Release Documentatie Linux Server



Author: Oscar Veldman 0879062  
Mert Ozturk 0900994

Klas: INF2C

Vak: INFDEV016B

Assignment: C

Inhoud

[Inleiding 3](#_Toc440921296)

[Core requirements 3](#_Toc440921297)

[Stakeholders 3](#_Toc440921298)

[Configuration management 3](#_Toc440921299)

[Technology Deployment 4](#_Toc440921300)

[Deployment pipeline 4](#_Toc440921301)

[Release plan 4](#_Toc440921302)

[Enumeration of the environments available 4](#_Toc440921303)

[Upgrade plan 5](#_Toc440921304)

[Application Monitoring 5](#_Toc440921305)

### Inleiding

In het document wordt besproken we ons product gaan uitrollen. Hier gaan we uitzoeken hoe we het doen om vervolgens te bespreken. De applicatie is C++ en daar gaan we ook de release documentatie over schrijven. De PHP gedeelte worden door andere programmeurs gedaan. Het is misschien 1 project maar proces gedeelte kan los ervan draaien. Dit gedeelte van het project is vooral voor het uploaden van de CityGis data over de voertuigen.

### Core requirements

We hebben drie onderdelen in ons project. We hebben de input, processen en output. De input is de aanlevering van de data. Deze wordt aangeleverd in een CSV bestand. Omdat de programma op een server staat of moet komen, wordt de CSV bestand geüpload. Dit is geschreven in PHP.

Daarnaast is er processen. Die zorgt ervoor dat de gegevens van de CSV goed in de database komt te staan en netjes gesorteerd.

Als laatste hebben we ook nog een output systeem. Deze zorgt voor het uiterlijk van de server. Dat de klanten de gegevens kunnen kopen en zien. Verder zorgt er ook voor dat de gegevens uit de database komen, waar de proces heeft opgeslagen.

### Stakeholders

Hieronder staan alle belangrijke stakeholders die bij het project betrokken zijn. Hier moet ook rekening gehouden worden bij testen en bij het opzetten van de omgeving.

* CityGis
  + Die wordt de eigenaar van de applicatie
* Klanten van CityGis
  + Die gaan de applicatie gebruiken door de data te kopen
* Beheerder van de website
  + Die moet het onderhouden
* Beheerders van de auto’s
  + Die moeten de gegevens van de auto’s zien op de site.
* Analisten
  + Die moeten de gegevens van CityGis om te analyseren.

### Configuration management

We gaan “Ansible” gebruiken als Configuration management. Hier kan je makkelijk zien wat er allemaal gebeurt van de applicatie die je gemaakt hebt. Bijvoorbeeld hij slaat op hoeveel er iets fout gaat. Of je kan heel snel even data invoeren in de database met van te voren gemaakte scripts. Verder zijn er ook Configuration files voor Postgres. Deze zorgen ervoor dat de programma aan de goede database wordt gekoppeld.

### Technology Deployment

We gaan de volgende technologie gebruiken:

* Postgres 9.4
* Laravel
* Docker
* Linux Ubuntu Server
* C++
* Ansible
* Homestead Vagrant Box

Je hebt niet hele goede technologie om deze applicatie te kunnen draaien. Pas als er ineens heel veel bestanden geüpload worden is het een probleem. De Linux server staat al geïnstalleerd voor alle opdrachten. De Postgres is makkelijk met een commando installeren. De programma zelf moet je met de make file compiler.

### Deployment pipeline

Alles wordt op de Linux Ubuntu Server getest, gecompileerd en uitgevoerd. Op de eerste plaats doen we de commitment stage. Daar wordt de code van GitHub gehaald met een script. Vervolgens wordt de code geanalyseerd op syntax errores. Dan kan de code worden gecompileerd. Hier worden de fouten of juist de goede dingen rapporteert.

De volgende stage is de test stage. Hier wordt een acceptance test en unit testen uitgevoerd. De unit test is de framework van CppUnit. In deze testen worden zonder database getest. Dus alleen testen dat de applicatie goed werkt.

In deze stage worden de configuratie files ook goed gezet. Dit zijn kleine file bestandjes die aangepast moet worden. Zoals naar de goede database verbinding maken. Verder wordt nog de technologie waar het op draait gecheckt. Bijvoorbeeld met hoeveel geheugen moet het draaien en met hoeveel harde schrijf. Daarbij moet het ook met schrijfsnelheid.

### Release plan

De release wordt elk uur van de dag gedaan. Hierdoor weet je dat het niet ineens kapot gaat. In ieder geval weet je vanaf welke datum en uur het kapot gaat. Er wordt een script gemaakt zodat alles netjes geïnstalleerd wordt of geüpdatet. Ansible wordt ook van git gehaald. Waardoor het ook makkelijk up-to-date te houden is. Verder worden de testen zoals CppUnit test Classen van git gehaald. Zodat je in Ansible kan zien of er fout meldingen zijn.

Verder hebben we nog Dead Code & Copy Paste Detector. Dit zorgt ervoor dat alle codes die niet gebruikt worden, worden gevonden. Daarnaast zoekt het ook nog naar dubbele code. Ook gebruiken we CPPCheck. Dit is een check dat onderzoekt je statische code. Deze check zoekt eigenlijk naar dezelfde fouten als een compiler, zoals Memory leaks, variabele die niet zijn goed zijn aan gemaakt of nul pointer die misschien kunnen komen.

### Enumeration of the environments available

Doordat deze app door vele gebruikers gebruikt kunt worden, moeten we een realistisch omgeving maken. Deze Linux Ubuntu Server draait op een Cloud Server. Hier zetten we Docker op om te laten zien dat we dat we parallel kunnen draaien. Dat er meerdere postgres databases kunnen werken. De Cloud server heeft wat een gemiddelde server kan verwachten qua snelheden ram en processoren. Daarbij moeten we ook verwachten dat er honderden gebruikers de gegevens van CityGis kunnen ophalen.

### Upgrade plan

Het programma wordt opnieuw gecompileerd, zodat we de nieuwste onderdelen hebben. Alle andere programma’s worden ook gecheckt. De programma’s zoals Postgres worden alleen geüpdatet als er toestemming gegeven is. Want anders kan het zien dat sommige functies ineens niet meer werken. Als er een echte nieuwe release is dan moeten we een nieuwe docker omgeving maken. Met een deployment script is alles zo geïnstalleerd inclusief alle testen. Dit wordt verzorgt bij Jenkins.

Om de data in de database over te zetten, gaan we eerst de database back-uppen. Zodat als er iets mis gaat. Dat we opnieuw kunnen beginnen. Verder zitten in Laravel functies die je kan gebruiken om de data over te zetten. In feite wordt de structuur alleen verandert in de database.

### Application Monitoring

Om alles in de gaten te houden als de app draait gebruiken we New Relic. Dit is een programma die zowel voor PHP als voor C++ kan werken. Met deze applicatie kan je zien of achter of vooruit gaat met een upgrade. Het helpt ook voor eventuele gebreken of zoekt waarom het slechte performance geeft. Het probeert het ook om de bottleneck te verwijderen. Het belangrijkste is dat er een rapport wordt gegenereerd voor je prestaties.