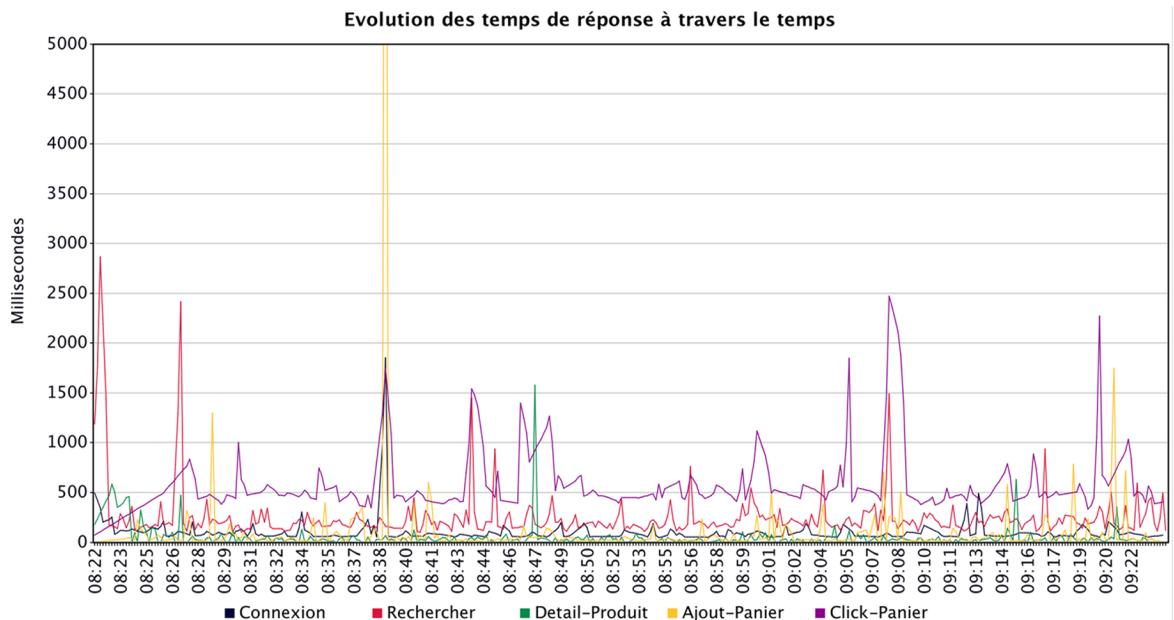


# Maîtriser Apache JMeter

Du test de charge  
à Devops



Antonio Gomes Rodrigues  
Bruno Demion (Milamber)  
Philippe Mouawad



# Maîtriser Apache JMeter

Du test de charge à Devops

Antonio Gomes Rodrigues, Bruno Demion (Milamber) et  
Philippe Mouawad

Ce livre est en vente à

<http://leanpub.com/maitriser-jmeter-du-test-de-charge-a-devops>

Version publiée le 2018-09-30

ISBN 978-2-9555036-1-4



Leanpub

Ce livre est publié par [Leanpub](#). Leanpub permet aux auteurs et aux éditeurs de bénéficier du Lean Publishing. [Lean Publishing](#) consiste à publier à l'aide d'outils très simples de nombreuses itérations d'un livre électronique en cours de rédaction, d'obtenir des retours et commentaires des lecteurs afin d'améliorer le livre.

© 2014 - 2018 Antonio Gomes Rodrigues, Bruno Demion (Milamber) et Philippe Mouawad

# Tweet ce livre !

S'il vous plaît aidez Antonio Gomes Rodrigues, Bruno Demion (Milamber) et Philippe Mouawad en parlant de ce livre sur [Twitter](#) !

Le tweet suggéré pour ce livre est :

Je viens d'acheter Maîtriser Apache JMeter : Du test de charge à #Devops par @ra0077, @milamberspace, @philmdot sur <https://leanpub.com/maitriser-jmeter-du-test-de-charge-a-devops>

Le hashtag suggéré pour ce livre est [#jmeter](#).

Découvrez ce que les gens disent à propos du livre en cliquant sur ce lien pour rechercher ce hashtag sur Twitter :

[#jmeter](#)

*Couverture et quatrième de couverture conçues par Cécile Platteeuw (C'grafic)*

# Table des matières

<b>Droits</b> . . . . .	<b>1</b>
<b>Présentation des auteurs</b> . . . . .	<b>2</b>
Antonio Gomes Rodrigues . . . . .	2
Bruno Demion (Milamber) . . . . .	2
Philippe Mouawad (Philippe M.) . . . . .	3
<b>L'écosystème d'Apache JMeter</b> . . . . .	<b>5</b>
Introduction . . . . .	5
Plugin polyvalent . . . . .	5
JMeter Plugins . . . . .	5
JMeter dans le cloud . . . . .	18
BlazeMeter . . . . .	19
Tricentis Flood . . . . .	23
Redline 13 . . . . .	28
OctoPerf (anciennement Jelly.IO) . . . . .	38
JMeter EC2 . . . . .	46
DevOps . . . . .	46
Aide à la supervision et au diagnostic . . . . .	46
Dynatrace APM . . . . .	47
Loadosophia.org . . . . .	54
D'autres protocoles . . . . .	60
UbikLoadPack . . . . .	60
DSL (Domain specific language) . . . . .	65
Ruby based DSL for JMeter . . . . .	66
Conclusion . . . . .	68

# Droits

Aucune partie de cette publication ne peut être reproduite, archivée ou transmise sous quelque forme ou par quelque moyen que ce soit (électronique, mécanique, par photocopie, sous forme enregistrée ou autre) sans autorisation préalable des auteurs.

Apache ActiveMQ, Apache Ant, Apache HTTP server, Apache JMeter et Apache Maven sont des marques déposées par la fondation Apache (ASF).

UbikLoadPack est une marque déposée d'Ubik-Ingénierie.

Tricentis Flood est une marque déposée par Tricentis.

OctoPerf est une marque déposée par OctoPerf.

BlazeMeter est une marque déposée par CA Technologies.

Byteman est une marque déposée par RedHat.

JProfiler est une marque déposée par ej-technologies GmbH.

Firefox est une marque déposée par Mozilla.

Spring et RabbitMQ sont des marques déposées par Pivotal Software.

Netbeans, Oracle et MySQL sont des marques déposées par Oracle Corporation.

JMeter Plugins est une marque déposée par Andrey Pokhilko.

Redline 13 est une marque déposée par RedLine13.

Dynatrace, PurePath sont des marques déposées de Dynatrace.

YourKit Java Profiler est une marque déposée par YourKit.

SoapUI est une marque déposée par SmartBear.

Swagger est une marque déposée par SmartBear.

Postman est une marque déposée par Postdot Technologies, Inc.

# Présentation des auteurs

## Antonio Gomes Rodrigues

Antonio Gomes Rodrigues est expert dans le domaine des performances applicatives depuis plus de 10 ans. Ses missions l'ont amené à travailler :

- Sur les performances des sites WEB à fort trafic
- Sur les performances d'une application pour courtiers
- Sur les performances de clients lourds, d'applications dans le cloud, d'application WEB, etc.
- Avec divers profilers : *JProfiler*, *Yourkit*, *Perfview*, etc.
- Avec divers APM : *Dynatrace*, *AppDynamics*, *Introscope*, *NewRelic*, etc.
- Avec divers outils de test de charge : *JMeter*, *LoadRunner*, etc.
- Dans diverses missions : tests de charge, mise en place de stratégies de performance, formations, audits de performance, diagnostics, etc.

Il partage ses connaissances de la performance applicative lors de conférences, sur son [blog](#)<sup>1</sup> et lors de relectures techniques de livre.

Il est actuellement 'committer' et membre PMC du projet [JMeter](#)<sup>2</sup> au sein de la fondation [Apache Software - ASF](#)<sup>3</sup>.

## Bruno Demion (Milamber)

Bruno Demion, plus connu dans la communauté JMeter sous le pseudonyme **Milamber** est un informaticien français expatrié au Maroc depuis 2002, vivant actuellement à Témara (à côté de Rabat). Il travaille dans une société d'expertises et de conseils

---

1. <http://arodrigues.developpez.com/>

2. <http://jmeter.apache.org/>

3. <http://www.apache.org/foundation/how-it-works.html#what>

technologiques, en tant qu'associé, architecte et expert technique senior sur les technologies Web et Cloud.

De par son travail et sa passion qu'est l'informatique, Milamber a de solides compétences dans le domaine des performances, la détection des problèmes (troubleshooting), la sécurité informatique ainsi que les architectures techniques pour les solutions Web et Cloud.

Depuis décembre 2003, il travaille avec JMeter pour effectuer des tests de charge dans diverses missions de performances et donne également des formations sur ce sujet. Il contribue autant que possible au projet JMeter sur son temps libre, notamment sur la traduction en français de l'interface graphique, des corrections d'anomalies ainsi que certaines évolutions (proxy https, nouvel arbre de résultats, barre d'icônes, etc.).

Il est actuellement 'committer', membre PMC et PMC Chair (secrétaire) du projet [JMeter](#)<sup>4</sup> au sein de la fondation [Apache Software - ASF](#)<sup>5</sup>. Il est également [ASF member](#)<sup>6</sup> officiel. Son ID Apache est [milamber](#)<sup>7</sup>.

Milamber a également un [blog](#)<sup>8</sup> personnel avec de nombreux articles et tutoriels sur JMeter, dont certains ont inspiré ce livre.

## Philippe Mouawad (Philippe M.)

Philippe Mouawad est expert technique en environnement Java EE et Web au sein de la société Ubik-Ingénierie. Il utilise JMeter depuis 2009 dans le cadre de missions de stabilisations, tests de charge de sites intranet ou eCommerce et de formations sur JMeter.

Il contribue à JMeter depuis 2009 d'abord sous forme de patches puis en tant que 'committer'. Parmi ses principales contributions, l'**extracteur CSS/Jquery**, le **Backend Listener** (permettant d'interfacer entre autres *Graphite*, *InfluxDB* ou *ElasticSearch*), l'optimisation des performances du noyau et sa stabilisation et diverses améliorations ergonomiques, à son actif plus de 200 bugs/améliorations.

---

4. <http://jmeter.apache.org/>

5. <http://www.apache.org/foundation/how-it-works.html#what>

6. <http://www.apache.org/foundation/how-it-works.html#roles>

7. <http://people.apache.org/~milamber/>

8. <http://blog.milamberspace.net/>

Il participe également au projet *JMeter-Plugins*, parmi ses contributions figurent **Redis DataSet** et **Graphs Generator Listener**.

Il est actuellement ‘committer’ et membre PMC du projet **JMeter**<sup>9</sup> au sein de la fondation **Apache Software - ASF**<sup>10</sup>. Son ID Apache est **pmouawad**<sup>11</sup>.

Il est également développeur principal de la solution *Ubik Load Pack*, ensemble de plugins prenant en charge des protocoles non supportés nativement par JMeter. Enfin, il contribue au **blog de Ubik-Ingénierie**<sup>12</sup>.

---

9. <http://jmeter.apache.org/>

10. <http://www.apache.org/foundation/how-it-works.html#what>

11. <http://people.apache.org/~pmouawad/>

12. <http://www.ubik-ingenierie.com/blog/>

# L'écosystème d'Apache JMeter

JMeter dispose d'un éco système foisonnant, aussi comme le dit Bregson, « Choisir, donc exclure ». Nous avons donc dû faire des choix subjectifs.

À vous de découvrir et approfondir les bibliothèques que nous avons écartées.

## Introduction

Comme nous avons pu le voir dans les chapitres précédents, JMeter est assez riche nativement, mais les technologies et les besoins évoluant, JMeter peut dans certains cas ne pas suffire.

Difficulté résolue par son écosystème dont l'existence est possible grâce à sa licence Apache et son architecture modulaire.

Dans ce chapitre, nous verrons les principaux plugins, services et outils permettant de compléter JMeter afin de répondre à tous les défis que nous rencontrerons.

## Plugin polyvalent

Commençons par le plugin le plus connu et le plus polyvalent : JMeter Plugins.

### JMeter Plugins

[JMeter Plugins](#)<sup>13</sup> est un ensemble de plugins gratuits et open source améliorant certaines parties de JMeter.

Pour cela il va ajouter un certain nombre de :

- récepteurs

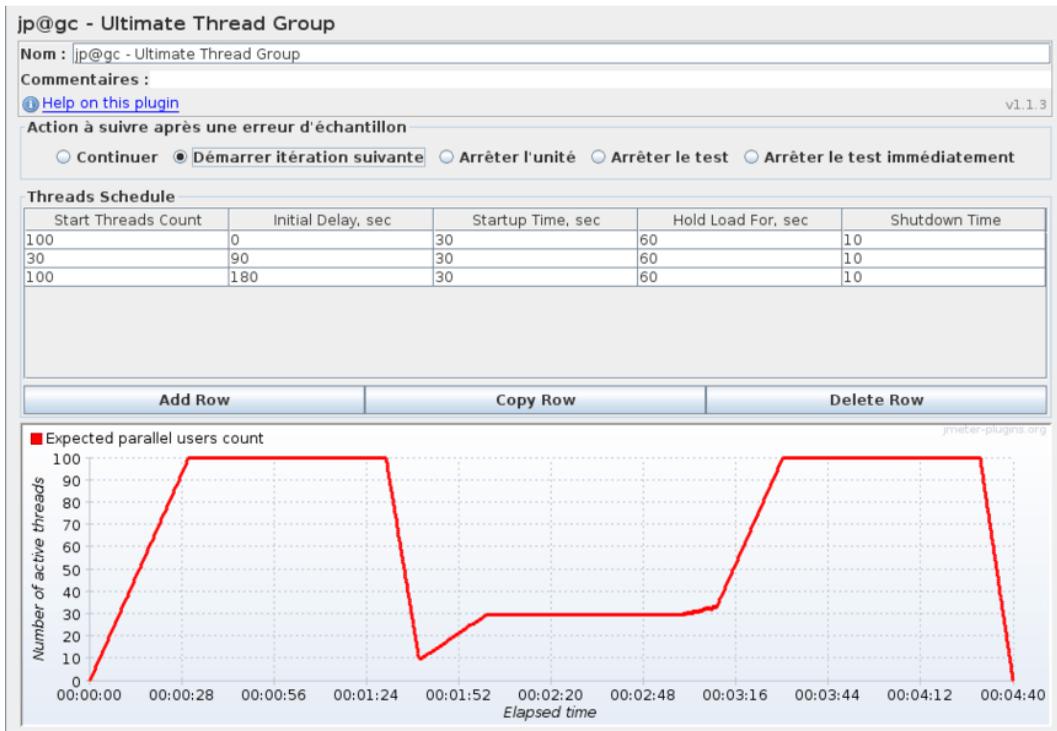
---

13. <http://jmeter-plugins.org/>

- graphiques
- fonctions
- échantillons
- outils
- groupe d'unités
- compteurs de temps
- assertions
- protocoles
- etc.

