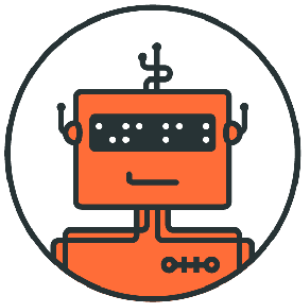
Vagrant en de opvolger, Otto

Een onderzoek naar de functionaliteit van Vagrant en Otto, de opvolger van Vagrant

VAGRANT



Vagrant en de opvolger, Otto

Een onderzoek naar de functionaliteit van Vagrant en Otto, de opvolger van Vagrant

Auteur:	J. Kaan
Cursus:	Vrije Studiepunten
Docent:	W. Everse
Cursuscode:	CU14845
Datum:	19-02-2016
Versie:	1.0

Versiebeheer

Dit hoofdstuk dient als een manier om gemakkelijk in te zien welke wijzigingen zijn aangebracht en wat de meest recente versie is.

Versie	Auteur	Beschrijving
0.1	J.K.	Initieel document
0.2	J.K.	Eerste conceptversie van portfolio met daarin alle resultaten voor elk leerdoel en voorlopige urenverantwoording.
1.0	J.K.	Officiële eerste versie van portfolio met daarin de STARRT-formulieren en de suggesties aan de hand van de feedback van W. Everse

Inhoud

1. Inleiding

1.1 Aanleiding

Het lectoraat maakt gebruik van Vagrant om zo buitenstaanders gemakkelijk te laten bijdragen aan de semantische wiki. Op deze manier is het gemakkelijk om een ontwikkelomgeving op te zetten die gelijk is voor elke ontwikkelaar.

Overigens is veel meer mogelijk met Vagrant dan waar het lectoraat deze technologie tot op heden voor gebruikt.

1.2 Probleemstelling

De volgende probleemstelling zal voor dit onderzoek gehanteerd worden:

"Het lectoraat gebruikt voor de ontwikkeling van de semantische wiki Vagrant. Ze gebruiken een minimaal deel van de functionaliteit van Vagrant, terwijl er veel meer mogelijk is met Vagrant. Het probleem is dat geen middelen beschikbaar zijn om onderzoek te doen naar de volledige functionaliteit van Vagrant en de opvolger Otto"

1.2.1 Doelstelling

Het doel van dit onderzoek is kennis en inzicht te vergaren zodat deze gebruikt kan worden om een eenduidige manier op te stellen die het ontwikkelingsproces van de semantische wiki ondersteunt.

1.2.2 Vraagstelling

1.2.2.1 Centrale vraag

De vraag die dit onderzoek probeert te beantwoorden luidt:

"Hoe kan Vagrant en eventueel de opvolger, Otto, gebruikt worden om een eenduidige manier op te stellen die dient als ondersteuning bij het ontwikkelingsproces van de semantische wiki waar rekening gehouden moet worden met het debuggen van PHP-code."

1.2.2.2 Deelvragen

De hiervoor genoemde centrale vraag valt uiteen in de volgende deelvragen:

- Welke mogelijkheden dienen voor het lectoraat en de semantische wiki aanwezig te zijn bij het ontwikkelproces?
- Wat is Otto precies en wat zijn de voordelen en nadelen ten opzichte van Vagrant?
- Welke manieren van PHP debuggen zijn beschikbaar?
- Hoe zijn de manieren van PHP debuggen samen te voegen met Otto/Vagrant?

1.2.3 Leerdoelen

De leerdoelen die worden behandeld in dit portfolio zijn als volgt:

- Aan het einde van de periode van 70 uur wil ik weten welk probleem Otto oplost, wat de voor- en nadelen zijn ten opzichte van Vagrant en hoe Otto ondersteunt bij het ontwikkelingsproces van een softwareproject.
- Aan het einde van de periode van 70 uur wil ik weten welke mogelijkheden er zijn op het gebied van het debuggen van PHP-code.
- Aan het einde van de periode van 70 uur wil ik met behulp van Vagrant/Otto een Vagrantfile/Appfile kunnen opstellen die ondersteunt bij het ontwikkelingsproces van een softwareproject.

2. Resultaten

Voor elk leerdoel zijn hieronder de resultaten beschreven en de reflectie op basis van de STARRT-methode.

2.1. Leerdoel 1: Vagrant en Otto

2.1.1. Resultaten

Om een antwoord te kunnen geven op deze deelvraag is het belangrijk om eerst alle betrokken begrippen uit te werken. Bij deze deelvraag zijn dat Otto en Vagrant. Aangezien Otto de opvolger is van Vagrant, wordt eerst een beschrijving gegeven van Vagrant.

2.1.1.1. Wat is Vagrant?

Vagrant wordt door HashiCorp (HashiCorp, 2012), ontwikkelaar van Vagrant, omschreven als volgt: *"Vagrant is a tool for building complete development environments. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases development/production parity, and makes the "works on my machine" excuse a relic of the past. "*

De omschrijving van HashiCorp beschrijft Vagrant als een tool waarmee complete ontwikkelomgevingen gebouwd kunnen worden. Door middel van een makkelijke workflow en een focus op automatisering zorgt Vagrant ervoor dat de tijd om een ontwikkelomgeving op te zetten veel minder wordt. Hiernaast zorgt Vagrant er ook voor dat de ontwikkelomgeving identiek is aan de productie omgeving en zorgt ervoor dat "op mijn omgeving werkt het wel" geen excuus meer is.

Een aantal voorbeelden hiervan zijn (HashiCorp, 2012): Voor softwareontwikkelaars zorgt Vagrant ervoor dat een consistente ontwikkelingsomgeving gemaakt kan worden, zodat altijd dezelfde dependencies worden gebruikt door verschillende softwareontwikkelaars. Dit allemaal komt niet ten koste van de tools die een softwareontwikkelaar gewend is om te gebruiken. Vagrant is een aanvulling op het bestaande proces, geen vervanging.

Hiernaast biedt Vagrant voor operations engineers een verwisselbare omgeving waarmee het mogelijk is om shell scripts, Chef cookbooks en Puppet modules lokaal te testen. Als het lokaal functioneert, kan met dezelfde configuratie de omgeving worden verplaatst naar bijvoorbeeld AWS of Rackspace.

Als laatste is het voor designers makkelijker om hun designs te testen door middel van de applicatie. Als een developer de Vagrantfile heeft geconfigureerd, dan kan de designer meteen aan de slag. Hierdoor is het eenvoudig om designs te maken die perfect aansluiten bij de applicatie.

2.1.1.2. Wat is Otto?

Otto wordt gekenmerkt als de opvolger van Vagrant. Dit is overigens de halve waarheid. Otto richt zich weliswaar op hetzelfde domein als Vagrant, maar Otto gebruikt hierbij ook Vagrant. Otto is als het ware een superset van Vagrant.

Waar Vagrant zich alleen richtte op het opzetten van een ontwikkelomgeving biedt Otto een complete oplossing voor het ontwikkelen, lanceren en configureren. Otto maakt dit mogelijk door verschillende services afkomstig van HashiCorp te bundelen: Vagrant (ontwikkelomgevingen), Terraform (infrastructuur) en Consul (services- en configuratie-management).

