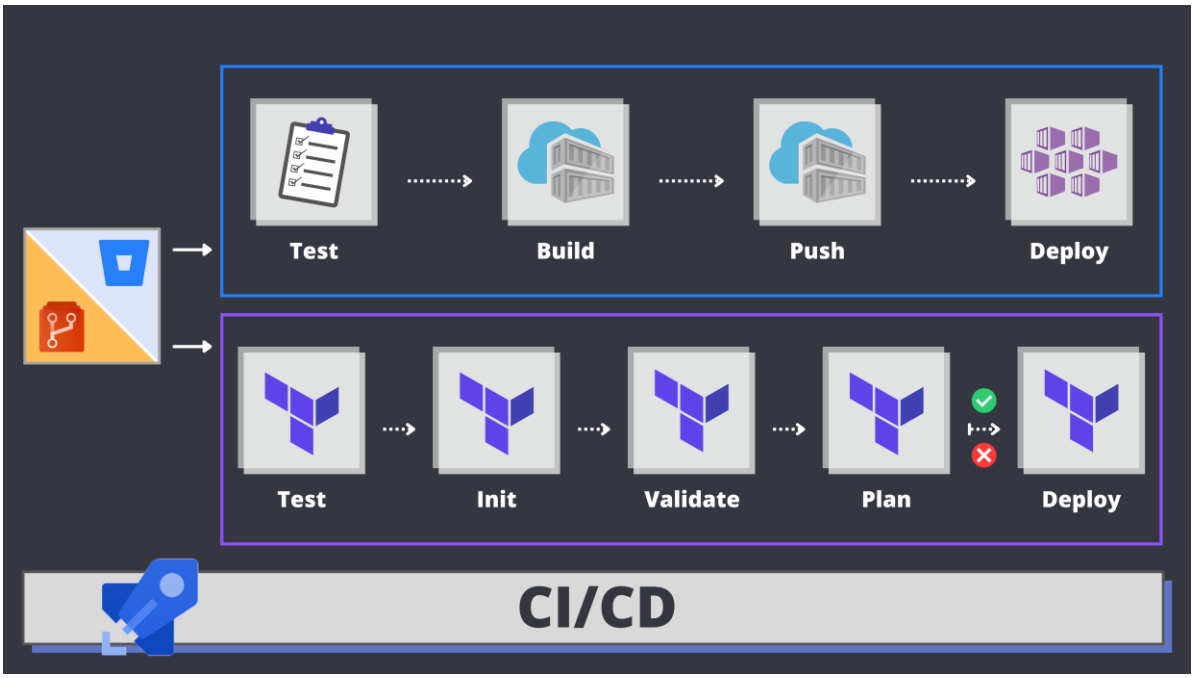
2.1.3 Azure Pipelines

Gezien Azure de rode draad is binnen mijn stage project moet ik ook vertrouwd raken met de software die de applicatie code online gaat zetten. Hiervoor zette ik een eerste een basic pipeline op. Gezien ik hier nog geen kennis van had baseerde ik mij hiervoor op een filmpje van coder dave. Met deze pipeline heb ik altijd verder gewerkt, stap voor stap een nieuwe job er aan toegevoegd. Eerst een build stap dat de applicatie gaat builden, hierna deze container in de cloud zetten en deze daarna in de kubernetes kluster zetten. Dat was het minimum dat ik wou bereiken in mijn onderzoeksfase.

2.1.4 Workflow

De workflow is belangrijk om een beeld te krijgen van wat je gaat doen en om dit uit te leggen aan andere tijdens een pitch. Hiervoor gebruikte ik canva om duidelijke visualisaties te maken voor de pitches. Developers coderen de applicatie en zetten deze in de GIT repository, hierna zal de pipeline deze in azure zetten.





2.2 ConceptFase

In de eerste stappen van de concept fase begint de automatisatie te vormen, de pipeline krijgt een trigger (webhook) deze zal een nieuwe run van de pipeline triggeren. De stappen binnen de pipeline worden ook uitgebreid.

2.2.1 Automatisatie:

Triggers worden aangestuurd door webhooks van de GIT service, in de pipeline kunnen we bepaalde waardes meegeven. Deze waardes kunnen gaan van branch specificeren tot een pipline dat na een andere pipeline moet uitgevoerd worden.

```
1  resources:
2    repositories:
3    - repository: [redacted]
4      name: [redacted]
5      type: git
6
```

De variabelen gebruikt bij de triggers is GIT service exclusief, deze moeten aangepast worden wanneer een ander GIT service gebruikt wordt, deze worden uitgebreid omschreven in volgende documentatie: [Azure Devops Pipelines](#).

