RELEASE DOCUMENTATIE

Strategie voor het releasen van software

Coen den Engelsman / Jordi Roër / Arjan Schouten / Ditmar Commandeur

0878733 / 0874316 / 0874652 / 0882224 - INF2A - 2015

# Inleiding

Dit verslag omschrijft een release strategie voor de PCConfigurator en alle aanverwanten onderdelen van deze applicatie. PCConfigurator stelt gebruikers in staat zelfstandig een computer te kunnen configureren op basis van componenten. Deze applicatie bevat meerdere componenten die een samenhangend geheel zijn van het gehele proces. In dit verslag wordt omschreven hoe deze componenten worden gereleased. Daarbij wordt omschreven hoe de configuratie eruit ziet, hoe de deployment pipeline eruit komt te zien, hoe verschillende omgevingen eruit komen te zien zoals een acceptatie en omgevingen.

De release van de software en de omgevingen worden erop gericht dat het een capaciteit van minstens 10.000 bezoekers per dag kan verwerken, met optimale performance en zonder downtime voor software-updates.

Inhoudsopgave

[Inleiding 2](#_Toc409699653)

[1. Stakeholders 4](#_Toc409699654)

[2. Configuraties en omgevingen 5](#_Toc409699655)

[2.1 Web applicatie 5](#_Toc409699656)

[2.1.1 Web applicatie 5](#_Toc409699657)

[2.1.1 Scraper 5](#_Toc409699658)

[2.2 Deployment pipeline 5](#_Toc409699659)

[2.2.1 Web applicatie 5](#_Toc409699660)

[2.2.2 Scraper 6](#_Toc409699661)

[3. Deployment 7](#_Toc409699662)

[3.1 Web applicatie 7](#_Toc409699663)

[3.1.1 Handmatige deployment 7](#_Toc409699664)

[3.2 Scraper 8](#_Toc409699665)

[3.3 Implementatie van de deployment pipeline 9](#_Toc409699666)

[3.3.1 Web applicatie 9](#_Toc409699667)

[3.3.2 Scraper 10](#_Toc409699668)

[4. Vermogensbeheer en configuratie management 14](#_Toc409699669)

[5. Upgrades en bugs 15](#_Toc409699670)

[5.1 Hardware 15](#_Toc409699671)

[5.2 Software 15](#_Toc409699672)

[5.3 Applicatie 15](#_Toc409699673)

[6. Data migratieplan 16](#_Toc409699674)

[6.1 Neo4j gegevens migratie 16](#_Toc409699675)

[6.2 MongoDB gegevens migratie 16](#_Toc409699676)

[6.3 TFS gegevens migratie 16](#_Toc409699677)

[7. Release plan 18](#_Toc409699678)

[7.1 Initiële release 18](#_Toc409699679)

[7.2 Smoke Test 18](#_Toc409699680)

[7.2.1 Web applicatie 18](#_Toc409699681)

[7.2.2 Scraper 19](#_Toc409699682)

[7.3 Release problemen 19](#_Toc409699683)

[7.4 Monitoring en loggen 19](#_Toc409699684)

# Stakeholders

Er zijn meerdere betrokkenen en belanghebbende bij deze applicatie. De belanghebbende zijn onder andere:

* Kevin van Ingen van de Hogeschool Rotterdam als Opdrachtgever
* De groep BigBananas als opdrachtnemer
* De uiteindelijke gebruikers van het product
* De beheerders van de omgeving waar het product op moet gaan draaien
* Hosting bedrijf dat de applicatie zal hosten op hun server.

# 2. Configuraties en omgevingen

Voor onze omgevingen zijn verschillende soorten software met bepaalde versies nodig, om het proces van deployment en continious integration goed te laten verlopen. Hieronder is aangegeven wat voor software er draait.

## 2.1 Web applicatie

De software en versies die gebruikt worden op de de server zijn de volgende:

### 2.1.1 Web applicatie

|  |  |
| --- | --- |
| Software | Versie |
| Windows Server | 2008 R2 Enterprise |
| Team foundation server | Express 2013 |
| Visual studio | 2013 Professional |
| MongoDB | 2.6.5 Standard |
| MS SQL server | 2012 Express |
| Neo4j | 2.2 Community |
| Internet Information Services | 7 |

### 2.1.1 Scraper

|  |  |
| --- | --- |
| Software | Versie |
| Ubuntu Server | 14.04 LTS |
| Docker | 1.4.1 |
| Python + pip | 2.7 |
| Scrapy | 0.24 |
| Jenkins | 1.596 |
| Scrapyd | 0.17 |
| Sonarsource Community Edition + Nosetests | 4.2 & 1.3.4 |
| Bitbucket of Github/Git | - |