HashiCorp heeft zelf een aantal zaken aangegeven die aanzienlijk zijn verbeterd in Otto ten opzichte van Vagrant (HashiCorp):

- **Applicatie-level vs. machine-level configuratie –** Met de Vagrantfile wordt de machine beschreven die gebruikt wordt voor het ontwikkelen. De Appfile die bij Otto gebruikt wordt beschrijft de applicatie. Het eerste wat in de Appfile wordt beschreven is het type van de applicatie (PHP, Ruby, enz.) en Otto gebruikt dit om een ontwikkelomgeving op te zetten.
- **Volledige ondersteuning voor dependencies –** Met Vagrant was het ingewikkeld om alle dependencies die nodig waren te beschrijven. Otto maakt het mogelijk om alleen de daadwerkelijke dependencies te beschrijven en Otto installeert dan automatisch deze dependencies met alles wat nodig is om deze dependencies te kunnen uitvoeren. Dit leidt ertoe dat de Appfile, zelfs met een grote hoeveelheid dependencies, nog steeds leesbaar blijft.
- **Deployment –** Otto maakt het mogelijk om de applicatie te deployen naar welke provider wenselijk is. Gebruikers van Vagrant wilden al jaren een manier om dit mogelijk te maken, alleen was het niet mogelijk om in de Vagrantfile de nodige informatie in te vullen die nodig was om een correcte productie-omgeving op te zetten. In de Appfile is dit wel mogelijk en daardoor is deployment heel erg eenvoudig.
- **Performance –** Alle zaken die zijn geleerd tijdens het ontwikkelen en beheren van Vagrant zijn toegepast bij het ontwikkelen van Otto: Een commando dat de status opvraagt van de ontwikkelomgeving duurde bij Vagrant een paar seconden; bij Otto milliseconden.

2.1.1.3. Voordelen en nadelen van Otto ten opzichte van Vagrant

Otto maakt gebruik van de beste onderdelen van Vagrant om zo automatisch ontwikkelomgevingen op te zetten voor de gebruiker. Het grootste voordeel aan Otto is het feit dat het hierbij niet nodig is om een Appfile aan te maken. De Appfile wordt aangemaakt door Otto op basis van de bestaande code. Als het gaat om een PHP-project dan zal Otto een Appfile aanmaken die specifiek is voor PHP. Het is weliswaar mogelijk om op het begin een Appfile aan te maken in de gevallen dat Otto niet in staat is er een automatisch te genereren. Otto is zelfs in staat om ontbrekende delen van een Appfile in te vullen. Zo is het bijvoorbeeld mogelijk om de naam van de applicatie in te vullen zonder het type van het project (PHP, Ruby etc.). Otto zal zodoende het type van het project aan de Appfile toevoegen.
Voor het gebruik van Vagrant om een ontwikkelomgeving op te zetten, was het vaak noodzakelijk om op GitHub op zoek te gaan naar een Vagrant machine die was gemaakt welke paste bij het project. Hiernaast was er de mogelijkheid om een eigen Vagrant machine op te zetten, wat veel tijd kost. Otto zorgt ervoor dat dit niet meer nodig is, wat het opstarten van een bepaald project veel gemakkelijker maakt.

De performance van Otto is nagenoeg gelijk aan die van Vagrant. Dit is te verwachten aangezien Otto gebruik maakt van Vagrant op de achtergrond.

Techniek	Requests/sec	Transfer grootte/sec
Otto	157,73	1,57 MB
Vagrant	141,37	1,52 MB

Tabel 1: Performance Otto vs. Vagrant

Otto heeft een aantal nieuwe functionaliteiten welke niet mogelijk zijn m.b.v. Vagrant. Zo is het mogelijk om met Otto de ontwikkelde applicatie meteen te builden en deployen naar bijvoorbeeld AWS[1]. Het deployen van een applicatie gebeurt in twee stappen:

- De applicatie wordt gebouwd aan de hand van de specificaties van het betreffende platform. Zo wordt er voor AWS een AMI (Amazon Machine Image) gegenereerd die wordt herkend door AWS. Het omzetten van de applicatie naar een deployment artifact gebeurt door middel van Packer; een tool waarvan Otto gebruik maakt.
- Dit zogenaamde deployment artifact wordt overgezet naar de betreffende provider. Hierbij worden alle instellingen goed gezet e.d. Zodoende wordt de applicatie op een veilige manier gehost bij een bepaalde provider, zonder dat kennis nodig is van het betreffende platform. (Sitepoint, 2016)

Otto maakt het mogelijk om directe dependencies van je applicatie te beschrijven en Otto installeert dan alle dependencies van deze dependencies. Dit wordt mogelijk gemaakt doordat de dependencies die geïnstalleerd worden allemaal een eigen Appfile hebben waarin beschreven staat wat zij als dependencies hebben. Dit zorgt ervoor dat alleen de nodige informatie in een Appfile staat. Een nadeel hiervan is dat een dependency moet

[1] Amazon Web Services: https://aws.amazon.com

worden omgezet naar een Otto project; wat dus betekent dat deze dependency een Appfile moet bevatten die beschrijft hoe Otto met deze dependency moet omgaan. Overigens maakt Otto het heel erg gemakkelijk om een Appfile aan te maken: bestaande applicaties zijn dus gemakkelijk om te zetten naar Otto-projecten.

De voor en nadelen zijn heel erg gericht op de eisen aan het ontwikkelproces. Otto maakt het makkelijker om een standaard ontwikkelomgeving op te zetten. Als de wens is om verschillende externe dependencies te installeren, dan is het minder gemakkelijk om deze toe te voegen door middel van Otto. Voor Otto moet elke dependency een otto-project zijn, dit wil zeggen dat het project een Appfile moet bevatten waarmee Otto weet hoe deze dependency gebruikt moet worden. Aangezien Otto een relatief nieuwe techniek is, zijn veel dependencies niet beschikbaar als otto-project en moeten deze eigenhandig omgezet worden. In sommige gevallen is dit gemakkelijk, maar dit kan ook veel problemen met zich meebrengen.

