Abstract

Door de huidige verschuiving naar cloudgebaseerde technologiën zal het belang van netwerkprotocollen en beschikbaarheid van netwerken alleen maar toenemen. Om deze verschuiving vlot te laten verlopen is er meer en meer onderzoek nodig naar netwerktechnologiën. Voor dit onderzoek wordt er dan ook veelvuldig gebruik gemaakt van testbeds. Testbeds worden aangestuurd via jFed en worden gebruikt om netwerken te simuleren. De situatie heeft enkele nadelen. Het is voor een onderzoeker soms erg moeilijk om te bepalen of een bepaald gedrag in hun experiment te wijten is aan de eigen ontwikkelingen, of aan het testbed zelf.

Deze masterproef zal trachten dit probleem te verhelpen door de testbed monitoring te automatiseren. Via een webservice zal deze informatie dan aangeboden worden. Deze webservice zal weergeven hoe betrouwbaar een testbed is. Hiervoor komen verschillende ontwikkelingstools aan bod. Uiteindelijk zal deze service ervoor zorgen dat een onderzoeker via zijn primaire gebruikersinterface op de hoogte gebracht wordt van eventuele problemen.

Abstract

Due to the new cloud-based technologies, network protocols and network reachability are now more important than ever. To make this change quick and clean, we need more and more network research. This reseach heavily relies on testbeds to simulate networks. These testbeds are controlled via software tools. In this thesis we will look at jFed, a java tool developed in the European FED4FIRE project. Unfortunately, the current situation has a major downside in that it is very hard for researchers to determine if a certain behavior is caused by the testconfiguration or by the testbed.

This thesis will try to solve the aforementioned problem by automating the testbed monitoring. A webservice will then share this information and show how reliable a testbed is. After having determined the reliability of the testbed, the webservice will notify the researcher of any problems through the primary user interface.

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Bestaande situatie

iMinds is een onafhankelijk onderzoekscentrum dat opgericht werd door de Vlaamse overheid. Het is voornamelijk bezig met onderzoek omtrent ICT-innovatie. Bij dit onderzoek wordt veelvuldig gebruik gemaakt van testbeds. Een testbed is een verzameling van nodes waarmee netwerkopstellingen kunnen gesimuleerd worden.

Een goed voorbeeld van een testbed is de Virtual Wall. De Virtual Wall is ontworpen voor het experimenteren met nieuwe netwerkoplossingen voor de volgende generatie van het Internet. De kracht van deze infrastructuur is dat ze de mogelijkheid biedt om op een eenvoudige wijze specifieke netwerkopstellingen op te zetten volgens de noden van het onderzoek.

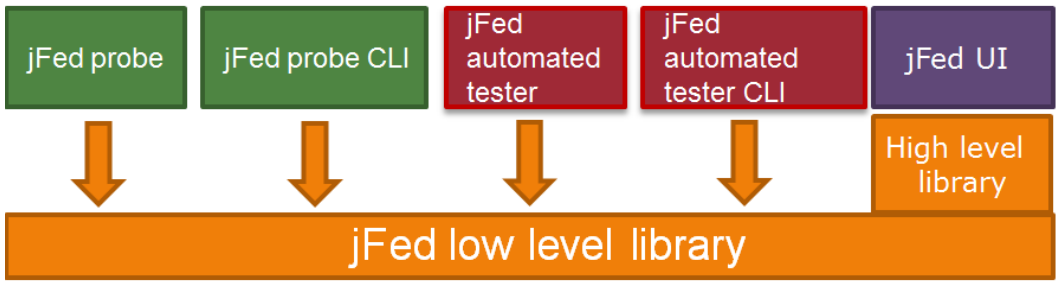
Een voorbeeld waarvoor dit testbed wordt gebruikt is dat men een server en een aantal cliënts definieert. Deze worden verbonden met een aantal tussenliggende routers. Vervolgens wordt een videostream opgestart. Op deze videostream kan men storing introduceren door pakketten te droppen. Deze storing zal ervoor zorgen dat het beeld aan de cliÃ≪nt-side hapert. Er kunnen technieken ingebouwd worden aan cliënt-side om deze storing op te vangen. Zo kan er overgeschakeld worden naar een lagere kwaliteit indien blijkt dat de beschikbare bandbreedte onvoldoende is. Testen van degelijke technieken verloopt dan ook aan de hand van testbeds.