Comme nous pouvons le voir, la liste est longue, nous ne présenterons donc que certains plugins.

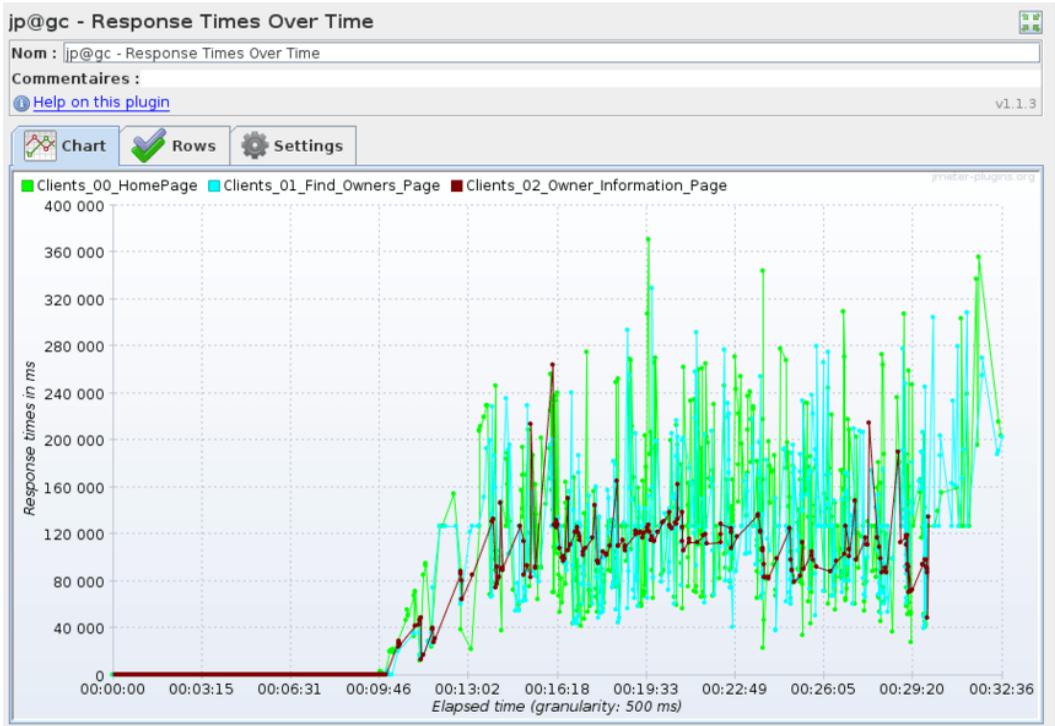
Commençons par le groupe d'unités **Ultimate Thread Group**. Ce plugin va nous permettre de contrôler facilement et visuellement notre injection.



Ultimate Thread Group

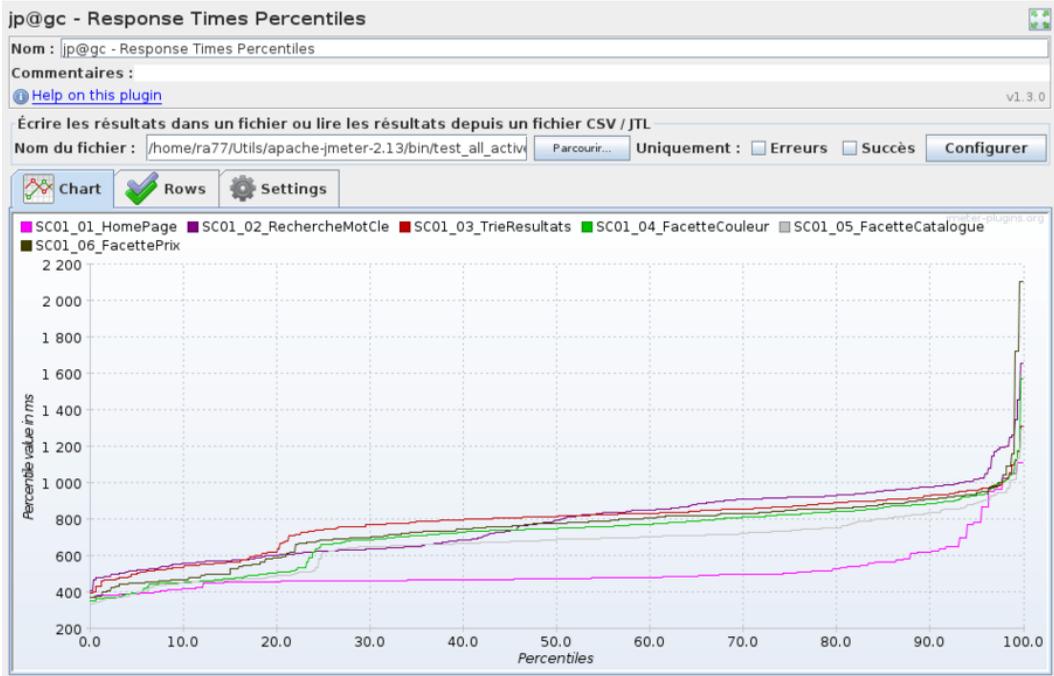
Regardons d'un peu plus près le point fort de JMeter Plugins : les graphiques.

**Response Times Over Time** nous permet de suivre l'évolution des temps de réponse au fil du tir.



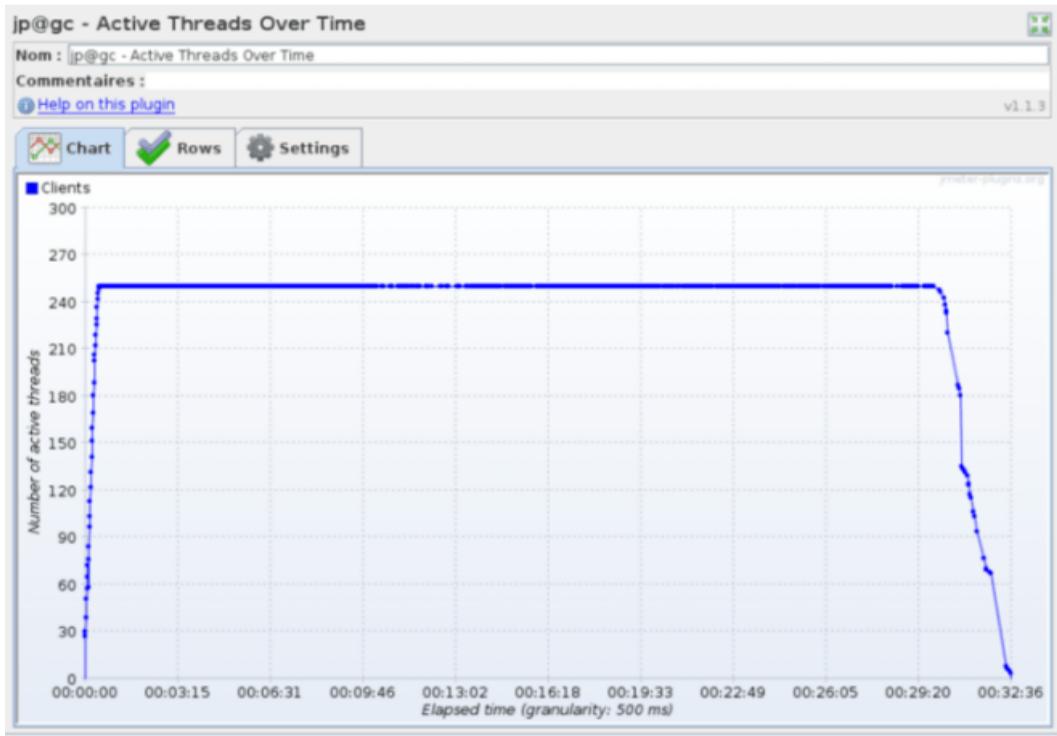
**Response Times Over Time**

De même pour **Response Times Percentiles**, mais concernant les percentiles.



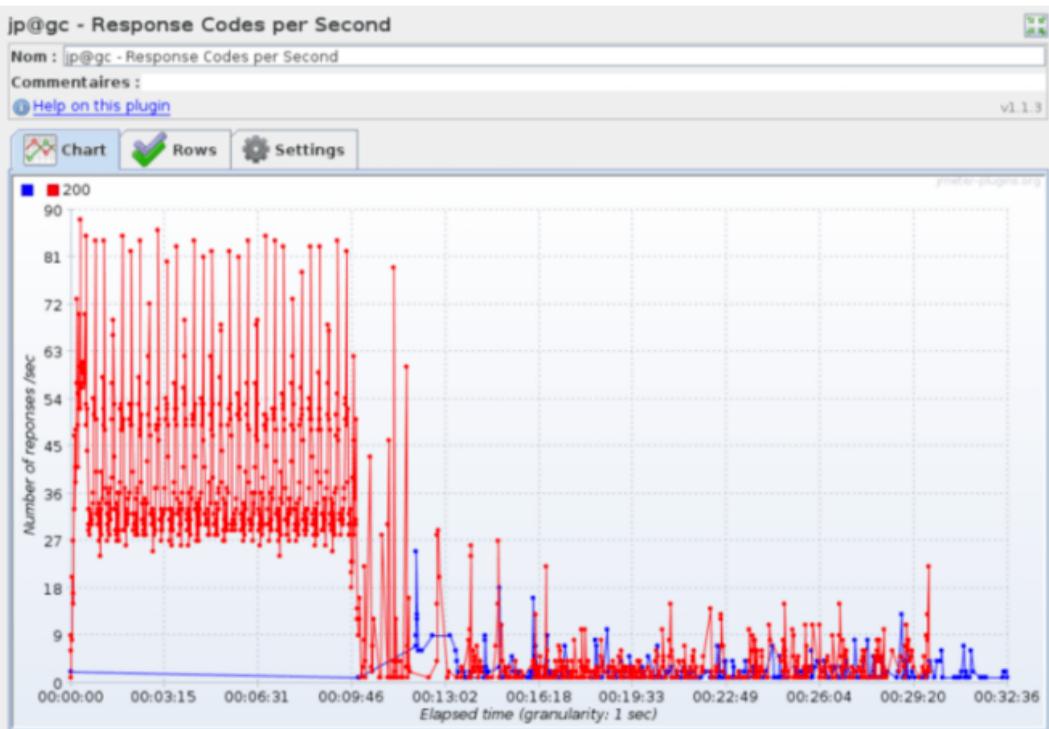
Response Times Percentiles

La même chose pour suivre le nombre d'utilisateurs virtuels avec le graphique **Active Threads Over Time**.



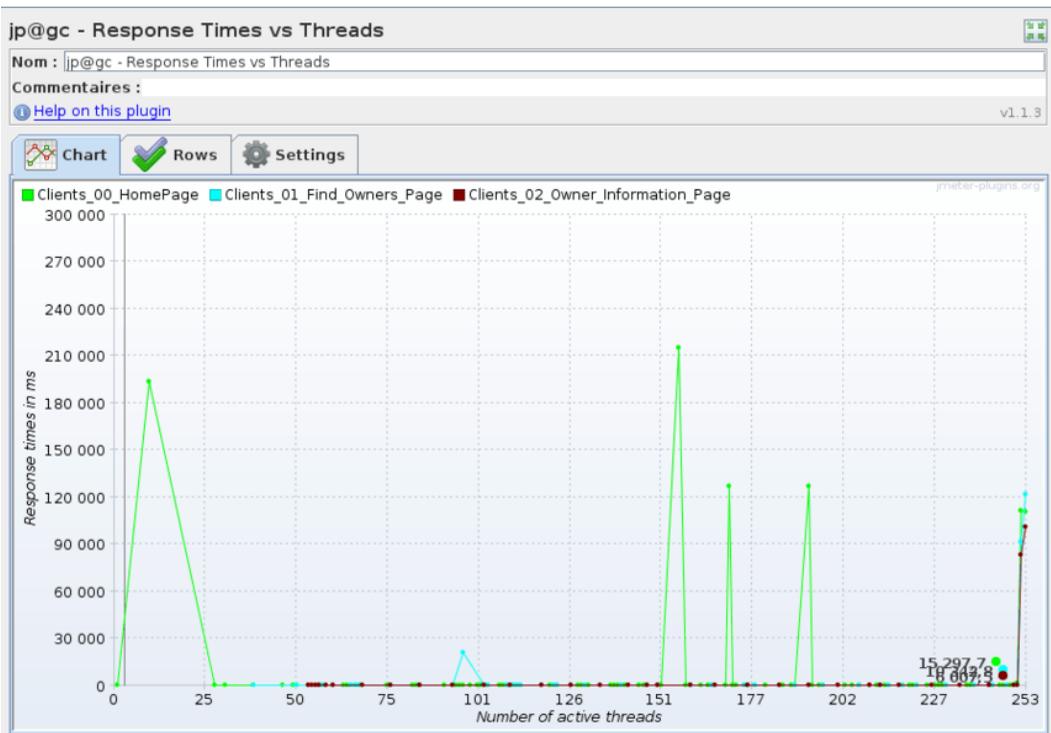
Active Threads Over Time

Si nous voulons suivre l'évolution des codes de retour HTTP au fil du tir, nous utiliserons **Response Codes per Second**.



