Professionele Bachelor Toegepaste Informatica

A.C.A IT-Solutions

Automatiseren van de creatie van een AWS-account

Ugur Akkar

Promotoren:

Peter Jans — ACA IT-Solutions

Gert Van Waeyenberg — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

A.C.A IT-Solutions

Automatiseren van de creatie van een AWS-account

Ugur Akkar

Promotoren:

Peter Jans — ACA IT-Solutions

Gert Van Waeyenberg — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Dankwoord

Eerst en vooral wil ik Peter Jans van het OpsKlaar team van ACA-IT Solutions bedanken voor zijn uitstekende begeleiding, feedback en alle kennis die hij heeft doorgegeven. Zonder Peter zijn begeleiding had ik deze stageopdracht niet volledig kunnen afronden. Ik wil ook al mijn collega's van het OpsKlaar team bedanken voor de leuke en leerrijke momenten.

Ten tweede wil ik de hogeschoolpromotor, Gert Van Waeyenberg, bedanken voor zijn feedback en begeleiding tijdens het schrijven van mijn eindwerk en mijn onderzoek.

Als laatst wil ik mijn ouders bedanken voor hun eindeloze steun tijdens mijn opleiding en stage. Ook wil ik mijn vriendin bedanken voor haar steun en motivatie.

Abstract

De stageopdracht betreft het schrijven van een workflow dat door een Jira issue automatisch een AWS-account creëert in de Organizations Service van Amazon Web Services. De Organizations service van Amazon Web Services geeft de mogelijkheid om accounts te creëren en beheren en deze te groeperen tussen verschillende categorieën. Het proces dat doorlopen moet worden om een account te creëren duurt erg lang en is niet efficiënt. Er kunnen ook verschillende configuratiefouten gemaakt worden. Vervolgens wordt Jira software pakket op het aangemaakte AWS-account automatisch geïnstalleerd en geüpdatet. Het upgrade proces verloopt in een zero downtime upgrade zodat het softwarepakket nooit offline gaat.

In deze bachelorpaper vinden twee onderzoeken plaats: eerst wordt het verschil onderzocht tussen de verschillende AWS SDK's dat Amazon Web Services aanbiedt. Vervolgens worden de implementatiemogelijkheden onderzocht om integratie te creëren tussen Jira en Jenkins.
De nood kwam vanuit de stageopdracht, hier moesten API-calls gebruikt worden om bepaalde interacties te maken tussen het script en AWS. Het onderzoek focust zich op de meest bekende en meest gebruikte programmeertalen om AWS SDK's te gebruiken.

Om AWS SDK's gelijk te evalueren zijn er criteria opgesteld. Alle implementatiemogelijkheden en de verschillende SDK's zijn in een testomgeving ingezet die een productieomgeving simuleert. Deze testomgeving bevat de laatste versie van Jenkins en Jira software. De testomgeving draait op een Docker-container die door Kubernetes beheerd en door Helm charts geconfigureerd wordt.

Inhoudsopgave

Lijst van gebruikte figuren

Lijst van gebruikte tabellen

Lijst van gebruikte afkortingen

API	*Application programming interface*
AWS	Amazon Web Services
ACA-IT Solution	Het bedrijf dat de stagiaire zijn stage loopt
SDK	*Software development kit*
POC	*Proof of concept*
OS	Operating System
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
RDS	Relational Database Service
EFS	Elastic File System
EKS	Elastic Container Service for Kubernetes

Inleiding

ACA-IT Solutions is een IT-bedrijf dat eerlijke en hoogkwalitatieve consultancy, implementatie en integratie van softwareoplossingen op maat aanbiedt voor bedrijven.
Deze stage kwam tot stand omdat ACA-IT Solutions nood had aan een script dat automatisch accounts aanmaakt bij elke nieuwe klant.

Doordat ACA-IT Solutions een sterk groeiend bedrijf is binnenin de IT-wereld, komen er elk jaar nieuwe klanten bij. Voor elk van deze klanten wordt er een nieuw account aangemaakt binnenin de Organization service van de Amazon Web Services.
Op dit moment worden deze accounts handmatig aangemaakt en daardoor duurt dit proces erg lang en is het niet efficiënt. Ook kunnen er verschillende configuratiefouten gemaakt worden tijdens het creëren van een account.

Om dit proces op een snellere en efficiëntere manier te behandelen wordt er een script geschreven dat door een Jira issue automatisch een Jenkins *build* aanroept en het script uitvoert.

Amazon Web Services biedt verschillende SDK (*Software development kit*) -mogelijkheden aan die API (*Application programming interface*) -calls behandelen. Dit onderzoek focust zich op het vergelijken van SDKs en de implementatiemogelijkheden. Het heeft de meest bekende, meest gebruikte en modernste SDKs gekozen om zo een vergelijking te maken. De SDKs zijn beoordeeld door middel van criteria en worden in een testomgeving ingezet dewelke een productieomgeving simuleert.

I. Stageverslag

1 Bedrijfsvoorstelling

In dit hoofdstuk wordt er kort omschreven wie ACA-IT Solution is en wat er allemaal gebeurt.

1.1 ACA-IT Solutions

ACA-IT Solutions is een IT-bedrijf dat in 1989 opgericht werd. Na het binnenstappen van de voormalige CEO Ronny Ruyters is het hoofdkantoor verhuisd naar Hasselt. In het begin focuste ACA-IT Solutions op Java Consultancy en de implementatie van projecten op een Agile manier. Na een sterke groei van het bedrijf heeft ACA-IT Solutions zich verspreid over verschillende locaties binnen België. Deze bedrijven bevinden zich in Hasselt, Olen en Gent. Er zijn ook twee bedrijven overgenomen.

In 2015 heeft het bedrijf een grote stap verdergezet: ACA-IT Solutions veranderde zijn organisatie in een podulaire organisatie. Ieder pod heeft zijn eigen expertise en kan als een autonoom bedrijf behandeld worden. Momenteel zijn er in het totaal negen pods beschikbaar.
ACA-IT Solutions telt 175 werknemers die verspreid zijn over verschillende bedrijfslocaties en klanten. De stagiair bevindt zich op twee locaties: maandag, woensdag en vrijdag in Hasselt en dinsdag en donderdag in Olen.

De pod waarin de stagiair bevindt is OpsKlaar. OpsKlaar biedt een geavanceerde en een persoonlijke cloud omgeving voor zijn klanten: starten van een cloud omgeving, cloud migratie en optimalisatie, managed cloud, beveiligen van cloud, cloud automatiseren en ten slotte DevOps. Deze pod telt acht Cloud Engineers.

Figuur 1: ACA-IT Solutions logo

1.2 Partners

In dit hoofdstuk volgt een korte omschrijving over de partners van ACA-IT.

1.2.1 Alfresco Gold Partner

Alfresco is een opensource softwarebedrijf dat zich specialiseert in het digitaliseren en beheren van content. Bovendien zijn ze ook gespecialiseerd in het veilig beheren van bedrijfsinformatie.

Figuur 2: Alfresco logo

1.2.2 Atlassian Gold Solution Partner

Atlassian is een Australisch bedrijf dat producten ontwikkelt voor softwareontwikkelaars, projectmanagers en content managers. Binnenin ACA-IT Solutions worden er verschillende producten van Atlassian gebruikt om te communiceren en beheren van projecten.

Figuur 3: Atlassian logo

Producten die ACA-IT Solutions gebruikt:

- Jira Software;
- BitBucket;
- Confluence.

1.2.3 Liferay Service Partner Platinum

Liferay is een opensourcebedrijf dat gratis documentatie en/of betalend professionele service aanbiedt aan bedrijven die hulp vragen bij het creëren van digitale ervaringen op het web. De onderneming richt zich voornamelijk op Enterprise-portal technologieën.

Figuur 4: Liferay logo

1.2.4 New Relic

New Relic is een cloud gebaseerd platform dat aan hun ontwikkelaars, ingenieurs, operations en management een duidelijk beeld geeft over wat er gaande is in de complexe softwaretoepassingen van vandaag. Hierdoor worden de problemen sneller opgelost en een goed presterend DevOps-team ontwikkeld.

1.2.5 ORACLE

Oracle is één van de grootste software development bedrijven die verschillende producten aanbiedt. Het bedrijf telt wereldwijd liefst 130.000 werknemers.
De bekendste producten die Oracle aanbiedt zijn:

- Databases: onder andere Oracle 11g, MySQL, JAVA DB, InnoDB, …;
- Developer tools;
- Oracle Fusion Middleware;
- E-Bussiness Suites (ERP/CRM Pakket).

1.2.6 BACKBASE

Backbase is de maker van de *Omni-Channel Banking Platform*, een software voor digitaal bankieren die gegevens en functionaliteit van traditionele kernsystemen verenigt in een naadloze, digitale klantervaring.
Backbase geeft financials de snelheid en flexibiliteit om klantervaringen op elk apparaat te creëren en te beheren, en meetbare bedrijfsresultaten te leveren.

1.2.7 Amazon Web Services Partner Network

AWS (Amazon Web Services) bestaat uit verschillende cloud computing-producten en -services. Amazon Web Services kan worden onderverdeeld in twee hoofdproducten: EC2, virtuele machineservice en S3, opslagsysteem.
Amazon Web Services is wereldwijd enorm populair voor verschillende toepassingen. Met Amazon Web Services kan elk bedrijf gebruik maken van de meest geavanceerde technologieën tegen zeer beperkte kosten. De kosten worden namelijk berekend op basis van het verbruik, waardoor er geen grote investeringen vooraf nodig zijn.
Amazon.com is ook het tweede grootste bedrijf op aarde met een marktwaarde van 777.8 Biljoen dollar.

Figuur 5: AWS-logo

Ranking of the companies rank 1 to 100	Market value in billion U.S. dollars
Apple	926.9
Amazon.com	777.8
Alphabet	766.4
Microsoft	750.6
Facebook	541.5

Tabel 1: Ranglijst AWS-marktwaarde

1.2.8 Proximus

Proximus is het grootste telecommunicatiebedrijf van België dat vroeger Belgacom heette. Proximus is een groep van verschillende bedrijven, onder andere:

- Proximus nv;
- Scarlet;
- Skynet;
- Tango;
- BICS;
- PingPing.

Proximus biedt verschillende telecommunicatieservices aan zoals telecommunicatie, mobiele telefonie, digitale televisie en internet.

1.2.9 Axway

Axway levert softwaretools voor Enterprise Application Integration. Ook zorgt het bedrijf voor monitoring van bedrijfsactiviteiten, bedrijfsanalyses, ontwikkeling van mobiele applicaties en web API-beheer.

Figuur 6: Axway logo

1.2.10 SAP Silver Partner

SAP is de wereldleider in bedrijfstoepassingen op het gebied van software- en software gerelateerde service-inkomsten. SAP is ook de werelds derde grootste onafhankelijke softwarefabrikant.

Figuur 7: SAP logo

1.2.11 Adobe Community Solution Partner

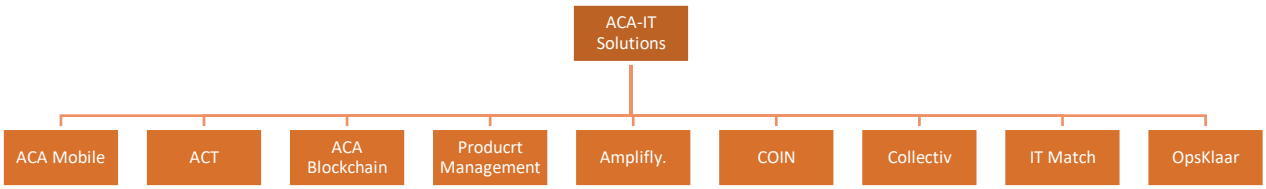
Adobe is een Amerikaans bedrijf dat 18.000 werknemers telt. Adobe werd bekend met zijn beeldbewerkingsprogramma Photoshop, dat na een jaar de grootste werd op de markt. Adobe telt meer dan dertig programma's die elk gespecialiseerd zijn in hun eigen categorie. Een paar bekende producten zijn:

- Adobe Photoshop;
- Adobe Reader;
- Adobe After Effects.

Figuur 8: Adobe logo

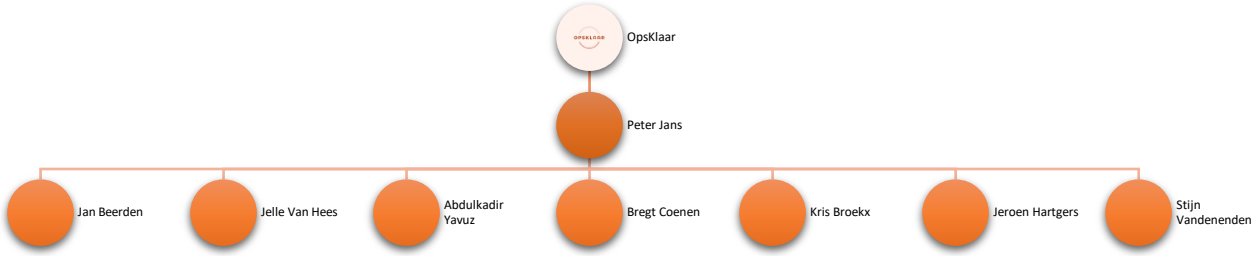
1.3 Organigram

ACA-IT Solutions bestaat uit negen businessunits(pods). Deze businessunits hebben elk hun expertises en kunnen als onafhankelijk bedrijf behandeld worden.
In figuur 9 is het organigram van ACA-IT Solutions weergegeven. Dit organigram is actief sinds de verandering in 2015.



Figuur 9: Organigram ACA-IT Solutions

De stageopdracht werd in de OpsKlaar businessunits gevolgd. Deze pod beheert de cloud omgeving en beschikt over acht ervaren Cloud Engineers. Grotendeels draaien de servers op Amazon Web Services.
In figuur 10 is het organigram van OpsKlaar weergegeven.



Figuur 10: Organigram OpsKlaar

2 Projectopdracht

In dit hoofdstuk worden de gebruikte technologieën van het project toegelicht.

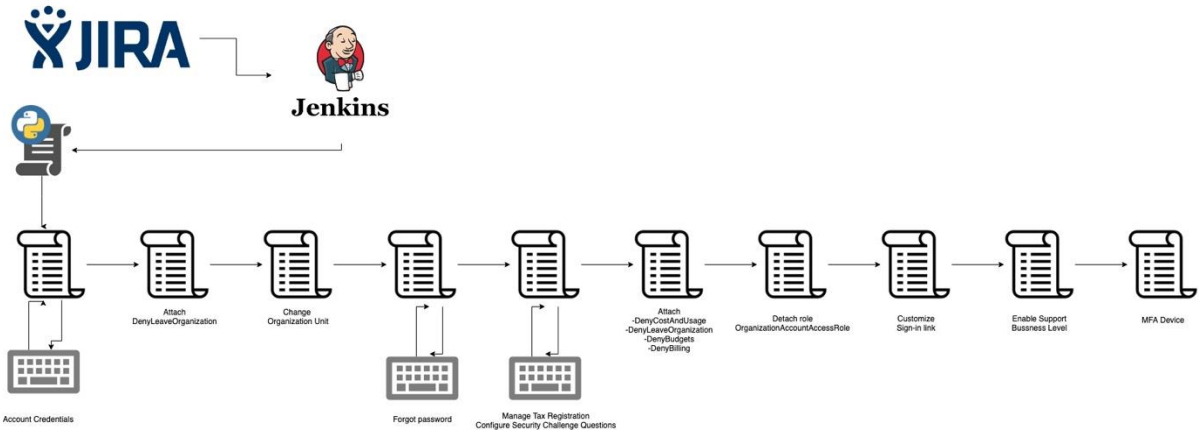
2.1 Probleemstelling

Deze stage kwam tot stand omdat ACA-IT Solutions nood had/heeft aan een script dat automatisch accounts moet aanmaken in de Organization Service van Amazon Web Services.