Onderzoekers bij iMinds hebben niet alleen toegang tot hun Virtual Wall, maar door samenwerkingsakkoorden met diverse onderzoekscentra hebben ze ook wereldwijd toegang tot andere en soms erg verschillende testbeds. Deze zijn elk apart ontwikkeld. Hierdoor maakt elk testbed gebruik van een eigen interface. Deze interface wordt aangeboden door middel van een federation API. Een federation API is een interface die alle mogelijk functionaliteiten van een specifiek testbed aanbiedt. Hij staat in voor de communicatie tussen een programma en het testbed. Onderzoekers die met meerdere testbeds werken, moeten dan ook de werking

van elk testbed apart bestuderen alvorens ermee aan de slag te kunnen. Ook rechtstreekse verbindingen leggen tussen verschillende testbeds wordt hierdoor moeilijker.

Daarom werd in het Europese onderzoeksproject Fed4FIRE beslist om een aantal federation API's vast te leggen die gelijk moeten zijn op elk testbed. Dit maakt het mogelijk om tools te ontwikkelen die bruikbaar zijn op alle testbeds verenigd in dit project. Hiervoor werd de tool jFed ontwikkeld door iMinds1. jFed biedt een wrapper aan rond de specifieke federation API's die in dat project worden toegepast. Daarenboven voorziet jFed in verschillende applicaties om enerzijds te kunnen testen of testbeds deze API's correct ondersteunen, en anderzijds om als onderzoeker op een eenvoudige wijze een experiment te kunnen opzetten.

De kernmodule waar alle andere modules gebruik van maken is de "jFed library". De voornaamste functie van deze library is om de communicatie met de testbeds te vereenvoudigen. De hiervoor bruikbare testbeds implementeren enkele gestandaardiseerde API's, die de benodigde functionaliteiten aanbieden. De jFed library implementeert de client kant van deze API's, en voorziet Java interfaces voor de API-calls. De gebruiker zal hierdoor niet geconfronteerd worden met deze details. Zo zal de library o.a. de achterliggend SSL-connectie en de bijhorende certificaten en private keys beheren. Verder zullen ook de nodige omzettingen gebeuren. Daarnaast bevat deze library vele hulpklassen en methoden om de API's aan te spreken.



Figuur 1.1: Opbouw van jFed

Een goed voorbeeld van een API die door jFed word geïmplementeerd is de Aggregate Manager API, versie 2 en 3. Een voorbeeld van een call, is de "CreateSliver"call in de AMv2 API. De AMv2 API specificeert dat alle calls gedaan worden over een SSL verbinding met client-side certificate. Verder specificeert de API de exacte argumenten die vereist zijn en het formaat van het resultaat. De jFed library biedt een Java methode "createSliverïn de klasse "AggregateManager2äan. De argumenten worden opgegeven in een Java formaat. Zo zal bijvoorbeeld een datum-argument als een Java Date-object doorgegeven worden. De jFed library zal de datum naar het RFC3339 formaat omzetten voor het naar de server verstuurd wordt. Ook het resultaat van de call wordt verwerkt door de jFed library terug omgezet.

Steunend op deze library zijn enkele tools zoals jFed probe en jFed compliance tester gebouwd. Deze hebben als doel het valideren of testbeds de federation API's correct ondersteunen. Tenslotte is er de jFed GUI tool. Deze tool maakt het mogelijk om als eindgebruiker een experiment op een intuïtieve manier op te zetten. De jFed GUI heeft momenteel echter een relatief complexe interface die gebruiksvriendelijker gemaakt kan worden.

1.2 Probleemstelling

Er is echter nog ruimte voor verbetering in jFed. Zo worden er op geregelde tijdstippen verschillende testen automatisch uitgevoerd op testbeds. Deze testen geven o.a. weer of testbeds online zijn en of het aanmelden via een SSH-verbinding gelukt is. Deze resultaten worden momenteel nog redelijk ad hoc bijgehouden. Men maakt geen gebruik van een databank, maar van bash scripts om de resultaten te archiveren. Resultaten op deze manier bijhouden, is uiteraard inefficiënt. Bovendien staat deze informatie momenteel enkel op een enkele specifieke website vermeld.

Een onderzoeker wordt dus niet via zijn primaire gebruikersinterface op de hoogte gebracht van eventuele storingen met de testbeds opgenomen in zijn experiment. Dit maakt het voor onderzoekers soms moeilijk om te identificeren of een bepaald gedrag in hun experiment te wijten is aan de eigen ontwikkelingen, of aan de testbeds zelf. Hierdoor kan veel tijd onnodig verloren gaan tijdens de uitvoering en de analyse van het experiment. Een betere oplossing zou zijn dat bij het selecteren van een testbed de betrouwbaarheid weergegeven wordt. Zo kan een betrouwbaar testbed gekozen worden, om dergelijke situaties zoveel mogelijk te vermijden.