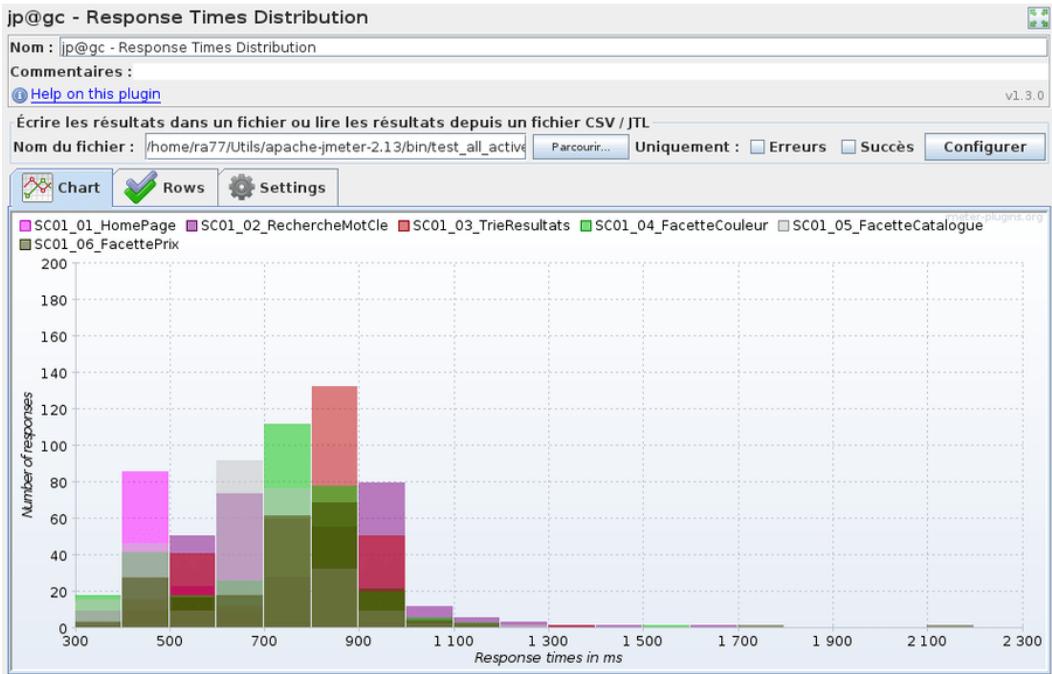
Response Codes per Second

Pour vérifier l'impact du nombre d'utilisateurs virtuels sur les temps de réponse, il y a le graphique **Response Times vs Threads**.



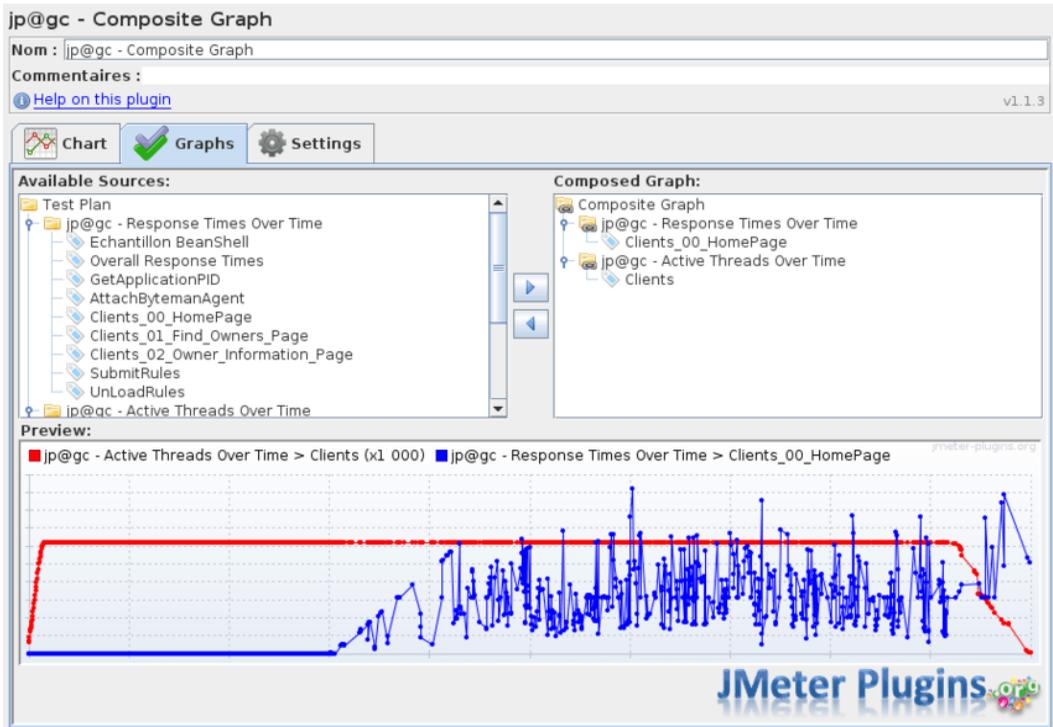
Response Times vs Threads

Pour la répartition des temps de réponse, nous utiliserons **Response Times Distribution**.



Response Times Distribution

Si cela ne suffit toujours pas, il est possible de superposer les graphes à l'aide des données de tous les autres graphes en utilisant **Composite Graph**.



Composite Graph

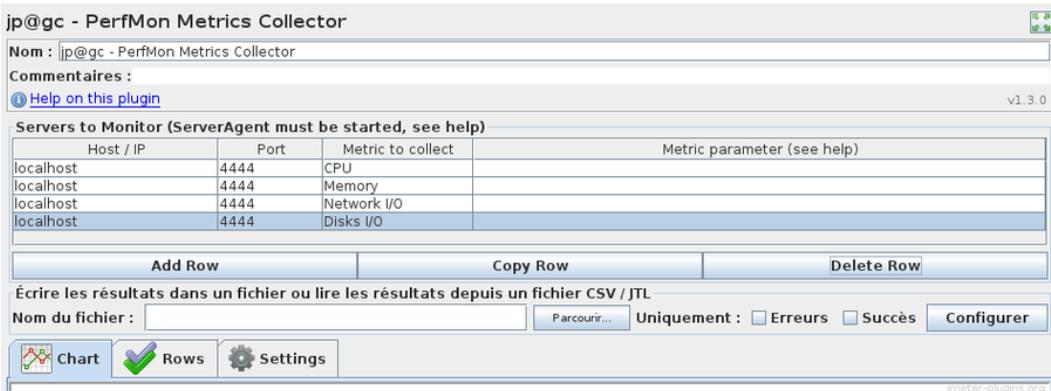
Pour l'instant, nous avons utilisé des graphiques se basant sur les données du test, mais il est possible de récupérer des données externes.

Comme des données JMX (par exemple de votre serveur d'application Java) à l'aide de JMXMon Samples Collector.



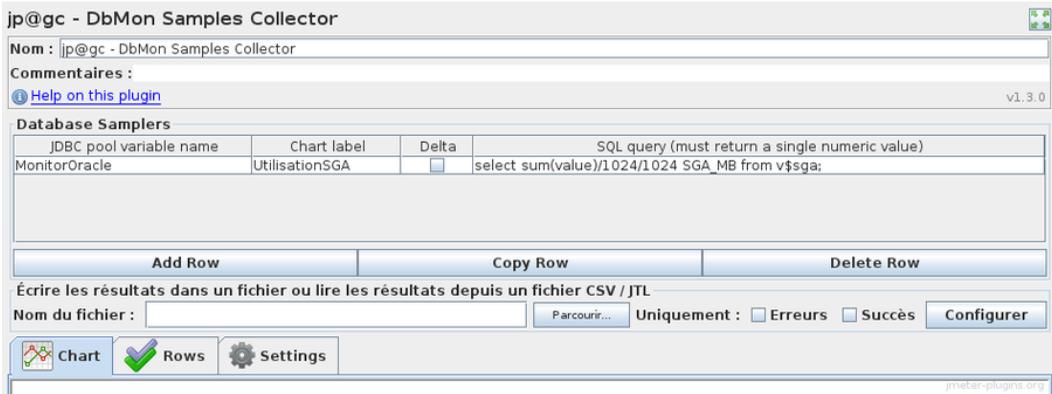
### JMXMon Samples Collector

Et pour savoir si les problèmes de performance de l'application testée viennent de l'infrastructure, nous pourrions utiliser **PerfMon Metrics Collector** afin de récupérer les métriques systèmes (CPU, réseau, mémoire, etc.).



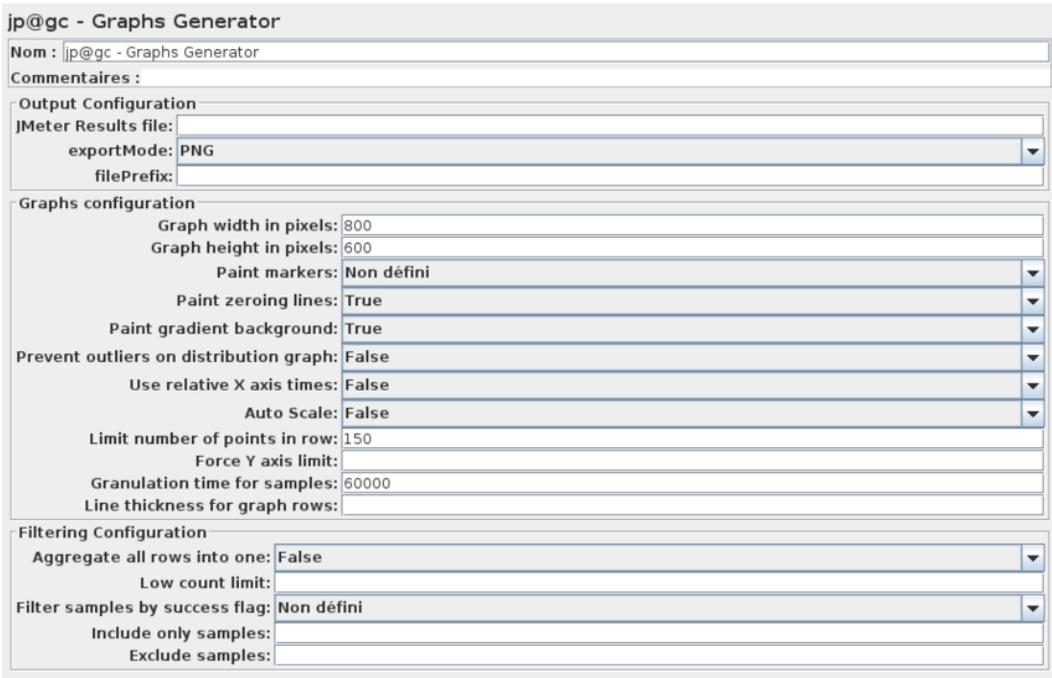
### PerfMon Metrics Collector

Si'il y a un problème avec la base de données, **DbMon Samples Collector** pourra nous aider.



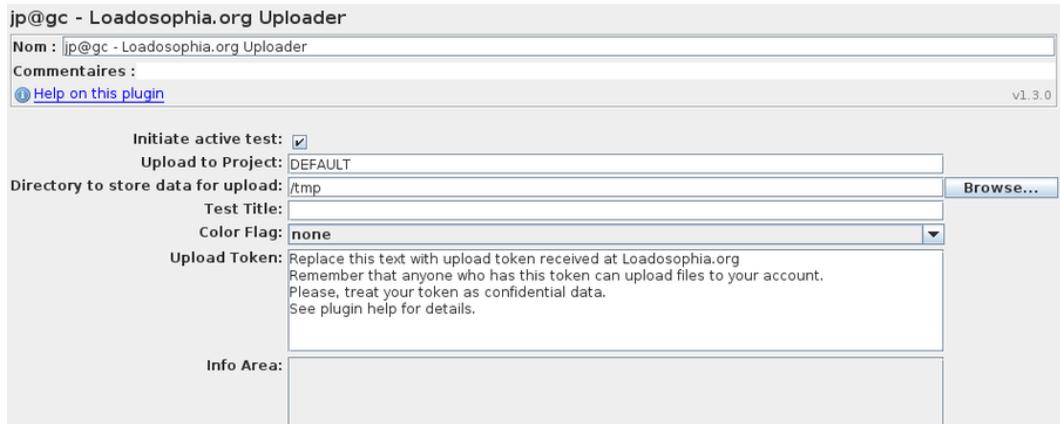
### DbMon Samples Collecto

Le tout avec la possibilité de générer les graphiques voulus à la fin de notre test de charge avec **Graphs Generator**. Méthode préconisée puisque le mode GUI doit être dédié au scripting.



### Graphs Generator

Une autre option consiste à uploader ses résultats sur Loadosophia.org à l'aide de **Loadosophia.org Uploader**.