Doordat ACA-IT Solutions een sterk groeiend bedrijf is binnen de IT-wereld, komen er elk jaar nieuwe klanten bij. Voor elk van deze klanten wordt er een nieuwe account aangemaakt binnen de Organization service van Amazon Web Services.

Met de huidige infrastructuur van ACA-IT Solutions moet de Cloud Engineer deze accounts handmatig aanmaken. Bij een nieuwe klant wordt er een issue in Jira aangemaakt. Dit issue wordt door een Cloud Engineer bekeken en gevalideerd om verder de manuele stappen te volgen om een account aan te maken. Dit proces neemt te veel werkuren in beslag en is niet efficiënt.
Ook kunnen er verschillende configuratiefouten gemaakt worden tijdens het creëren van het account.

Het doel van de opdracht is om deze stappen zo efficiënt en geautomatiseerd mogelijk te behandelen.



Figuur 11: Process flow

In figuur 11 wordt de *process flow* voorgesteld. Er wordt een Jirra issue aangemaakt voor een nieuwe klant die een *Jenkins build* aanroept. Vervolgens wordt het script uitgevoerd om een account aan te maken.

2.2 Gebruikte technologieën

Bij deze opdracht wordt er verschillende technologieën aangesproken. Om een geautomatiseerd proces te bekomen worden er verschillende technologieën bij elkaar geïmplementeerd om zo een automatisch *process flow* te creëren.

2.2.1 Amazon Web Services

Amazon Web Services is een grote cloud *vendor* die meerdere services aanbiedt. Bij ACA-IT Solution worden verschillende virtuele machines en verschillende services via Amazon Web Services aangeboden.

Organizations service geeft de mogelijkheid om centraal het account te beheren en te configureren. Bij dit project wordt de verschillende API-calls gebruikt om zo accounts automatisch te creëren en te beheren.
Deze service zorgt ervoor dat er verschillende accounts binnen een organisatie aangemaakt kunnen worden. De aangemaakte accounts krijgen hun eigen policy's om de rechten binnenin het account te beheren. Ook geeft Organization service de mogelijkheid om accounts te groeperen om zo een scheiding te kunnen maken tussen verschillende accounttypes.

Figuur 12: AWS cloud logo

2.2.2 Python

Dit project is in Python geschreven. Python is een programmeertaal die in 1991 ontworpen is. Momenteel is Python een sterk groeiende programmeertaal binnen in de programmeercommunity. Door zijn sterke groei is Python verkozen tot de beste programmeertaal van 2018. Ook biedt Python verschillende API-call mogelijkheden in Amazon Web Services.

Voor elke stap die er nodig is bij het creëren van het account wordt er een functie geschreven. Deze functies bevatten verschillende *error handling* zodat er bij een fout of *error* verstaanbare tekst tevoorschijn komt.

Om Python te gebruiken zijn er verscheidene packages nodig. Deze zijn Boto3 en AWSCLI.
Deze packages worden in het hoofdstuk over implementatie verder uitgelegd.

2.2.3 Jenkins

Jenkins is een opensource automatiseringsserver voor het automatiseren van *non-human* processen. In dit project wordt Jenkins gebruikt om een script uit te voeren. Dankzij de grote community beschikt Jenkins over duizend plug-ins. Bij dit project wordt er een plug-in gebruikt om integratie tussen Jenkins en Jira tot stand te brengen.

Om een gestructureerde werking te verkrijgen wordt er een Jenkins-pipeline geschreven. Dit is omdat er verschillende stappen zijn die niet geautomatiseerd kunnen worden. Bij niet-geautomatiseerde stappen verschijnt er een boodschap in het venster met de verdere instructies die de Cloud Engineer moet volgen.

2.2.4 Jira

Jira is een ticketing-systeem waarbij werknemers verschillende issues plaatsen en bekijken om zo de nodige handelingen uit te voeren. In dit project is het de bedoeling dat bij elke nieuwe klant een issue aangemaakt wordt. Vervolgens roept deze issue een Jenkins *build* aan. Wanneer de *build* succesvol behandeld wordt, wordt de issue op *'Solved'* gezet. Hiervoor worden er verschillende REST API-calls geschreven om services te beheren.

2.2.5 Docker

Docker is een computerprogramma om softwarepakketten uit te voeren. Deze softwarepakketten heten 'containers'. Deze containers zijn van elkaar geïsoleerd en hebben geen invloed op elkaar.

Deze tool helpt om applicaties en services in containers te draaien in plaats van op de hostmachine zelf. Containers hebben veel minder computerspecificaties nodig dan normale VM's, omdat enkel een applicatie op de containers draait en niet een heel nieuw besturingssysteem met daarbovenop nog de applicatie zelf.

Containers worden aangemaakt door images die online te vinden zijn. Tijdens de stageopdracht werden de containers beheerd door Kubernetes in de 'Docker For Mac'-software.

2.2.6 Kubernetes

Kubernetes is een systeem om de *deployment*, *scaling* en beheer van gecontaineriseerde applicaties te automatiseren. De meeste services en applicaties van ACA-IT Solutions draaien in een Kubernetes cluster op de Amazon Elastic Container Service for Kubernetes van AWS.
De kleinste bouwsteen van Kubernetes is een *pod*. Een *pod* bevat de Docker container(s) waar de applicaties en services in draaien. In de *pods* kunnen ook volumes ingesteld worden om opslag te garanderen.

Kubernetes beschikt over verschillende *deployment* soorten met elk zijn eigen doel. De drie *deployments* die Kubernetes ondersteundt heten: *Deployment*, *DaemonSet* en *StatefulSet*. De standaard *deployment* garandeert de beschikbaarheid van de *pods*. De *daemonSet* zorgt ervoor dat elke node van de cluster ten minste één instantie van die pods draait.
En als laatste is er *StatefulSet:* dit zorgt ervoor dat elke pod een unieke ID krijgt. Met dit soort *deployment* worden er vaak databases gecreëerd.

2.2.7 Helm

Om de pods te *deployen* in een Kubernetes-cluster is Helm gebruikt. Helm gebruikt Helm *charts* om pods te *deployen* op de cluster. Deze *charts* worden vooraf geconfigureerd. In deze *charts* zitten:

- Soort van de *chart*;
- Welke containers in de *pods* draaien;
- Welke volumes de pod nodig heeft;
- Welke configuratie-instellingen op de pod worden toegekend, etc.

In figuur 14 wordt de verschillende vooraf gedefinieerde eigenschappen weergegeven.

```yaml
releases:
  - name: jira
    namespace: jira
    chart: ./atlassian-jira-software
    values:
      - ingress:
          enabled: false
        jira:
          javaHeapSize: 1536m
          # Disable websudo for SAML or OIDC
          javaOptions: -Xmx2g -Xms2g
        resources:
          limits:
            # for single core CPU
            cpu: 800m
            memory: 2560Mi
          requests:
            cpu: 0
            memory: 2560Mi
```

Figuur 14: Jira Helm Chart

Figuur 13: Helm-logo

Elk tool van de testomgeving heeft een eigen *helm chart* waarop alle nodige configuratiebestanden zich bevinden.

2.2.8 Slack

Slack is Cloudbased software die de mogelijkheid geeft om in groepen of individueel met elkaar te communiceren binnen het bedrijf en/of buiten het bedrijf. Binnen ACA-IT Solutions wordt Slack gebruikt om interne en externe communicatie te voeren tussen medewerkers. Ook wordt van Slack gebruikgemaakt om verschillende applicaties te integreren.
Een voorbeeld van integratie is Jira. Bij deze integratie worden nieuwe issue-meldingen als een bericht getoond in Slack.

2.3 Uitwerking stageopdracht

In dit hoofdstuk wordt de werking en de implementatie van het project toegelicht.

2.3.1 Manueel verloop

Om een account aan te maken binnen Organization service van AWS worden de onderstaande stappen manueel behandeld:

Input Credentials	*Creating Account*	*Add Policy*	*Change Organization Unit*	*Create Password* ➔

➔ *Edit 'Manage Tax Registration'*	*Configure Security Challenge Questions*	*Add Policy's*	*Detach role*	*Customize the sign-in link*

1. Gegevens
   In dit proces wordt de benodigde accountinformatie aangevraagd om een account aan te maken. Deze informatie bestaat uit: *email*, *account name*, *role name* en *access to billing.*

2. Account aanmaken
   Als de benodigde informatie (stap a) binnengehaald is, maakt het script het account aan.

3. Koppelen van een policy aan een account
   Elk aangemaakt account wordt in een DenyLeaveOrganization-policy gekoppeld zodat het account de organisatie niet kan verlaten.

4. Veranderen van organisatie unit (groep)
   Hier wordt het account in de juiste OU gezet. Elk OU heeft zijn eigen rol en eigen policy.

OU	Beschrijving
ACA/TEST-ACCOUNTS	Interne testaccounts
ACA/PRODUCTION-ACCOUNTS	Interne productieaccounts
CUSTOMER/ACA-OWNED-ACCOUNTS	Klantaccounts waarbij ACA-IT Solutions de eigenaar is van de rootgebruiker
CUSTOMER/CUSTOMER-OWNED-ACCOUNTS	Klantaccounts waarbij de klant eigenaar is van de rootgebruiker

5. Een sterk wachtwoord aanmaken
   Tijdens deze stap wordt een wachtwoord aangemaakt dat aan de volgende vereisten voldoet:
   a. Lengte van het wachtwoord moet bestaan uit 32 tekens;
   b. Bevat zowel grote als kleine letters;
   c. Bevat zowel cijfers als speciale tekens.

6. Instellen van *tax* gegevens
   a. *Country*: Belgium
   b. *Tax Registration Number*: BEXXXXXXXXX
   c. *Business Legal Name*: ACA IT-Solutions NV
   d. *Address*: Herkenrodesingel 8B 2.01
   e. *City*: Hasselt
   f. *State*:/
   g. *Postal code*: 3500
   h. *Country*: Belgium

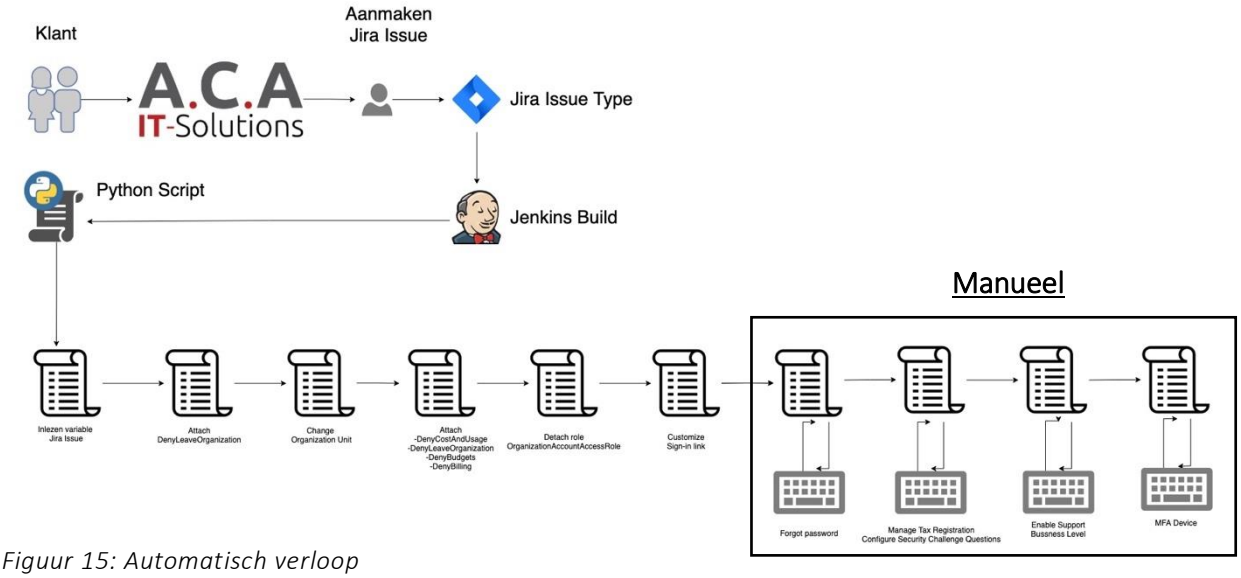
7. Veiligheidsvragen instellen
   Antwoorden op de volgende vragen moeten 32 tekens bevatten. Moeten uit: kleine letters, grote letters en cijfers bestaan.
   a. Question: Security Challenge Response 1?
   b. Answer: <secret 1>
   c. Question: Security Challenge Response 2?
   d. Answer: <secret 2>
   e. Question: Security Challenge Response 3?
   f. Answer: <secret 3>

8. Policy toevoegen
   Om verschillende rechten toe te kennen aan het account worden de volgende policy's gekoppeld.
   a. FullAWSAccess
   b. DenyLeaveOrganization
   c. DenyCostAndUsageReport
   d. DenyBudgets
   e. DenyBilling

9. Rol verwijderen
   Wanneer de configuratie van het account is voltooid wordt de OrganizationAccountAccessRole rol van het account afgehaald.

10. Aangepaste aanmeldlink maken
    Om in te loggen wordt er een aangepaste aanmeldlink gegenereerd door Amazon Web Services.

2.3.2 Automatische verloop



Figuur 15: Automatisch verloop

Bij een nieuwe klant wordt er een nieuwe issue aangemaakt in Jira. Dit issue wordt vervolgens aan een speciale issue type toegekend. Dit issue type heet 'CreateAccount'.

Vervolgens wordt er drie parameters ingevuld. Deze drie parameters worden in de *Custom fields* geschreven.