Niet enkel de betrouwbaarheid is van belang, ook het 'comfort'is van belang. Een testbed met een snellere verbinding geniet de voorkeur op een testbed met een trage verbinding. Naast de betrouwbaarheid en comfort moet ook gekeken worden naar de vereisten. Indien een betrouwbaar testbed niet beschikt over voldoende resources kan het niet gebruikt worden voor een experiment.

Door rekening houden met de betrouwbaarheid, het comfort en de vereisten kan jFed de onderzoeker melden dat een gekozen testbed niet betrouwbaar is. Hiervoor is een monitoring service nodig die bijhoudt hoe vaak en hoelang een testbed offline is.

Hoofdstuk 2

Analyse

2.1 Bestaande uitwerking

De ontwikkeling van de huidige situatie is door de sneller ontwikkeling, minder gestructureerd verlopen. Hierdoor bestaat de software uit een basis versie gevolgd door een aantal 'quick and dirty' toevoegingen. Er moet opgemerkt worden dat de huidige situatie werkt, maar het kan beter. Het gemist van structuur in opbouw zal op lange termijn leiden tot code die zeer moeilijk aan te passen is.

De oplossing van dit probleem en tevens het onderwerp van mijn masterproef is een webservice die de monitoring automatiseert. Hierbij komen een aantal vragen naar boven. Welke informatie moet bijgehouden worden? Wat bepaald de betrouwbaarheid van een testbed? Hoe nauwkeurig moet deze informatie bijgehouden worden?

2.1.1 Databanken

De huidige situatie voorziet niet in een centrale webservice. Wat er wel bestaat is een verzameling websites die rechtstreeks verbinding maken met een of meerdere databanken. Er zijn 3 databanken voorzien:

1. flsmonitoring
2. flsmonitoring-international
3. scenarios

Flsmonitoring en flsmonitoring-internation databank

In de eerste en de tweede databank bestaan uit een tabel waarin de laatste resultaten van elke test bijgehouden worden. Het verschil tussen deze 2 databanken komt overeen met de toegewezen categorie waarin het testbed zich bevind. De eerste databank bevat de lokale testbeds, de tweede bevat de internationale testbeds. Deze tabellen bevatten volgende kolommen:

- testbedid
- testbedname
- testbedurl
- pinglatency
- getversionstatus
- aggregatetestbedstate
- last-check

De eerste 3 parameters zijn duidelijk. 'Pinglatency' houdt de waarde van de pingtest bij. De kolommen 'getversionsStatus' en 'aggregatetestbedstate' worden gebruikt om de uitkomst van de getVersion test bij te houden. Deze test bevat o.a. het versie nummer van de aggregate manager. Doordat er geen ssl authenticatie nodig is voor deze test, wordt hij vaak gebruikt om de status van een server op te vragen. De kolom 'last-check' bevat een timestamp om bij te houden waneer de lijn laatste werd aangepast.

Scenario databank

De laatste databank bestaat uit 3 tabellen. Het doel ervan is het bijhouden van informatie over de scenariotesten. Scenariotesten of stitchingtesten zijn complexe testen die uit meerdere subtesten bestaan. Eenvoudig gezegd zal een stitching test de verbinding tussen verschillende testbeds testen. Hiervoor worden op elk testbed meerdere resources aangevraagd. Deze zullen dan trachten naar elkaar te pingen. Indien een testbed offline is wordt de volledige test afgebroken. Hieronder staan de opeenvolgende stappen die een stitching of scenariotest doorloopt.

1. setUp
2. getUserCredential
3. generateRspec
4. createSlice

5. initStitching
6. callSCS
7. callCreateSlivers
8. waitForAllReady
9. loginAndPing
10. callDeletes

De inhoud van elke subtest wordt hier buiten beschouwing gelaten. Wat wel opgemerkt kan worden is dat we de tests kunnen opdelen in 3 groepen. Zo zijn testen 1-6 voorbereidende testen. Ze dienen om de configuratie voor testen 7-9 klaar te zetten. Test 10 is de cleanup die de opgebouwde configuratie van stappen 1-6 terug ongedaan maakt. Elke subtest heeft een resultaat. Een stitching test zou dus minstens 10 resultaten hebben. In de huidige versie zijn er slechts 3 statussen gedefiniëerd. De stitching test is volledig gelukt, dit komt overeen met 10 geslaagde subtesten. De status is gedeeltelijk gelukt, dit komt overeen met de voorbereiding die wel gelukt is, maar de stappen 7-9 zijn niet of slechts gedeeltelijk gelukt. De laatste status geeft aan dat alle subtesten mislukt zijn.