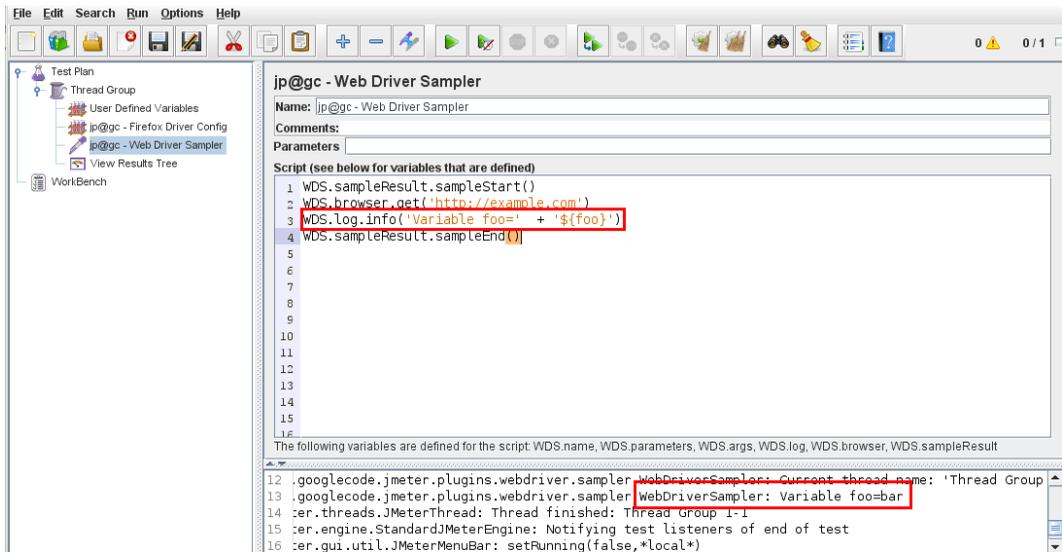


The screenshot shows the configuration window for the 'Loadosophia.org Uploader' plugin. The title bar reads 'jp@gc - Loadosophia.org Uploader'. Below the title bar, there is a text field for 'Nom' containing 'jp@gc - Loadosophia.org Uploader' and a 'Commentaires' section with a link to 'Help on this plugin'. The main configuration area includes a checked 'Initiate active test' checkbox, an 'Upload to Project' dropdown set to 'DEFAULT', and a 'Directory to store data for upload' field set to '/tmp' with a 'Browse...' button. Other fields include 'Test Title', a 'Color Flag' dropdown set to 'none', and an 'Upload Token' field with a text area containing instructions: 'Replace this text with upload token received at Loadosophia.org. Remember that anyone who has this token can upload files to your account. Please, treat your token as confidential data. See plugin help for details.' At the bottom, there is an empty 'Info Area'.

### Loadosophia.org Uploader

Si nous souhaitons connaître le ressenti utilisateur dans le browser (les temps de réponse JMeter n'intègrent pas le rendu dans le browser), nous pourrions utiliser dans notre test :

- L'élément **HTTP Request** pour injecter massivement la charge
- L'élément **WebDriverSampler** pour obtenir les temps dans le browser



### Web Driver Sampler

Une problématique qui peut se poser dans le cadre de tirs distribués avec JMeter est la distribution du jeu de données sur les injecteurs. Une solution à ce problème consiste à utiliser Redis et le plugin **Redis DataSet**. Il est même possible de garantir qu'une donnée une fois utilisée est supprimée du jeu de test.

RedisDataSet	
Name:	RedisDataSet
Comments:	
<b>Redis data configuration</b>	
Redis key:	ccList
Variable Names (comma-delimited):	cardNumber
Delimiter (use '\t' for tab):	,
Wether Getting a row removes it from list:	RANDOM_REMOVE
<b>Redis connection configuration</b>	
Redis server host:	localhost
Redis server port:	6379
Timeout for connection in ms:	2000
Password for connection:	
Database:	0
<b>Redis Pool Configuration</b>	
minIdle:	0
maxIdle:	10
maxActive:	20
maxWait:	30000
whenExhaustedAction:	GROW
testOnBorrow:	False
testOnReturn:	False
testWhileIdle:	False
timeBetweenEvictionRunsMillis:	30000
numTestsPerEvictionRun:	0
minEvictableIdleTimeMillis:	60000
softMinEvictableIdleTimeMillis:	60000

### Redis Plugin

Comme vous pouvez le voir, cet ensemble de plugins couvre un large périmètre et est très utile dans la vie de tous les jours d'un utilisateur de JMeter.

## JMeter dans le cloud

Cette catégorie répondra au problème d'infrastructure nécessaire pour l'injection de la charge.

En effet lors de certains tests, il peut être nécessaire d'avoir beaucoup de puissance afin de simuler une charge importante.

Il peut être également nécessaire de distribuer l'injection depuis plusieurs endroits du pays ou de la planète.

Dans d'autres cas, pour des tests ponctuels, l'achat de serveurs pour l'injection n'est pas justifié.

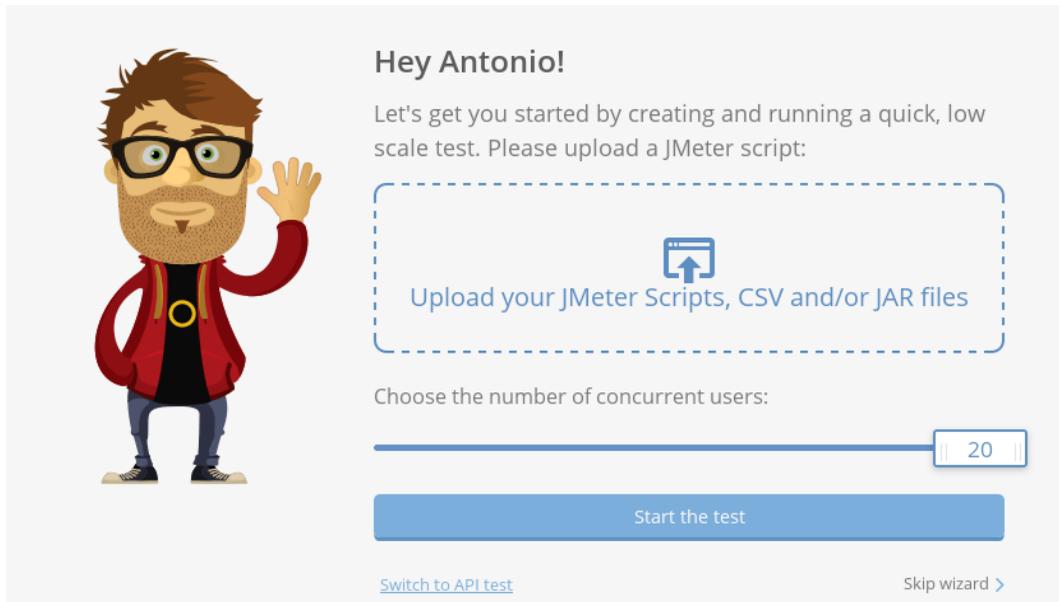
Pour cela, une des solutions possibles est l'utilisation de services commerciaux dans le cloud.

## BlazeMeter

À noter que [BlazeMeter](http://blazemeter.com/)<sup>14</sup> est un contributeur actif du projet Apache JMeter à travers les contributions d'Andrei Pokhilko (responsable du projet JMeter-Plugins).

L'utilisation de BlazeMeter est assez simple.

Lorsque nous nous connectons sur le site pour créer un test, un assistant apparaît.



Hey Antonio!

Let's get you started by creating and running a quick, low scale test. Please upload a JMeter script:

Upload your JMeter Scripts, CSV and/or JAR files

Choose the number of concurrent users:

20

Start the test

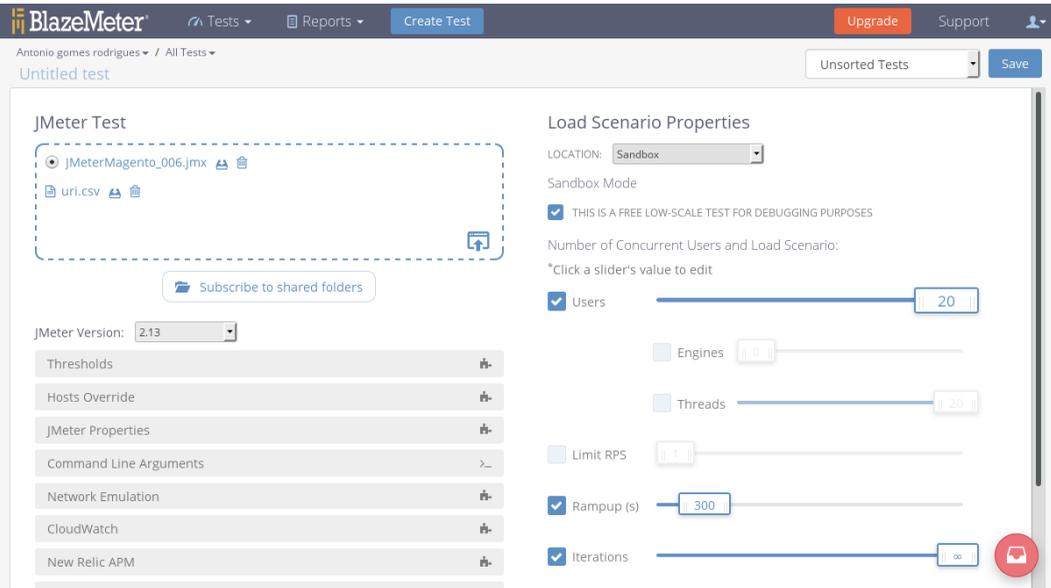
[Switch to API test](#) [Skip wizard >](#)

### Assistant BlazeMeter

Dans notre cas, nous n'allons pas l'utiliser. Un écran très complet nous permet de créer notre test.

---

14. <http://blazemeter.com/>

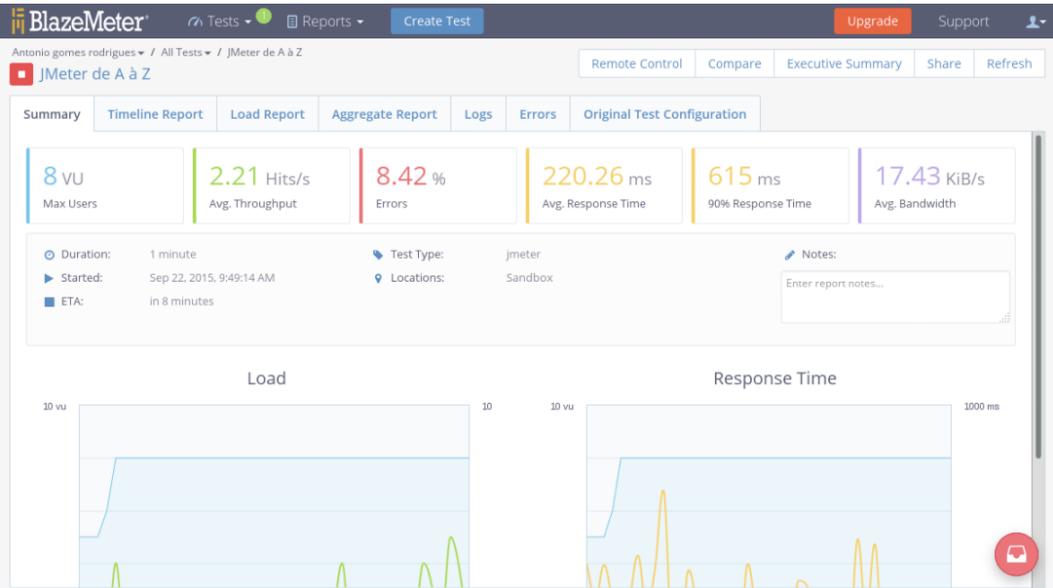


### BlazeMeter - Création d'un test

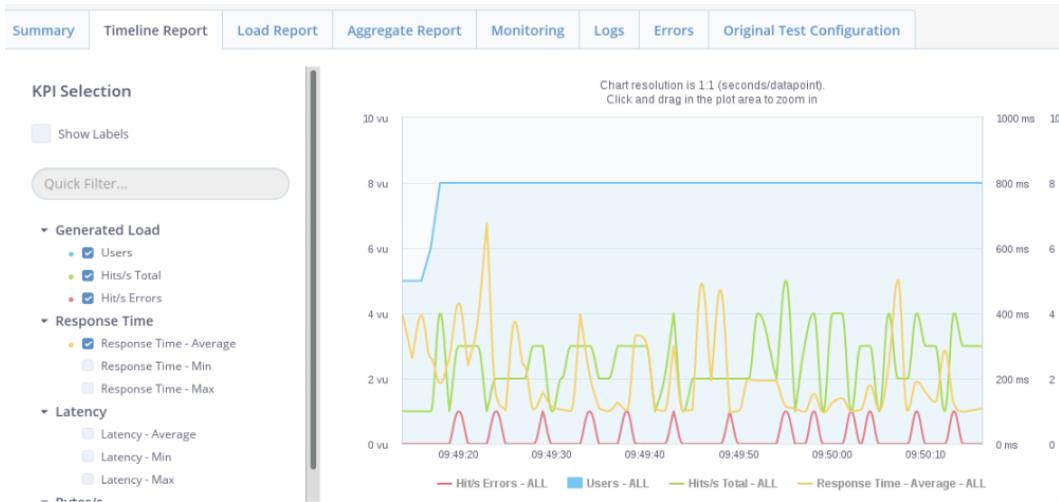
Nous pouvons :

- Importer notre script (fichier jmx) avec son jeu de données (fichier CSV)
- Régler le nombre d'utilisateurs et leur localisation géographique
- Régler notre plan de test (durée du test, durée de la montée en charge, etc.)
- Simuler une bande passante réseau particulière (4G, ADSL, Fibre, etc.)
- Nous interfacier avec des outils d'APM (Newrelic, Dynatrace)
- Surcharger des valeurs de propriétés
- Etc.

Lançons notre test et suivons son évolution en temps réel.