2.1.2. Reflectie

STARRT-Formulier	
Student: Joey Kaan Nummer: 64808	
Opleiding: HBO-ICT Afstudeerdocent (1e examinator): Wouter Everse	
Competentie/leerdoel: Aan het einde van de periode van 70 uur wil ik weten welk probleem Otto oplost, wat de voor- en nadelen zijn ten opzichte van Vagrant en hoe Otto ondersteunt bij het ontwikkelingsproces van een softwareproject.	
Datum: 12-05-2016	
Titel en nummer van bewijs/bewijzen: • Paragraaf 2.1.1 (p. 3-5)	
S	***Geef voorbeelden van opdrachten (situaties) waarmee je kunt aantonen dat je de competentie hebt verworven. Beschrijf kort wat er aan de hand was of om welke opdracht het ging.*** Voor de cursus VCC heb ik de opdracht gekozen om onderzoek te doen naar Vagrant en de opvolger hiervan genaamd Otto. Dit onderzoek had als doel om voorkennis te vergaren die later bij deze opdracht werd gebruikt om een ontwikkelomgeving op te zetten. Tevens moest hieruit een advies komen waarin de beste keuze staat beschreven voor het lectoraat met betrekking tot Vagrant en/of Otto.
T	***Beschrijf de exacte rol/taak die jij had. Geef aan of het om een complexe taak ging en waaruit dat bleek. Wat moest jij doen?*** Mijn taak bij deze situatie was om onderzoek te doen naar Vagrant en de opvolger Otto. Aangezien Vagrant gebruikt werd door het lectoraat moest gekeken worden of deze vervangen kon worden door Otto. Door middel van de voor- en nadelen van Vagrant en Otto op te sommen heb ik een keuze kunnen maken welke techniek de beste keuze is voor het lectoraat.
A	***Beschrijf de activiteiten die jij achtereenvolgens hebt ondernomen in het kader van deze opdracht. Wat heb je concreet gedaan?*** Om tot het gewenste resultaat te komen heb ik de volgende activiteiten uitgevoerd: • Onderzoek uitgevoerd naar Vagrant • Onderzoek uitgevoerd naar Otto • Test opgezet om performance van Vagrant en Otto te meten • Aanvullend onderzoek naar overige voor- en nadelen van Vagrant/Otto.
R	***Beschrijf het resultaat van de opdracht en hoe de betrokkenen erop reageerden. Wat is er vervolgens met het resultaat gebeurd?*** Het resultaat was een verslag van het onderzoek met een conclusie waaruit kwam dat de beste keuze Vagrant was. Vervolgens is dit resultaat verwerkt in een portfolio en ter validatie voorgelegd bij een medewerker van het lectoraat. De reactie was heel positief. Op het algehele portfolio kreeg ik op een aantal punten feedback, maar dat was hierbij niet het geval.
R	***Wat heb je ervan geleerd? Wat zou je volgende keer anders aanpakken en waarom?*** Ik ben tevreden met het eindresultaat. De reden hiervoor is dat dit precies aansloot bij de verwachtingen van het lectoraat aangezien er geen feedback op heb gekregen. Van deze opdracht heb ik geleerd dat het goed is om allereerste goed onderzoek te doen naar de verschillende concepten die behandeld worden. Zo heb ik eerst Vagrant en Otto goed onderzocht voordat ik een vergelijking kon maken tussen deze.

	Het tweede wat ik heb geleerd is dat door middel van een praktijkvoorbeeld een vergelijking maken zeer wenselijk is. Op deze wijze worden meteen bewijzen geleverd waarmee gemakkelijk een conclusie getrokken kan worden. De reden daarvoor is dat met de meeste praktijkvoorbeelden meteen duidelijke resultaten worden vergaard.
T	***Geef een voorbeeld van een andere situatie waarin je deze competentie kunt toepassen.*** In het vervolg wil ik mij meer proberen in te leven in het oogpunt van het lectoraat. Zo kan ik in de toekomst de resultaten nog beter aansluiten bij hetgeen wat het lectoraat verwacht.

2.2. Leerdoel 2: Mogelijkheden op gebied van debuggen PHP-code

2.2.1. Resultaten

2.2.1.1. Meest gebruikte PHP-debug tools

De meest bekende tool om PHP-code te debuggen is XDebug. Zo populair dat deze vaak wordt meegeleverd met PHP. XDebug maakt het niet alleen mogelijk om te debuggen, maar ook om te profilen. Dit wil zeggen dat hiermee met XDebug inzicht verkregen kan worden in de prestaties van de PHP-code. (Rethans, 2016)

Naast XDebug zijn er nog een aantal andere manieren:

- PHPdbg (Standaard meegeleverd met PHP 5.6)
- Firebug en FirePHP (Sitepoint, 2010)

PHPdbg lijkt in veel opzichten op XDebug: je definieert breakpoints in je code en de debugger luistert hiernaar, als een breakpoint tegen wordt gekomen stopt de executie van de code en kun je hier stap voor stap doorheen lopen. (PHPdbg, 2016)

Firebug was in het begin een extensie exclusief voor FireFox. Deze tool heeft veel weg van de developer tools die beschikbaar zijn binnen Google Chrome. Door FirePHP te gebruiken is het mogelijk om gegevens te loggen naar Firebug. Deze gegevens zijn dan zichtbaar in de console. Dit heeft veel weg van het toevoegen van print/echo-statements in de PHP-code.

2.2.1.2. Hoe zijn de manieren van PHP debuggen samen te voegen met Vagrant/Otto?

De manier van samenvoegen van PHP-debug tools en de ontwikkelomgevingen die op te zetten zijn met Vagrant/Otto, hangt af van de manier waarop de ontwikkelomgeving is opgezet. Bij Vagrant is er de mogelijkheid om door middel van een bepaalde provisioner de ontwikkelomgeving op te zetten. Een provisioner is bijvoorbeeld:

- Ansible
- Puppet
- Shell scripts

De eerste twee hiervan zijn technologieën die speciaal ontwikkeld zijn voor het beheren van een server en/of het gemakkelijk installeren. De laatste is een script waarin commando's staan die direct worden uitgevoerd binnen de ontwikkelomgeving. Als het gaat om een ontwikkelomgeving die gebruik maakt van Ubuntu dan zullen hier verschillende sudo apt-get commando's instaan. Bij het mediawiki project wordt gebruik gemaakt van een shell script om de ontwikkelomgeving op te zetten. De PHP-debug tools zijn op te zetten binnen de ontwikkelomgeving op dezelfde manier als dat deze te installeren zijn op een ubuntu-systeem. Voor ubuntu is XDebug te installeren door 'sudo pecl install xdebug' uit te voeren. Hierna zijn nog een aantal commando's vereist om XDebug te configureren zodat code gedebugged kan worden die draait binnen de vagrant machine.

De PHP-debug tools zijn toe te voegen aan een Otto-project door een dependency toe te voegen aan de Appfile. Deze dependency moet een zogenaamd Otto-project zijn. De allereerste stap is om onderzoek te doen naar bestaande otto-projecten van de dependency die gewenst is.

Als deze niet beschikbaar zijn, dan moet een eigen otto-project aangemaakt worden welke de dependency installeert. Een Otto-project kan verschillende types hebben (HashiCorp, 2016). Een app type genaamd docker-external is goed te gebruiken voor het beschikbaar maken van een dependency voor Otto. Het docker-external type wijst naar een docker image die al ontwikkeld zijn en gereed zijn voor gebruik.