De database is opgebouwd uit 3 tabellen.

- test-results
- test-context
- testbeds

De eerste resultaat houdt informatie bij over de testresultaten. De tweede tabel houdt de context van de test bij. De laatste tabel houdt de testbeds bij. De concrete invulling van de tabellen is van minder belang en wordt hier niet vermeld.

2.1.2 Webpagina's

In de huidige versie zijn een aantal webinterfaces voorzien. Er is een webinterface die enkel de status van de lokale testbeds weergeeft. Daarnaast er een webinterface die de internationale testbeds weergeeft. Deze 2 webinterfaces geven de naam van het testbeds met de ping, een veld dat aanduid of de getVersion call gelukt is en het aantal beschikbare resources.

De laatste webinterface geeft de resultaten van de scenariotesten weer. Er is ook de mogelijkheid om de log files van een test te bekijken. Tevens is het ook mogelijk om de geschiedenis van een scenariotest te bekijken.

2.2 Wat kan anders

2.2.1 Samenvoegen databases

Een eerste situatie die beter kan is het bijhouden van de laatste resultaten. Deze resultaten zitten in een databank, die voor elke testbed een lijn bevat. Nieuwe waarde overschrijven die lijn. Meteen kan opgemerkt worden dat het niet mogelijk is om statistieken over een lange termijn weer te geven.Het aanpassen van de monitoring waarden gebeurd door shell scripts. Deze worden periodiek uitgevoerd en lezen een configuratiefile in. Indien het testbed waarop de test uitgevoerd moet worden al in de databank zit is het id vermeld in de configuratiefile. Indien er geen id staat zal het script zelf een nieuwe lijn aanmaken en vervolgens de id wegschrijven naar de file. Eenmaal uitgevoerd, geeft de test een resultaat terug in de vorm van een xmlfile die door geïnstaleerde commando's geparset wordt. Vervolgens wordt de data weggeschreven naar de databank.

Door nieuwe resultaten toe te voegen op een nieuwe lijn kunnen we wel statistieken over langere termijn bijhouden. Ook de configuratiefiles kunnen opnemen in de databank. Toevoegen van een nieuwe test is dan eenmalig een entry toevoegen aan de database. Deze entry bevat dan het commando en de benodigde commando's.

Een tweede punt is de opdeling flsmonitoring en flsmonitoring-international. Deze 2 databanken zijn eigenlijk gelijk en kunnen gemakkelijk ondergebracht worden in een database. Dit kan door een extra punt toe te voegen dat weergeeft in welke categorie het testbed zich bevind. Een andere mogelijkheid is de internationale testbeds hardcoderen op de monitoring site.

Door de databanken scenario's en flsmonitoring in een databank onder te brengen, kunnen we het beheer vereenvoudigen. Beide gegevens zijn eenmaal resultaten van testen. Deze meer generische aanpak zal er toe leiden dat testen eenvoudiger gedefiniëerd kunnen worden.

2.2.2 Structuur database

De huidige opbouw van de databanken is niet flexibel genoeg. Als we de databanken zouden samenvoegen is er een complexere structuur nodig die meer flexibiliteit toelaat. Het moet mogelijk zijn dat een tests meerdere argumenten meekrijgt. Dit aantal is verschillend per type tests, werken met extra kolommen is dus niet aangeraden. Ook geven sommige testen meerdere resultaten terug. Dit aantal is ook variabel en afhankelijk van het type test dat gebruikt wordt. De structuur van de databank wordt later in deze scriptie verder uitgewerkt.

2.2.3 Webservice uitbouwen

Doordat de webinterfaces rechtstreeks contact maken met de databank is er minder overhead. Als de databank echter ook vanuit jFed bereikbaar moet zijn, is het beter om een webservice tussen te voegen. Dit vermijd duplicatie van code. Deze service zal complexere zaken zoals filteren van resultaten voorzien. Daardoor kan zowel jFed als de webinterfaces en eventuele toekomstige toepassingen met een eenvoudige call de informatie ophalen en moet niet elke mogelijke applicatie terug zelf de filtering voorzien.

Eenmaal de webservice uitgebouwd is kunnen er sites gemaakt worden die deze webservice contacteren om informatie op te halen. De huidige webinterfaces moeten dus ook vervangen worden door nieuwe versies die wel gebruik maken van de webservice. Er kan ook integratie in jFed voorzien worden. Dit kan als een onderdeel van de masterproef, of als een latere uitbreiding voor jFed.