## 2.2 Deployment pipeline

Bij iedere commit naar het versie beheer systeem word geautomatiseerd een build gedaan van de nieuwe versie van de software, ook worden de beschikbare unit test en acceptatie test geautomatiseerd gedraaid. Als de uitkomst van deze tests positief is, is de build succesvol en word de versie gedeployed naar de “test” omgeving. In de andere gevallen krijgen de ontwikkelaars een melding in Visual studio of een notificatie in Jenkins dat de build van de laatste versie mislukt is.

### 2.2.1 Web applicatie

1. Ontwikkelaar commit nieuwe versie
2. TFS start nieuwe build
3. TFS draait unit tests
4. TFS draait acceptatie tests
5. Als 2, 3 en 4 succesvol zijn, deploy versie naar acceptatie omgeving

### 2.2.2 Scraper

1. Ontwikkelaar commit nieuwe versie
2. Jenkins start nieuwe build
3. Jenkins start unit tests
4. Jenkins maakt code coverage rapport
5. Jenkins opent testomgeving
6. Jenkins doet een smoketest naar testomgeving
7. Jenkins released de software naar de testomgeving via scrapyd
8. Wanneer correct deploy naar productie

# 3. Deployment

De deployment van een nieuwe versie van een onderdeel van de applicatie is een geautomatiseerd proces, dat uitgevoerd wordt door middel van de *continuous integration* techniek. Hierbij wordt de software gebouwd, getest en eventueel direct gedeployed naar de productie server(s).

## 3.1 Web applicatie

De *continious integration* techniek wordt gebruikt in samenstelling met Team Foundation Server. Team Foundation Server, hierna noemende TFS, wordt gebruikt voor het beheren van projecten, het configureren van deployment en test strategieën en kent nog veel meer functionaliteit dit in dit verslag niet benoemd zullen worden.

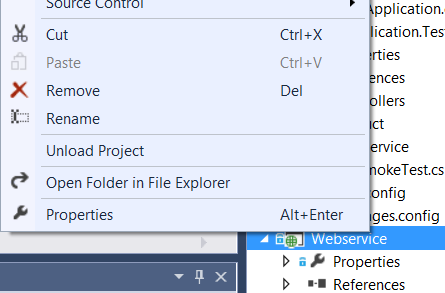
TFS is een van de fundamentele bouwstenen van het project. De TFS omgeving biedt de mogelijkheid om met behulp van continious integration een automatische deployment pipeline op te stellen met daarin een build rapport, test rapport en zorgt voor een automatische deployment van de web applicatie en de webservice op verschillende servers.

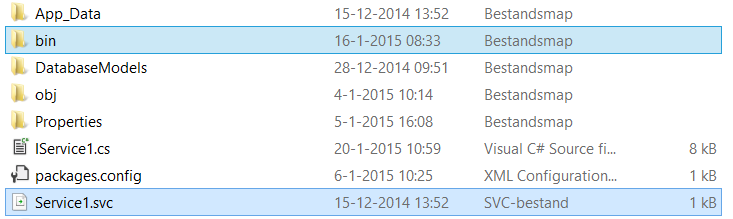
### 3.1.1 Handmatige deployment

Mochten er, vervelend genoeg, handmatig een wijziging moeten worden doorgevoerd aan de webservice, dan raad ik aan de volgende stappen te ondernemen om deze uit te voeren. Het is belangrijk om voor het handmatig deployen de hele solution in Visual Studio te builden voor het uitvoeren van de volgende stappen.

Stap 1: Build de solution in Visual Studio. De build is succesvol als er geen errors zijn. De status van de build is ook te vinden in een informatie balk onderaan Visual Studio.

Stap 2: Ga naar de solution explorer en zoek de web service project. Rechtermuisknop op het project en klik op ‘Open Folder in File Explorer’.



Stap 3: Er wordt in windows een nieuw File Explorer Window geopend met daarin de bestanden van de webservice. De webservice applicatie in IIS heeft alleen een .svc bestand nodig en alle dll’s in de bin map, dus de bin map wordt mee gekopiëerd. De bestanden die zijn gekopiëerd worden meegenomen naar de server. IIS heeft een map op de volgende locatie: *C:\inetpub\wwwroot\WCFServer\Services*. In deze map is al een bin map en .svc bestand te vinden en dienen overschreven te worden.

