Use Cases Monitoring jFed

Andreas De Lille

February 23, 2014

1 Inhoud

In dit document wordt beschreven wat de uiteindelijke webservice moet kunnen. Daarna worden ook een aantal opmerkingen over de uitwerking gegeven. Merk op dat in deze versie enkel uitlezen van info uitgewerkt wordt. In de latere uitwerking moet het ook mogelijk zijn om de tests die uitgevoerd moeten worden te beheren via een database. Deze database zal dan periodiek overlopen worden om de monitoring-resultaten actueel te houden. De webservice moet de bestaande situatie opkuisen en een duidelijke interface aanbieden. Het uiteindelijke doel is dat jFed gebruik maakt van deze interface om zo een ranking van de testbeds weer te geven. Verder zal ook de huidige webinterface gebruik maken van deze interface. Indien er vragen of opmerkingen zijn bij een use case, zullen ze onder de titel opmerkingen staan.

2 Cases

2.1 Ping

De ping van de verschillende testbeds moet opgevraagd kunnen worden. Hiervoor zou elk testbed een id moeten hebben. Het voordeel van een testbedid (en later testid) is dat men meerdere versies tegelijk kan ondersteunen. Hierdoor kan men 2 tests van het zelfde type opslaan zonder dat men problemen krijgt omdat een test van dat type al bestaat voor dat testbed. Verder moet duidelijk gemaakt worden of men enkel de laatste ping wil of het gemiddelde over een periode. Hiervoor zou een optionele parameter voorzien worden. Indien deze parameter niet ingevuld is wordt de actuele ping teruggegeven. Indien men een gemiddelde wil, moet een tijdsperiode meegeven worden. Indien de ping niet opgevraagd kan worden, wordt een foutboodschap teruggegeven.

2.2 getVersion status

Dit zou de Aggregate Manager server opvragen. Dit bevat o.a. het versie nummer van de aggregate manager. Doordat er geen ssl authenticatie voor nodig is, wordt hij vaak gebruikt om de status van de server op te vragen. Dit zou terug werken via een testbedid dat weergegeven wordt en zal enkel de actuele informatie terug geven. Bij een fout zal zoals altijd een foutboodschap teruggegeven worden.

2.3 Free Resources

Deze functie zal het aantal beschikbare resources van een testbed terug geven. Hiervoor is een testbedid nodig. Indien geen 2e parameter wordt weergegeven zal de actuele informatie getoond worden. Indien een tijdsperiode meegegeven wordt zal het gemiddelde beschikbare resources over die periode teruggegeven worden. Bij een fout (als het aantal niet opgevraagd kan worden) zal een foutboodschap teruggestuurd worden.

2.4 Last check

Dit zal terug geven wanneer een test voor de laatste keer is uitgevoerd op een testbed. Hiervoor zijn dus 2 parameters nodig een testbedid en een testid. De testid kan optioneel gemaakt worden, indien er geen testid gegeven zal het tijdstip van de laatste test op dat testbed teruggegeven worden.

2.5 International testbed monitoring status

Op elk testbed draait een stuk monitoring software dat periodiek zijn status aanpast in de databank. Bij deze call moet er teruggegeven worden of status in orde is of niet, maar ook of de status periodiek aangepast wordt.

2.6 Duration

Dit geeft weer hoelang de laatste test op een testbed duurde. Hiervoor is dus weer een testbedid en testid nodig. Verder kan ook een gemiddelde opgevraagd worden, een tweede paramater door te geven die de periode voorstelt.

2.7 Status van test

Het moet mogelijk zijn om te zien of een test gelukt is. Indien hij niet gelukt is moet er snel en duidelijk zichtbaar zijn bij welke stap de test is gestopt. Daarnaast moet men ook een log kunnen op vragen van de test. Deze functie zal weer per test en testbed werken en heeft dus terug een testbedid en testid nodig.

2.8 History van een test op een testbed

Het moet mogelijk zijn om per testbed en per test de geschiedenis op te vragen. Hierdoor kan men snel kijken naar de duur en de frequentie van een fout. Zoals hierboven is ook hier weer een testid en testbedid nodig.

2.9 History van een testbed

Verder kan het handig zijn om per testbed een overzicht weer te geven. Dit overzicht kan data bevatten zoals wanneer was de laatste keer dat het testbed offline was en hoeveel resources zijn er gemiddeld vrij. Dit moet echter niet als functie uitwerkt worden, men kan ook meerdere calls doen om tot deze informatie te komen. Dit kan echter wel in een functie gestoken worden om het aantal calls te beperken. Zie later bij besluit.

3 besluit

3.1 Componenten

Ik zou werken met een database en daarboven een webservice. Deze webservice geeft dan toegang tot alle informatie. De opbouw van mijn masterproef zal bestaan in het uitschrijven van use cases gevolgd door een interface van mijn api beschikbaar te stellen. Vervolgens zal ik op basis van de service-interface een database ontwerpen. Eenmaal de database en service gekoppeld zijn kunnen loadtests uitgevoegd worden. Er zullen 2-3 grote componenten uitgewerkt worden. Een webinterface enerzijds en anderzijds integratie in jFed.