2.3 Besluit

Om aan de verwachten te voldoen zijn er 2 grote veranderingen nodig. Er zullen twee grote veranderingen gebeuren. De eerste eerst is het samenvoegen van de databanken. Om het beheer van de data te vereenvoudigen zou 1 grote databank het werk van de 3 kleinere databanken over nemen. Deze zou zo opgebouwd zijn dat we testen met een variabel aantal argumenten en resultaten kunnen toevoegen, zonder iets te wijzigen aan de structuur van de databank. Zowel de configuratie van de testen als de resultaten zouden in deze databank opgenomen zijn.

Hoofdstuk 3

Ontwerp service

In dit hoofdstuk wordt de interface van de webservice besproken. Daarna worden er enkele voorbeelden van calls uitgewerkt.

3.1 Api Calls

De calls kunnen opgedeeld worden in 2 grote groepen.

- Results: Het opvragen van resultaten van een test.
- Configuration: Het opvragen, aanpassen en aanmaken van testen en de bijhorende configuratie.

3.2 Calls - Results

Voor het opvragen van resultaten wordt gebruik gemaakt van een generische functie.

3.2.1 Functies

Er zijn een aantal functies voorzien voor het opvragen van de resultaten.

- last: Deze functie geeft het laatste/ de laatste resultaat/resultaten terug (per test,testbed).
- list: Deze functie geeft een lijst resultaten die voldoen aan de opgegeven parameters. Zo kan deze lijst bijvoorbeeld beperkt worden tot een resultaten die overeenkomen met een bepaalde status of vallen tussen bepaalde data.
- detail: Deze functie geeft een detailweergave terug van een test. Deze detail weergave kan bijvoorbeeld ook logfiles bevatten om debugging te vergemakkelijken.

3.2.2 Parameters

Hier worden de parameters besproken. Deze parameters zullen het resultaat van de functies hierboven filteren.

- testbed: Deze parameter duidt een of meerdere testbeds aan.
- testtype: Deze parameter duidt het type van de test aan.
- status: Deze parameter duidt de status aan. Let op dit is per subtest.
- count: Deze parameter geeft het aantal laatste resultaten aan dat teruggegeven moet worden. Ook wanneer er enkel een till opgegeven is, worden de laatste resultaten voor de opgegeven tijd teruggegeven. Wanneer er enkel een from parameter opgegeven is, dan worden de eerste x resultaten startend vanaf de from terug gegeven.
- from: Geeft aan vanaf welke timestamp er gezocht moet worden.
- till: Geeft aan tot welke timestamp de resultaten gezocht moeten worden.
- resultid: Geeft een of meerdere id's van een testresultaat.
- testname: De naam van de test. Voornamelijk nodig bij stitching tests omdat er daar meerdere testbeds in betrokken zijn.
- instanceId: Filteren op de testinstantie, gelijkaardig aan de testname.
- format: Deze parameter dient niet om de resultaten te filteren, maar om het formaat van het resultaat te bepalen. Standaard is dit json, later kan eventueel xml en/of csv toegevoegd worden.

Indien bij tests of bij testbeds de waarde all wordt gegeven, worden respectievelijk alle tests of alle testbeds teruggegeven. Ook opgeven van lijsten is mogelijk. Zo kunnen de resultaten van pingtest voor 2 verschillende testbeds opvraagd worden met een call, zie ook voorbeelden.

3.2.3 Voorbeelden

Hieronder worden een aantal voorbeelden van calls en bijhorende antwoorden geven. Voorlopig zijn de voorbeelden beperkt tot het opvragen van data met behulp van HTTP GET requests.

last?testbed=urn-testbed0&testtype=ping&count=2

Deze call geeft voor testbed0 de laatste 2 ping resultaten terug.


```
{
    "177": {
        "testinstanceid": "1",
        "testtype": "ping",
        "testname": "ping voor testbed0",
        "log": "http:\/\/f4f-mon-dev.intec.ugent.be\/logs\/1\/1386",
        "timestamp": "2014-03-18 19:29:12.876569",
        "testbeds": [
            "urn-testbed0"
        ],
        "results": {
            "pingValue": "166"
        }
    },
    "155": {
        "testinstanceid": "1",
        "testtype": "ping",
        "testname": "ping voor testbed0",
        "log": "http:\/\/f4f-mon-dev.intec.ugent.be\/logs\/1\/1615",
        "timestamp": "2014-03-18 19:29:09.138309",
        "testbeds": [
            "urn-testbed0"
        ],
        "results": {
            "pingValue": "223"
        }
    }
}
```