Als dat voltooid is, wordt er een nieuwe versie van de webservice online gezet en kan deze meteen gebruikt worden. Dit veroorzaakt geen downtime.

## 3.2 Scraper

Door gebruik te maken van Scrapyd voor het implementeren is het deployen van de scraper zeer vergemakkelijkt. Scrapyd dient op zowel de server kant als op de kant van de ontwikkelaar geïnstalleerd te zijn naast Scrapy en Python 2.7. Wanneer dit het geval is kan de ontwikkelaar in zijn Scrapy project in de root folder in Scrapy.cfg aangeven wat het Scrapyd ip is waar naartoe gestuurd moet worden. In ons geval is dit: **149.210.188.85**,wij gebruiken de standaard Scrapyd port: **6800**.

Wanneer de configuratie in het Scrapy project correct is neergezet kan doormiddel van onderstaande Scrapyd commando het project ingepakt en gedeployed worden;

“scrapyd**-**deploy”

Scrapyd zal het project nu inpakken en direct naar de server sturen. Daar zal de serverversie van Scrapyd het huidige project archiveren, het project verwijderen en het nieuwe project uitpakken voor gebruik.

Doordat er van te voren getest wordt alvorens er doorgegaan wordt met de pipeline is er voor de scraper zero-downtime ingebouwd. Wel moet er in acht genomen worden dat deployment gebeurt voor 01:00 (’s nachts).

## 3.3 Implementatie van de deployment pipeline

Voor alle onderdelen van de software zijn pipelines opgebouwd om het builden, testen en deployen van de software te vergemakkelijken. Hieronder is voor de web applicatie en de scraper beschreven hoe die pipelines opgezet kunnen worden.

### 3.3.1 Web applicatie

De pipeline word opgezet met behulp van de volgende software: Team Foundation server Express (heeft SQL server nodig) en IIS met het web deploy toolkit.

Voor het opzetten van de deployment pipline is een Windows server nodig met daarop minimaal Windows server 2008 R2. De server moet minimaal de volgende specificaties hebben, Core 2 Quad, 4GB ram en 50GB schijfruimte.

1. IIS moet geïnstalleerd worden. Dit gebeurd door de server role voor IIS toe te voegen.
2. Team Foundation server moet geïnstalleerd worden. Dit mag in de meest eenvoudige uitvoering zonder sharepoint integratie. TFS installeert SQL express vanzelf voor u als deze nog niet op de server aanwezig is.
3. Als TFS geïnstalleerd is start de “Team foundation server Express Administration Console”
4. Ga in de console naar “Application Tier” en dan “Team Project Collections”
5. Maak een nieuwe collectie aan door op “Create Collection” te klikken
6. Geef de collectie een naam en maak de collectie aan.
7. Ga in de webbrowser op de server naar “localhost:8080/tfs”
8. Klik rechts boven op het tandwieltje
9. Selecteer de aangemaakte colletie
10. Klik op “View the collection administration page”
11. Selecteer het project
12. Selecteer het team
13. Klik op “Add” om gebruikers toe te voegen aan het project, voeg alle benodigde gebruikers toe.
14. Start op een pc Visual Studio
15. Klik in het menu op “team”
16. Klik rechts op “select team projects”
17. Voeg de server toe onder “servers”
18. selecteer de juiste connectie en connect
19. Maak in de build explorer een nieuwe Build configuration aan
20. Geef in de instellingen op dat deze getriggerd word bij een nieuwe commit
21. Geef op dat de aanwezige unit test gedraaid worden
22. Geef een publish file op en geef aan dat er alleen gedeployed word als alle testen geslaagd zijn.
23. Sla het build profiel op.

Vanaf nu word bij elke commit een build gemaakt, de tests gedraaid en als deze slagen alles gedeployed.

### 3.3.2 Scraper

Voor de implementatie pipeline wordt gebruik gemaakt van Docker, Jenkins, Python 2.7, Scrapy (+ libraries), Scrapyd, Git, nosetests, Sonar, Bitbucket. Hieronder is een handleiding voor het opzetten van de complete pipeline inclusief tests en release mogelijkheden. Uiteindelijk kan hiermee een automatische pipeline nagebootst worden voor de crawler.

Voordat er begonnen kan worden met de pipeline is er een server benodigd met een publiek IP adres waarop Ubuntu 14.04 draait. De minimale specificatie van deze server bedragen: 2GB aan Ram, 30GB aan opslag, 2 CPU cores, tenminste 25mbit/s aan internet verbinding en een SSH mogelijkheid.