Dockers Hub is een verzamelplaats voor alle images die beschikbaar zijn gemaakt door anderen. Tenzij de gewilde dependency onbekend is, zal in de meeste gevallen hierop een image te vinden zijn die de dependency bevat. Als bijvoorbeeld mongodb als voorbeeld wordt genomen, dan kan deze omgezet worden naar een otto-project als volgt:

Een Appfile toevoegen met de volgende content:

```
application {
        name = “mongodb”
        type = “docker-external”
}

customization {
        image = “mongo:3.0”
        run_args = “-p 27017:27017”
}
```

Het is belangrijk dat deze Appfile in een aparte folder wordt geplaatst. Alleen de Appfile is voldoende. Als deze is aangemaakt, dan moet eenmaal ‘otto compile’ worden gerund in de folder waarin deze Appfile zich bevindt. Hiermee wordt het Otto-project gebuild worden en bruikbaar zijn binnen andere projecten.

Bij deze dependency is het gemakkelijk om deze om te zetten naar een Otto-project. Vaak is het overigens zo dat dit niet het geval is. Dit is te verklaren doordat mongodb een standalone techniek is. Als bijvoorbeeld XDebug wordt genomen, wat geen standalone techniek is, maar juist een toevoeging aan PHP dan is het minder gemakkelijk om deze te vinden in een docker image.

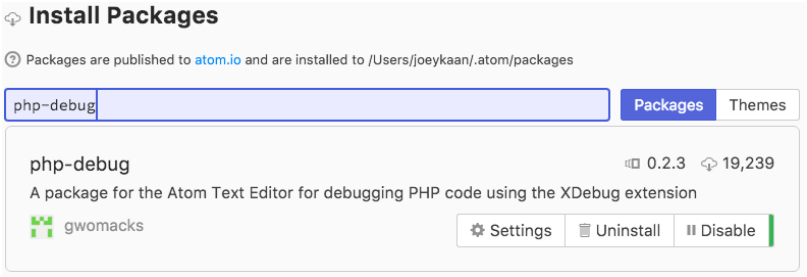
2.2.1.3 Handleiding gebruik XDebug

Om de code die draait op de Vagrant machine te debuggen moeten de volgende stappen worden ondernemen, hoe de stappen precies zijn ligt aan de ontwikkelomgeving:

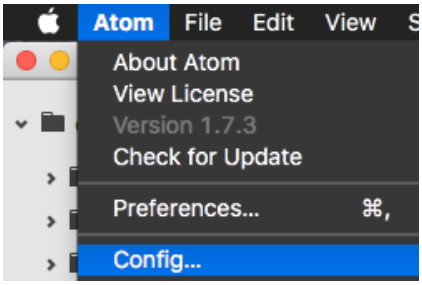
- Een XDebug client opzetten die luistert naar poort 9000.
- Pad-mapping opzetten die de lokale bestanden linkt naar de bestanden op de Vagrant machine.
- Breakpoint definiëren in de code.
- Het PHP-script uitvoeren waarin het betreffende breakpoint staat.
- De XDebug geïnstalleerd op de Vagrant-omgeving maakt vervolgens verbinding met de client die luistert naar poort 9000.

Hieronder is een korte introductie te zien die is gericht op de ontwikkelomgeving genaamd Atom:

Allereerste de package genaamd php-debug installeren:



Vervolgens moet de path mapping geconfigureerd worden, dit gebeurt bij Atom door een aantal regels toe te voegen aan het config bestand:

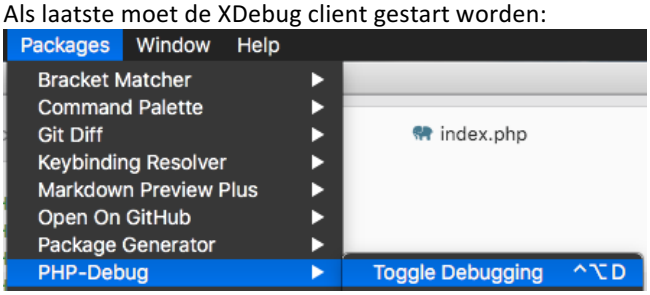


De volgende regels moeten worden toegevoegd aan dit bestand waarbij het pad naar lokale bestanden ingevuld moet worden naar het correcte pad:

```
"php-debug":
  PathMaps: [
    "/vagrant/;<pad_naar_lokale_bestanden>"
  ]
```

Vervolgens moet een breakpoint gedefinieërd worden:

Als laatste moet de XDebug client gestart worden:



Hiermee is de configuratie voor de Atom ontwikkelomgeving compleet. Als nu de URL bezocht wordt, wat in dit geval http://localhost:5555 is (aangezien de index.php in de root wordt gedebugged), zal XDebug verbinding maken met de client en kan er begonnen worden met debuggen.

XDebug maakt verbinding met de XDebug client en niet andersom; dit wordt door veel ontwikkelaars als verrassend gezien. Precies om deze reden hoeft de poort 9000 niet geforward te worden naar de host, dit is te zien doordat hier geen entry voor is in de Vagrantfile. Als poort 9000 niet beschikbaar is, omdat deze al voor andere doeleinden wordt gebruikt is het wijzigen van de poort zo simpel als in de XDebug-configuratie (bootstrap.sh) de poort aanpassen en vervolgens te zorgen dat de XDebug client luistert naar deze specifieke poort.

2.2.2. Reflectie

STARRT-Formulier	
Student: Joey Kaan Nummer: 64808	
Opleiding: HBO-ICT Afstudeerdocent (1e examinator): Wouter Everse	
Competentie/leerdoel: Aan het einde van de periode van 70 uur wil ik weten welke mogelijkheden er zijn op het gebied van het debuggen van PHP-code.	
Datum: 13-05-2016	
Titel en nummer van bewijs/bewijzen: • Paragraaf 2.2.1 (p. 8-11)	
S	***Geef voorbeelden van opdrachten (situaties) waarmee je kunt aantonen dat je de competentie hebt verworven. Beschrijf kort wat er aan de hand was of om welke opdracht het ging.*** Voor de VCC-opdracht moest ik onderzoek uitvoeren naar het debuggen van PHP-code, hoe het debuggen te integreren is binnen Vagrant en Otto en een handleiding op te zetten voor het gebruik van XDebug.
T	***Beschrijf de exacte rol/taak die jij had. Geef aan of het om een complexe taak ging en waaruit dat bleek. Wat moest jij doen?***