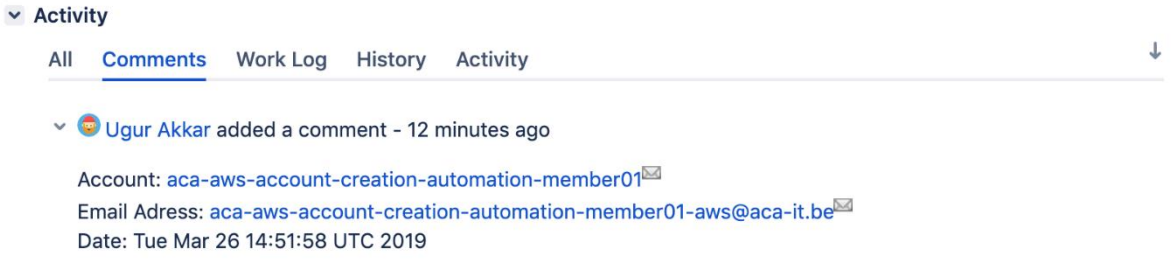
Voorbeeld:

AccountName:	aca-aws-account-creation-automation-member01
AccountEmail:	aca-aws-account-creation-automation-member01-aws@aca-it.be
Organization Unit:	ACA/PRODUCTION-ACCOUNTS

Figuur 16: Voorbeeld description sectie

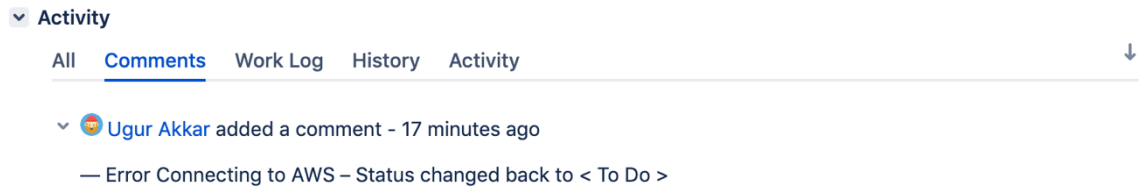
Eén van de Cloud Engineers controleert en valideert deze velden, en vervolgens de status van de issue op '*In Progress*' gezet.

Wanneer de status geüpdatet wordt, wordt er een *webhook* gestuurd naar de Jenkins-server die vervolgens een pipeline aanroept. Deze pipeline doorloopt verscheidene stappen en maakt een account aan. Als deze script succesvol behandeld wordt, wordt er een boodschap toegevoegd aan de *Comments* sectie en de status van de issue op 'Done' gezet.



Figuur 17: Voorbeeld Comments sectie 'Success'
Figuur 17 toont aan dat het account succesvol gecreëerd is.

Als er een fout of probleem tevoorschijn komt tijdens het creëren van het account, wordt de oorzaak van de fout getoond in de *Comments* sectie en de status van de issue terug op 'To Do' gezet.



Figuur 18: Voorbeeld Comments sectie 'Error'

Om een optimale interactie te bekomen worden deze meldingen ook naar Slack doorgestuurd.

Als laatst moeten er nog vier stappen manueel uitgevoerd worden. Voor deze stappen geeft AWS geen ondersteuning via API-calls:

- Een sterk wachtwoord aanmaken;
- Instellen van taks gegevens;
- Veiligheidsvragen instellen;
- Rol verwijderen.

Mogelijke oplossingen voor niet automatiseerbare stappen:

Instellen van taks gegevens:
Voor het wijzigen van de taks gegevens bestaat een package genaamd 'coto'. Deze package stuurt AWS API-calls naar het *billing resource* van AWS. Maar omdat deze package geen officiële AWS-package is, is deze package gevaarlijk en niet veilig voor gebruik. Het package kan ook ieder moment stuk gaan.

2.3.3 Jira Issue

Figuur 19: Workflow

Bij een nieuw probleem of taak wordt er een Jira issue aangemaakt. Via deze issue kunnen de Cloud Engineers zien wat er allemaal nog opgelost moet worden. Ook kunnen de Cloud Engineers de tijd die ze spenderen aan een issue hierop koppelen. Bij dit project wordt er een nieuw issue type aangemaakt die 'CreateAccount' heet. Voortaan worden de nieuwe accounts onder deze issue type aangemaakt en bijgehouden.

Deze issue heeft een workflow dat in twee richtingen kan doorlopen. Bij het begin wordt de issue in de *'To Do'* fase gezet. Als de vereiste parameters ingevuld wordt gaat de Cloud Engineers dit controleren en vervolgens valideren en in de *'In Progress'* fase zetten. Als het script foutloos verwerkt wordt, gaat deze issue automatisch in de *'Done'* fase gezet worden. Maar als er een fout ontstaat tijdens het uitvoeren van het script wordt de fase terug op *'To Do'* gezet.

In figuur 19 wordt de workflow visueel voorgesteld.

2.3.4 Jira Webhook

Een *webhook* wordt getriggerd door een event in de Jira issue en levert een actie op in de Jenkin-pipeline. Bij deze opdracht wordt de *webhook* getriggerd wanneer de issue status van '*To Do*' naar *'In Progress'* gezet wordt. Deze actie activeert de *webhook* en roept vervolgens een Jenkins-pipeline aan.

2.3.5 Jenkins-pipeline

Jenkins-pipeline wordt gebruikt om verschillende stappen te automatiseren. Bij deze opdracht werden de volgende stappen geautomatiseerd:

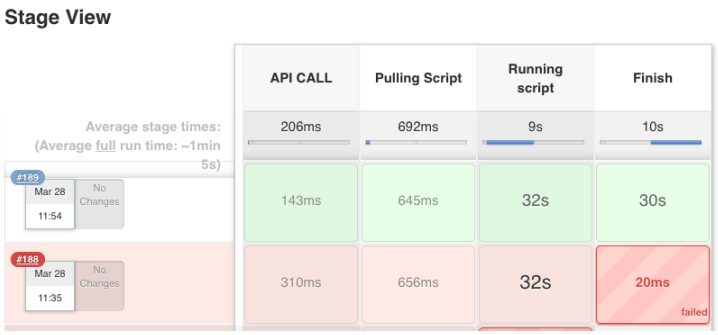
- API-call tegen Jira om informatie op te vragen van de issue;
- Het binnenhalen van het script van een STASH *repository;*
- Het uitvoeren van een script;
- API-call tegen Jira om een boodschap te tonen in de *Comments* sectie;
- API-call tegen Slack om een boodschap te tonen in een *workspace/channel;*
- Bij een fout of mislukte bouw poging een melding weergeven in Jira issue;
  Bij een fout of mislukte bouw poging een melding weergeven in de Slack *workspace/channel.*

Figuur 20 visualiseert twee stappen. In de eerste stap wordt het script binnengehaald via een SSH-connectie en in de tweede stap wordt er eerst een error file gecreëerd en vervolgens het script uitgevoerd met het parameters dat wordt meegegeven.

```
stage('Pulling Script') {
    git branch: 'master',
            credentialsId: 'StashSSH',
            url: 'ssh://git@stash.aca-it.be:7999/aaaca/aws-account-creation-automation-scripts.git'
}
stage('Running script') {
    sh 'touch error.txt'
    sh 'python3 ./Script/Python/CreateAccount.py ' + account + ' ' + email + ' ' + mfa + ' ' + ou + ' 2> errors.txt'
}
```

*Figuur 20: Voorbeeld jenkinsfile (*pipeline*)*

Jenkins-pipelines worden meestal door vooraf geschreven bestanden beheerd. Het bestand dat Jenkins-pipelines beheerd wordt Jenkinsfile genoemd.



Figuur 21: Jenkins-pipeline

2.3.6 Jenkins Plug-ins

2.3.7 HTTP Request Plug-in

HTTP Request plug-in geeft de mogelijkheid om POST API-calls uit te voeren binnen een Jenkinsfile. Zonder deze plug-in kunnen er geen POST calls uitgevoerd worden om informatie binnen te halen van Jira of informatie te verzenden naar Jira of Slack.

```
def responseStatus = httpRequest consoleLogResponseBody: false, \
httpMode: 'POST', contentType: 'APPLICATION_JSON', requestBody: status, \
url: "http://192.168.64.11:32036/rest/api/2/issue/${issueid}/transitions", \
customHeaders: [[name: 'Authorization', value: "Basic ${auth}"]]
```

Figuur 22: HTTP-Request plug-in

Bij figuur 22 wordt er een variabel aangemaakt. In deze variabel wordt het *HttpRequest* functie aangeroepen, bij het aanroepen wordt er parameters gedefinieerde zodat er het gewenste resultaat verkregen wordt.

2.3.8 Global Slack Notifier plug-in & Slack Notification plug-in

Met deze twee plug-ins wordt er een connectie gemaakt tussen de Slack server en de Jenkins-server. Bij het gebruik van deze plug-in wordt er eerst een API-token gecreëerd zodat de plug-in over voldoende permissies beschikt om een API-call te kunnen uitvoeren. Deze permissies zijn nodig om successen of foutmeldingen door te sturen naar Slack.

```
def notifyBuild(String buildStatus = 'STARTED', slackComment) {

    def colorName = 'RED'
    def colorCode = '#FF0000'
    def subject = "${buildStatus}: Job '${env.JOB_NAME} [${env.BUILD_NUMBER}]'\n" + slackComment
    def summary = "${subject} (${env.BUILD_URL})"

    if (buildStatus == 'STARTED') {
        color = 'YELLOW'
        colorCode = '#FFFF00'
    } else if (buildStatus == 'SUCCESS') {
        color = 'GREEN'
        colorCode = '#00FF00'
    } else {
        color = 'RED'
        colorCode = '#FF0000'
    }

    slackSend(color: colorCode, message: summary)
}
```

Figuur 23: Global Slack Notifier plug-in

In figuur 23 wordt er een functie aangemaakt dat '*notifyBuild*' heet. Deze functie creëert een boodschap; als Jenkins de pipeline foutloos heeft uitgevoerd gaat deze boodschap in een groene lay-out tevoorschijn komen. Maar als er een fout is in de pipeline wordt de boodschap in het rood gevisualiseerd.

2.3.9 Matrix Authorization Strategy plug-in

Matrix Authorization plug-in zorgt ervoor dat er genoeg rechten aan een account gekoppeld wordt om een *build* te starten via een API-call. Niet elke gebruiker heeft dezelfde rechten.

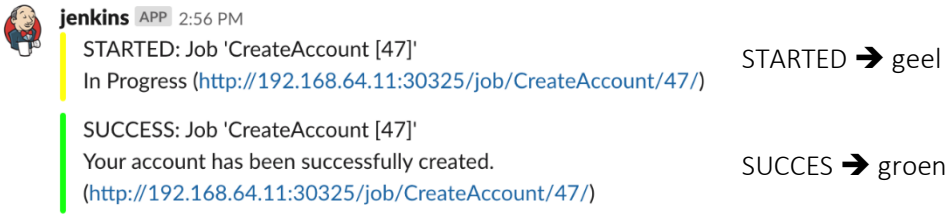
Figuur 24: Matrix Auth. Strategy plug-in

In figuur 24 worden de verschillende groepen en gebruikers geïllustreerd. De gebruiker 'Ugur Akkar' heeft een administrator recht die alles kan beheren. De gebruiker 'jenkinswebhook' krijgt enkel *read* rechten toegekend. Dit geeft de mogelijkheid dat de gebruiker alleen de resources kan bekijken maar niet aanpassen. Ook kan de gebruiker API-calls uitvoeren op Jenkins *builds*.

2.3.10 Slack plug-ins

Om Jenkins notificaties te tonen op een Slack channel moet er eerst een plug-in geïnstalleerd worden. Deze plug-in zorgt ervoor dat er een connectie opstand gebracht kan worden tussen Jenkins en Slack. Jenkins Plug-in registreert de API-calls dat vervolgens de aangegeven functies uitvoert. Bij deze opdracht werd er een API-call gestuurd vanuit Jenkins met de nodige boodschap. De boodschap dat in de API-call zit wordt vervolgens naar de Slack channel doorgestuurd en getoond.

Voorbeeld:



Figuur 25: Slack Jenkins CI plug-in

2.4 Wijzigingen voor productieomgeving ‘AWS Account creatie’

Om van testomgeving naar een productieomgeving over te stappen werd er kleine wijzigingen in Jenkins, Jira en de Python script aangebracht. Deze wijzigingen worden in dit hoofdstuk vermeld.

2.4.1 Jenkins

In de test omgeving wordt alleen issue id als parameter meegegeven, maar voor de productie omgeving zijn er drie nieuwe parameters bij toegevoegd.

Deze drie nieuwe parameters zijn: AWS_SECRET_ACCESS_KEY, AWS_SESSION_TOKEN en AWS_ACCESS_KEY. Deze parameters worden gebruik gemaakt bij de connectie tussen Jenkins en AWS. Om deze parameters in te vullen moet er eerst een rol *assumed* worden. Deze *assume* functie gaat een JSON-formaat tekst uitprinten, in deze JSON-file zitten de nodige parameters.

Bij het Jenkinsfile worden de testomgeving URL’s van Jira en Slack platform gewijzigd naar de URL’s van ACA-IT Solutions platform.

Door security redenen kan er geen API-call gestuurd worden naar de Jenkins-pipeline. Om dit probleem optelossen wordt er een plug-in gebruikt namelijk Jira Trigger plug-in. Deze plug-in wordt gedetailleerd uitgelegd in de onderzoek gedeelte.

2.4.2 Jira

Bij het workflow worden extra stappen toegevoegd. Omdat elke workflow hetzelfde verloop heeft binnen het bedrijf, wordt er een extra block toegevoegd aan het begin van de workflow.

2.4.3 Python script

Om geen root users *credentials* mee te geven aan het script wordt er een rol gecreëerd, deze rol gaat automatisch een token creëren voor het Jenkins-user binnen AWS. Vervolgens gaat de user deze token gebruiken om het script uit te voeren.

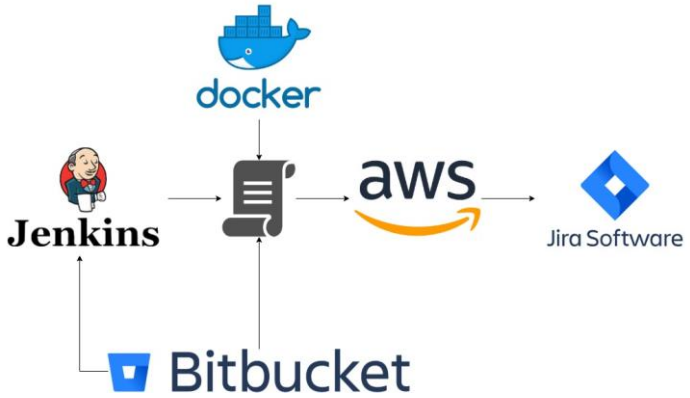
```
print("----------------------")
print("> Connecting to AWS")
print("----------------------")
#   Connecting to Organization unit

jenkinsManagerClient = boto3.client('iam',
                        aws_access_key_id=AWS_ACCESS_KEY_ID,
                        aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
                        aws_session_token=AWS_SESSION_TOKEN)

response = jenkinsManagerClient.list_access_keys(
    UserName=jenkinsManagerRole
)
if response['AccessKeyMetadata']:
    for x in response['AccessKeyMetadata']:
        response = jenkinsManagerClient.delete_access_key(
            UserName=jenkinsManagerRole,
            AccessKeyId= x['AccessKeyId']
        )

responseJenkinsUserKeys = jenkinsManagerClient.create_access_key(
    UserName= jenkinsManagerRole
)

closeConnectionAfter = 0;
while True:
    try:
        organizationManagerClient = boto3.client('sts',
                                            aws_access_key_id=responseJenkinsUserKeys['AccessKey']['AccessKeyId'],
                                            aws_secret_access_key=responseJenkinsUserKeys['AccessKey']['SecretAccessKey']
                                             )
        response = organizationManagerClient.get_caller_identity()

        responseOrganizationManager = organizationManagerClient.assume_role(
            RoleArn='arn:aws:iam::'+response['Account']+':role/OrganizationManager',
            RoleSessionName=sessionNameRole
        )

        break
    except ClientError as e:
        closeConnectionAfter = closeConnectionAfter + 1
        print("Access Key is not Active!")
        sleep(2)
        if (closeConnectionAfter == 150):
            sys.exit("Could not use Access Key!")

credentials = responseOrganizationManager['Credentials']

client = boto3.client('organizations',
                        aws_access_key_id=credentials['AccessKeyId'],
                        aws_secret_access_key=credentials['SecretAccessKey'],
                        aws_session_token=credentials['SessionToken']
                        )
```

Figuur 26: Productieomgeving credentials

Als laatst worden de policy en Organization unit id's gewijzigd naar het productie omgeving id's.