last?testbed=ALL&testtype=ping

Deze call geeft voor elk testbed de laatste ping resultaten terug.


```
{   "177": {
        "testinstanceid": "1",
        "testtype": "ping",
        "testname": "ping voor testbed0",
        "log": "http:\/\/f4f-mon-dev.intec.ugent.be\/logs\/1\/1386",
        "timestamp": "2014-03-18 19:29:12.876569",
        "testbeds": [ "urn-testbed0" ],
        "results": { "pingValue": "166" }
    },"178": {
        "testinstanceid": "2",
        "testtype": "ping",
        "testname": "ping voor testbed1",
        "log": "http:\/\/f4f-mon-dev.intec.ugent.be\/logs\/2\/9739",
        "timestamp": "2014-03-18 19:29:12.889286",
        "testbeds": [ "urn-testbed1" ],
        "results": { "pingValue": "112" }
    },"187": {
        "testinstanceid": "11",
        "testtype": "ping",
        "testname": "ping voor testbed10",
        "log": "http:\/\/f4f-mon-dev.intec.ugent.be\/logs\/11\/4280",
        "timestamp": "2014-03-18 19:29:13.039074",
        "testbeds": [ "urn-testbed10" ],
        "results": { "pingValue": "106" }
    },"188": {
        "testinstanceid": "12",
        "testtype": "ping",
        "testname": "ping voor testbed11",
        "log": "http:\/\/f4f-mon-dev.intec.ugent.be\/logs\/12\/2792",
        "timestamp": "2014-03-18 19:29:13.055714",
        "testbeds": [ "urn-testbed11" ],
        "results": { "pingValue": "156" }
    }
    ...
}
```

last?testname=stitching2

Deze functie geeft een detail resultaat terug van de stitchingtest met naam 'stitching2'.

`Voorbeeld invoegen`

list?from=2014-03-18T19:29:00&till=2014-03-18T19:29:10&testtype=ping&testbed=urn-testbed1

Deze call geeft een lijst van de pingtests tussen 2014-03-18 19:29:00 en 2014-03-18 19u29:10 op testbed1 terug.


```
{
    "156": {
        "testinstanceid": "2",
        "testtype": "ping",
        "testname": "ping voor testbed1",
        "log": "http:\/\/f4f-mon-dev.intec.ugent.be\/logs\/2\/2306",
        "timestamp": "2014-03-18 19:29:09.152291",
        "testbeds": [
            "urn-testbed1"
        ],
        "results": {
            "pingValue": "66"
        }
    },
    "134": {
        "testinstanceid": "2",
        "testtype": "ping",
        "testname": "ping voor testbed1",
        "log": "http:\/\/f4f-mon-dev.intec.ugent.be\/logs\/2\/1991",
        "timestamp": "2014-03-18 19:29:05.413795",
        "testbeds": [
            "urn-testbed1"
        ],
        "results": {
            "pingValue": "24"
        }
    },
    ...
}
```

3.3 Calls - configuration

Deze calls hebben het doel om de configuratie van een test op te vragen.

3.3.1 Functies

Er zijn (uiteraard) ook een aantal functies voor het opvragen van de testconfiguratie.

- TestDefinition : Geeft de definitie van een testweer. Deze bevat informatie over de naam, het commando. Verder duidt deze definitie ook aan welke parameters er nodig zijn en welke waarden er terug gegeven worden. Deze waarden zijn echter niet ingevuld. Het is de bedoel om hier bijvoorbeeld te definiëren dat een pingtest bijvoorbeeld bestaat uit een ping commando en dat deze een time-out en een testbed moet meekrijgen. Samengevat kan gesteld worden dat dit de beschrijving van een type test is.
- TestInstance : Hierbij wordt een instance teruggegeven. Dit is de concrete invulling van een testDescription. Daarbij wordt de test gecombineerd met de ingevulde parameters. Zo zal hier ingevuld worden dat er op testbed2 om de 5 minuten een pingtest moet uitgevoerd worden.

Deze opsplitsing is noodzakelijk om flexibiliteit aan te bieden. Een ping test wordt op meerdere testbeds uitgevoerd. Door deze opbouw moet een ping test maar eenmalig gedefiniëerd worden. Als we de uitwerking vergelijking met het model-view-controller pattern, geeft de definition de klasse, de instance kan gezien worden als een object van die klasse.

3.3.2 Parameters

- testtype: Deze parameter duidt aan welke testtype opgevraagd wordt.
- testdefinitionId / testinstanceId: Deze parameter duidt aan welke definitie of instance opgevraagd wordt.