	Mijn exacte taak was om onderzoek te doen naar de verschillende manieren van PHP-code debuggen. Als dit duidelijk was, moest ik kijken naar hoe deze manieren te integreren zijn met behulp van Vagrant en Otto. Als laatste had ik een handleiding opgesteld voor het gebruik van XDebug.
A	***Beschrijf de activiteiten die jij achtereenvolgens hebt ondernomen in het kader van deze opdracht. Wat heb je concreet gedaan?*** Om de resultaten te vergaren voor deze opdracht zijn de volgende activiteiten uitgevoerd: Onderzoek doen naar verschillende manieren van PHP-code te debuggen. • Uitgebreider onderzoek doen naar verschillende manieren van PHP-code te debuggen. • Onderzoek doen naar integratie van Vagrant/Otto met PHP-code debuggen. • Onderzoek doen naar gebruik van XDebug met Vagrant om zo tot een handleiding te komen.
R	***Beschrijf het resultaat van de opdracht en hoe de betrokkenen erop reageerden. Wat is er vervolgens met het resultaat gebeurd?*** Het resultaat was een verslag van het onderzoek waarin de verschillende manieren van PHP-code debuggen staan beschreven, de integratie met Vagrant/Otto staat beschreven en een handleiding voor XDebug staat. Het resultaat is vervolgens verwerkt in het portfolio dat opgeleverd moest worden voor de cursus VCC. Het resultaat werd als positief bevonden, er was geen feedback op hetgeen wat opgeleverd was. Allereerste was overigens de handleiding voor XDebug niet aanwezig. Deze is later toegevoegd aan de hand van de feedback die was ontvangen.
R	***Wat heb je ervan geleerd? Wat zou je volgende keer anders aanpakken en waarom?*** Ik heb geleerd dat het goed is om uitgebreid onderzoek te doen naar de verschillende zaken die aan bod kwamen. Ik had geen rekening gehouden met het moeten opstellen van de handleiding XDebug. Overigens had ik uitgebreid onderzoek gedaan naar XDebug en daarom was het zeer gemakkelijk voor mij om alsnog de handleiding op te stellen. Een positief punt aan het onderzoeken van de integratie van Vagrant/Otto met PHP-code debuggen is dat ik dit heb onderzocht aan de hand van proof of concepts op te zetten. Aan de hand van deze proof of concepts was het gemakkelijk om snel door te hebben op welke wijze debug tools te integreren waren met Vagrant en/of Otto. De volgende keer zou ik meer onderzoek doen naar de alternatieven naast XDebug. XDebug is de meest bekende debugger, maar dat wil overigens niet zeggen dat deze het beste is. Als ik meer onderzoek had gedaan naar de alternatieven, had ik misschien een keuze gevonden die beter aansloot bij de eisen van het lectoraat. De reden hiervoor is dat ik nu met het onderzoek was begonnen met de gedachte **dat** XDebug uiteindelijk de keuze zou worden. Hierdoor heb ik een bepaalde voorkeur voor XDebug. Dit zorgt er uiteindelijk voor dat ik met een sceptisch oog kijk naar alternatieven.

T	***Geef een voorbeeld van een andere situatie waarin je deze competentie kunt toepassen.*** Bugs vinden in software door middel van debuggen kan heel erg verhelderend zijn. Met behulp van een debugger kun je in verschillende methodes stappen en daardoor echt de oorzaak van een probleem zichtbaar maken. Bij toekomstige softwareprojecten wil ik meer gebruik maken van XDebug. Hiernaast wil ik tijdens het gebruik van XDebug tijd vrij maken om te kijken naar alternatieven. De wereld van softwareontwikkeling ontwikkelt zich voortdurend en daarom zal er uiteindelijk een beter alternatief op de markt komen.

2.3. Leerdoel 3: Opzetten ontwikkelomgeving m.b.v. Vagrant/Otto

2.3.1. Resultaten

2.3.1.1. Eisen en wensen aan de ontwikkelomgeving

Zoals is op te maken uit de bootstrap.sh die wordt gebruikt om de VM op te zetten wordt er bij het lectoraat gebruik gemaakt van de programmeertaal PHP. Dit is te zien aan de hand van het feit dat PHP5 wordt geïnstalleerd. Naast PHP worden nog een aantal packages geïnstalleerd, namelijk:

- Apache
- MySQL
- PHPMyAdmin
- Git
- Composer (PHP)

Tevens is er naast deze functionaliteit een behoefte aan het debuggen van code geschreven met PHP wat ook aangeeft dat PHP de voorkeur belieft.

Naast de analyse van het script dat de nodige bestanden/applicaties installeert, is er een vragenlijst opgesteld die is beantwoord door een vertegenwoordiger van het lectoraat. Hieruit is het volgende gekomen:

- De code (de wikifarm met alles erop en eraan) zit nu in de gedeelde folder van de Vagrant machine. Dit verhindert het packen en mooi delen van de machine.
- De performance mag beter.
- Kibana voor Elastic Search.

2.3.1.2. *Implementatie ontwikkelomgeving*

Om een ontwikkelomgeving op te zetten, moet allereerste een keuze gemaakt worden tussen Vagrant en Otto.

Uit de analyse van Vagrant en Otto is duidelijk geworden dat Otto de voorkeur heeft als de standaard PHP-ontwikkelomgeving die hiermee wordt geleverd voldoende is voor het project. Als een aantal aangepaste zaken gewenst zijn, dan zijn deze niet gemakkelijk te implementeren door middel van Otto. Otto is als het ware een abstractie laag over Vagrant heen, die het makkelijker moet maken om met Vagrant ontwikkelomgevingen op te zetten.

Aangezien de lijst met eisen en wensen aan de ontwikkelomgeving een aantal aanpassingen omvat, is gekozen voor Vagrant om de ontwikkelomgeving op te zetten. Vagrant biedt de maakt het gemakkelijker om een aangepaste ontwikkelomgeving te ontwerpen, dit sluit aan bij de eisen en wensen van het lectoraat.

Het uitgangspunt is de Vagrantfile die is ontworpen door een medewerker van het lectoraat voor het mediawiki project.

De volgende wijzigingen zijn aangebracht aan de Vagrantfile en de virtuele machine:

- Een aantal performance verbeteringen:
  - Het gebruik van NFS voor het syncen van folders met de virtuele machine.
  - Memory wordt nu dynamisch gealloceerd aan de virtuele machine. Een vierde van het beschikbare geheugen wordt beschikbaar gemaakt aan de virtuele machine.
- XDebug is geïnstalleerd en geconfigureerd zodat remote debuggen mogelijk is, deze maakt verbinding op poort 9000.
- Kibana is geïnstalleerd en is beschikbaar op poort 5601.

De Vagrantfile en bootstrap.sh die nodig zijn om de virtuele machine op te zetten met Vagrant zijn aanwezig in de bijlagen (bijlage 1: Vagrantfile & bijlage 2: Bootstrap.sh).

2.3.1.2.1. Gedeelde folder door middel van NFS

Het gebruik van NFS heeft een beduidend nadeel en dat is dat hiermee niet de owner/group-attributen meegegeven kunnen worden wat zonder het gebruik van NFS wel mogelijk is. Dit kan ervoor zorgen dat doordat permissies niet goed geconfigureerd zijn bepaalde mappen niet toegankelijk zijn. Het is overigens zo dat NFS de permissies van de host weerkaatst. Zodoende is het daarom mogelijk om de permissie-problemen op te lossen door de permissies op het host systeem goed te configureren. Aangezien het hier gaat om een ontwikkelomgeving is de gemakkelijkste manier om het volgende command in te voeren op Mac OSX/Linux: chmod 777 <naam>. Bij Windows kan er geen gebruik gemaakt worden van NFS, dus is dit probleem niet van toepassing.

Hiernaast is er een applicatie genaamd vagrant-bindfs[2] die het mogelijk maakt om toch owner/group-attributen te setten als er gebruik wordt gemaakt van NFS. Het nadeel hieraan is overigens dat dit een extra stap toevoegt bij het syncen van de folders waardoor de performance omlaag zal gaan. Daarom heeft de eerste optie de voorkeur. De performance van NFS + vagrant-bindfs zal overigens wel beter zijn dan als er helemaal geen gebruik wordt gemaakt van NFS.