2.5 Automatisatie van Jira Software deployment

2.5.1 Elastic Container Service for Kubernetes (EKS)

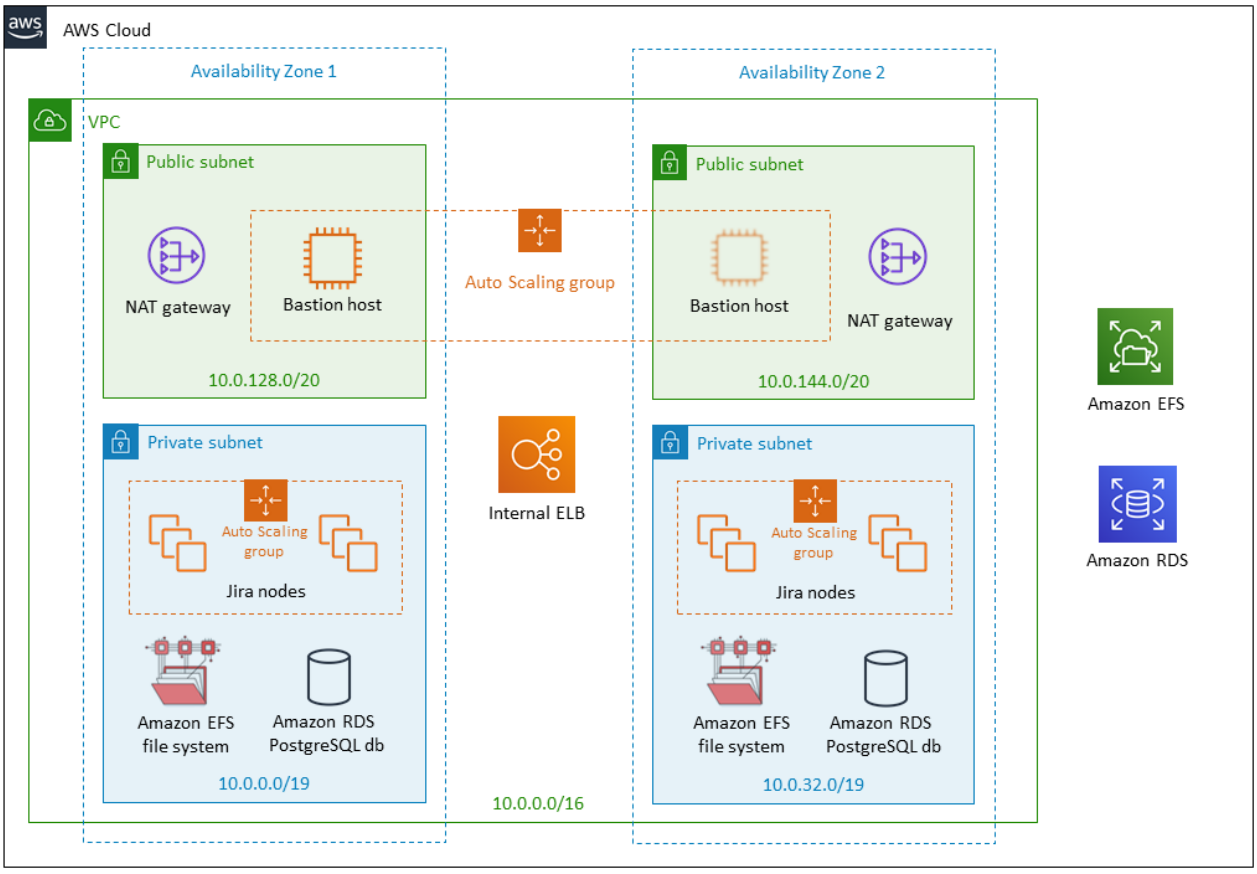


Elastic Container Service for Kubernetes zorgt ervoor dat Kubernetes rechtstreeks op AWS Docker containers kan deployen. Bij deze opdracht worden Docker containers gedployed met behulp van Kubernetes op de aangemaakte AWS-account.

Figuur 27: EKS Cluster en Atlassian Jira

Met EKS wordt het installeren en het onderhouden van software makkelijker en efficiënter. Een manuele installatie van Jira software duurt ongeveer één tot twee uur soms wat langer, met EKS gaat dit verminderen tot 5 minuten effectieve werktijd.

2.5.2 Terraform



Figuur 28: AWS-Configuatie

Om EKS te gebruiken moeten verschillende componenten geconfigureerd worden. Bij de bovenstaande afbeelding worden de componenten visueel tevoorschijn gebracht. VPC, subnets, Elastic IPs, EKS, RDS, EFS, etc. Deze configuratie wordt in Terraform geschreven en vervolgens door Terraform uitgevoerd.

In figuur 29 wordt de RDS visueel voorgesteld. Hier worden verschillende parameters ingevuld zodat Terraform RDS (*Relational Database Service*) kan aanmaken. Deze database wordt gebruikt bij Jira software.

```
resource "aws_db_instance" "Jira_RDS" {
  allocated_storage       = 20 # gigabytes
  backup_retention_period = 7   # in days
  db_subnet_group_name    = "${aws_db_subnet_group.Jira_RDS_Subnet_Group.name}"
  engine                  = "postgres"
  engine_version          = "9.6.12"
  identifier              = "jirards"
  instance_class          = "db.t3.micro"
  multi_az                = false
  name                    = "jiraRDS"
  parameter_group_name    = "${aws_db_parameter_group.Jira_RDS_Parameter_Group.name}" # if you have tuned it
  password                = "${trimspace(file("${path.module}/secrets/mydb1-password.txt"))}"
  port                    = 5432
  publicly_accessible     = true
  storage_encrypted       = true # you should always do this
  storage_type            = "gp2"
  username                = "ugur"
  vpc_security_group_ids  = ["${aws_security_group.Jira_RDS_Security_Group.id}"]
}
```

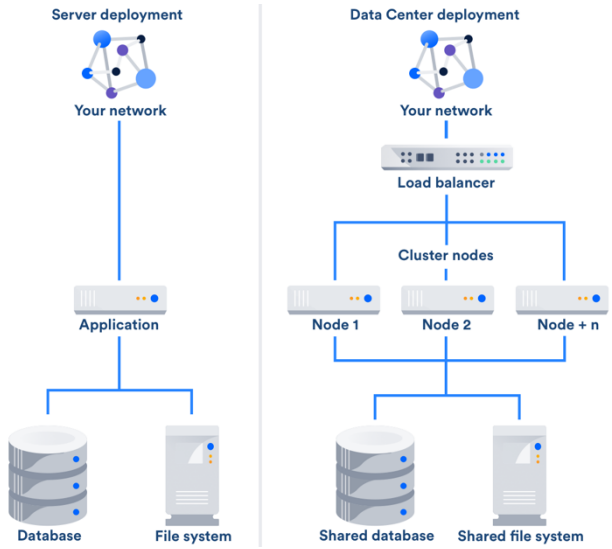
Figuur 29: Terraform RDS-configuratie

Wanneer Terraform al de nodige componenten geconfigureerd en klaargemaakt heeft kan Kubernetes vervolgens de Docker containers opspinnen op de EKS *nodes*.

2.5.3 Kubernetes

Om een werkende Jira software te deployen moeten er verschillende Kubernetes componenten geconfigureerd worden. Als eerst wordt er een NGINX-controller gecreëerd om binnen komende traffic op de juiste pods te lijden en de pods aan een LoadBalancer te koppelen.

Vervolgens wordt er een volume gecreëerd, in deze volume wordt de pod opgezet en geconfigureerd. Deze volume gaat niet verwijderd worden wanneer de pod volledig gewist wordt. Ook wordt er een sharedFolder aan de pod gekoppeld, deze sharedFolder is nodig bij het opzetten van een datacenter versie van Jira. Wanneer de pod moet *gescaled* worden naar 2 of meer *replicas*, zullen deze pods hetzelfde configuratie bestanden gebruiken door de sharedFolder.



Figuur 30: Jira datacenter

Bij het eerste deployment wordt er maar 1 pod op gespind, na dat deze pod is geconfigureerd en foutloos werkt, wordt er een tweede, derde, vierde, etc. pod opgezet.

Deze pods gaan met de hulp van een sharedFolder hetzelfde configuratie bestanden gebruiken en onderling herkennen als Nodes.

2.5.4 Jenkins

Deze configuraties worden met behulp van Jenkins geautomatiseerd. Jenkins gaat de nodige componenten voor EKS installeren en configureren vervolgens gaan de pods uitgevoerd worden door een Jenkins build.

Bij figuur 31 wordt de pipeline visueel voorgesteld, hier kan er gezien worden dat de automatisatie in verschillende stappen is onder gedeeld. Als deze pipeline succesvol is afgehandeld kan de gebruiker op de Jira softwarewebsite inloggen en gebruiken.

Figuur 31: Jenkins-pipeline Jira software

3 Conclusie

Bij het lopen van mijn stage heb ik veel dingen bijgeleerd van verschillinde technologieën tot de aanpak van taken. Deze leerrijke stageperiode zal loop van mijn carrière veel positieve bijdragen geven. Het aangeleerde technologieën zijn kernpunten voor mijn toekomst en carrière.

Tijdens deze stageperiode ben ik verschillende problemen tegen gekomen, deze problemen hielpen mij om in andere perspectieven te denken en in een nieuwe aanpak methodes te behandelen. Ook had ik soms problemen bij het durven van sommige dingen te zeggen of te doen, deze stageperiode hielp mij om in een verstandig en logische manier deze problemen te behandelen.

Het opleiding Toegepaste Informatica – Systeem en Netwerkbeheer heeft mij verschillende vaardigheden gegeven, deze vaardigheden hebben mij sterk geholpen in mijn stage verloop. Dankzij deze opleiding heb ik mij kunnen verdiepen in verschillende technologieën in mijn stage zoals: AWS, Kubernetes, Jenkins, etc.

Ook heb ik geleerd om verslagjes te schrijven en te documenteren. Hiermee heb ik geleerd om alles te documenteren en bij te houden, om achteraf een verslag of documentatie te schrijven. Voortaan probeer ik een documentatie te schrijven over de bereikte activiteit.

Het overzetten van testomgeving naar de productieomgeving ging niet vlotjes als gewenst. Bij het overzetten komen er verschillende securityproblemen, deze permissies zijn erg belangrijk voor het security en privacy van het bedrijf. Met heel wat testen en zo secure mogelijk proberen te houden is het overzet succesvol gelukt.

Ik ben zeer tevreden over het opgeleverde product, het automatisch aanmaken van een account binnen Organization services van AWS vervolgd door het automatisch deployment van Jira datacenter met Kubernetes (EKS), voor mijn stagebedrijf. Het opgeleverde eindresultaat van mijn stage is het gewenste eindresultaat. Het bedrijf zal een groot aantal werkuren besparen, van ongeveer 45 minuten voor het aanmaken van een account naar ongeveer 3 minuten.

II. Research topic

In dit hoofdstuk wordt de onderzoeksvraag bondig toegelicht en vervolgens worden de verschillende onderzoeksmethoden aangehaald.

1 Onderzoeksvraag

Doordat ACA-IT Solutions een sterk groeiend bedrijf is binnenin de IT-wereld, komen er elk jaar nieuwe klanten bij. Voor elk van deze klanten wordt er een nieuw account aangemaakt binnenin de Organization service van Amazon Web Services.

Op dit moment worden deze accounts handmatig aangemaakt. Omdat deze accounts handmatig aangemaakt worden, duurt dit proces erg lang en is het niet efficiënt. Ook kunnen er verschillende configuratiefouten gemaakt worden tijdens het creëren van het account.

Om dit proces op een snellere en efficiëntere manier te behandelen wordt er een script geschreven. Er zijn verschillende SDK-mogelijkheden binnen Amazon Web Services die API-calls behandelen. Dit onderzoek focust zich op het vergelijken van SDK's en hun API-Calls. Het heeft de meest bekende, meest gebruikte en modernste SDK's gekozen om zo een vergelijking te maken.

Uiteindelijk probeert dit onderzoek de volgende vragen te beantwoorden:

- Welke SDK is het meest geschikt voor het beheren van AWS API-calls en wat zijn de voor- en nadelen van de SDK ten opzichte van de andere SDK's?
- Wat zijn de beste integratiemethodes om een Python-script uit te voeren via een Jira Issue?

2 Onderzoeksmethode

Dit onderzoek is een vergelijkende studie tussen de verschillende SDK's samen met een literatuurstudie. Om alle tools gelijk te evalueren, zijn criteria opgesteld.

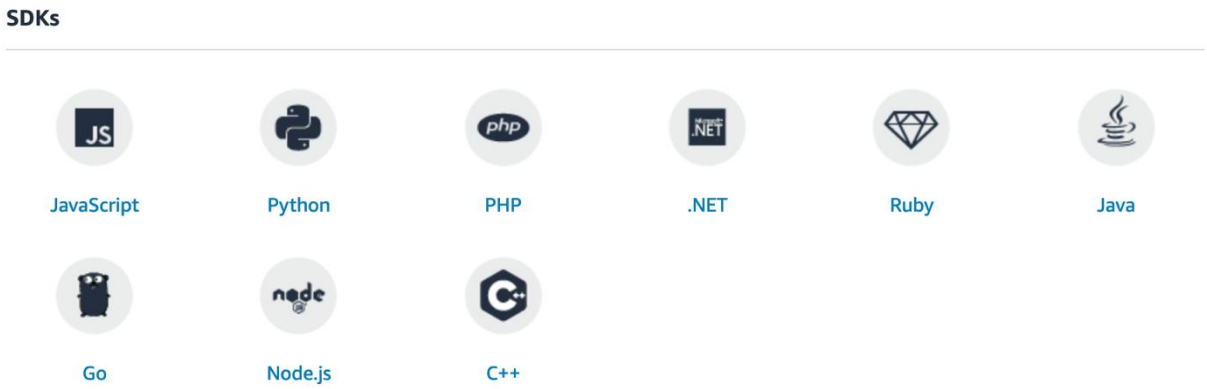
Om deze SDK's te testen wordt er gekeken in hoeverre het proces geautomatiseerd kan worden. Het proces moet door verschillende stappen gaan om zo een automatisch gecreëerd account te verkrijgen.