BlazeMeter - Suivi d'un test



BlazeMeter - Suivi d'un test

Label	# Samples	Avg. Latency	Avg. Response Time	Geo. Mean Response Time	StDev	90% Line	95% Line	99% Line	Min	Max	Avg. Bandwidth (Bytes/s)	Avg. Throughput (Hits/s)	Error %	Duration (hh:mm:ss)
ALL	256	1 ms	171.92 ms	137.1 ms	163.79 ms	288 ms	615 ms	887 ms	94 ms	916 ms	19868.62	2.49	8.98	00:01:43
SC01_01_HomePage	96	1 ms	129.43 ms	118.96 ms	66.92 ms	263 ms	279 ms	394 ms	95 ms	394 ms	6849.36	0.93	0	00:01:43
SC01_02_Univers	92	1 ms	126.05 ms	114.89 ms	92.09 ms	181 ms	256 ms	284 ms	94 ms	898 ms	6617.36	0.94	25	00:01:38
SC01_03_SousUnivers	68	1 ms	293.97 ms	212.77 ms	250.81 ms	716 ms	878 ms	908 ms	96 ms	916 ms	7289.29	0.72	0	00:01:35

### BlazeMeter - Suivi d'un test

Une supervision des injecteurs est proposée, ce qui est pratique pour valider de la pertinence de nos résultats.



### BlazeMeter - Supervision des injecteurs

Une fois notre test fini, un rapport peut être généré.



## LOAD TEST REPORT

JMeter de A à Z



Report created by Antonio gomes rodrigues  
Date of Run: Tue, 09/22/2015 — 09:48  
9 minutes test master

HIDE SUMMARY

### DESCRIPTIVE SUMMARY / CONCLUSIONS



Average Throughput

**3**

Hits/s



Avg. Response Time

**126**

Milliseconds



90% Response Time

**182**

Milliseconds



Error Rate

**9.14**

%

### TOP 5 SLOW RESPONSES 5

Request	% of executions	Avg Time	90% Time	Max Time
SC01_03_SousUnivers	27.08%	152.891 ms	268 ms	916 ms
SC01_01_HomePage	36.63%	120.719 ms	257 ms	394 ms
SC01_02_Univers	36.3%	111.805 ms	113 ms	898 ms

### TOP 5 ERRORS 5

BlazeMeter - Rapport de test

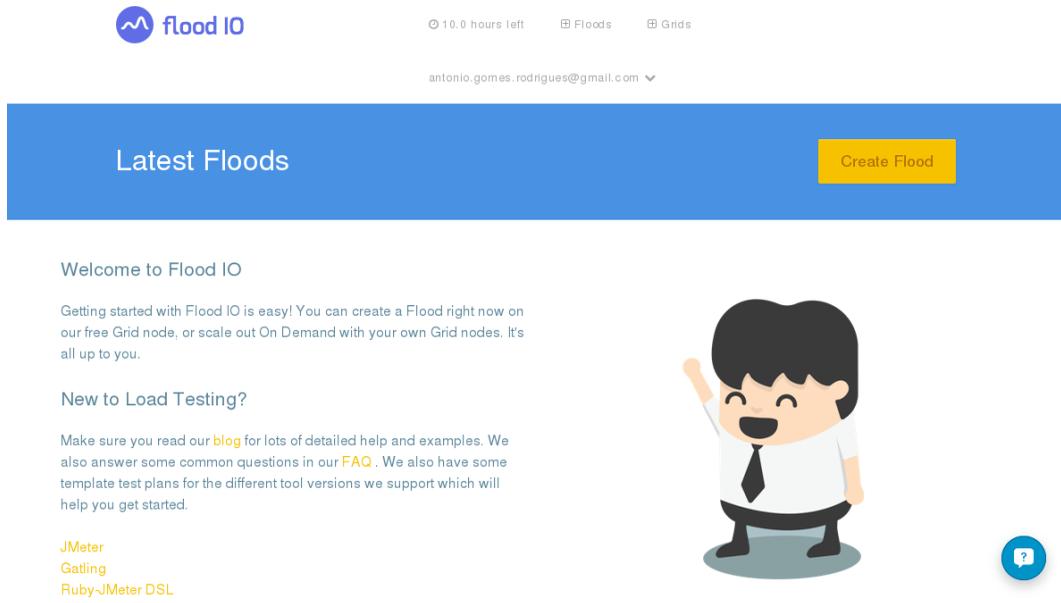
## Tricentis Flood

Tricentis Flood<sup>15</sup> est une autre solution qui utilise JMeter pour exécuter des tests de charge dans le cloud.

À noter que Flood.IO a développé un DSL pour JMeter (<https://github.com/flood-io/ruby-jmeter>) qu'il est possible d'utiliser directement dans l'outil.

15. <http://flood.io/>

Son interface est très simple d'utilisation.



### Tricentis Flood

Nous importons d'abord notre script accompagné de son jeu de données.  
Puis nous définissons les paramètres de notre test.

# Create Flood

Upload Files

JMeterMagento\_006.jmx  
Remove file

uri.csv  
Remove file

[+ or edit steps](#)

Tool

JMeter 2.13

Name

Apache JMeter de A à Z

Grids ?

1 Grid Selected

Threads

8

Rampup

8

Duration

600

Validation ?

Enabled

[+ advanced](#)

Start Flood

Cancel

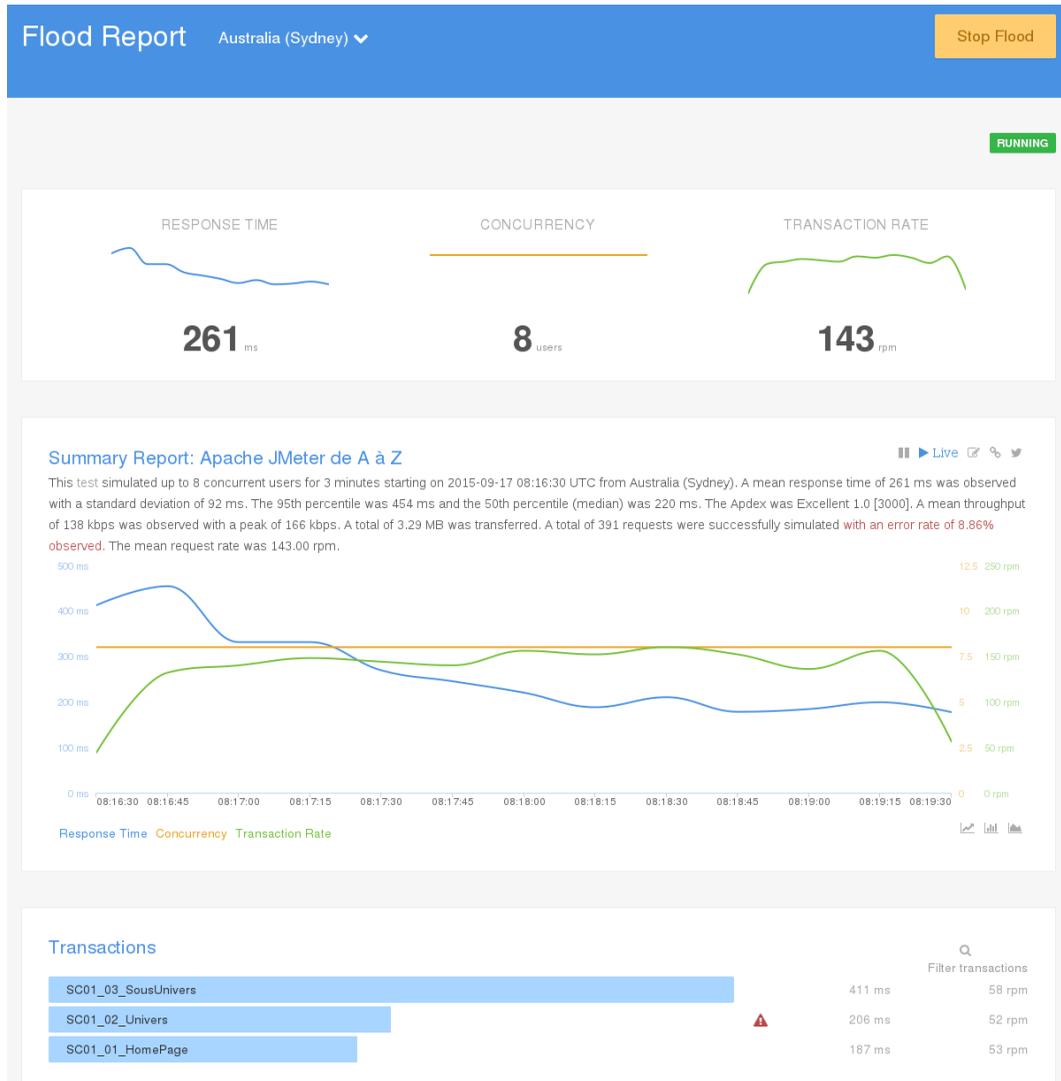
Si nécessaire, la partie *advanced* nous permet d'affiner notre test (simuler une bande passante réseau particulière, surcharger des valeurs de propriétés, etc.).

[+ advanced](#)

Project	<input type="text"/>
Tags	<input type="text" value="optional"/>
Network Emulation	<input type="text" value="None"/>
Notes	<input type="text" value="optional"/>
Privacy	<input type="text" value="Private. Account users can view results."/>
Hosts	<input type="text" value="example.com/0.0.0.0, ..."/>
Parameters	<input type="text" value="-Dparam=value, ~Jparam=value, ..."/>
URL	<input type="text" value="optional"/>
Start	<input type="text"/> 
Archive Results ?	<input checked="" type="checkbox"/> Enabled

### Tricentis Flood - Création d'un test : advanced

Lançons notre test et suivons son exécution en temps réel.



### Tricentis Flood - Suivi d'un test

Une supervision des injecteurs est proposée.

**Grid**  
Australia (Sydney)  
ap-southeast-2a

**1 Grid Node**  
m3.xlarge  
celestial-farm

Shared  
Stop Manually

CPU MEM DISK

### Tricentis Flood - Supervision des injecteurs

À la fin de notre test, nous pouvons télécharger notre rapport ainsi que les fichiers de résultats au format CSV et fichiers de log.

#2 Apache JMeter de A à Z **FINISHED** Repeat Started 17 Sep 2015 08:16 UTC for 5 minutes ACTIONS

Response Time	Concurrency	Throughput	Apdex
268 ms	8 users	102 kbps	1.0 [3000]

JMETER x AUSTRALIA (SYDNEY) x

- View
- Edit
- Stop
- Delete
- Repeat
- Download
- Archived Results
- Summary Results
- Files
- JMeterMagento\_006.jmx
- uri.csv

Blog Help Privacy Terms Status

### Tricentis Flood - Rapport de test

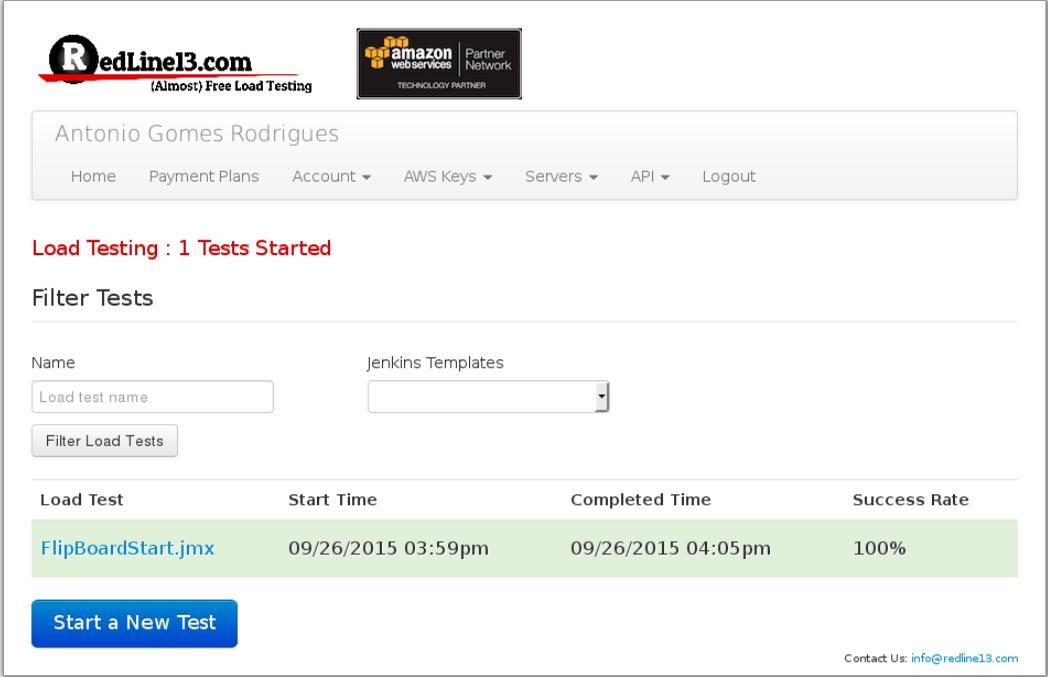
## Redline 13

**Redline 13**<sup>16</sup> est une autre solution un peu particulière qui utilise JMeter pour exécuter des tests de charge dans le cloud. En effet, il va utiliser notre propre clé AWS (cloud Amazon et bientôt d'autres) afin d'instancier des serveurs et y déployer JMeter et tout le nécessaire (scripts, plugins, jeux de données, etc.) pour exécuter notre test de charge.

La première chose à faire est d'entrer notre clé AWS.

16. <https://www.redline13.com/>

Une fois cela fait, nous nous retrouvons sur une page d'accueil assez simple avec tous les tests exécutés précédemment.