[2] Vagrant BindFS: https://github.com/gael-ian/vagrant-bindfs

2.3.1.2.2. Packaging van de Vagrant machine

Naast dit probleem blijft er nog één probleem over: op dit moment wordt de applicatie beschikbaar gesteld in de Vagrant box door middel van een shared folder. De ontwikkelaar moet daarom de code van de applicatie zelf uitchecken vanuit het versiebeheer op zijn/haar ontwikkelsysteem.

Een betere manier zou zijn om tijdens het opstarten van de Vagrant machine meteen de nieuwste versie uit te checken vanuit het versiebeheer. Dit zou ervoor zorgen dat alleen de Vagrant machine opgestart moet worden en daarna meteen begonnen kan worden met ontwikkelen. Overigens moet dan nog wel een shared folder opgezet worden die het mogelijk maakt om de bestanden aan te passen op de computer van de ontwikkelaar. Kortom; een gedeelde folder zal altijd nodig zijn.

Een laatste optie hiervoor is om de Vagrant machine en de applicatie in een repository onder te brengen. Het resultaat hiervan is een zogenaamde monolithic repository (Szorc, 2014). Een monolithic repository is in het kort een repository waar alles in aanwezig is om de applicatie te kunnen ontwikkelen. Het tegenovergestelde hiervan zijn meerdere kleine repositories die elk een bepaald onderdeel van de applicatie bevatten. In dat geval zou dus de Vagrant machine een aparte repository krijgen en de mediawiki (en bijbehorende wiki's) ook een aparte repository. Aangezien de Vagrant machine twee bestanden beslaat, is het een logische keuze om deze te bundelen bij de applicatie zelf. Dat wil zeggen: een monolithic repository is de meest logische keuze. Monolithic repositories zorgen er tevens voor dat het heel gemakkelijk is om te ontwikkelen aan een project: één repository moet worden uitgecheckt en er kan meteen begonnen worden. Google gebruikt precies om deze reden ook monolithic repositories. (@Scale, 2015)

Het uiteindelijk advies gebaseerd op deze argumenten is dat een monolithic repository opgezet wordt met daarin de Vagrant machine en de code van de applicatie. Als de applicatie dan uitgecheckt wordt kan meteen de Vagrant machine worden gestart die de meest up-to-date versie heeft van de applicatie.

2.3.2. Reflectie

STARRT-Formulier	
Student: Joey Kaan Nummer: 64808	
Opleiding: HBO-ICT Afstudeerdocent (1e examinator): Wouter Everse	
Competentie/leerdoel: Aan het einde van de periode van 70 uur wil ik met behulp van Vagrant/Otto een Vagrantfile/Appfile kunnen opstellen die ondersteunt bij het ontwikkelingsproces van een softwareproject.	
Datum: 13-05-2016	
Titel en nummer van bewijs/bewijzen: • Paragraaf 2.3.1 (p. 14-16) • Bijlage 1: Vagrantfile (p. 22) • Bijlage 2: Bootstrap.sh (p. 23-25)	
S	***Geef voorbeelden van opdrachten (situaties) waarmee je kunt aantonen dat je de competentie hebt verworven. Beschrijf kort wat er aan de hand was of om welke opdracht het ging.*** Het eindproduct voor deze VCC-opdracht was een ontwikkelomgeving die met behulp van Vagrant en/of Otto opgezet moest worden. Op basis van voorgaand onderzoek is allereerste een keuze gemaakt tussen Vagrant en Otto. Vervolgens zijn de bestaande voorzieningen meegenomen en verbeterd aan de hand van de eisen van het lectoraat.
T	***Beschrijf de exacte rol/taak die jij had. Geef aan of het om een complexe taak ging en waaruit dat bleek. Wat moest jij doen?*** Mijn rol binnen deze situatie en deze specifieke deelopdracht was om een ontwikkelomgeving op te zetten die aansloot bij de eisen en wensen van het lectoraat. Dit moest gebeuren aan de hand van de informatie die was vergaard uit de vorige deelopdrachten: de eisen en wensen van het

	lectoraat, het onderzoek naar Vagrant/Otto en het onderzoek naar de manieren van PHP-code debuggen.
A	***Beschrijf de activiteiten die jij achtereenvolgens hebt ondernomen in het kader van deze opdracht. Wat heb je concreet gedaan?*** Hieronder zijn de activiteiten te zien die ik heb uitgevoerd om tot het resultaat te komen: Bestaande voorzieningen onderzocht en gekeken wat verbeterd kon worden aan de hand van de eisen en wensen van het lectoraat. • Onderzoek gedaan naar hoe de performance van Vagrant te verbeteren was. (Gebruik van NFS & dynamisch alloceren van beschikbare geheugen aan de Vagrant machine). • Toepassen van deze performance verbeteringen op de bestaande Vagrant machine. • Aanvullend onderzoek uitgevoerd naar installatie en configuratie van XDebug en Kibana. Toegepast op de Vagrant machine. • Aangepaste versie van Vagrant machine testen met behulp van de deltaexpertise en hzportolio wiki's. • Onderzoek doen naar de manier van Vagrant machine en de applicatie packagen.
R	***Beschrijf het resultaat van de opdracht en hoe de betrokkenen erop reageerden. Wat is er vervolgens met het resultaat gebeurd?*** Het resultaat van deze deelopdracht was een ontwikkelomgeving opgezet met Vagrant waarin alle eisen en wensen van het lectoraat zijn verwerkt. De betrokken reageerden erg enthousiast hierover. De enige feedback was dat een goede aanvulling hierop was om onderzoek te doen naar de manier van packagen aangezien dat een probleemgebied was. Dit resultaat is vervolgens verwerkt in het portfolio en zal in de toekomst worden gebruikt door het lectoraat om verder te ontwikkelen aan de wiki.
R	***Wat heb je ervan geleerd? Wat zou je volgende keer anders aanpakken en waarom?*** Ik ben uitermate tevreden met het resultaat aangezien deze goed werd ontvangen. Tijdens het uitvoeren van de activiteiten heb ik geleerd dat het goed is om door te blijven vragen totdat een bepaalde eis/wens helemaal duidelijk is. Door het lectoraat werd aangekaart dat het packagen van de applicatie een probleem was. Ik was van de veronderstelling dat ik een oplossing hiervoor moest bieden, overigens was dit niet het geval. Van mij werd verwacht dat ik hier onderzoek naar deed en een advies hierover gaf. Het daadwerkelijke uitwerken wordt uiteindelijk door het lectoraat uitgevoerd. Hierdoor zou ik de volgende keer meer duidelijkheid vragen over de eisen als dit nodig is. Uiteindelijk is dit alsnog goed gekomen, alleen had wel zo kunnen zijn dat ik werk had uitgevoerd wat niet nodig was geweest. Een negatief punt aan deze deelopdracht was dat op het begin de eerste versie van de wiki niet werkte op mijn ontwikkelomgeving. Uiteindelijk heeft het lectoraat mij een nieuwe versie toegestuurd, welke wel goed functioneerde. In het vervolg zou ik daarom meteen aan de bel trekken als blijkt dat de voorzieningen niet goed functioneren op mijn lokale ontwikkelomgeving. Ik wist overigens dat het lectoraat een nieuwe versie had gemaakt van de applicatie en ik dacht dat deze de problemen zou oplossen. In deze situatie heeft het goed uitgepakt, maar het had veel slechter kunnen uitpakken. Een positief punt aan de hele reeks activiteiten is dat ik gaandeweg veel contact heb gehad met W. Everse van het lectoraat. Hierdoor werd het voor mij duidelijk of ik goed bezig was en zo was het makkelijker om de ontwikkelomgeving zo op te zetten dat deze goed aansloot bij wat het lectoraat wilde.