3 Onderzoek SDK

In dit hoofdstuk worden de verschillende SDKs opgesomd en vervolgens een vergelijking gemaakt tussen de meest populaire SDK-talen. Met deze vergelijking worden de top vier SDKs gebruikt om een script te schrijven en deze te testen op een testomgeving.

3.1 Beschikbaar SDKs

Amazon Web Services beschikt over verschillende SDK's om toegang te verlenen tot hun services en deze te beheren. Deze services worden *software development kits* genoemd.



Figuur 32: Beschikbare SDKs

3.2 Programmeertaal

In dit hoofdstuk wordt de verschillende programmeertalen onderzocht die momenteel beschikbaar gesteld zijn bij Amazon Web Services. Vervolgens worden deze programmeertalen bestudeert op populariteit en de functionaliteit die nodig is om een script te schrijven.

3.2.1 Meest gebruikte programmeertalen

3.2.1.1 TIOBE Index

TIOBE Index [1] is gespecialiseerd in het beoordelen en volgen van de kwaliteit van software. TIOBE Index meet de kwaliteit van een softwaresysteem door algemeen aanvaarde coderingsnormen toe te passen. Bij het analyseren en verzamelen van data worden er 25 verschillende zoekmachines gebruikt.

Volgens TIOBE Index zijn de top vijf meest opgezochte programmeertalen: Java, C, Python, C++ en Visual Basic .Net.

Bij de onderstaande lijst is te zien dat Java en C momenteel de meest opgezochte programmeertalen zijn en dat Python en Visual Basic .Net een sterke groei kennen.

Tabel 2: TIOBE Index TOP 5

Mar 2019	Mar 2018	Change	Programming Language	Ratings	Change
1	1		Java	14.880%	-0.06%
2	2		C	13.305%	+0.55%
3	4	▲	Python	8.262%	+2.39%
4	3	▼	C++	8.126%	+1.67%
5	6	▲	Visual Basic .NET	6.429%	+2.34%

Het jaar 2018 was een succes voor Python.
Python kreeg de Hall of Fame award van TIOBE Index voor de sterkst groeiende programmeertaal van dat jaar.

Tabel 3: Hall of Fame TIOBE Index

Year	Winner
2018	Python
2017	C
2016	Go
2015	Java

3.2.1.2 Google search index

Volgens Google search index die door een pagina van Pierre Carbonnelle [2] is geanalyseerd, door de zoekterm [programmeertaal] + *tutorial* in te geven, zijn de top vijf meest opgezochte programmeertalen: Python, Java, JavaScript, C# en PHP.

Bij de onderstaande lijst is te zien dat Python zich aan de top bevindt (met 26%) en gevolgd wordt door Java (21%), JavaScript (8%), C# (7%) en PHP (7%).

Tabel 4: Google Search Index TOP 5

Worldwide, Feb 2019 compared to a year ago:

Rank	Change	Language	Share	Trend
1	↑	Python	26.42 %	+5.2 %
2	↓	Java	21.2 %	-1.3 %
3	↑	Javascript	8.21 %	-0.3 %
4	↑	C#	7.57 %	-0.5 %
5	↓↓	PHP	7.34 %	-1.2 %

Het is ook te zien dat Python een stijging kent van 5% ten opzichte van vorig jaar en dat de andere programmeertalen een daling inzetten ten opzichte van vorig jaar.

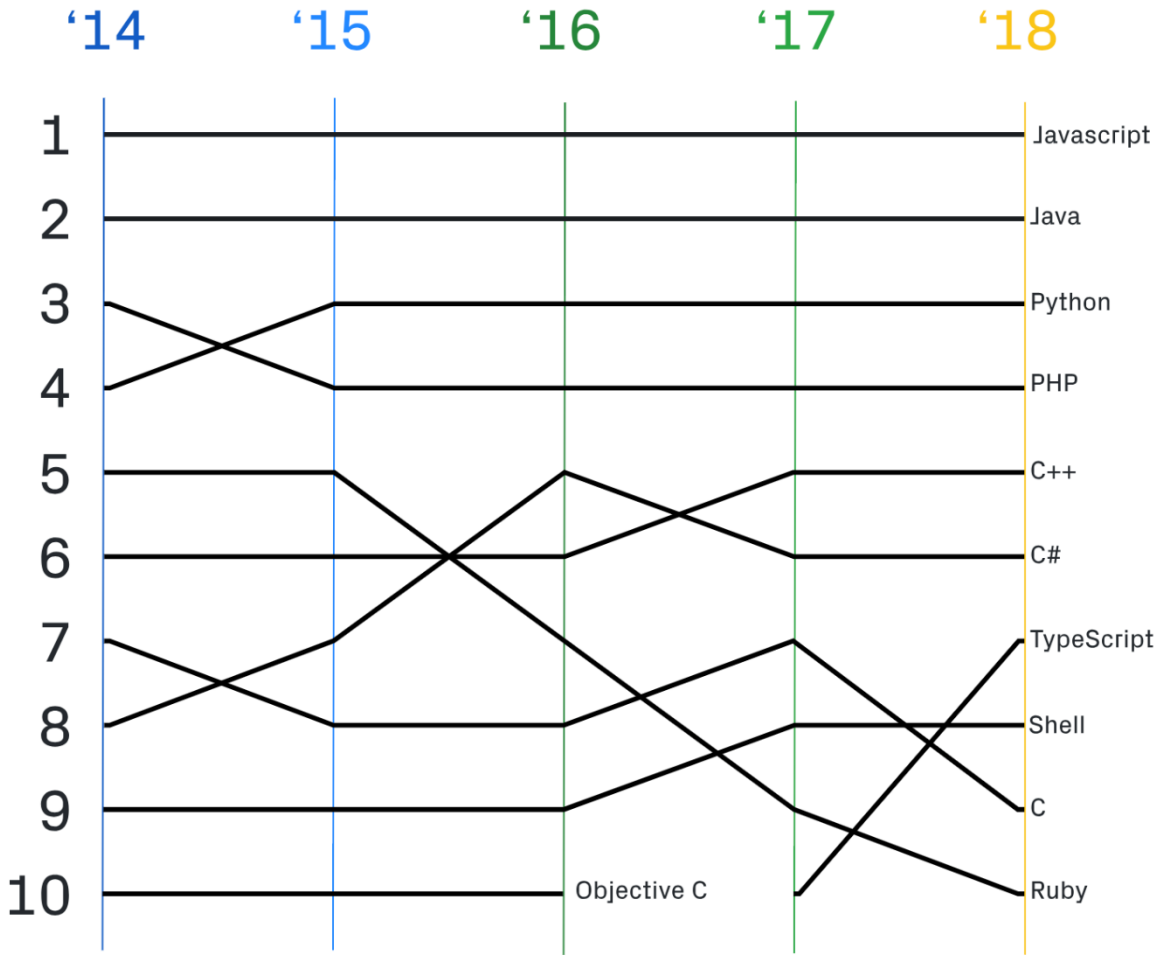
3.2.1.3 GitHub

GitHub [3] is een website waar software op kan worden geplaatst.
Op GitHub wordt opensourcesoftware geplaatst waar anderen in kunnen kijken of die anderen met elkaar kunnen delen. Ook geeft GitHub de mogelijkheid om parallel te werken aan een softwareontwikkeling.

In dit project wordt GitHub gebruiken, maar niet de onlineversie. In deze project wordt de interne GitHub gebruikt van ACA-IT Solutions om code privé te houden. Hierbij wordt BitBucket gebruikt. Dit is een subversie van GitHub.

Volgens GitHub zijn de meest gebruikte talen binnenin GitHub: JavaScript, Java, Python, PHP en C++.

Tabel 5: GitHub TOP 10



In deze lijst is te zien dat dit ongeveer overeenkomt met de TIOBE Index en de Google Search Index.

3.2.1.4 Conclusie

Bij het onderzoek is het gebleken dat Java, JavaScript, Python, C en PHP de meest gebruikte programmeertalen zijn.

- Java en Javascript blijven de grootste talen;
- Python is een sterk groeiende taal;
- C en PHP blijven een sterke taal.

Deze talen worden ook beschikbaar gesteld bij Amazon SDKs. Vervolgens gebeurt er dieper onderzoek gebeuren naar deze talen wat betreft de functionaliteit en moeilijkheid en de integratie binnenin de software.

3.2.2 Criteria

Voor het vergelijken van de SDK's is er een criteria opgesteld. Al deze SDKs worden op macOS getest met de beschikbare Amazon Web Services API-calls.

3.2.2.1 Algemeen

- Kostprijs: Is de SDK gratis of betalend? Zijn bepaalde functies alleen aanwezig in de betalende versie, etc.?
- API-calls: Bevat de SDK voldoende API-calls om het proces te automatiseren? (Procentueel voorstel)
- Installatie: Hoe makkelijk is de installatie van de benodigde software om de SDK te gebruiken?
- Gebruikte technologie: Wordt de SDK-taal gebruikt binnenin:
    - Sector?
    - Bedrijf?
    - Team?
- Toekomstgericht: Heeft de SDK een toekomst?
- Learning curve: Wat is de moeilijkheidsgraad van de SDK?

3.2.3 SDK's

3.2.3.1 JavaScript Node.js

Om deze programmeertaal te gebruiken is er geen extra kost nodig. Dit is een opensourceprogrammeertaal. Het script is in Node.js geschreven.

Figuur 33: Node.Js logo

De documentatie [4] die Amazon Web Services aanbiedt is laatst geüpdatet op 11 oktober 2016. Dit betekent dat er drie jaar lang geen extra AWS API's toegevoegd zijn. JavaScript Node.js heeft in totaal 43 API-calls om de Organization service te beheren. Verder heeft de IAM-service 138 API-calls ter beschikking.

Om Node.js te gebruiken zijn er een paar vereisten. Als eerst moet Node.js op de computer geïnstalleerd worden. Deze installatieprocedure is niet moeilijk en duurt ook niet lang.

Vervolgens is er 'NPM' bij nodig. 'NPM' zorgt ervoor dat verschillende *packages* gedownload en geïnstalleerd kan worden: dit wordt automatisch geïnstalleerd tijdens de installatie van Node.js.

Als laatste zijn de SDK-bestanden nodig van Amazon Web Services. Dit kan gemakkelijk teruggevonden worden op de website van AWS.

Voor een volledig werkend script zijn er twee packages nodig. Het installatieproces van deze *packages* is niet zo eenvoudig: er kunnen verschillende conflicten en problemen ontstaan tijdens de installatie. Dit veroorzaakt in het begin veel tijdverlies en werkuren.

Packages die gebruikt zijn:

- AWS-SDK
- Readline-sync

'AWS-SDK' package wordt gebruikt om connectie te maken met verschillende services dat Amazon Web Services aanbiedt. Zonder deze package is het niet mogelijk om de API-calls te gebruiken.

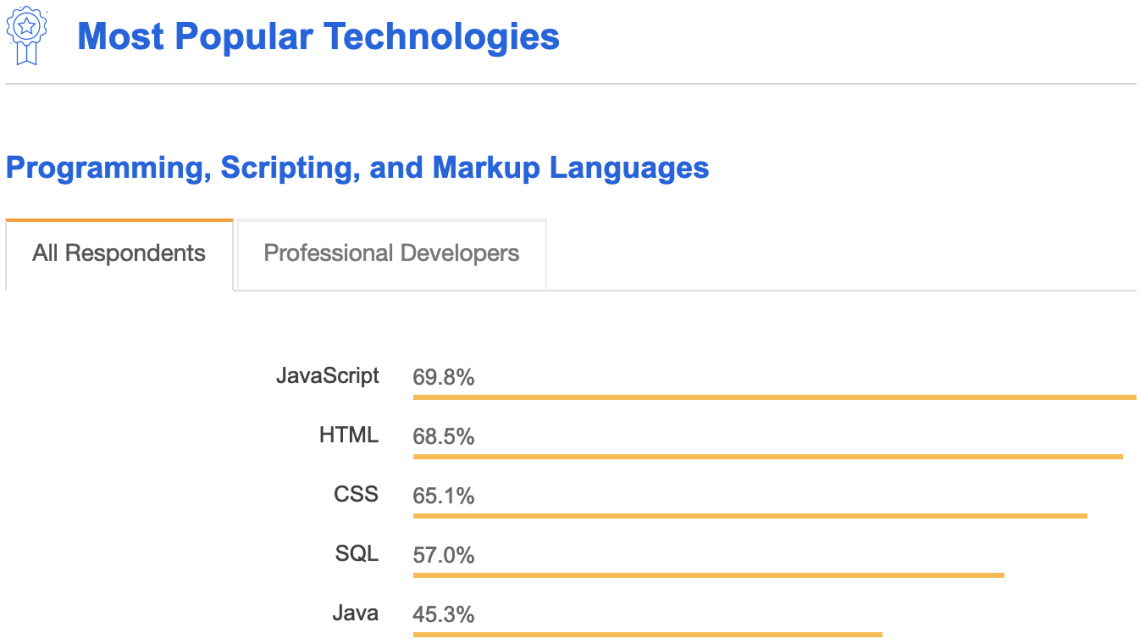
Een voorbeeld:
In figuur 34 wordt er getoond hoe er connectie gemaakt wordt met de AWS Organizations Service.

```
var organizations = new AWS.Organizations({
    apiVersion: '2016-11-28' ,
    region: 'us-east-1' ,
    accessKeyId: stsAssume.Credentials.AccessKeyId,
    secretAccessKey: stsAssume.Credentials.SecretAccessKey,
    sessionToken: stsAssume.Credentials.SessionToken
});
```

Figuur 34: JavaScript Organization code

AWS STS is een service dat verschillende rechten aan een account kan toekennen. Met deze service kunnen er verschillende accountinstellingen gewijzigd worden. Als deze service niet opgeroepen wordt kan er een *Access Denied* foutmelding tevoorschijn komen.

JavaScript speelt een belangrijke rol in de IT-sector. Bij figuur 35 is te zien dat JavaScript de meest gebruikte programmeertaal is. De onderstaande lijst komt van één van de grootste code community websites, namelijk Stack Overflow.



Figuur 35: Stack Overflow lijst

Ook ACA-IT Solutions loopt mee met de IT-sector. Het bedrijf telt meer dan 50 Java Developers in Hasselt die hard werken aan verschillinde projecten.
Het Cloud team van ACA-IT Solutions heeft ook lichte ervaring met JavaScript, maar zijn geen experten. Team zelf weet de basis van JavaScript.

Omdat de taal zo veel wordt gebruikt en nog steeds een verhoging heeft in populariteit is deze taal toekomstgericht. Acht op de tien nieuwe projecten worden op JavaScript gecodeerd. Dit maakt de toekomst van de SDK sterk. Hierdoor gaat Amazon Web Services ondersteuning blijven geven bij het behandelen van API-calls via JavaScript.

Voor een nieuwe programmeur die nooit Javascript heeft gebruikt is dit een moeilijke taal. In het begin moet er veel opzoekwerk gepresteerd worden om zo stuk voor stuk de code te schrijven.