Het is ook mogelijk om als waarde ALL of een lijst van id's op te geven.

3.3.3 Voorbeelden

Hieronder worden een aantal mogelijke calls weergegeven. Voorlopig zijn deze voorbeelden beperkt tot het opvragen van data via http get requests.

testDefinition?testtype=stitch

Deze call geeft de beschrijving van een stitching test terug.

```
{   "stitch": {
        "testcommand": "stitch",
        "parameters": {
            "topology": {
                "type": "string",
                "description": "ring | line"
            },"testbedId": {
                "type": "testbedId[]",
                "description": "multiple testbeds for ping test"
            }
        },"return": {
            "callDeletes": {
                "type": "string", "description": "status of subtest"
            },"loginAndPing": {
                "type": "string", "description": "status of subtest"
            },"waitForAllReady": {
                "type": "string", "description": "status of subtest"
            },"callCreateSlivers": {
                "type": "string", "description": "status of subtest"
            },"callSCS": {
                "type": "string", "description": "status of subtest"
            },"initStitching": {
                "type": "string", "description": "status of subtest"
            },"createSlice": {
                "type": "string", "description": "status of subtest"
            },"generateRspec": {
                "type": "string", "description": "status of subtest"
            },"getUserCredential": {
                "type": "string", "description": "status of subtest"
            },"setUp": {
                "type": "string", "description": "status of subtest"
}}}}
```

testInstance?testbed=urn-testbed1,urn-testbed2

Hierbij worden alle testinstanties van testbed1 en testbed2 teruggegeven. Dit zijn dus de testen die effectief draaien op testbed1 en/of testbed2.


```
{    "2": {
        "testname": "ping voor testbed1",
        "testtype": "ping",
        "frequency": "60",
        "parameters": [
         {"testbedId": "urn-testbed1"},
            {"timeout": "300"}
        ]
    },"3": {
        "testname": "ping voor testbed2",
        "testtype": "ping",
        "frequency": "60",
        "parameters": [
            {"testbedId": "urn-testbed2"},
            {"timeout": "300"}
        ]
    },"18": {
        "testname": "stiching0",
        "testtype": "stitch",
        "frequency": "3600",
        "parameters": [
            {"testbedId": "urn-testbed2"},
            {"testbedId": "urn-testbed1"},
            {"testbedId": "urn-testbed0"},
            {"topology": "ring"}
        ]
    },
    ...
}
```

testInstance?testtype=ping

Deze call geeft alle geplande tests terug.

```
{
    "1": {
        "testname": "ping voor testbed0",
        "testtype": "ping",
        "frequency": "60",
        "parameters": [
            { "testbedId": "urn-testbed0" },
            { "timeout": "300" }
        ]
    },
    "2": {
        "testname": "ping voor testbed1",
        "testtype": "ping",
        "frequency": "60",
        "parameters": [
            { "testbedId": "urn-testbed1" },
            { "timeout": "300" }
        ]
    },
    "3": {
        "testname": "ping voor testbed2",
        "testtype": "ping",
        "frequency": "60",
        "parameters": [
            { "testbedId": "urn-testbed2" },
            { "timeout": "300" }
        ]
    },
    ...
}
```

testInstance?Plan=2

Deze call geeft de instance met id 2 terug.

`Voorbeeld invoegen`

Hoofdstuk 4

Ontwerp Databank

In dit hoofdstuk wordt de interface van de databank besproken.

4.1 Tabellen

De database zal bestaan uit volgende tabellen.

- testbeds: Deze tabel zal informatie over de testbeds bijhouden.
- testDefinitions: Deze tabel zal de testbeschrijving bijhouden.
- parameterDefinitions: Deze tabel houdt de parameters bij die de test nodig heeft.
- returnDefinitions: Deze tabel houdt de returnwaardes bij die een test genereert.
- testInstances: Hier worden de instanties van de testdefinities bijgehouden. Deze tabel zorgt voor de link tussen de testdefinitie en de ingevulde parameters.
- parameterInstances: Deze tabel houdt de ingevulde waarden van de parameters bij.
- results: Deze tabel zorgt voor de link tussen de testinstantie en de subresults.
- subResults: Deze tabel houdt de verschillende resultaten van een testbij.

Merk hier ook de opsplitsing testDefinition <=> testInstance. TestDefinition is een definitie van een test, testinstantie is de test met de ingevulde waarden.