T	***Geef een voorbeeld van een andere situatie waarin je deze competentie kunt toepassen.*** Tegenwoordig wordt er veel gebruik gemaakt van Vagrant en Docker om ontwikkelomgevingen op te zetten. Hierdoor is het gemakkelijk voor een geheel team om met dezelfde configuratie te werken en te ontwikkelen. Hiermee is het ook mogelijk om een omgeving op te zetten die identiek is aan de productie omgeving. Dit zorgt ervoor dat er geen onverwachtse zaken aan het licht komen als de applicatie naar productie wordt gedeployed. In het vervolg zou ik nog meer informatie vergaren met betrekking tot de eisen en wensen van een softwareproject.

3. Literatuur

@Scale. (2015, September 14). *Why Google Stores Billions of Lines of Code in a Single Repository*. Opgeroepen op Mei 13, 2016, van YouTube: https://www.youtube.com/watch?v=W71BTkUbdqE
HashiCorp. (2016, Mei 1). *Otto App Types*. Opgeroepen op Mei 1, 2016, van Otto Project: https://www.ottoproject.io/docs/apps/
HashiCorp. (sd). *The Successor to Vagrant*. Opgeroepen op April 19, 2016, van Otto: https://www.ottoproject.io/intro/vagrant-successor.html
HashiCorp. (2012). *Vagrant*. Opgeroepen op Februari 23, 2016, van About Vagrant: https://www.vagrantup.com/about.html
PHPdbg. (2016, Mei 1). *PHPdbg*. Opgeroepen op Mei 1, 2016, van PHPdbg: PHPdbg
Rethans, D. (2016, Mei 12). *XDebug*. Opgeroepen op Mei 12, 2016, van XDebug: https://xdebug.org/
Sitepoint. (2010, Februari 9). *Debug PHP with Firebug and FirePHP*. Opgeroepen op Mei 1, 2016, van Sitepoint: http://www.sitepoint.com/debug-php-firebug-firephp/
Sitepoint. (2016, Maart 21). *Otto Makes Development and Deployment a Breeze*. Opgeroepen op April 25, 2016, van Sitepoint: http://www.sitepoint.com/otto-makes-development-and-deployment-a-breeze/
Szorc, G. (2014, September 9). *On Monolithic Repositories*. Opgeroepen op Mei 12, 2016, van Gregory Szorc's Blog: http://gregoryszorc.com/blog/2014/09/09/on-monolithic-repositories/

4. Urenverantwoording

Hieronder is een overzicht te zien van de bestede uren aan dit portfolio.

Student: Joey Kaan			**Studentnummer:** 64808		
Activiteit	**Uren**				
Opdracht bespreken met opdrachtgever	1				
POP & PAP: Eerste versie	3				
POP & PAP: Tweede versie op basis van feedback opdrachtgever	2				
POP & PAP: Derde versie op basis van feedback opdrachtgever	1				
Eerste opzet portfolio	1				
Onderzoek doen naar Vagrant & Otto en verschillen hiertussen	12				
Test opstellen voor performance Vagrant vs. Otto	2				
Onderzoek naar mogelijkheden PHP-debuggen	5				
Informatie verwerken in portfolio	5				
Onderzoek naar integreren PHP-debuggen met Vagrant/Otto	4				
Eisen en wensen inzichtelijk maken van lectoraat	5				
Onderzoek doen naar opzetten ontwikkelomgevingen met behulp van Vagrant	8				
Opzetten van ontwikkelomgeving m.b.v. Vagrant	13				
Testen van ontwikkelomgeving	4				
Informatie verwerken in portfolio	1				
Portfolio aanpassen aan de hand van de feedback/suggesties	4				
STARRT-formulieren opstellen	4				
Totaal:	75				

BIJLAGEN

Bijlage 1: Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

  config.vm.provider "virtualbox" do |v|
    v.cpus = 2

    host = RbConfig::CONFIG['host_os']

    if host =~ /darwin/
      mem = `sysctl -n hw.memsize`.to_i / 1024
    elsif host =~ /linux/
      mem = `grep 'MemTotal' /proc/meminfo | sed -e 's/MemTotal://' -e 's/ kB//'`.to_i
    elsif host =~ /mswin|mingw|cygwin/
      mem = `wmic computersystem Get TotalPhysicalMemory`.split[1].to_i / 1024
    end

    mem = mem / 1024 / 4
    v.customize ["modifyvm", :id, "--memory", mem]
  end

  # Every Vagrant virtual environment requires a box to build off of.
  config.vm.box = "ubuntu/trusty64"

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine.
  config.vm.network "forwarded_port", guest: 80, host: 5555
  config.vm.network "forwarded_port", guest: 8142, host: 5142
  config.vm.network "forwarded_port", guest: 9200, host: 5200
  config.vm.network "forwarded_port", guest: 5601, host: 5601

  # Setup a private network to be able to use the NFS-way of syncing folders.
  # This will be ignored on Windows since Windows doesn't support NFS without a workaround (https://github.com/winnfsd/vagrant-winnfsd).
  config.vm.network "private_network", ip: '192.168.50.50'

  # Share an additional folder to the guest VM. The first argument is the path on the host to the actual folder.
  # The second argument is the path on the guest to mount the folder.
  # config.vm.synced_folder "./", "/var/www/html"
  config.vm.synced_folder "./", "/vagrant",
    nfs: true
    # owner: "vagrant",
    # group: "www-data",
    # mount_options: ["dmode=775,fmode=764"]

  # Define the bootstrap file: A (shell) script that runs after first setup of your box (= provisioning)
  config.vm.provision :shell, path: "bootstrap.sh"
  #  config.vm.provision :evmwikis, path: "evmwikis.sh"

  # Post up message
  config.vm.post_up_message = "Enjoy the EVM Vagrant machine. It is meant for development and testing of the EVM wikisystems. It contains

end
```

Bijlage 2: Bootstrap.sh

Bootstrap.sh is het script dat gebruikt wordt om de virtuele machine opgezet met Vagrant te configureren.