Conclusie JavaScript

JavaScript speelt een belangrijke rol in de IT-wereld en beschikt over een grote community die veel ondersteuning aanbiedt. De documentatie van Amazon Web Services om SDK te gebruiken is goed verduidelijkt en gedocumenteerd.

Het gebruik van JavaScript is niet zo eenvoudig, wie nog nieuw is in de programmeerwereld wordt JavaScript niet aangeraden als eerste programmeertaal. Het installatieproces van de nodige packages zijn niet eenvoudig. Er kunnen verschillende problemen ontstaan tijdens het binnenhalen en verwerken van de *packages*.

3.2.3.2 PHP

Figuur 36: PHP-logo

PHP is een gratis programmeertaal die vaak gebruikt wordt bij het creëren van dynamische websites. De documentatie die AWS [5] aanbiedt is erg gedetailleerd, de documentatie zelf bevat 42 API-calls die Organizations service beheren, en 137 API-calls om IAM-service te beheren.

Het SDK dat PHP ondersteunt is als laatst op 28 November 2016 geüpdatete. Na deze datum is er geen nieuwe API-call aangemaakt of een wijziging in de API-call verricht.

Om PHP te gebruiken is er PHP-software nodig, vaak wordt deze software automatisch geïmplementeerd in het *operating system*. Als dit toch niet beschikbaar wordt gesteld door de OS (Operating System) kan de PHP-software gemakkelijk afgehaald en geïnstalleerd worden. Vervolgens zijn de SDK-bestanden nodig voor het gebruiken van de PHP API-calls, dit kan gemakkelijk terug te vinden zijn op de officiële website van Amazon Web Services.

Bij het gebruiken van de API-calls moet er als eerst een website gecreëerd worden om verschillende unieke parameters door te sturen. De computer moet ook geconfigureerd worden om PHP-scripts uit te voeren. Als deze configuratie niet correct geconfigureerd is, gaat de PHP-script niet werken en gaat de API-call niet doorgestuurd worden.

Een voorbeeld:

```
$accountCreated = $client->createAccount([
    'AccountName' => $_POST["name"],
    'Email' => $_POST["email"],
    'IamUserAccessToBilling' => 'DENY',
    'RoleName' => 'OrganizationAccountAccessRole'
]);
```

Figuur 37: PHP createAccount functie

Bij dit voorbeeld wordt er getoond hoe een nieuw account aangemaakt wordt: er wordt een API-call aangeroepen om een account aan te maken. Vervolgens worden er verschillende parameters ingevuld en verstuurd naar AWS om op deze manier een account aan te maken. Bij een foutieve parameter wordt er een error melding doorgestuurd die vervolgens op het scherm getoond wordt.

PHP is een belangrijk en een populair programmeertaal om een website te creëren, deze taal wordt dagdagelijks gebruikt om verschillende functies aan te roepen en uit te voeren. Hierdoor heeft PHP een grote waarde en functionaliteit in de IT-wereld.

Toch wordt er binnenin ACA-IT Solutions heel weinig met PHP gewerkt. ACA-IT Solutions gebruikt andere programmeertalen om dynamischere websites te creëren. Het bedrijf zelf gebruikt onder andere Java en JavaScript. Deze talen worden ook gebruikt bij andere projecten. Dit maakt de taal gemakkelijker te gebruiken binnenin het bedrijf. In het team wordt er zelden tot nooit met PHP gewerkt.

Figuur 38: Tabel afbeelding

PHP is uitgegroeid tot een algemene programmeertaal om dynamische web interfaces te creëren. Bij het aanleren en uitoefenen van het creëren van dynamische interfaces wordt er meestal PHP als eerste programmeertaal gekozen. Maar dit wilt niet zeggen dat PHP de leider gaat blijven. PHP wordt elk jaar minder en minder gebruikt omdat het een oud programmeertaal is. Toch krijgt PHP dikwijls nieuwe updates om toch nog op de markt te blijven.

Het aanleren van de programmeertaal is niet zo moeilijk maar ook niet gemakkelijk. Om deze taal te gebruiken moet er ook HTML (*Hypertext Markup Language*) /CSS (*Cascading Style Sheets*) aangeleerd worden. Bij deze talen worden er vaak fouten gemaakt waardoor de website niet meer als gewenst wordt getoond. Eén verkeerde code die geschreven is kan de volledige website vernielen. Om een dynamische website aan te maken moeten deze programmeertalen samen gebruikt worden met PHP.

Conclusie PHP

Het gebruik van de programmeertaal is niet zo moeilijk maar er is voorkennis nodig om deze taal te gebruiken bij het creëren van een dynamische website. De documentatie van het SDK op Amazon Web Services werd grondig uitgelegd met verschillende voorbeelden. Hoewel PHP een belangrijke rol speelt in de IT-wereld wordt er binnen het bedrijf heel weinig PHP gebruikt, en dit maakt de waarde van de SDK binnen in het bedrijf heel laag.

3.2.3.3 Python

Figuur 39: Python-logo

Het gebruik van deze programmeertaal is volledig gratis, ook bevat Python geen uitbreidbare betalende software. Amazon Web Services biedt verschillende API-call mogelijkheden in verschillende services. In de Organization service biedt de SDK 46 unieke API-calls en in de IAM-service 141 unieke API-calls. De documentatie [6] die AWS aanbiedt wordt regelmatig geüpdatet.

De installatie van Python is zeer eenvoudig maar er zijn verschillende benodigdheden bij vereist. Als eerst moet PIP3 geïnstalleerd worden. Met PIP3 kunnen er Python packages geïnstalleerd worden. De installatie is vrij eenvoudig en duurt ook niet lang.

Vervolgens is Boto3 nodig. Met Boto3 kunnen er API-calls uitgevoerd worden aan AWS. De benodigde bestanden voor de SDK zijn terug te vinden op de website van Amazon Web Services.

Als laatst moet AWSCLI geïnstalleerd worden. Deze package zorgt ervoor dat er connectie gemaakt wordt met Amazon Web Services via tokens die verkrijgbaar zijn via de AWS-account. Als deze tokens niet geconfigureerd zijn kan er geen connectie gemaakt worden en gaat het script niet werken.

Bij de onderstaande code is het te zien dat er een connectie op stand wordt gebracht met de Organization service, met de verkregen AWS-tokens van Amazon Web Services.

```
client = boto3.client('organizations',
    aws_access_key_id=responseCredentials['Credentials']['AccessKeyId'],
    aws_secret_access_key=responseCredentials['Credentials']['SecretAccessKey'],
    aws_session_token=responseCredentials['Credentials']['SessionToken'])
```

Figuur 40: Python Organization code

Als er een succesvolle connectie op stand gebracht wordt, kunnen de API-calls aanroepen worden om een account aan te maken.

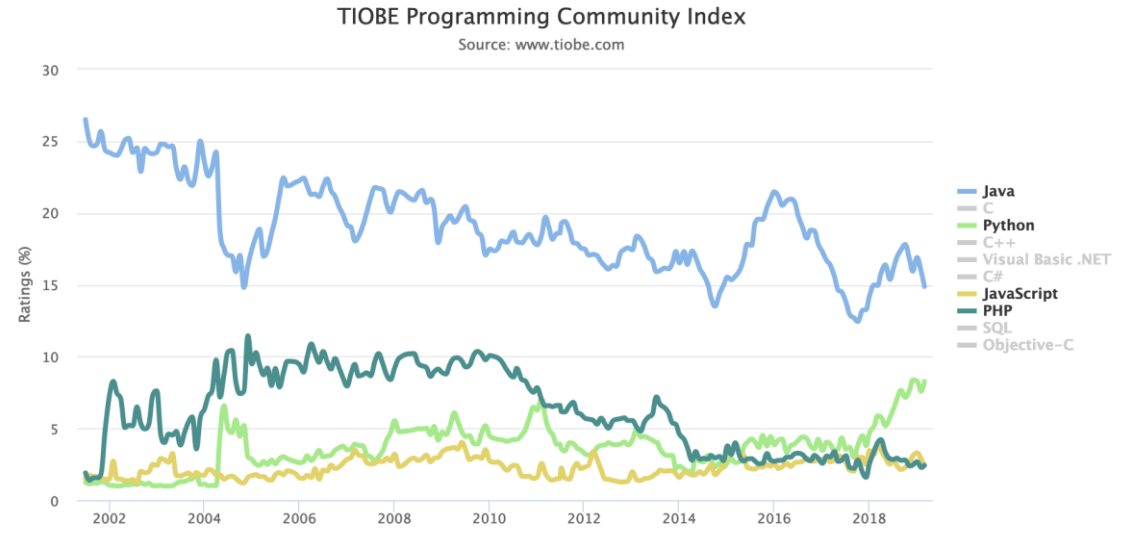
```
CreateAccount = client.create_account(
    Email=email,
    AccountName=accountName,
    RoleName='OrganizationAccountAccessRole',
    IamUserAccessToBilling='DENY'
)
```

Figuur 41: Python createAccount code

Python is momenteel een sterk groeiende programmeertaal binnen in de programmeer community en dit veroorzaakt dat Python een grote waarde heeft in de IT-wereld. Door zijn sterke groei is Python verkozen tot de beste programmeertaal van 2018.

Bij de onderstaande grafiek is het te zien dat Python een sterke groei heeft ten opzichte van JavaScript, PHP en Java.

Tabel 6: Python vergelijking



ACA-IT Solutions loopt mee met deze groei, ACA-IT Solutions beschikt over geavanceerde Python programmeurs.
Er worden verscheidene projecten opgestart met als programmeertaal Python. Ook beschikt de Cloud team over voldoende kennis en vaardigheid om een script te schrijven in Python.

Python is een zeer toekomstgerichte programmeertaal. Omdat het een sterke groei heeft, worden er meer en meer projecten gestart in Python. Ook beschikt Python over een grote community. Hierdoor gaat Amazon Web Services ondersteuning blijven geven.

Python is verkozen tot makkelijkste programmeertaal. Het aanleren en gebruiken van de taal is zeer eenvoudig. De geschreven code is een leesbare code. Dit wil zeggen dat iemand die nooit heeft geprogrammeerd gemakkelijk de code kan lezen en begrijpen.

Conclusie Python

Python is een gemakkelijke programmeertaal dat verschillende functies aanbiedt. Dankzij de sterke groei beschikt Python aan een grote community en speelt Python een belangrijke rol in de IT-wereld. Hierdoor kunnen de problemen sneller opgelost en teruggevonden worden op het internet. Voor een nieuwe programmeur dat nooit heeft geprogrammeerd is dit taal een goed begin punt. De documentatie dat AWS aanbiedt is grondig uitgelegd en gedocumenteerd.

3.2.3.4 Java

Java is het populairste en meest gebruikte programmeertaal in de IT-wereld. Om Java te gebruiken is er geen betalende software nodig. Er zijn wel betalende softwarepakketten die Java ondersteunen om in Java te programmeren het vergemakkelijken.

Amazon Web Services biedt een geavanceerde documentatie [7] over Java SDK. De laatste update van de Organizations services was op 28 november 2016. De Organization service bevat 43API-calls en de IAM-service bevat 364 API-calls.

Figuur 42: Java-logo

Het gebruik van Java is niet eenvoudig, er moeten verschillende benodigdheden geïnstalleerd worden. Als eerst moet er Java geïnstalleerd worden op onze OS. Deze software kan door de officiële website [8] gemakkelijk afgehaald en geïnstalleerd worden. Bij dit onderzoek werd Eclipse Tool Kit gebruikt als Java IDE-development. Om AWS SDK te gebruiken in Eclipse Tool Kit zijn er AWS SDK-bestanden nodig, deze bestanden zijn terug te vinden op de officiële website van AWS. Vervolgens moeten er verschillende configuraties uitgevoerd worden om een connectie op stand te brengen tussen AWS en het OS.

Als de nodige software geïnstalleerd is kan het script geschreven worden.

Java wordt dagdagelijks gebruikt in verschillende applicaties: Banking, Android, Big Data, Retail, etc. Java speelt een hele grote rol in de IT-wereld. Ook speelt Java een grote rol in het bedrijf. ACA-IT Solutions focust zich vooral op Javaconsultancy en de implementatie van Javaprojecten op een Agile manier. Het Cloud team beschikt over voldoende kennis en vaardigheid om een script of applicatie te schrijven in Java.

Java is momenteel het meest gebruikte programmeertaal op de markt, en dit maakt de toekomst van de programmeertaal zeer sterk. Ook worden grote softwarepakketten zoals Android, Blue Ray, Minecraft, etc. met Java geschreven.

Een script schrijven in Java is niet zo gemakkelijk. Om in Java in een vloeiende manier te schrijven dient er veel tijd in gestoken te worden. Ook moet er veel onderzoek en kennis opgedaan worden. Bij

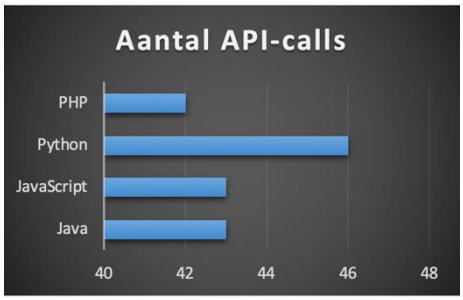
het begin was het moeilijk om het script te schrijven, maar na een paar keer proberen en informatie op te zoeken is het uiteindelijk gelukt en is de script volledig geschreven.

Conclusie Java

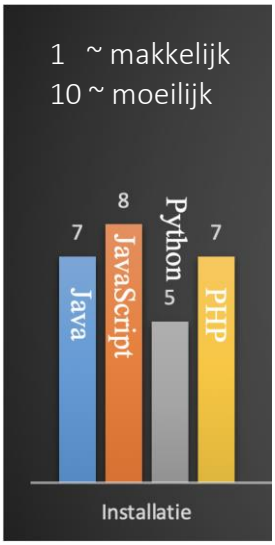
Java is een populaire taal binnen het bedrijf en ook in de IT-wereld. De populariteit geeft de mogelijkheid om snel een oplossing te vinden op het internet. De installatie en de configuratie van Java is eenvoudig, maar het gebruik van Java is niet zo eenvoudig voor wie nooit heeft geprogrammeerd. Het gebruik van AWS SDK is in het algemeen gemakkelijk.

3.2.3.5 Conclusie

Het SDKs dat onderzocht en getest zijn, zijn allemaal gratis. Er zijn geen extra betalende tools nodig om het SDK te gebruiken.



In het algemeen wordt er hetzelfde aantal API-calls ter beschikking gesteld om verscheidene functies te gebruiken. In grafiek 7 is het te zien dat Python SDK een kleine voorsprong heeft op de andere SDKs.