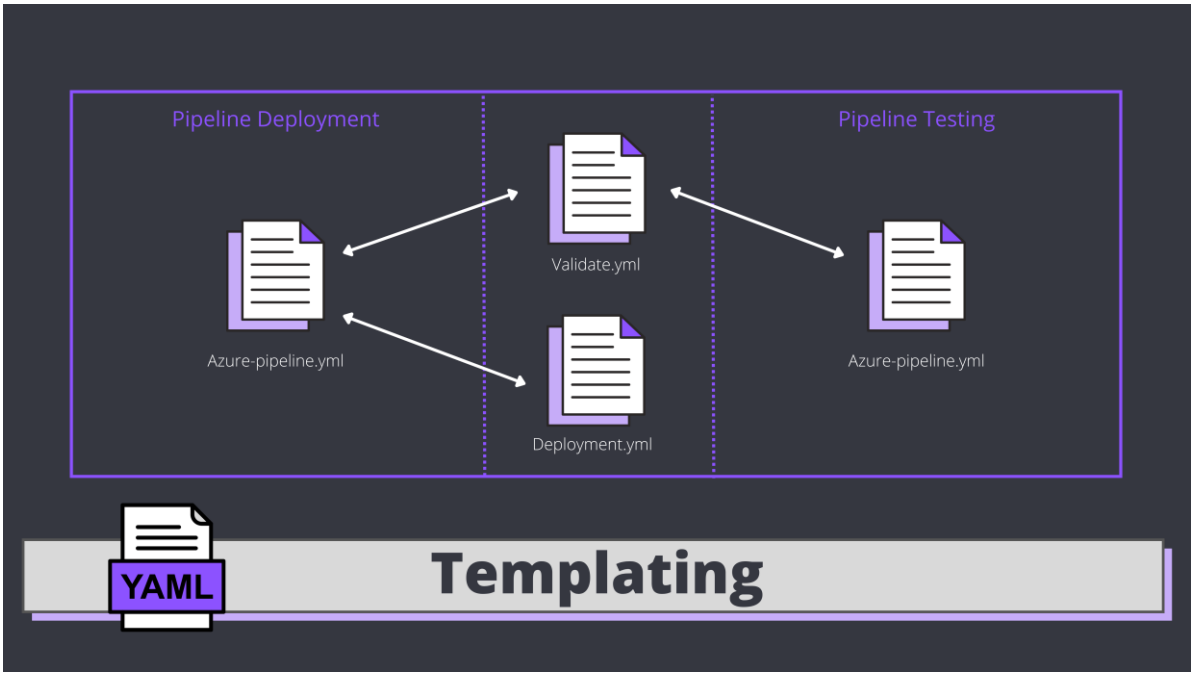
2.2.2 Infrastructure as code

Gezien bepaalde microservice applicaties ook extra eigen infrastructuur nodig hebben moet er ook terraform code uitgerold worden. Dit was zeker nieuw voor mij gezien ik nog nooit terraform heb gebruikt. Nadat ik onderzoek had gedaan waren extra plugins nodig voor deze code uit te voeren. Om alle terraform code uit te voeren moeten we bepaalde commandos uitvoeren, deze moeten ook in de pipeline uitgevoerdt worden daarvoor dienen deze extenties. Wanneer de terraform code wordt uitgevoerd zal ook een TFstate file opgeslagen worden in een Blob Container. [Terraform Azure DevOps Extentie](#)

```
- task: TerraformTaskV2@2
  displayName: Terraform - Init
  inputs:
    provider: 'azurerm'
    workingDirectory: [redacted]
    command: 'init'
    backendServiceArm: [redacted]
    backendAzureRmResourceGroupName: [redacted]
    backendAzureRmStorageAccountName: [redacted]
    backendAzureRmContainerName: [redacted]
    backendAzureRmKey: [redacted]
- task: TerraformTaskV2@2
  displayName: Terraform - Apply
  inputs:
    provider: 'azurerm'
    workingDirectory: [redacted]
    command: 'apply'
    environmentServiceNameAzureRM: [redacted]
```