```
#!/usr/bin/env bash

# Use single quotes instead of double quotes to make it work with special-character passwords
PASSWORD='12345678'

# prevent stdin errors and warnings
export DEBIAN_FRONTEND=noninteractive

# Add the source list for Kibana to the apt-get's list of sources
wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
echo "deb http://packages.elastic.co/kibana/4.5/debian stable main" | sudo tee -a /etc/apt/sources.list

# update / upgrade
sudo apt-get update
sudo apt-get -y autoremove
sudo apt-get -y upgrade

# install apache 2.5 and php 5.5
sudo apt-get install -y apache2
sudo apt-get install -y php5

# install mysql and give password to installer
sudo debconf-set-selections <<< "mysql-server mysql-server/root_password password $PASSWORD"
sudo debconf-set-selections <<< "mysql-server mysql-server/root_password_again password $PASSWORD"
sudo apt-get -y install mysql-server
sudo apt-get -y install php5-mysql php5-mcrypt php5-curl php5-apcu
sudo php5enmod mcrypt curl

# install phpmyadmin and give password(s) to installer
# for simplicity I'm using the same password for mysql and phpmyadmin
sudo debconf-set-selections <<< "phpmyadmin phpmyadmin/dbconfig-install boolean true"
sudo debconf-set-selections <<< "phpmyadmin phpmyadmin/app-password-confirm password $PASSWORD"
sudo debconf-set-selections <<< "phpmyadmin phpmyadmin/mysql/admin-pass password $PASSWORD"
sudo debconf-set-selections <<< "phpmyadmin phpmyadmin/mysql/app-pass password $PASSWORD"
sudo debconf-set-selections <<< "phpmyadmin phpmyadmin/reconfigure-webserver multiselect apache2"
sudo apt-get -y install phpmyadmin

#### create and import 2 specific dbs for wikis ####
    # create users and dbs (deltaexpertise and hzportfolio)
    CREATE_USERS_DBS_SQL='/vagrant/create-users-dbs.sql.gz'
    gzip -dc < $CREATE_USERS_DBS_SQL | mysql -u root --password=$PASSWORD

    # do this iff the previous command returns 0 (no errors). (>=2nd time provisioning will skip this (since db exist error)
    if [ "$?" -eq 0 ]; then
        # import deltaexpertise and hzportfolio tables and data
        gzip -dc < /vagrant/deltaexpertise.sql.gz | mysql -u root --password=$PASSWORD deltaexpertise
        gzip -dc < /vagrant/hzportfolio.sql.gz | mysql -u root --password=$PASSWORD hzportfolio
    else
        echo "Error - dbs already exist, skip importing databases"
    fi
####
```

```
# set VHOST for apache
VHOST=$(cat <<EOF
<VirtualHost *:80>
    DocumentRoot "/vagrant"
    <Directory "/vagrant/">
 Options -Indexes +FollowSymLinks +MultiViews
        AllowOverride All
        Order allow,deny
        allow from all
        Require all granted
    </Directory>

    # WME July2015 workaround for elasticsearch:
    # this enables the javascript search suggestions, cannot be done directly
    # since that is considered unsafe (from https to http)
    ProxyRequests off
    ProxyPass /elasticsearch/ http://localhost:9200/
    <Location /elasticsearch/>
        ProxyPassReverse /
    </Location>
</VirtualHost>
EOF
)
echo "${VHOST}" > /etc/apache2/sites-available/000-default.conf

# enable mod_rewrite & mod_proxy
sudo a2enmod rewrite proxy_http

# Install XDebug
sudo apt-get -y install php-pear php5-dev
sudo pecl install xdebug

# Create the folder that is needd for xdebug
mkdir -p -v /vagrant/log/xdebug

if [ ! -f /vagrant/xdebug/configured ]; then
  echo "[xdebug]" >> /etc/php5/apache2/php.ini
  echo 'zend_extension="/usr/lib/php5/20121212/xdebug.so"' >> /etc/php5/apache2/php.ini
  echo "xdebug.remote_enable=1" >> /etc/php5/apache2/php.ini
  echo "xdebug.remote_connect_back=1" >> /etc/php5/apache2/php.ini
  echo "xdebug.remote_autostart=1" >> /etc/php5/apache2/php.ini
  echo 'xdebug.remote_log="/vagrant/log/xdebug/xdebug.log"' >> /etc/php5/apache2/php.ini
  touch /vagrant/xdebug/configured
fi

# Restart apache for the enabled modules (xdebug, mod_rewrite, mod_proxy)
sudo service apache2 restart

# install git
sudo apt-get -y install git

# install composer
curl -s https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer

# install oracle java 8 http://www.webupd8.org/2012/09/install-oracle-java-8-in-ubuntu-via-ppa.html
sudo add-apt-repository -y ppa:webupd8team/java
#auto accept license:
```

```
#auto accept license:
echo oracle-java8-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-selections
sudo apt-get -y update && sudo apt-get -y install oracle-java8-installer

# install parsoid https://www.mediawiki.org/wiki/Parsoid/Setup
sudo apt-key advanced --keyserver keys.gnupg.net --recv-keys 664C383A3566A3481B942F007A322AC6E84AFDD2
sudo apt-add-repository -y "deb https://releases.wikimedia.org/debian jessie-mediawiki main"
sudo apt-get -y update && sudo apt-get -y install parsoid
PARSOIDPATH='/etc/mediawiki/parsoid'
sudo mv $PARSOIDPATH/settings.js $PARSOIDPATH/settings.js.original
sudo cp /vagrant/search/parsoidsettings.js $PARSOIDPATH/settings.js
sudo service parsoid restart

# install and start elasticsearch www.elastic.co
wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
echo "deb http://packages.elastic.co/elasticsearch/2.x/debian stable main" | sudo tee -a /etc/apt/sources.list.d/elasticsearch-2.x.list
sudo apt-get -y update && sudo apt-get -y install elasticsearch=2.3.0
sudo update-rc.d elasticsearch defaults 95 10

#config setting which allows for port forwarding (see Vagrantfile)a local port to 9200 of this vagrant
sudo echo "network.bind_host: 0" >> /etc/elasticsearch/elasticsearch.yml

#install analyzer plugin: https://github.com/jprante/elasticsearch-analysis-skos
sudo /usr/share/elasticsearch/bin/plugin install http://xbib.org/repository/org/xbib/elasticsearch/plugin/elasticsearch-analysis-skos/2.3.0.0/elasticsearch-analysis-skos-2.3.0.0-plugin.zip

#start elasticsearch
sudo service elasticsearch start

# run update maintenance scripts and reset Admin passwords of the wikis
# deltaexpertisewiki
cd /vagrant/wikis/deltaexpertise
php wiki/maintenance/update.php --quick
php wiki/maintenance/changePassword.php --user Admin --password _vagrant

# hzportfoliowiki
cd /vagrant/wikis/hzportfolio
php wiki/maintenance/update.php --quick
php wiki/maintenance/changePassword.php --user Admin --password _vagrant

# Install kibana
sudo apt-get -y install kibana

# Startup kibana when booting
sudo update-rc.d kibana defaults 95 10

# Start kibana for first time
sudo service kibana start
```