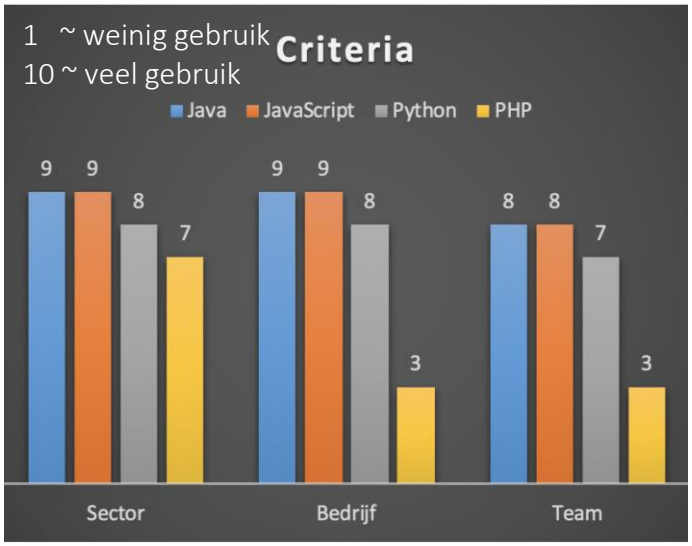
Tabel 7: Aantal API-calls

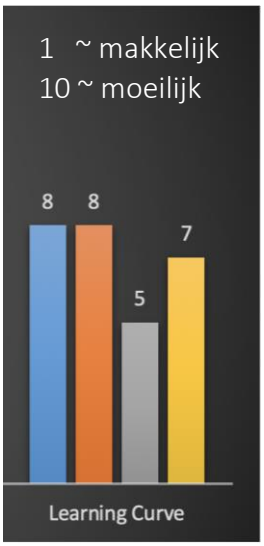
Bij het installeren van de Java en JavaScript SDKs kwamen er verschillende keren problemen voor. Er moesten verschillende componenten geïnstalleerd worden om de SDK volledig functioneel te maken.

De installatie van PHP was in het algemeen gemakkelijk omdat de software al in het OS geïntegreerd is. Maar er moesten configuratie aanpassingen gebeuren om het SDK volledig te kunnen gebruiken.

De installatie en configuratie van Python SDK was gemakkelijk en snel. Er werden geen extra vereiste packages of configuraties uitgevoerd om de SDK te gebruiken.

Algemeen worden de programmeertalen evenveel gebruikt in de IT-wereld. Omdat ACA-IT Solutions een Javaconsultancy bedrijf is, wordt er heel veel met Java geprogrammeerd gevolgd door Python. PHP wordt door het bedrijf niet ondersteund.





De *learning curve* van Java, JavaScript en PHP liggen hoog. Deze drie programmeertalen zijn niet gemakkelijk te gebruiken. Je hebt goede voorkennis nodig voordat er een script kan geschreven worden. Python, in tegendeel, is een makkelijke programmeertaal. Het aanleren en het gebruiken van deze taal is niet moeilijk.

4 Integratie Jira

In dit hoofdstuk worden de verschillende implementatiemogelijkheden ingelicht.

Voor het vergelijken van de plug-ins is er een criteria opgesteld. Al deze plug-ins worden in de testomgeving getest.

4.1 Criteria

- Prijs: is het een gratis plug-in?
- Geschikt voor de opdracht?
- Complexiteit: is het plug-in gemakkelijk te configureren?

4.1.1 Jenkins plu-gins

4.1.1.1 Jira Trigger

Jira Trigger plug-in is een gratis plug-in dat geïnstalleerd kan worden via de Jenkins plug-in portaal. Deze plug-in zorgt ervoor om binnen komende API-calls te identificeren en vervolgens acties uit te voeren.

Met deze plug-in wordt er een build gestart wanneer er een commentaar geschreven wordt op het issue of wanneer de issue van status wijzigt. Als de build succesvol uitgevoerd wordt kan er een commentaar geschreven worden bij het issue. Ook geeft de plug-in de mogelijkheid om *customfields* te parametreren, dit is handig omdat er 3 nieuwe *customfields* toegevoegd werd aan de Jira issue.

Er moet één keer een globale configuratie gebeuren zodat de plug-in inkomende API-calls kan waarnemen en uitvoeren. Het opzetten van build parameters, build configuraties zijn eenvoudig, via JQL kunnen de API-calls gefilterd worden. Dit zorgt ervoor dat de build niet elke inkomende API-calls als actie gaat waarnemen.

Figuur 43: JQL Jira Trigger

Conclusie

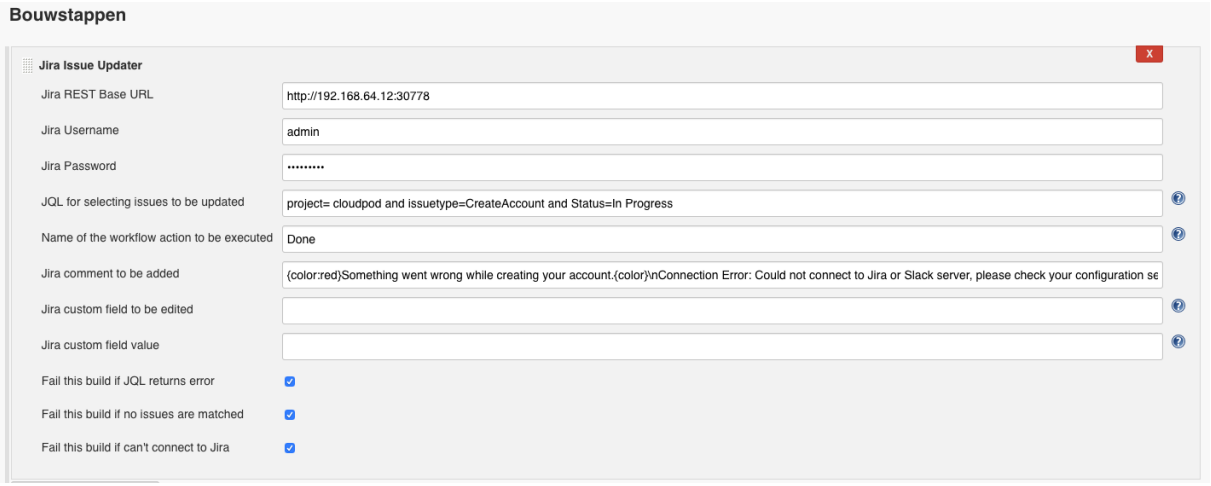
Gratis en gemakkelijk te configureren en kan gebruikt worden bij deze opdracht.
Deze plug-in is geschikt voor het gebruik bij deze opdracht.

4.1.1.2 Jira Issue Updater

Jira Issue Updater is een gratis plug-in dat geïnstalleerd kan worden via de Jenkins-portaal.

Het doel van de plug-in is om een Jira issue te updateten. Er kan geen velden ingelezen worden dit is een min punt voor de plug-in. Toch kan deze plug-in gebruikt worden op het einde van de build om een issue te updateten. Na het installeren hoef je geen configuratie te maken aan de plug-in, het kan direct gebruikt worden bij een build.

Bij elk nieuw build wordt de connectie gegevens en de issue gegevens ingevuld.



De plug-in heeft een **ernstig securityprobleem**. De plug-in slaat het wachtwoord van de Jira user in een leesbaar tekst in zijn *config* bestand. Hierdoor kan dit plug-in niet in de productie omgeving gebruikt worden.

Jira Issue Updater Plugin stores credentials in plain text

SECURITY-837 / CVE-2019-1003054

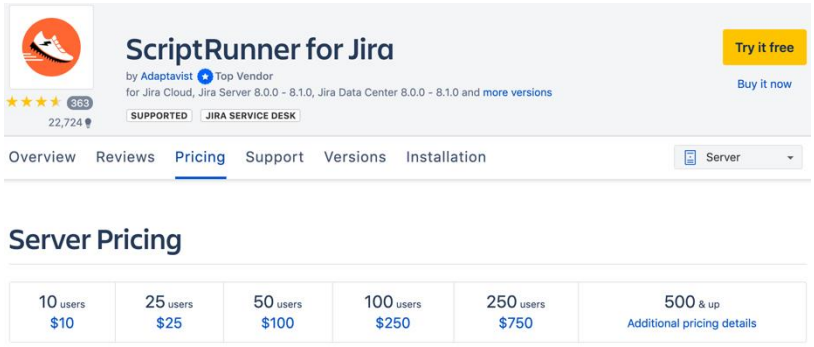
Jira Issue Updater Plugin stores credentials unencrypted in job `config.xml` files on the Jenkins master. These credentials can be viewed by users with Extended Read permission, or access to the master file system.

<u>Conclusie</u>

Deze plug-in kan gebruikt worden met samenwerking met andere plug-ins. Maar omdat deze plugin een securityprobleem heeft, kan deze plugin **nooit** in de productie omgeving gebruikt worden. Deze plug-in is niet geschikt voor deze opdracht.

4.1.2 Jira plug-ins

4.1.2.1 ScriptRunner



Plug-in is niet gratis aan te schaffen, er is een 30 dagen probeer tijd, na dat deze tijd is verlopen moet de plug-in aangeschaft worden. Prijzen verschillen tussen de versies van de software die geïnstalleerd zijn: Server, Cloud en Datacenter.

Figuur 44: Prijs ScriptRunner

Figuur 45:Prijs Jenkins IntegrationFiguur 46: Prijs ScriptRunner

De geïnstalleerde versie van de testomgeving is Server, hier worden de prijzen per aantal gebruikers berekent.

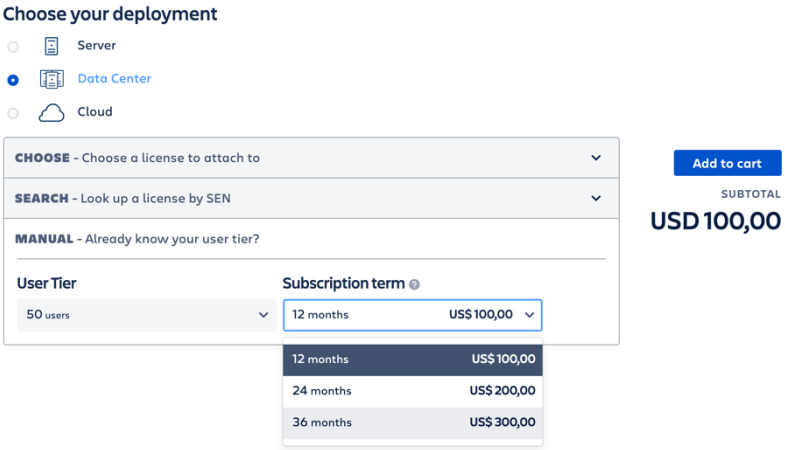
Via deze plug-in kan er vooraf geschreven scripts gerund worden. Het taal dat de plug-in ondersteunt is Groovy. Met deze plug-in kunnen er verschillende opdrachten door gevoerd worden, er kunnen issues geüpdatet, gewijzigd of verwijderd worden. Ook kan er een API-call gestuurd worden naar AWS. Maar omdat de opdracht het doel heeft om over Jenkins te werken heeft deze plug-in een minder waarden.

Een ander optie is dat er een API-call gestuurd kan worden vanuit de plug-in naar het Jenkins build en zo een build uitgevoerd kan worden. Maar dit gaat een dubbel werk en onnodige kosten verrichten. Jira heeft een *build-in* webhook systeem dat API-calls stuurt naar Jenkins of andere *sources*.

Plug-in bevat een uitgebreide documentatie met verschillende voorbeelden, toch is het schrijven van het script voor de plug-in niet zo eenvoudig. Er moeten verschillende packages geïnstalleerd en verwijst worden vanuit de plug-in.

<u>Conclusie:</u>
Deze plug-in is geschikt voor het aansturen van API-calls naar AWS, maar omdat het dubbel werk en onnodige kosten verricht geeft dit plug-in geen extra waarden om te gebruiken.

4.1.2.2 Jenkins Integration for Jira



De plug-in is gratis te installeren voor het server versie. Voor het Data Center versie hangt de prijs af van de aantal gebruikers en de duur van de subscriptie. Met deze plug-in kunnen er *builds* op opgevraagd en visueel voorgesteld worden. Deze plug-in is niet geschikt voor de opdracht omdat de plug-in geen Jenkins *build*/pipeline aanroept.

Figuur 47:Prijs Jenkins Integration

Figuur 48:Prijs Jenkins Integration

Met deze plug-in kan er gezien worden of de *build*/pipeline succesvol is afgehandeld. Deze plug-in geeft niet de mogelijkheid om de gewenste verrichtingen door te voeren.

Het configureren en installeren is vrij simpel, bij het gebruik van de installatie wizard wordt alles stap voor stap getoond. Ook beschikt de plug-in van een uitgebreide documentatie.

<u>Conclusie:</u>
Deze plug-in is niet geschikt voor deze opdracht.
Toch kan deze plug-in gebruikt worden bij het tonen van de Jenkins *build*/pipeline status. Doordat deze plug-in geïntegreerd is in Jira hoeft er niet van omgeving veranderd te worden om te zien of de *build* gefaald of succesvol afgerond is.

4.1.3 Conclusie

Naam	Platform	Prijs	Geschikt	Complexiteit	Security
Jira Trigger	Jenkins	Gratis	Ja	Makkelijk	In orde
Jira Issue Updater	Jenkins	Gratis	Nee	Makkelijk	Laag
ScriptRunner	Jira	Betalend	Nee	Moeilijk	In orde
Jenkins Intergration	Jira	Betalend	Nee	Makkelijk	In orde

Jira *Trigger* en Jira Issue *Updater* kunnen bij deze opdracht gebruikt worden maar omdat Jira Issue *Updater* een security lek heeft kan deze plug-in in een bedrijf omgeving niet gebruikt worden.

ScriptRunner kan bij deze opdracht gebruikt worden, toch gaat deze plug-in niet gebuikt worden omdat de nodige functies binnen Jira al bestaan. Dit gaat alleen dubbel werk en extra kosten verrichten. Jenkins Intergration is geen geschikte plug-in, deze plug-in gaat geen extra waarden opleveren en het opdracht niet vergemakkelijken.

5 Literatuurstudie

Onderzoek zoals verricht door Joseph K. Ziegler [9] (2014) heeft aangetoond dat de keuze van de juiste SDK te maken heeft met het gebruik en voorkennis van de programmeertaal.
Tijdens het onderzoek is gebleken dat er verschillende beïnvloedingsfactoren zijn. Dat de keuze voor het juiste AWS SDK programmeertaal uit drie factoren bestaat. Deze zijn: *Language Characteristics , Problem Domain* en *Local Ecosystem*.

Language Characteristics:

Wanneer er voor Cloud gericht opdracht ontwikkeld wordt, richt op interpretatieve, dynamische en open-source programmeertalen om voor een snelle en kosteneffectieve ontwikkeling te bekomen. Voor bedrijven met security vereisten of voor integraties met oudere omgevingen kan gecompileerde talen gebruikt worden.

Problem Domain:

Een voor onderzoek doen op het zelefde probleem of doel om deze opdracht uit te voeren, en kijken of er reeds oplossingen bestaan en of deze overgenomen kan worden.

Local Ecosystem:

Richt op het taal dat het meest in het bedrijf wordt gebruikt. Een onderzoek doen op ervaring en kennis van de werknemers binnen het bedrijf om zo de juiste programmeertaal te kiezen, en vervolgens voor een lager budget deze SDK te gebruiken. Dit speelt een belangrijk rol als er bijvoobeeld een nieuwe taal moet aangeleerd worden, dit gaat het kosten en fout margen vergroten.