1. Er dient via een SSH verbinding ingelogd te worden op de server.
2. Docker dient geïnstalleerd te worden op de server doormiddel van de volgende commando’s:  
   *[ -e /usr/lib/apt/methods/https ] || {  
    apt-get update  
    apt-get install apt-transport-https  
   }  
     
   sudo apt-key adv –keyserver hkp://keyserver.ubuntu.com:80 –recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9  
     
   sudo sh –c “echo deb* [*https://get.docker.com/ubuntu docker main\*](https://get.docker.com/ubuntu%20docker%20main\) *> /etc/apt/sources.list.d/docker.list”  
     
   sudo apt-get update  
   sudo apt-get install lxc-docker*
3. Om te controleren of de installatie van Docker goed verlopen is kun je het volgende commando uitvoeren:  
   *sudo docker run –i –t ubuntu /bin/bash*
4. Nadat Docker geïnstalleerd is en met bovenstaande commando een nieuwe Ubuntu container gedraaid wordt, wordt het tijd voor het installeren van Jenkins in deze container. Het installeren van Jenkins kan doormiddel van de volgende commando’s:  
   *wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -  
     
   sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ > /etc/apt/sources.list.d/jenkins.list'  
     
   sudo apt-get update  
   sudo apt-get install jenkins*
5. Hierna kan Jenkins gestart worden, dit is te doen met:  
   *nohup java –jar jenkins.war > $LOGFILE 2>&1*
6. Door naar het ipadres van de server te gaan met :8080 erachter kom je in de Jenkins omgeving. Hierin dienen verschillende onderdelen ingesteld te worden alvorens we verder gaan. Selecteer “Manage Plugins” en update alle plugins, laat Jenkins hierna opnieuw opstarten.
7. Ga terug naar “Manage Plugins” en zoek de volgende plugins om te installeren en installeer de plugins, laat Jenkins daarna opnieuw opstarten:  
   - Bitbucket Plugin  
   - Build Pipeline Plugin  
   - SiteMonitor Plugin  
   - Violations Plugin
8. Zorg er vervolgens voor dat Jenkins afgeschermd wordt, door via “Configure Global Security”, “Enable security” aan te vinken. Hierna selecteer je onder Security Realm: “Jenkins own user database” en laat “Allow users to sign up” nog aangevinkt!, klik op “Save”.
9. Maak een nieuw account voor jezelf aan en log daarmee in.
10. Selecteer vervolgens onder “Authorization”: “Matrix-based security”, selecteer je zojuist gemaakte account en vink alles aan, klik vervolgens op “Save”.
11. Voeg een nieuwe build stap toe aan Jenkins en noem deze “Build”.
12. Geef als Source Code Manangement op dat het om een “Git” repository gaat.
13. Vul de benodigde Bitbucket of Git gegevens in.
14. Geef een Build Trigger aan en selecteer hiervoor “Trigger Builds remotely”, voer hiervoor de Authentication Token van Bitbucket of Git in.
15. Voeg één extra build step toe aan deze build, maak hiervan een “Execute shell” met de volgende commando’s:  
    *pip install –q Scrapy  
    pip install –q pymongo*
16. Voeg nog één extra build stap toe, maak hiervan een “Post-build projects”, als projects to build dient: “Unit Testing” ingevuld te worden, trigger only if the build is stable.
17. Sla deze Build nu op.
18. Maak een nieuwe Build aan en noem deze “Unit Testing”.
19. Klik op “Advanaced” onder Advanaced Project Options.
20. Selecteer “Use customer workspace” en vul hier in: *“/var/lib/jenkins/jobs/Build/workspace/”*
21. Voeg een extra build step toe en maak hier een execute shell van, vul hierin de volgende code in:  
    *PYTHONPATH='/var/lib/jenkins/jobs/Build/workspace/'*

*nosetests --with-xunit --all-modules --traverse-namespace --with-coverage --cover-package=crawler --cover-inclusive*