3.1.1 Webinterface

De huidige webinterface zal ook gebruik maken van de webservice. Het voornaamste doel van deze interface is een duidelijk overzicht tonen van de actuele situatie. Dit zou moeten zorgen dat men snel kan zien welke tests fout lopen.Indien een testfout loopt moet het snel duidelijk zijn tot welke stap hij is geraakt. Er moet dus gelinkt worden naar de logs. Deze webinterface kan verbinding maken met de webservice of rechtstreeks met de databank verbinding maken. Het voordeel hiervan is dat er minder overhead is. Een nadeel is code-duplicatie en minder eenvoudig beheer.

3.1.2 Integratie in jFed

Een andere component is de integratie in jFed. Het moet mogelijk zijn om bij het selecteren van een testbed weer te geven of dat testbed online is en of er nog beschikbare resources zijn.

3.2 Opdeling

We kunnen de use cases opdelen in 2 grote groepen, namelijk de lange en korte termijn.

3.2.1 Lange termijn - Gemiddelde

Enerzijds is er de gemiddelde data die moet weergeven hoe betrouwbaar een testbed is op lange termijn. De duur van die termijn kan opgegeven worden in parameters. Deze gemiddelde data zal voor een ranking zorgen van een testbed. Deze ranking heeft niet als doel om de actuele informatie weer te geven, maar om de betrouwbaarheid over lange termijn weer te geven. jFed zal deze ranking weergeven zodat een onderzoeker snel in zijn gui kan zien welk testbed betrouwbaar is.

3.2.2 Korte termijn - Actueel

Anderzijds is er de actuele component die de huidige toestand moet weergeven. Dit zal de overeenkomen met de laatst gemeten waarde voor die test. Deze waarde kan eventueel gecached worden om de performantie te verbeteren. De webinterface zal deze actuele informatie weergeven en voorzien in een gedetailleerd overzicht voor elke test. Dit zou het voor ontwikkelaars makkelijk moeten maken om fouten op te sporen en te verbeteren. Deze informatie kan ook in jFed geïntegreerd worden. Zo kan het nuttig zijn om te zien of een testbed online is en/of er resources beschikbaar zijn.

3.3 Uitwerking

Naast de opdeling van de use cases en de opdeling in componenten moeten we ook kijken naar performantie. Deze wordt grotendeels bepaald door de uitwerking en de gebruikte technologiën.

3.3.1 Calls bundelen

Op de site https://flsmonitor.fed4fire.eu/ wordt een overzicht weergegeven van een aantal testbeds. Met de use cases hierboven kan dit overzicht perfect worden gegenereerd. Al moet met opmerken dat elke call slechts een waarde van een test weergeeft. Er zijn dus meerdere calls nodig. Dit kan voor overhead zorgen. Een mogelijke oplossing is calls bundelen. Zo kan men bv per testbedid direct een tabel met alle waarden die nodig zijn in de tabel terugsturen. Het voordeel is de snelheidswinst, een call per testbed volstaat nu. Het nadeel is dat dit de code minder proper maakt, meer functies toevoegen die dezelfde dingen doen maken de code onnodig complex. Een oplossing is een tussenweg waarbij we een algemene get-functie maken waaraan we kunnen zeggen welke waarden we terug willen krijgen. Zo kunnen we bijvoorbeeld een functie get oproepen met
- Een array parameters=ping,version,duration
- Een testbedid
waarop de call een hashmap zal teruggeven. In die hashmap zit dan:
- ping = getPing()
- version = getVersion()
- duration = getDuration()
Deze functie is gelijkaardig aan de history van een testbed.

3.3.2 Technologie - php

De tweede vraag die we ons moeten stellen is welke technologie die we gebruiken. Aangezien de taal gekend moet zijn en op elk platform moet kunnen draaien, liggen er 2 opties voor de hand, php en java.Beide talen zijn vrij te gebruiken en draaien op zo goed als elk platform.

De tool jFed is in java geschreven.Mijn ervaring met java is ook groter dan mijn ervaring met php. Het gebruiken van java zou eventuele integratie eenvoudiger kunnen maken. Al moet men opmerken dat het hier om een webservice gaat. Een webservice zal antwoorden in een taalonafhankelijk formaat zoals json of xml. Beide talen zullen dezelfde json genereren. De keuze voor java lijkt dus al gemaakt. Toch heb ik een voorkeur op php te gebruiken. De huidige uitwerking is namelijk al volledig in php. Php is een makkelijke taal om te leren. Een ander bijkomend voordeel is dat alles wat voorzien moet worden, nu al geïmplementeerd is.

3.3.3 Type webservice - REST

Er liggen 2 types voor de hand SOAP (Simple Object Access Protocol) en REST (Representational State Transfer). Er zijn een aantal verschillen tussen beide types. Beide zijn ondersteund door meerdere programmeertalen.Beide kunnen json/xml uitwisselen over het http protocol. Het grote verschil is dat er voor elke SOAP-service een WSDL file is. In deze file worden de functies en parameters volledig beschreven. Verder verpakt SOAP zijn berichten in een SOAP-envelop bestaande uit een SOAP-header en een SOAP-body. REST heeft geen file waarin alles beschreven wordt. De filosofie van rest is eenvoud. Rest is volledig statusloos en veel minder strict dan SOAP. Rest heeft geen overzichtsfile waarin alles tot in details beschreven staat. Als we beide vergelijken zien we dat REST een betere performantie biedt. SOAP voorziet in meer out-of-the-box functionaliteit zoals beveiliging en transacties. Dit echter wel ten koste van performantie. Aangezien de beveiliging van monitoring informatie niet de hoogste prioriteit is, lijkt REST hier de juiste keuze te zijn. Eventuele beveiliging kan dan later toegevoegd worden.