The screenshot shows the Redline13.com dashboard. At the top left is the Redline13.com logo with the tagline "(Almost) Free Load Testing". To the right is the Amazon Web Services Partner Network logo. Below the logos is a user profile section for "Antonio Gomes Rodrigues" with a navigation menu containing links for Home, Payment Plans, Account, AWS Keys, Servers, API, and Logout. A red notification banner states "Load Testing : 1 Tests Started". Below this is a "Filter Tests" section with a search box for "Name" (containing "Load test name") and a dropdown menu for "Jenkins Templates". A "Filter Load Tests" button is present. A table displays test results with columns for Load Test, Start Time, Completed Time, and Success Rate. The table contains one entry: "FlipBoardStart.jmx" with a start time of "09/26/2015 03:59pm", a completed time of "09/26/2015 04:05pm", and a success rate of "100%". A blue "Start a New Test" button is located below the table. In the bottom right corner, there is a contact link: "Contact Us: info@redline13.com".

Load Test	Start Time	Completed Time	Success Rate
<a href="#">FlipBoardStart.jmx</a>	09/26/2015 03:59pm	09/26/2015 04:05pm	100%

### Redline 13 - Page d'accueil

Cliquons sur le bouton « Start a New Test » pour commencer notre test.

Comme nous le voyons, plusieurs types de tests sont possibles :

- Test d'URL
- Test écrit en PHP ou NodeJS
- Test JMeter
- Test Gatling

## Start a Load Test



### Redline 13 - Type de test

Choisissons JMeter comme type de tests.

Ici, nous pouvons choisir :

- Notre script JMeter
- Le jeu de données associé (ici le fichier uri.csv)
- Le nombre de serveurs à instancier
- La version de JMeter
- Le nom de notre test

Start a Load Test

Help

Simple Test Custom Test **Jmeter Test** Gatling Test

This Test Type allows you to scale out running [JMeter](#) Tests in your cloud environment.

- Step 1 - Upload your JMX File
- Step 2 - Select how many servers you want to run your test from
- Step 3 - Optionally select size and location of your servers
- Step 4 - Optionally select if you would like to gather your results files.

For more information, click the Help Button on the top right.  
Have a question, ask in the [forums](#)

JMX File  JMeterMagento\_006.jmx Number of Servers  Servers

Include CSV data files, Plugins, Libraries, or other files.  
They will be placed in same directory as JMX

uri.csv  split ✕  
  Split

JMeter Version

Save Response Output and Calculate Percentiles  
 Save the response output from individual tests and calculate percentiles. ([Pro Version](#))

[Advanced Cloud Options](#)

[Advanced JMeter Test Options](#)

**Max Server Cost: \$0.07/hr**  
Does not include Data Transfer

Redline 13 - Création de notre plan de test

Dans la partie avancée du plan de test, il est possible de choisir les localisations AWS des serveurs et ainsi inclure plusieurs localisations si nous avons plus d'un serveur à instancier. Ceci nous permet potentiellement de voir le ressenti utilisateur en fonction du réseau, ce genre de besoins est plutôt réservé à des sites WEB déployés à l'international.

**Advanced Cloud Options**

**Tag Your AWS Instances**

Key  Value

**Load Agents**

Location:

Number of Servers:

Size:

Disk Size:  GB

Pro Version Only

Subnet ID:

Associate Public IP Address  
 *If running in a VPC subnet, associate a public IP address with this instance. If you're subnet has a route to an internet gateway, you should check this box. If the instance will be going through a NAT server, do NOT check this box.*

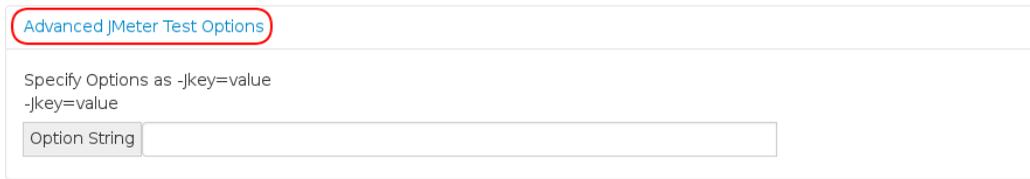
Security Group IDs:

To include multiple security groups, separate groups with a comma. If not set, the default security group is used.

Optional Description:

**Redline 13 - Création de notre plan de test - options avancées**

Enfin, nous pouvons spécifier des options JMeter à passer aux injecteurs :



Advanced JMeter Test Options

Specify Options as -jkey=value  
-jkey=value

Option String

### Redline 13 - Création de notre plan de test - options avancées JMeter

Notons au passage que nous pouvons sauvegarder notre plan de test comme modèle pour l'utiliser dans Jenkins avec le plugin Redline13.

Load Test: **LivreJMeter** Jenkins Enabled

[Clone Test](#)  
Beta Feature

This Load Test has been saved as a template!

### Details

#### Description

JMeter Test: JMeterMagento\_006.jmx

#### Test Key

f49402ba69644cc0589b6c7b7b3c3ca1

#### Language

jmeter

#### Test Server Specs

Size	Max Price	Location	No. Servers	Approx Users
m3.medium	\$0.0670 (On Demand)	Virginia	1	1

#### Load Resources (CSS, JS, images, etc.)

No

#### Created

10/05/15 08:15:06am

#### Comments

N/A

Want to get started with Jenkins and Redline13?

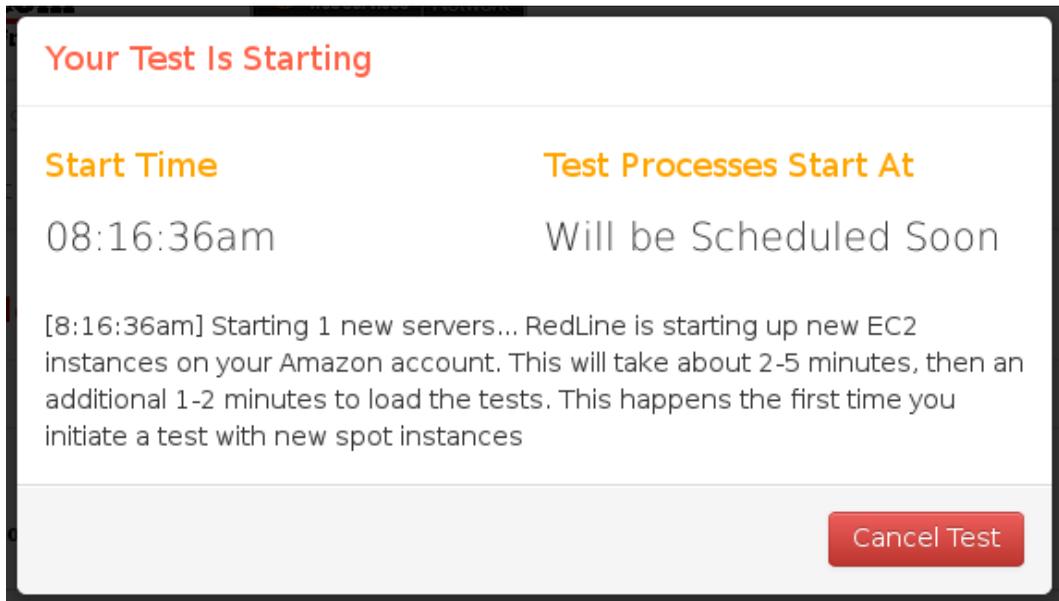
Download the Jenkins Plugin and install into your Jenkins instance.



[Download Now](#)

**Redline 13 - Modèle personnalisé pour Jenkins**

Exécutons notre test.



**Your Test Is Starting**

<b>Start Time</b>	<b>Test Processes Start At</b>
08:16:36am	Will be Scheduled Soon

[8:16:36am] Starting 1 new servers... RedLine is starting up new EC2 instances on your Amazon account. This will take about 2-5 minutes, then an additional 1-2 minutes to load the tests. This happens the first time you initiate a test with new spot instances

Cancel Test

Redline 13 - Démarrage de notre test

Et suivons-le.

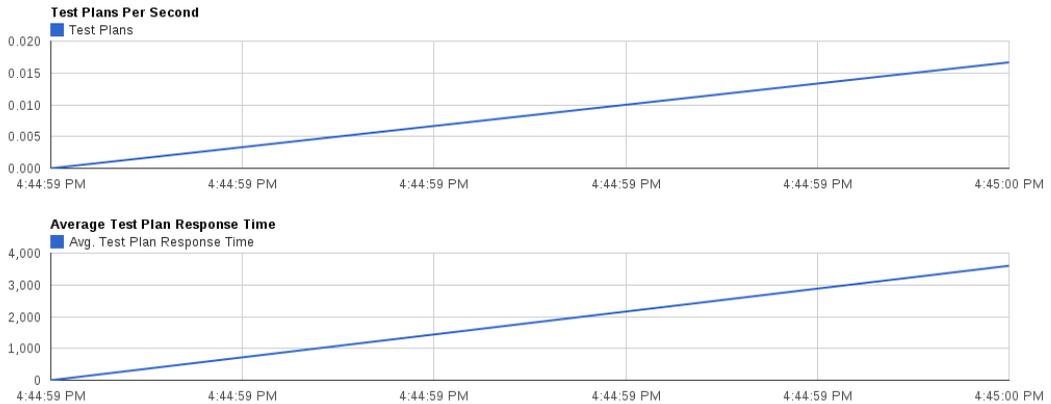
[Details](#)

Test Servers

Server	IP Address	Status	Size/Max Price	Test Plans Running	Test Plans Completed	Avg Test Plan Time	Download Size	Failed Pages
Server #00049134 i-67806bb3	54.227.85.166	Shutdown at 10/05/15 11:37am	m3.medium/\$0.0670	0	1	3600.808s	8.147MB	0

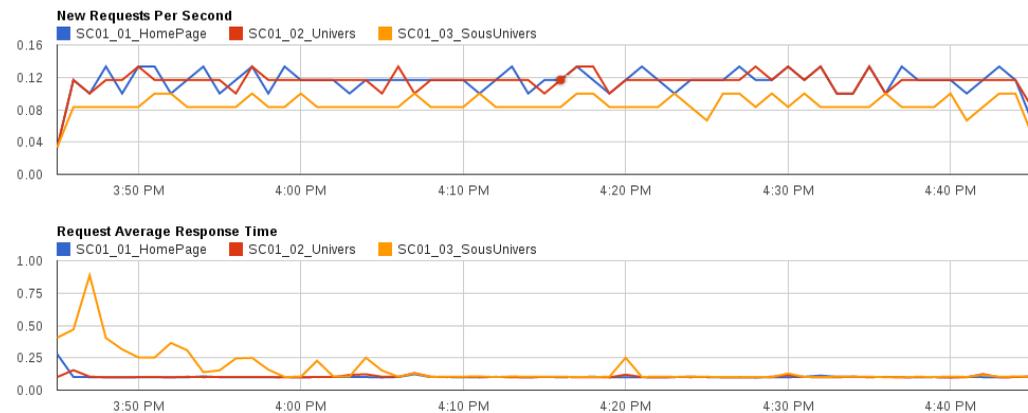
[Download as CSV](#)

Metrics



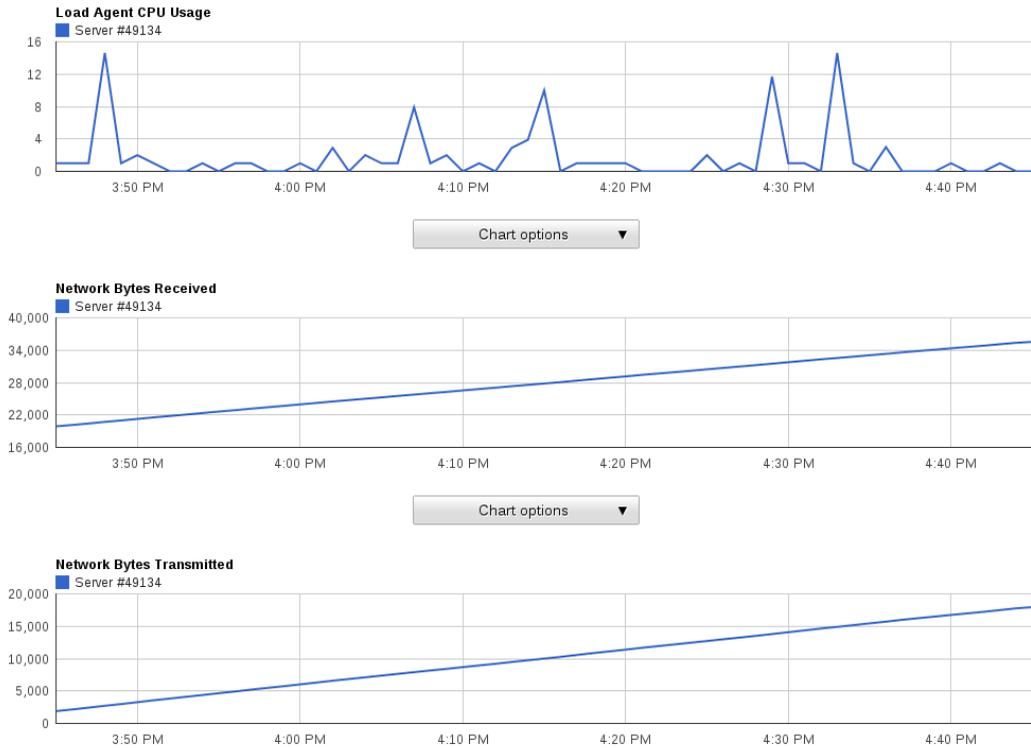
Redline 13 - Suivi de notre test

Les métriques de notre scénario sont disponibles.