1. Voeg nog een build step toe en maak hiervan een Post Build Action, selecteer vervolgens Publish JUnit test result report. Voer als XML in: *“nosetests.xml”.*
2. Voeg nog één extra build stap toe, maak hiervan een “Post-build projects”, als projects to build dient: “Code Coverage” ingevuld te worden, trigger only if the build is stable.
3. Sla deze Build nu op.
4. Maak een nieuwe Build aan en noem deze “Code Coverage”.
5. Klik op “Advanaced” onder Advanaced Project Options.
6. Selecteer “Use customer workspace” en vul hier in: *“/var/lib/jenkins/jobs/Build/workspace/”*
7. Voeg een extra build step toe en maak hier een execute shell van, vul hierin de volgende code in: *python -m coverage xml -i --include=crawler\**
8. Voeg nog een build step toe en maak hiervan een Post Build Action, selecteer vervolgens Publish Cobertura Coverage Report. Voer als XML in: *“coverage.xml”.*
9. Voeg nog één extra build stap toe, maak hiervan een “Post-build projects”, als projects to build dient: “Violations Test” ingevuld te worden, trigger even if the build is unstable.
10. Sla deze Build nu op.
11. Maak een nieuwe Build aan en noem deze “Violations Test”.
12. Klik op “Advanaced” onder Advanaced Project Options.
13. Selecteer “Use customer workspace” en vul hier in: *“/var/lib/jenkins/jobs/Build/workspace/”*
14. Voeg een extra build step toe en maak hier een execute shell van, vul hierin de volgende code in: *pylint -f parseable -d I0011,R0801,C0301,C0330,W0312 crawler | tee pylint.out*
15. Voeg nog een build step toe en maak hiervan een Post Build Action, selecteer vervolgens “Report Violations”. Voer onder pylint de volgende code in: *“\*\*/pylint.out”.*
16. Voeg nog één extra build stap toe, maak hiervan een “Release”, als projects to build dient: “Violations Test” ingevuld te worden, trigger even if the build is unstable.
17. Sla deze Build nu op.
18. Voordat we de Release build toevoegen moeten we een Dockerfile maken welke in deze Build aangeroepen kan worden om opnieuw gestart te worden, hiermee wordt een schone omgeving opgezet waarin de software uiteindelijk zal draaien.
19. Login op de server via SSH en open in de home directory van je gebruiker een nieuwe Nano voor “Dockerfile”, vul hierin de volgende gegevens in:

*FROM ubuntu  
MAINTAINER Big Bananas <https://bitbucket.org/DualCore>*

*RUN echo deb http://archive.ubuntu.com/ubuntu precise universe >> /etc/apt/sources.list  
RUN apt-get update  
RUN apt-get install –qy git  
RUN apt-get install –qy python python-dev python-distribute python-pip  
  
RUN pip install requests pymongo  
RUN apt-get install –qy libffi-dev libxml2-dev libxslt-dev lib32z1-dev libssl-dev  
RUN pip install lxml scrapy scrapyd  
  
EXPOSE 6800  
  
CMD [“scrapyd”]*

1. Sla “Dockerfile” op en sluit de ssh verbinding.
2. Maak een nieuwe Build aan en noem deze “Release”.
3. Klik op “Advanaced” onder Advanaced Project Options.
4. Selecteer “Use customer workspace” en vul hier in: *“/var/lib/jenkins/jobs/Build/workspace/”*
5. Voeg een extra build step toe en maak hier een execute shell van, vul hierin de volgende code in:  *docker stop crawler*

*docker rm crawler*

*docker run -d --name crawler -p 6800:6800 dualcore/crawler*

1. Voeg nog één extra build stap toe, maak hiervan een “Release”, als projects to build dient: “Smoke Test” ingevuld te worden, trigger only if the build is stable.
2. Sla deze Build nu op.
3. Maak een nieuwe Build aan en noem deze “Smoke Test”.
4. Voeg een extra build step toe en maak hier een Monitor Site van, vul hierna de url van de server in met :6800 erachter voor de scrapyd locatie.
5. Voeg nog één extra build stap toe, maak hiervan een “Release”, als projects to build dient: “Deployment” ingevuld te worden, trigger only if the build is stable.
6. Sla deze Build nu op.
7. Maak een nieuwe Build aan en noem deze “Deployment”.
8. Voeg een extra build step toe en maak hier een execute shell van, vul hierin de volgende code in:  *cd /var/lib/jenkins/jobs/Build/workspace*

*rm -R setup.py*

*rm -R build*

*rm -R project.egg-info*

*scrapyd-deploy*

1. Sla deze Build nu op.

De pipeline is nu volledig opgezet en kan in gebruik genomen worden, er zal bij elke commit automatisch een build gestart worden welke elke stap in de pipeline zal doorlopen. Het is mogelijk om de deployment niet automatisch te laten verlopen, maar deze handmatig in te voeren. Hiervoor kan er eventueel ingesteld worden dat een email gestuurd wordt wanneer de build klaar is voor het deployen.