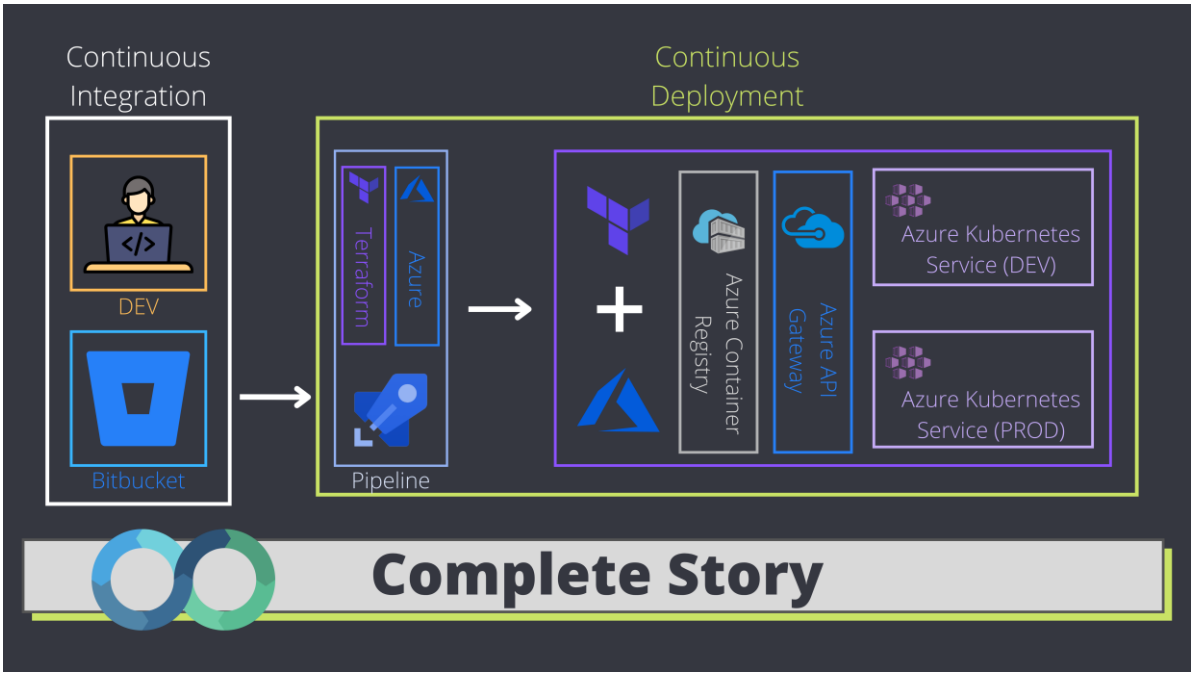
2.2.3 Azure Pipeline

In de concept fase heb ik alle YAML-files uitgewerkt met templating in gedachten.
Dit zorgt ervoor dat bestanden opnieuw aangesproken kunnen worden wanneer nodig, dit is vergelijkbaar met object oriented programming gezien men hier classes gebruikt voor hergebruik, hier kan men files in plaats van classes aanspreken om bepaalde opnieuw te gebruiken, in andere pipelines bijvoorbeeld.



2.3 RealisatieFase

In de realisatiefase ga ik van het idee en aparte puzzelstukjes in elkaar klikken om een totaal proof of concept te vormen zodat we een volledig CI/CD verhaal krijgen.
Als eerste integreer ik de GIT oplossing met de IaS (Infrastructure as code) zodat de terraform en de applicatie code in dezelfde repository staan. Hierna stel ik de automatisatie in zodat deze in de juiste volgorde triggerd.



2.3.1 Applicatie Integratie

Ik kreeg van de provincie Antwerpen een werkende applicatie om mijn project te testen, gezien ik werkende pipeline code klaar had staan was het makkelijk om deze applicatie te integreren binnen mijn DevOps project en deze was snel successvol zonder al te veel problemen. Deze applicatie werd containerized en weggeschreven in de Azure Container Registry met een tag voor versiebeheer.

2.3.2 Omgeving Integratie

Eerst was mijn proof of concept alleen gebaseerd op de development omgeving. Dit was niet handig omdat er meerdere branches waren binnen de repository voor elke branch nieuwe YAML files maken is overbodig en niet efficient.

2.3.2.1 Predefined variables

Hiervoor heb ik predefined variablen gebruikt om deze stap toe te voegen zodat waardes als repository naam en branch naam uit het project gehaald kunnen worden zonder al te veel configuratie werk. Deze waardes zullen gebruikt worden om te bepalen hoe de applicatie zal noemen in de kubernetes cluster en in welke omgeving deze uitgerold zullen worden

```
reponaam: '$(Build.Repository.Name)'
branchnaam: '$(Build.Repository.Name)'
```

2.3.2.2 Azure Key Vault

Om alle autenticatie waardes op een veilige manier op te bergen maken we gebruik van Azure Key vault, dit is een best practice. Gevoelige informatie als authenticatie naar Azure afschermen is belangrijk, hier kan men met een variable de juiste waardes meegeven zonder de programeur de waarde kan zien.

3 EINDRESULTAAT

Ik ben trots op wat ik gerealiseerd heb als stageproject.
Gezien deze stage zeer zelfstandig was en ik geen kennis van DevOps had voor ik hieraan begon heb ik toch enorm veel bijgeleerd. Zowel in zelfstandigheid als in het technische vak. Als resultaat heb ik een volledige CI/CD omgeving opgezet van de CI-deel met Bit Bucket naar het cd-deel met Azure DevOps Pipelines, het was een heel avontuur.

3.1 Workflow

Wanneer de programmeur een deel code pushed naar de Bit Bucket repository zal de pipeline in actie schieten. Als eerste zal men kijken of er Terraform code is veranderd, wanneer dit het geval is wordt een validatie en plan commando op uitgevoerd, hierbij wordt er data terug naar de gebruiker gestuurd in de terminal en wacht op een confirmatie voor deze verder gaat met de pipeline. Als deze validatie is goedgekeurd zal een apply commando uitgevoerd worden en de TFstate file in de Azure Blob Storage geüpdatet worden.

Hierna zal de applicatie gebouwd worden en in de Azure container Registry gezet worden als opslag. Wanneer de built fase succesvol afgerond is zal de pipeline beginnen met het uitrollen van de service naar Azure kubernetes service cluster. Deze zal de eerdere opgeslagen container uit de Azure Container Registry halen en deze container in de kubernetes kluster in de namespace dat is meegegeven uitrollen.