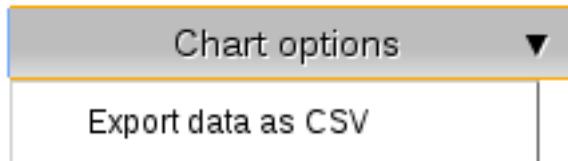
Redline 13 - Suivi de notre test : métriques de notre scénario

La supervision des injecteurs est disponible.



Redline 13 - Supervision des injecteurs

À la fin de notre test, il est possible pour chaque métrique de l'exporter au format CSV pour la travailler avec un autre outil.



Redline 13 - Export CSV



Beaucoup d'autres fonctionnalités sont disponibles, je vous laisse creuser le sujet.

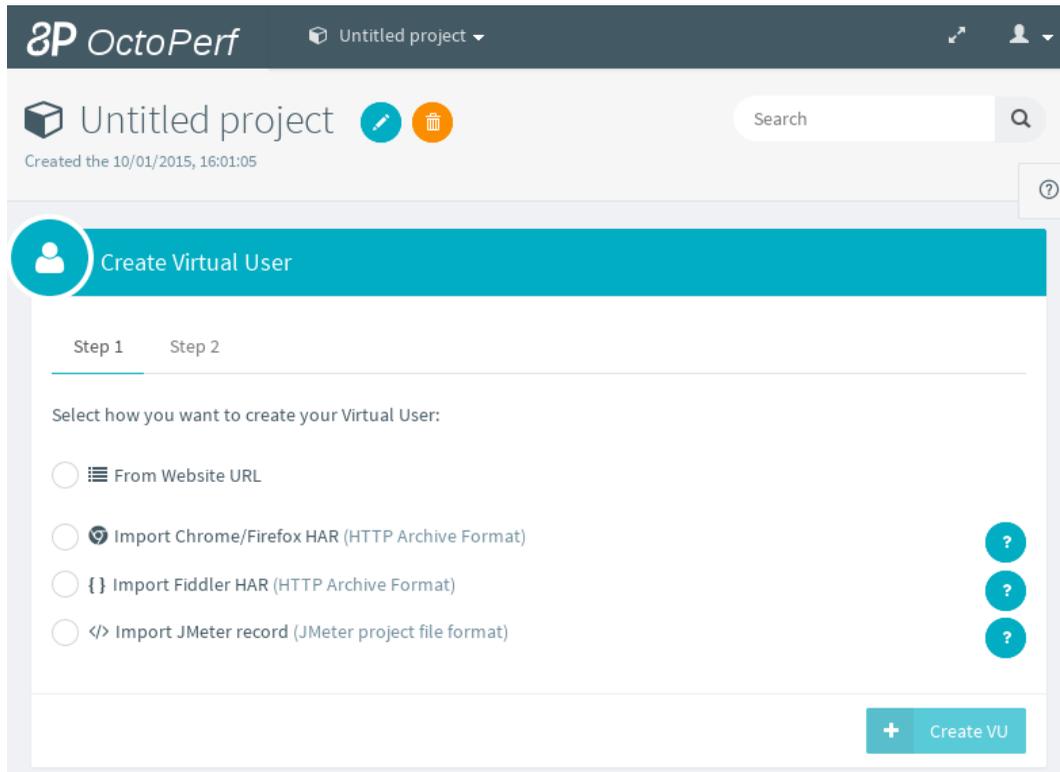
## OctoPerf (anciennement Jelly.IO)

Dernière solution qui a vu le jour, [OctoPerf<sup>17</sup>](http://octoperf.com/) permet d'exécuter un test de charge basé sur JMeter depuis le cloud.

OctoPerf dispose d'une interface très simple et sobre.

Plusieurs possibilités sont proposées :

- Tester une URL
- Utiliser un fichier HAR (HTTP Archive) comme script
- Importer un script JMeter

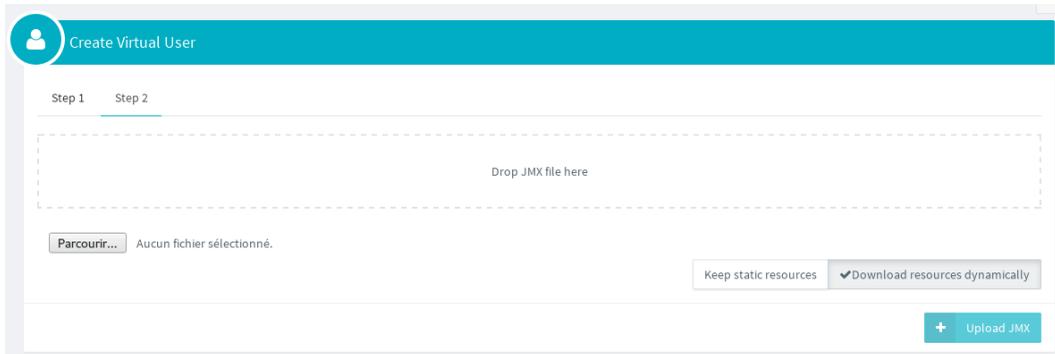


OctoPerf

---

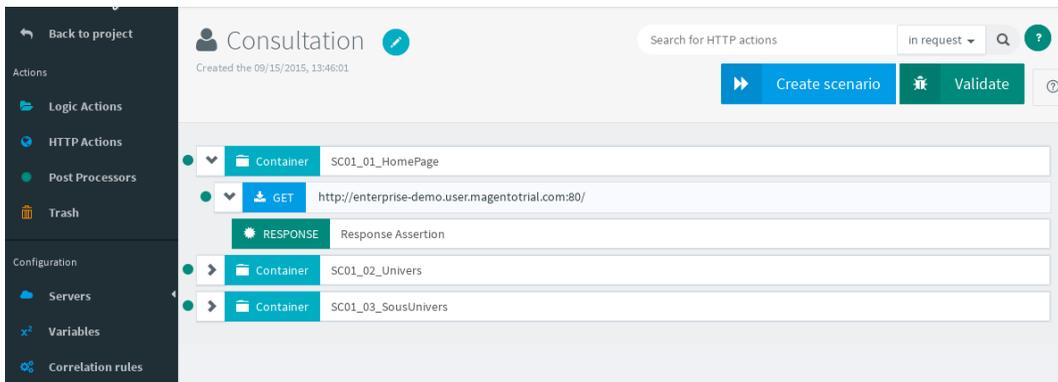
17. <http://octoperf.com/>

Nous choisissons d'importer notre script JMeter.



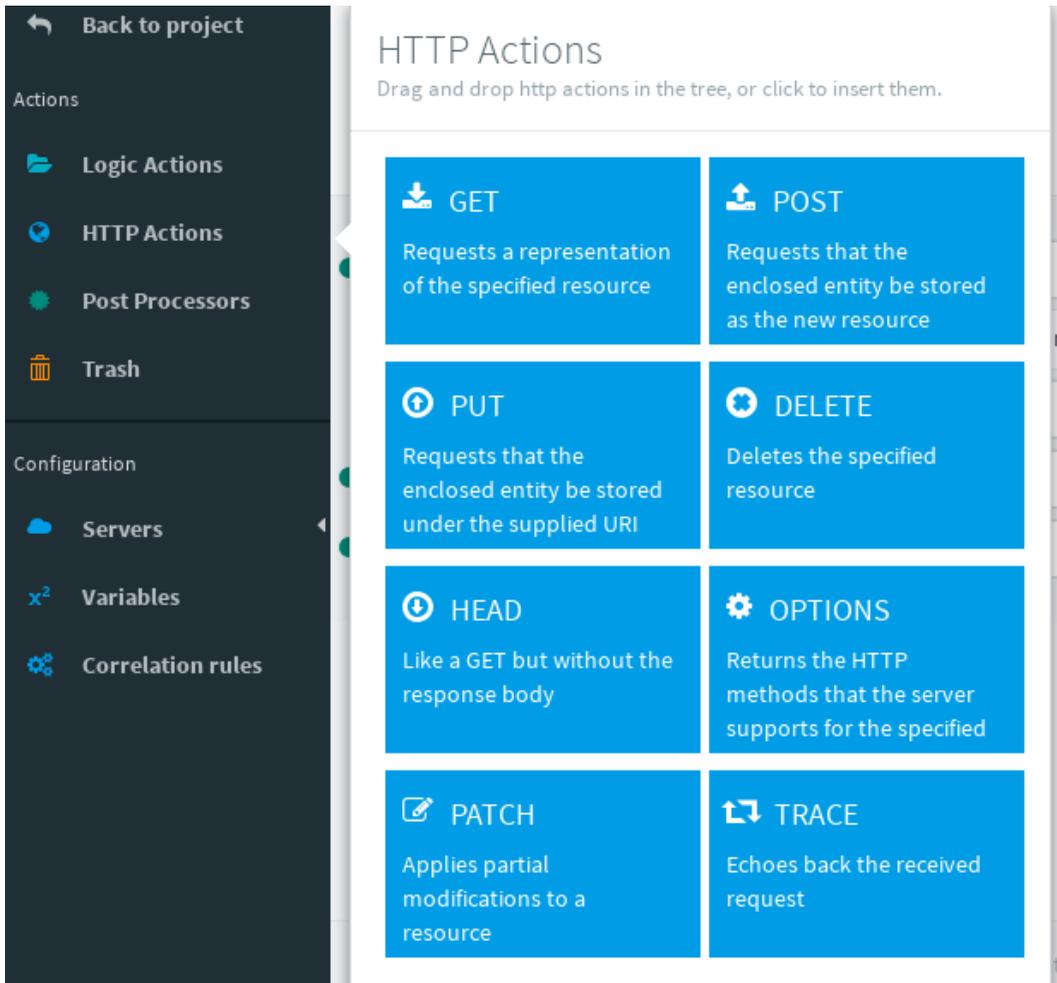
### OctoPerf - Import d'un script JMeter

Une fois l'importation réalisée, Jelly offre une représentation visuelle du script.

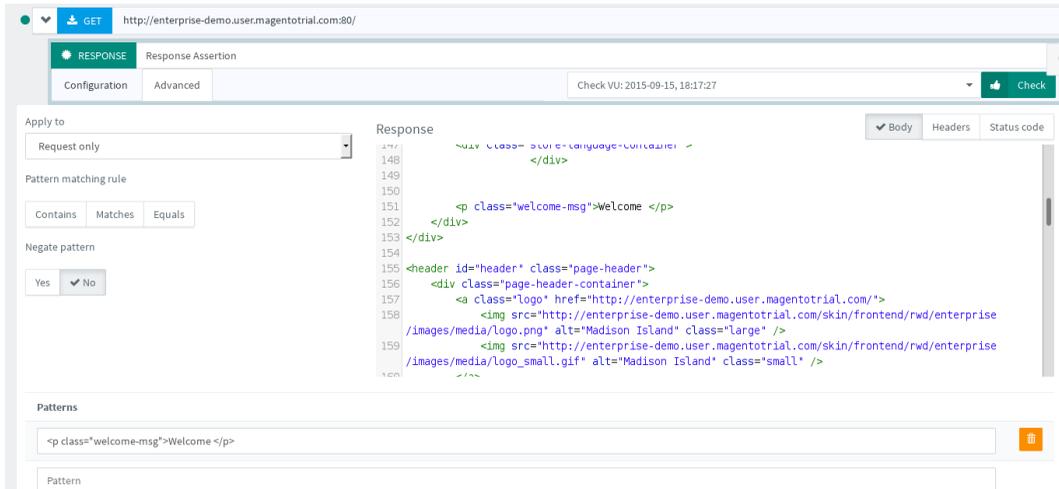


### OctoPerf - Notre script

Si nécessaire nous pouvons modifier notre script.

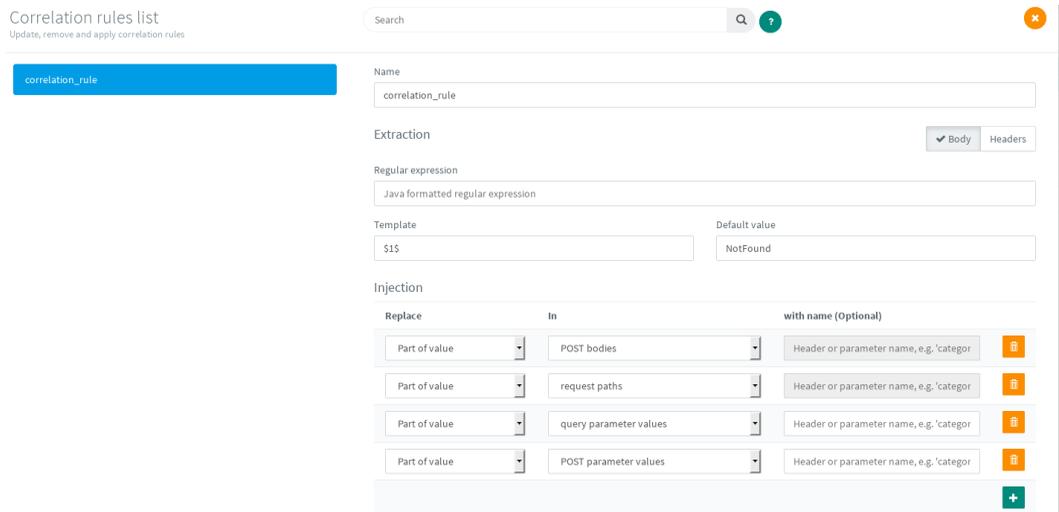


OctoPerf - Modification d'un script



### OctoPerf - Modification d'un script

Il est même possible d'utiliser la fonctionnalité *Règles de corrélation* qui est l'équivalent de l'auto-corrélation dans LoadRunner (HP) ou bien des paramètres de framework dans NeoLoad (Neotys).



### OctoPerf - Règles de corrélation

Définissons et importons notre jeu de données.