Doormiddel van bovenstaande pipeline is het mogelijk om nieuwe versies automatisch door te voeren in de productie omgeving, tijdens de build in de pipeline wordt er gecontroleerd of er voldaan wordt aan alle tests en alleen goedgekeurde software komt zo in productie. Mocht er onverhoopt iets fout gaan kan er via scrapyd terug gegaan naar de vorige build versie hiervoor gebruik je de volgende CURL commando:

curl http://localhost:6800/delversion.json -d project=crawler -d version=<versienummer>

Door bovenstaande commando zal scrapyd terugvallen in de laatste versie, de versie voor de gemaakte deploy.

# 4. Vermogensbeheer en configuratie management

De configuratie en de assets van de applicatie worden bewaard in configuratie bestanden. Deze configuratie bestanden zijn in versiebeheer aanwezig zodat ook hiervan eerdere versies teruggehaald kunnen worden. Zodra een nieuwe versie van de scraper gereleased wordt worden ook de benodigde configuratie bestanden opgehaald.

De installatie van software library’s worden uitgevoerd via de NuGet Packet Manager. Alle toekomstige updates aan deze library’s worden verzorgd door NuGet en zorgt ervoor dat alle projecten up-to-date blijven.

Toekomstige aanpassingen aan deze library’s hoeven geen dependecy problemen op te leveren. Bij het releasen van een nieuwe versie worden overbodige dll’s verwijderd en de applicaties krijgen geen verwijzing meer naar de dll’s. Door middel van deze methode betekent het ook dat toegevoegde libary’s moeiteloos toegevoegd worden en bruikbaar zijn. Mocht er eventueel iets misgaan met een van de library’s dan zal de build manager van TFS aangeven wat er mis gaan en hoe dit verholpen kan worden.

De scraper zal in de toekomst niet snel extra dependencies krijgen en zal de komende tijd gemakkelijk op één server kunnen draaien. Hierdoor is er niet de noodzaak om configuratiemanagement voor meerdere servers op te zetten. Mocht dit wel het geval zijn dan zou er met Chef meerdere servers beheerd kunnen worden waarop de installatie en deployment kan worden uitgevoerd. Dit is door de efficiëntie van de crawler niet nodig. In de huidige status van de scraper kan binnen 1 dag 30.000 producten voor het eerst worden opgehaald. Dit is in een clean status van de product historie. Daarna hoeven alleen updates uitgevoerd te worden die enkele minuten duren.

Grotere wijzigingen aan het project betekent dat dit doorgevoerd moet worden op drie omgevingen. Het wordt aangeraden om andere wijzigingen handmatig door te voeren. Dit is goedkoper en de management hiervan is ook niet geautomatiseerd.

# 5. Upgrades en bugs

Gedurende de levenslijn van de applicatie en alle onderdelen die erbij betrokken zijn worden er meerdere upgrades uitgevoerd aan de servers en de applicaties die erop draaien.

Er zijn meerdere soorten upgrades die plaatsen kunnen vinden. Dit kunnen zowel fysieke updates als softwarematige updates zijn. Dit hoofdstuk beschrijft deze updates en hoe ermee om wordt gegaan.

## 5.1 Hardware

Het upgraden van een fysieke server is een van de laatste mogelijkheden, omdat de kans bestaat dat de server gedurende een korte periode offline komt te staan. Gelukkig zijn er andere technieken om ervoor te zorgen dat de servers achter de applicatie een upgrade krijgen zonder daadwerkelijk down-time te veroorzaken.

De server kan gespiegeld worden, waardoor er tijdens het upgraden van de fysieke server alsnog naar de gespiegeld server kan worden gestuurd. Wanneer de oude server gereed is kan de koppeling weer teruggezet worden.

## 5.2 Software

Het upgraden van software is een bezigheid die regelmatig voorkomt. Het upgraden van de web service en web applicatie is een geautomatiseerd proces en creëert geen down-time. Het upgraden van software zal gebeuren tijdens nachtelijke updates.

Geadviseerd wordt om de upgrades eerst op de testserver door te voeren, hierop te testen of de applicatie nog werk alvorens dit op de productieserver door te voeren. Ook hiervoor is het mogelijk om de server eerst te spiegelen alvorens te updaten met de software. Door de server te spiegelen en de verbinding door te lussen naar de gespiegelde server tijdens het updaten kan er zero-downtime opgezet worden.