4.2 Testbeds

Deze tabel houdt de verschillende testbeds bij en bevat volgende kolommen :

- testbedId: Het id van het testbed. Aangezien elk testbed een uniek urn heeft zal dit waarschijnlijk gebruikt worden als testbedid. Een nadeel is wel dat een urn moeilijk te onthouden is, wat gebruik van de service zal bemoeilijken.

- name: de naam van het testbed.

4.3 TestDefinitions

Deze tabel houdt de definities van de tests bij. Hierbij wordt een test abstract omschreven als een commando met een aantal parameters en een of meerdere return waarden. De tabel bevat volgende kolommen:

- testtype : Dit is de naam van een testtype en moet tevens uniek zijn.
- testcommand : Het commando (of script) dat uitgevoerd moet worden.

4.4 ParameterDefinitions

Deze tabel houdt bij welke parameters een test nodig heeft. Hierbij wordt de naam, het type en een beschrijving van een parameter opgeslagen. De tabel bevat volgende kolommen:

- testtype: Houdt bij voor welke testtype de parameters van toepassing zijn.
- parameterName: De naam van de parameter.
- parameterType: Het type van de parameter.
- parameterDescrition: De beschrijving van een parameter.

4.5 ReturnDefinitions

Deze tabel bevat de returnwaarden die bij een testtype horen. De tabel bevat volgende kolommen:

- testtype: Houdt bij voor welke testtype de returnwaarden van toepassing zijn.
- returnName: Naam van de returnwaarde.
- returnType: Type van de returnwaarde.
- returnDescription: Beschrijving van de returnwaarde.

4.6 TestInstances

Deze tabel zal later de testdefinition koppelen aan de tabel met parameters. In de eerste versie zal hij de waarden hardgecodeerd bevatten. In deze tabel wordt er bijgehouden der er om de 5 minuten een ping test uitgevoerd moet worden op een testbedX met een timeout Y. De tabel bevat volgende kolommen:

- testInstanceId: Een id om de testinstance mee aan te duiden.
- testtype: Geeft het testtype aan.
- testname: Geeft de naam van een test aan.
- Frequency : De frequentie waarmee de test uitgevoerd moet worden.

4.7 ParameterInstances

Deze tabel houdt de ingevulde waarden van de parameters bij voor een testinstantie. De tabel bevat volgende kolommen:

- testinstanceId: Geeft aan voor welke instantie de waarde geld.
- parameterName: De naam van de parameter.
- parameterValue: De waarde van de parameter.

4.8 Results

Deze tabel vormt de link tussen de testinstantie en de subResults. De tabel bevat volgende kolommen:

- resultid : Later gebruikt om detail weergave op te vragen van een resultaat.
- testInstanceId: Geeft aan van welke testinstantie deze resultaten afkomstig zijn.
- log : De link naar de link van de log file. Deze zal beschikbaar staan op het testbed, via de url is het mogelijk om hem op te vragen.
- timestamp : Geeft aan wanneer die resultaat is bepaald.

4.9 SubResults

Deze tabel bevat de ingevulde resultaten.

- resultId: Geeft aan bij welk resultaat deze waarden horen.
- name: Geeft de naam van het resultaat aan.
- value: Geeft de waarde voor het resultaat aan.

Hoofdstuk 5

Keuzes

In dit hoofdstuk worden belangrijke keuzes bij de uitwerking uitgelegd en gemotiveerd.

5.1 Formaat Returnwaarden

Er kan gekozen worden tussen xml en json. Het verschil tussen xml en json is dat xml een boomstructuur beschrijft. De json voorstel komt meer overeen met een hashmap. Xml is meer geschikt voor grote geavanceerde structuren. Aangezien het hier om eenvoudige monitoring informatie gaat, is het gebruik van xml af te raden. Bovendien is de xml-notatie van een object langer dan de overeenkomstige json-notatie. Ook is het parsen van json eenvoudiger. Bijgevolg zal hier dus geen xml, maar json gebruikt worden. Door de model-view-controller opbouw die gebruikt zal worden, kan xml-functionaliteit achteraf makkelijk toegevoegd worden. Dit is al voorzien door de parameter format, die standaard ingevuld wordt met json.

5.2 Structuur returnwaarden

De teruggegeven data zal ook generiek gemaakt worden als een soort geneste hash/array waarin alle waarden zitten. Dit geeft het voordeel dat we ons geen zorgen moeten maken over attributen die niet ingevuld zijn, deze worden dan gewoon weggelaten. Het dan ook aan de client om de teruggegeven data om te zetten naar klassen, indien nodig.

5.3 Databank technologie

5.4 webservice technologie