Conclusie

Python is het meest geschikt programmeertaal voor deze opdracht, Java en JavaScipt kan gebruikt worden binnen het bedrijf maar omdat deze talen complexe talen zijn en omdat deze weinig gebruikt worden bij het team worden deze talen niet gebruikt bij het opdracht, ook heeft heet team weinig ervaring en kennis op deze talen. PHP wordt binnen het bedrijf niet gebruik wat het gebruik van deze taal voor deze opdracht helemaal elimineert. Toch kan PHP gebruikt worden voor een ander bedrijf omdat het installatie en het gebruik gemakkelijker is dan Java en JavaScript.

Het onderzochte Jenkins integraties voor Jira-platform zal niet gebruikt worden omdat deze niet geschikt zijn voor het opdracht, en bij een gebruik dubbel werk en extra kostte zal verrichten. Jenkins integratie for Jira is een goede plug-in om Jenkin-builds te visualiseren. Het geeft een breed schets van de build, maar het heeft geen extra voordeel voor deze opdracht. Met ScriptRunner kan er groovy scripts uitgevoerd worden. Ook kan er een Jenkins-Build aangeroepen worden. Maar omdat dit al met een Jira Webhook kan gedaan worden zonder extra kost, zal deze script onnodige kosten verrichten.

Jira Trigger is een geschikte plug-in voor deze opdracht, het plug-in kan Jira Webhook waarnemen en een Jenkins-build starten. Deze plug-in werd bij het bedrijf op productieomgeving gebuikt.

Jira Issue Updater en Jira Trigger plug-ins zijn praktisch identiek in werking en passen goed in de opdracht. Alleen heeft de Jira Issue Updater een security lek, deze plug-in slaat Jira gebruiker en wachtwoord in een leesbaar tekst in zijn configuratie bestand op, wat het gebruik onmogelijk maakt.

Bibliografie

[1]	T. Index, „Tiobe Index,” Tiobe, [Online]. Available: https://www.tiobe.com/tiobe-index/. [Geopend 03 2019].
[2]	P. Carbonnelle, „PopularitY of Programming Language,” Github, [Online]. Available: http://pypl.github.io/PYPL.html. [Geopend 03 2019].
[3]	GitHub, „Github Index,” GitHub, 2018. [Online]. Available: https://octoverse.github.com/projects#languages. [Geopend 03 2019].
[4]	Amazon, „JavaScript SDK API,” Amazon, 28 Oktober 2016. [Online]. Available: https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/Organizations.html. [Geopend 03 2019].
[5]	Amazon, „PHP SDK API,” Amazon, 28 November 2016. [Online]. Available: https://docs.aws.amazon.com/aws-sdk-php/v3/api/. [Geopend 03 2019].
[6]	Amazon, „Python SDK API,” Amazon, [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/index.html. [Geopend 27 02 2019].
[7]	Amazon, „Java SDK API,” Amazon, 2019. [Online]. Available: https://aws.amazon.com/sdk-for-java/. [Geopend 03 2019].
[8]	Oracle, „Oracle Java SE,” Oracle, [Online]. Available: https://www.oracle.com/technetwork/java/javase/downloads/index.html. [Geopend 04 2019].
[9]	J. K. Ziegler, „Choosing the Right Programming Language for your Startup,” Amazon, 07 November 2014. [Online]. Available: https://aws.amazon.com/blogs/startups/choosing-the-right-programming-language-for-your-startup/. [Geopend 04 2019].
[10]	A. Documentation, „SDK Docs Amazon,” Amazon, 2019. [Online]. Available: https://docs.aws.amazon.com/index.html#lang/en_us. [Geopend 02 2019].
[11]	D. M. J. Garbade, „Top 3 most popular programming languages in 2018,” Hackernoon, 30 augustus 2018. [Online]. Available: https://hackernoon.com/top-3-most-popular-programming-languages-in-2018-and-their-annual-salaries-51b4a7354e06. [Geopend 04 2019].

Bijlagen

A. **Python script (testomgeving) om AWS-accounts te creëren**

B. **Omschrijving Bijlage B**

C. **Omschrijving Bijlage C**

A. Python script (testomgeving) om AWS-accounts te creëren

```
import boto3
from botocore.exceptions import ClientError
import sys
from time import sleep

if __name__ == "__main__":
    accountName = str(sys.argv[1])
    email = str(sys.argv[2])
    mfa = str(sys.argv[3])
    destination = int(sys.argv[4])

# Variables
roleOrganizationManager = 'xxx'
sessionNameRole = 'mySession'
authSerialNumber = 'xxx'
rootOrganizationUnit = 'xxx

# Add policies in this list to grant policies to accounts MAX 5 policies per account
policies = ["p-xxx", "p-xxx", "p-xxx"]
# Add or change Organization unit IDs
organizationUnit = ["ou-xxx-xxx", "ou-xxx-xxx", "ou-xxx-xxx", "ou-xxx-xxx"]

regions = []


def attach_policies(alreadyCreated):
    # Adding policies to the Account
    print("Attaching Policies: Started! ")

    if alreadyCreated == True:
        while True:
            for i in policies:
                try:
                    AddPolicyAccount = client.attach_policy(
                        PolicyId=i,
                        TargetId=accountId)
                    print("-------- Policy: "+i + " :")
                    print(AddPolicyAccount)
                    print("-----------------")
                except ClientError as e:
                    print("Policie id "+ i +": already attached! ")
                    if not e.response['Error']['Message'] == "A policy with the specified name and type
already exists.":
                        print("Error: " + e.response['Error']['Message'])
                        print("-----------------")

            break
    else:
        try:
            for i in policies:
                AddPolicyAccount = client.attach_policy(
```

```
                PolicyId=i,
                TargetId=accountId)
            print(AddPolicyAccount)
            print("----------------")
        except ClientError as e:
            print("Error: " + e.response['Error']['Message'])

    print("Attaching Policies: Success! ")
    print("----------------")
    print("----------------")


def change_ou(alreadyCreated):
    # Changing Organization unit from root to desired ou
    print("Attaching OU: Started!")
    if alreadyCreated == True:
        while True:
            try:
                response = client.move_account(
                    AccountId=accountId,
                    SourceParentId=rootOrganizationUnit,
                    DestinationParentId=organizationUnit[destination - 1]
                )
                print("-------- OU: "+ organizationUnit[destination - 1] + " :")
                print(response)
            except ClientError as e:
                print("----------------")
                print("Attaching OU: already attached!")
                if not e.response['Error']['Message'] == "You specified an account that doesn't exist.":
                    print("Error: " + e.response['Error']['Message'])
            break
    else:
        response = client.move_account(
            AccountId=accountId,
            SourceParentId=rootOrganizationUnit,
            DestinationParentId=organizationUnit[destination - 1]
        )
        print(response)
        print("----------------")

    print("Attaching OU: Success!")
    print("----------------")
    print("----------------")

def change_role(responseIn, accountId, alreadyCreated):
    print("Change Alias: Started!")

    secondsRemaining = 120

    while True:
        try:
            # Changing to Security Token Service
            sts_client = boto3.client('sts',
```

```
                    aws_access_key_id=responseIn['Credentials']['AccessKeyId'],
                    aws_secret_access_key=responseIn['Credentials']['SecretAccessKey'],
                    aws_session_token=responseIn['Credentials']['SessionToken'])

    # Changing from role
    assumeRole = sts_client.assume_role(
        RoleArn='arn:aws:iam::' + accountId + ':role/xxx,
        RoleSessionName=sessionNameRole)

    credentials = assumeRole['Credentials']

    ec2_client = boto3.client('ec2',
                    aws_access_key_id=credentials['AccessKeyId'],
                    aws_secret_access_key=credentials['SecretAccessKey'],
                    aws_session_token=credentials['SessionToken'])

    regions = ec2_client.describe_regions()

    # Changing to IAM service
    iamuser = boto3.client('iam',
                    aws_access_key_id=credentials['AccessKeyId'],
                    aws_secret_access_key=credentials['SecretAccessKey'],
                    aws_session_token=credentials['SessionToken'])

    # Change account allias link to accountName
    response = iamuser.create_account_alias(
        AccountAlias=accountName
    )
    print("---Alias changing to: "+ accountName + " :")
    print(response)

    break
  except ClientError as e:
    sleep(10)
    secondsRemaining = secondsRemaining - 10
    if e.response['Error']['Message'] == "The account alias " + accountName + " already exists.":
        print(e.response['Error']['Message'])
        break
    if secondsRemaining == 0:
        print("Error: " + e.response['Error']['Message'])
        break
print("----------------")
print("Change Alias: Success!")
print("----------------")
print("----------------")
print("Delete VPC: Started!")
print("----------------")

try:
  for regions_id in regions['Regions']:
    ec2_client_regions = boto3.client('ec2',
                        region_name=regions_id["RegionName"],
                        aws_access_key_id=credentials['AccessKeyId'],
```

```
                aws_secret_access_key=credentials['SecretAccessKey'],
                aws_session_token=credentials['SessionToken'])

responseVPC = ec2_client_regions.describe_vpcs()
responseSubnets = ec2_client_regions.describe_subnets()
responseSG = ec2_client_regions.describe_security_groups()
responseACL = ec2_client_regions.describe_network_acls()
responseIG = ec2_client_regions.describe_internet_gateways()
responseRouteTable = ec2_client_regions.describe_route_tables()

if not responseVPC["Vpcs"]:
    print("-" +regions_id["RegionName"] + " has no VPC.")
else:
    for vpc in responseVPC["Vpcs"]:
        if not responseSubnets["Subnets"]:
            print("List is empty Subnets")
        else:
            for i in responseSubnets["Subnets"]:
                response = ec2_client_regions.delete_subnet(
                    SubnetId=i["SubnetId"]
                )

        if not responseIG["InternetGateways"]:
            print("There is no Internet Gateway Resources")
        else:
            for i in responseIG["InternetGateways"]:
                response = ec2_client_regions.detach_internet_gateway(
                    InternetGatewayId=i["InternetGatewayId"],
                    VpcId=vpc["VpcId"]
                )
                response = ec2_client_regions.delete_internet_gateway(
                    InternetGatewayId=i["InternetGatewayId"]
                )

        if not responseRouteTable["RouteTables"]:
            print("List is empty RouteTables")
        else:
            for i in responseRouteTable["RouteTables"]:
                for x in i["Associations"]:
                    if x["Main"] == False:
                        response = ec2_client_regions.delete_route_table(
                            RouteTableId=x["RouteTableId"]
                        )

        if not responseACL["NetworkAcls"]:
            print("List is empty NetworkAcls")
        else:
            for i in responseACL["NetworkAcls"]:
                if i["IsDefault"] == False:
                    response = ec2_client_regions.delete_network_acl(
                        NetworkAclId=i["NetworkAclId"]
                    )
```

```
                if not responseSG["SecurityGroups"]:
                    print("List is empty SecurityGroups")
                else:
                    for i in responseSG["SecurityGroups"]:
                        if not i["GroupName"] == "default":
                            response = ec2_client_regions.delete_security_group(
                                GroupId=i["GroupId"]
                            )

                responseVpcDelete = ec2_client_regions.delete_vpc(
                    VpcId=vpc["VpcId"]
                )

                print("Region: " + regions_id["RegionName"] + " VPC Deleted.")

    except ClientError as e:
        print("Error: " + e.response['Error']['Message'])

    print("----------------")
    print("Delete VPC: Success!")
    print("----------------")

try:
    if destination < 1 or destination > 4:
        print('Organization unit is not valid!')
        raise Exception ("Organization unit is not valid!")

    print("----------------")
    print("> Connecting to AWS")
    print("----------------")
    #  Connecting to Organization unit

    client = boto3.client('sts')

    responseCredentials = client.assume_role(
        RoleArn=roleOrganizationManager,
        RoleSessionName=sessionNameRole,
        SerialNumber=authSerialNumber,
        TokenCode=mfa,
    )

    client = boto3.client('organizations',
                          aws_access_key_id=responseCredentials['Credentials']['AccessKeyId'],
                          aws_secret_access_key=responseCredentials['Credentials']['SecretAccessKey'],
                          aws_session_token=responseCredentials['Credentials']['SessionToken'])

except ClientError as e:
    print("-------ERROR-------")
    print(e.response['Error']['Message'])
    sys.exit("Error: Could not connect to Amazon Web Services, Credentials are incorrect!")


try:
```

```
# Get a list of existing users
list_Accounts = client.list_accounts(
)
alreadyCreated = False
# Quick check if the user exists
print("----------------------")
print("> Creating account: Started!")
for i in list_Accounts['Accounts']:
    if i["Email"] == email:
        alreadyCreated = True
        accountId = i["Id"]
        print("-"+i["Email"], 'already in the Organization.')
        print("--------------------")
        try:
            attach_policies(alreadyCreated)
        except ClientError as e:
            print("--------ERROR--------")
            print(e.response['Error']['Message'])
            sys.exit("Error: Could not attach policies to the account.")

        try:
            change_ou(alreadyCreated)
        except ClientError as e:
            print("--------ERROR--------")
            print(e.response['Error']['Message'])
            sys.exit("Error: Could not change Organization unit.")

        try:
            change_role(responseCredentials, accountId, alreadyCreated)
        except ClientError as e:
            print("Account is not ready! Tried for 120 seconds!")

if alreadyCreated == False:
    # Create user account

    CreateAccount = client.create_account(
        Email=email,
        AccountName=accountName,
        RoleName=xxx,
        IamUserAccessToBilling='DENY'
    )
    # Get unique creation id to check if the user is created successfully
    createId = CreateAccount['CreateAccountStatus']['Id']

    status = client.describe_create_account_status(
        CreateAccountRequestId=createId
    )

    # Wait till the user status == Succeeded
    print("-----------------")
    print("--Waiting Creation --")
    print("-----------------")
    while status['CreateAccountStatus']['State'] == "IN_PROGRESS":
```

```
            status = client.describe_create_account_status(
                CreateAccountRequestId=createId
            )
        # Get created account id
        accountId = status['CreateAccountStatus']['AccountId']
        print("--------------------")
        print(CreateAccount)
        print("--------------------")
        print("Creating account: Success!")

        try:
            attach_policies(alreadyCreated)
        except ClientError as e:
            print("------ERROR------")
            print(e.response['Error']['Message'] + " Or the policies you specified are already attached.")
            sys.exit("Error: Could not attach policies to the account.")

        try:
            change_ou(alreadyCreated)
        except ClientError as e:
            print("------ERROR------")
            print(e.response['Error']['Message'])
            sys.exit("Error: Could not change Organization unit.")

        try:
            change_role(responseCredentials, accountId, alreadyCreated)
        except ClientError as e:
            print("------ERROR------")
            print(e.response['Error']['Message'])
            sys.exit("Error: Could not change account alias.")

    except ClientError as e:
        print("-------ERROR-------")
        print(e.response['Error']['Message'])
        sys.exit("Error: Something went wrong while creating your account")

    print("--------------------")
    print("------ FINISH ------")
    print("--------------------")
```

B. Omschrijving Bijlage B

C. Omschrijving Bijlage C