## 5.3 Applicatie

Het updaten van de Webapplicatie en de Webservice is eenvoudig en levert geen tot weinig downtime op.

In Visual studio kan dit gedaan worden met de “publish” methode. In de versie beheer zit voor zowel de webservice als de webapplicatie al een publish profile.

Als in Visual studio met de rechter muisknop gelikt word op de betreffende solution en hier gekozen word voor “publish” komt het publish web scherm tevoorschijn. Hier kan eventueel eerst gekozen worden voor “preview publish” om te kijken wat er precies gaat gebeuren. Als er op “publish” geklikt word, word de applicatie gedeployed naar de server die opgegeven is in het betreffende publish profile.

Het publishen zorgt met een vlotte internetverbinding, minimaal 50mbit, voor weinig tot geen vertraging. Enkele gebruikers zullen hoogstens een kleine vertraging ervaren.

Het updaten van de scraper gebeurd na de commit volledig automatisch, hier is geen enkele downtime bij gemoeid.

# 6. Data migratieplan

Het migreren van data voor zowel Neo4j als MongoDB is een makkelijke klus. Niet te vergeten is, dat ook de TFS database gemigreerd moet worden, maar daar heeft TFS een standaard procedure voor. Voor de scraper is geen datamigratie nodig, deze slaat gegevens niet op maar stuurt ze door naar de service, hierdoor blijft de scraper onafhankelijk.

## 6.1 Neo4j gegevens migratie

De meeste Neo4j gegevens staan opgeslagen in een folder binnen Windows. Bij het opstarten van de Neo4j Community applicatie wordt er aangegeven wat de locatie van de database folder is. Deze map kan gekopieerd worden en naar een externe locatie worden geplakt, om vervolgens op die locatie de Neo4j Community applicatie database in te stellen op de folder met de geplakte gegevens.

## 6.2 MongoDB gegevens migratie

De mongodb gegevens kunnen eenvoudig geëxporteerd en geïmporteerd worden. Dit kan met de volgende commando’s:

**Exporteren**

*mongoexport -d pcconfigwarehouse -c products -o <bestandsnaam>.json*

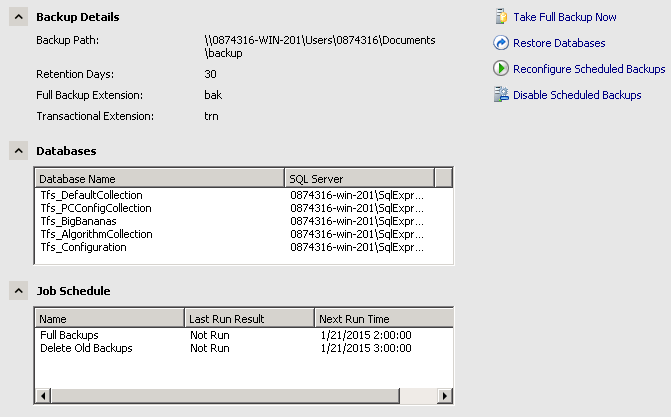
**Importeren**

*mongoimport -d pcconfigwarehouse -c products <bestandsnaam>.json*

Deze binaries bevinden zich in de bin map van de mongo db installaties. De commando’s zijn gebaseerd op Mongodb 2.6.

## 6.3 TFS gegevens migratie

TFS geeft beheerders de mogelijkheid om een back-up te maken van alle TFS collecties en de instellingen. Deze gegevens zijn te vinden in de door de beheerder aangegeven map. Deze map bevat alle informatie over de collecties, instellingen en databases die TFS bevat. De back-up folder kan moeiteloos worden gekopieërd naar een externe locatie.



Het gebruik maken van de back-up folder is niet heel moeilijk, mits dezelfde versie TFS als op de andere locatie wordt gebruikt. Backups moeten geconverteerd worden als ze gebruikt worden bij een lagere versie TFS.

Bij het instellen van de backups wordt er gevraagd om een *shared backup path*. Dit is een verwijzing naar een gedeelde folder in Windows of een gedeelde folder op een andere locatie. Een verwijzing naar deze map geeft TFS de kans om alle back-ups op een rijtje te zetten. De beheerder kan de laatste back-up terugzetten. Bij het voltooien van deze stap is de data correct gemigreerd.

# 7. Release plan

Het release plan beschrijft het proces voor het uitvoeren van een release. Er wordt getracht een zo geautomatiseerd systeem op te stellen. Het gebruik maken van het release plan voor een ander systeem wordt afgeraden, omdat sommige instellingen niet genoemd worden in dit document en de kans dat het nieuwe systeem niet werkt aanzienlijk groter is.

## 7.1 Initiële release

De eerste release is de belangrijkste release binnen de release cycle omdat dan de scripts voor het eerst uitgevoerd worden. Echter geldt dat de release voor het eerst uitgevoerd zal worden op een moment dat de scraper nog niet nodig is voor deployment waardoor er enige tijd over blijft om software aan te passen.

Voordat er gereleased kan worden moeten voorgaande stappen correct zijn uitgevoerd. Omdat de omgeving dan al compleet is opgezet hoeft alleen de software gedeployed te worden. De eerste keer hoeven oude bestanden niet verwijderd te worden.

## 7.2 Smoke Test

Er wordt voor beide pipelines gebruik gemaakt van een Smoke Test om te bapelen of de service of testomgeving naar behoren is opgezet. Hieronder staat voor elke van deze omgevingen wat die test inhoud.

### 7.2.1 Web applicatie

Er wordt gebruik gemaakt van een Smoke Test om te bepalen of de webservice beschikbaar is en de endpoints bereikbaar zijn. Deze test wordt behandeld als een Unit Test. Tijdens elke check-in wordt er een build gemaakt van de applicatie en worden alle Unit Tests uitgevoerd. Dus ook de Smoke Test.

Als de Smoke Test niet slaagd betekent het dat de webservice niet bereikbaar is. De webapplicatie zal zonder niet volledig functioneren en een nieuwe versie van de webservice kan hoogstwaarschijnlijk niet deployed worden naar de configureerde locatie.



Met de code hierboven wordt er een Smoke Test gemaakt en uitgevoerd bij een check-in. Als deze test samen met de andere Unit Tests slaagd wordt er groen licht gegeven voor het publishen van de componenten naar een acceptatie omgeving.

### 7.2.2 Scraper

Om de applicatie te kunnen smoke testen is het nodig dat er een fake scraper bestaat die uitgevoerd kan worden door het commando:

*‘scrapy crawl fakespider’*

Binnen deze fakespider wordt een url aangeroepen. Er kan worden geautomatiseerd worden nagegaan of deze url daadwerkelijk na een bepaald tijdstip is aangeroepen. Door voor iedere deployment een nieuwe url aan te maken en deze ook in de scraper geautomatiseerd te verwerken middels een automatisch gegenereerde configuratie file kan worden nagegaan of de laatste versie van de scraper is gedeployed.

## 7.3 Release problemen

Mochten er na of tijdens het releasen problemen op treden dan kan er een release van een oudere versie worden uitgevoerd. Dit kan gedaan worden zonder dat er extra stappen vereist zijn. Echter zullen wel de log files bewaard moeten worden om later in te kunnen zien waarom er een vorige versie van de scraper of de web applicatie gereleaset werd.

## 7.4 Monitoring en loggen

Het bijhouden van logs is een proces dat automatisch wordt uitgevoerd bij een check-in van de code. Gedurende verschillende stappen worden er logs geschreven. Hieronder een opsomming van de logs die verwacht worden:

* Pushen van changes in versiebeheer;
* Builden van de code;
* Testen van de code;
* Deployen van de code;
* Runnen smoke tests;
* Draaien van de applicatie (errors).

De webservice bevat een log systeem die alle requests naar de MongoDB bijhoudt. Daarbij is aangegeven om wat voor type request het gaat en wanneer de request is uitgevoerd. Logs zijn te vinden in een aparte collectie in de huidige MongoDB database.

Problemen op de server worden ook gelogd. Alle security of event logs worden bijgehouden in de System32 map. De logs worden overschreven als ze te groot worden. Het wordt aangeraden om deze logs goed bij te houden als het loggen van de server als zeer belangrijk wordt gezien.

De Internet Information Services, hierna IIS genoemd, houdt ook verschillende logs bij voor alle activiteiten die met IIS worden uitgevoerd. Deze logs zijn ook te vinden in de System32 map.

Met behulp van Jenkins CI wordt de source code gebuild. Binnen het build proces worden log gegevens verzameld. Deze logs worden bewaard door Jenkins en zijn later te bekijken via de interface van Jenkins.

Het deployment proces wordt vergemakkelijkt met behulp van het opensource project scrapyd. Met scrapyd kan de voortgang van de spiders gemonitord worden. Ook hoe de release van de spiders zelf verloopt kan hiermee gemonitord worden.