Variables list

Update, remove and add variables

ChoixDeLUnivers

Search

Name  
ChoixDeLUnivers

Description  
Enter description

Configuration Columns

File name  
uri.csv

Encoding  
The CSV file encoding, if not UTF-8

Delimiter  
,

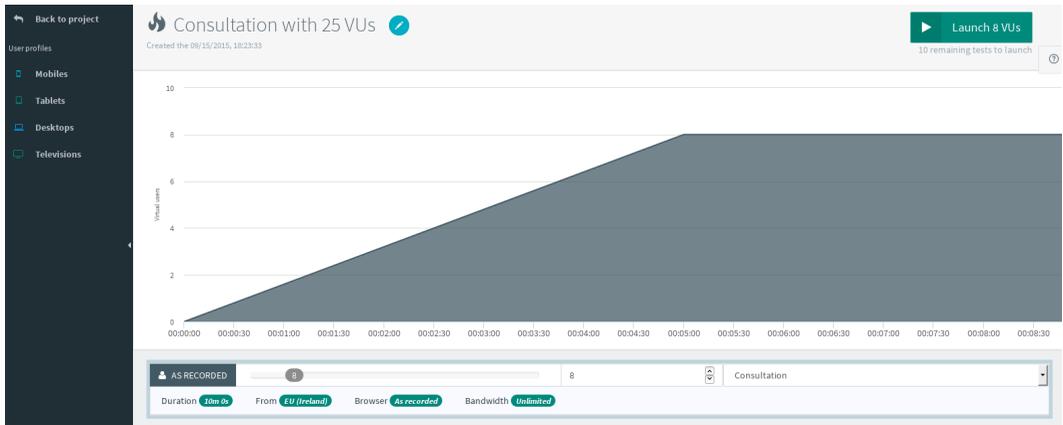
Allow quoted data?

Sharing mode *Whether the file is shared amongst all VUs or is private to each VU*  
Private  Shared

Recycle values on end of file?  Stop VU on end of file?

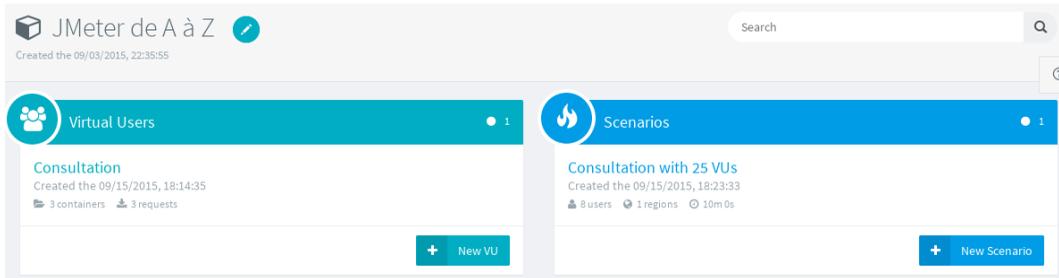
OctoPerf - Importation du jeu de données

Créons notre plan de tir.



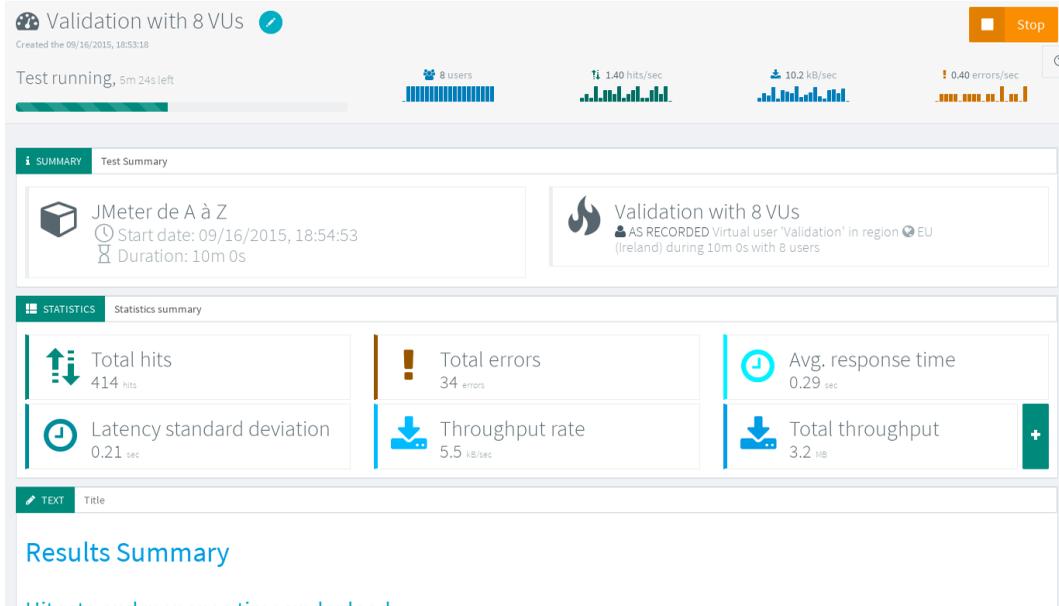
OctoPerf - Création de notre plan de test

Notre projet de test de charge est prêt.



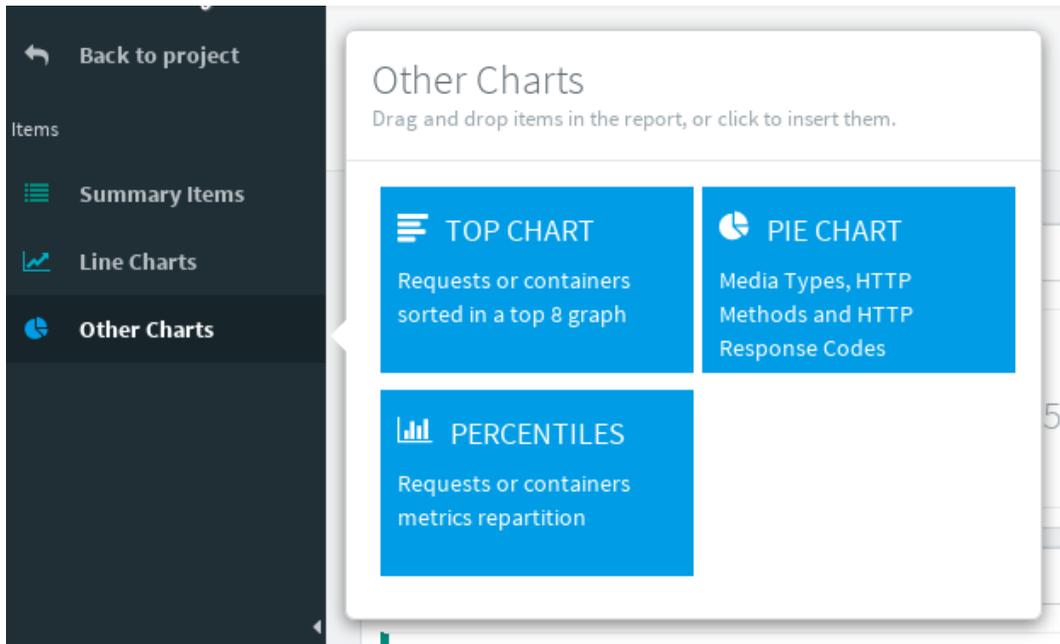
### OctoPerf - Projet de test de charge

Exécutons notre test pour le suivre en temps réel.



### OctoPerf - Suivi d'un test

De nombreux graphiques sont disponibles.



OctoPerf - Rapport et graphiques

Line Charts

Drag and drop items in the report, or click to insert them.

<b>HITS</b> Number of hits (requests) per second and userload over time	<b>THROUGHPUT</b> Bit rate in Bytes per second and userload over time
<b>ERRORS</b> Number of errors per second and userload over time	<b>ASSERTIONS</b> Total count of assertions in error and userload over time
<b>RESPONSE TIME</b> Average time between the request and the end of the response and userload	<b>LATENCY</b> Average time between the request and the first response byte (or Time To
<b>CONNECT TIME</b> Average time between the request and the server connection and userload	

### OctoPerf - Rapport et graphiques

Une fois notre test fini, un rapport au format PDF est disponible. Ce rapport peut être entièrement personnalisé avec les graphiques souhaités :

- statistiques avancées telles que les percentiles
- détail des réponses reçues du serveur en cas d'erreur
- etc.

## JMeter EC2

Une autre solution est d'utiliser le plugin gratuit dédié à AWS nommé [JMeter EC2](#)<sup>18</sup> qui permet simplement de :

- Démarrer les instances AWS
- Lancer le tir
- Récupérer et fusionner les résultats

## DevOps

Plus un problème de performance est détecté et corrigé tard, plus son coût a des chances d'être élevé. C'est pour cela qu'il est conseillé de faire des tests de charge le plus tôt possible en ayant quand même une application mature et un environnement représentatif en termes de données.

Un chapitre entier est consacré à l'intégration de JMeter dans le monde DevOps avec les plugins **Jenkins Performance Plugin** et **JMeter Maven Plugin**.

## Aide à la supervision et au diagnostic

Lors d'une campagne de tests de charge si la supervision n'est pas à la hauteur ou pire inexistante, la phase d'analyse risque d'être longue et/ou peu productive. Pour éviter ce souci, il existe de nombreux outils (APM, profiler, etc.).

Afin d'être le plus productif, deux critères de choix sont importants :

- être multi-technologie afin de couvrir des systèmes complexes ;
- s'intégrer facilement à JMeter afin de faire facilement le lien entre l'injection et son impact sur le système testé.

Il existe une solution répondant à ces deux critères.

---

18. <https://github.com/oliverlloyd/jmeter-ec2>

## Dynatrace APM

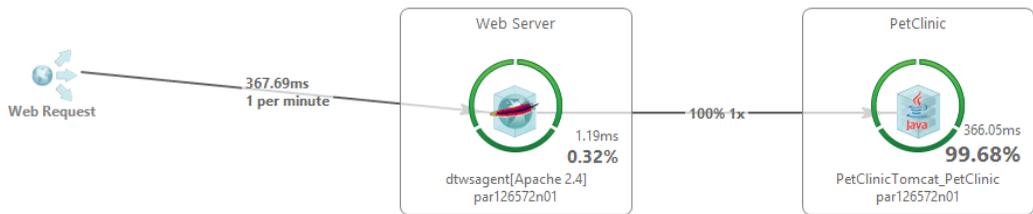
Cette solution est l'APM Dynatrace<sup>19</sup>.

Dynatrace permet de suivre la performance de bout en bout à l'aide de sa technologie PurePath. Cela va nous permettre d'aller jusqu'à la ligne de code problématique si un problème de performance est détecté.

Method	Argument	Exec Total [ms]	Exec (ms)	Breakdown	Class	API
Web request	/petclinic/owners/search	1848.96	0.58	io(100.0%)	Web server	Web server
Synchronous Invocation		-	-	-	-	-
Synchronous Path (Webserver Call)		-	-	-	-	-
Web request	http://localhost:8080/petclinic/owners/search	1848.39	3.66	io(100.0%)	Web server	Web server
Synchronous Invocation		-	-	-	-	-
Synchronous Path (partly asynchronous) (HTTP)		-	-	-	-	-
doFilter(ServletRequest request, ServletResponse response, FilterChain filter)	/petclinic/owners/search	1844.73	3.78	cpu(84.0%) io(96.0%)	OncePerRequestF...	Servlet
doGet(HttpServletRequest request, HttpServletResponse response)	/petclinic/owners/search	1840.95	0.00	cpu(64.0%) io(96.0%)	FrameworkServlet	Servlet
loadClass(String)		29.02	0.00	cpu(64.0%) io(96.0%)	WebappClassLoa...	ClassLoader...
defineClass(String, byte[], int, int, ProtectionDomain, String)		29.02	29.02	cpu(64.0%) io(96.0%)	ClassLoader	ClassLoader...
<init>(String)		29.02	29.02	cpu(64.0%) io(96.0%)	AbstractBindingR...	Servlet
service(HttpServletRequest, HttpServletResponse)		1782.90	0.00	cpu(64.0%) io(96.0%)	JspServlet	JSP
newInstance()		29.02	0.00	cpu(64.0%) io(96.0%)	DocumentBuilder...	XML Proce...
getResources(String)		29.02	0.00	cpu(64.0%) io(96.0%)	ClassLoader	Classloadi...
getEntry(long, byte[], boolean)		29.02	29.02	cpu(64.0%) io(96.0%)	ZipFile	Classloadi...
parse(InputSource)		29.02	0.00	cpu(64.0%) io(96.0%)	DocumentBuilder...	XML Proce...
endElement(QName, Augmentations)		29.02	29.02	cpu(64.0%) io(96.0%)	AbstractDOMParser	XML Proce...
getData()		29.02	0.00	cpu(64.0%) io(96.0%)	CharacterDataImpl	XML Proce...
synchronizeData()		29.02	29.02	cpu(64.0%) io(96.0%)	DeferredTextImpl	XML Proce...
loadClass(String)		29.02	0.00	cpu(64.0%) io(96.0%)	ClassLoader	Classloadi...
doPrivileged(PrivilegedExceptionAction, AccessControlContext)		29.02	29.02	cpu(64.0%) io(96.0%)	AccessController	Classloadi...
loadClass(String)		58.05	0.00	cpu(64.0%) io(96.0%)	ClassLoader	Classloadi...
doPrivileged(PrivilegedExceptionAction, AccessControlContext)		29.02	29.02	cpu(64.0%) io(96.0%)	AccessController	Classloadi...
fillInStackTrace(int)		29.02	29.02	cpu(64.0%) io(96.0%)	Throwable	Classloadi...
loadClass(String)		29.02	0.00	cpu(64.0%) io(96.0%)	ClassLoader	Classloadi...
defineClass(String, byte[], int, int, ProtectionDomain, String)		29.02	29.02	cpu(64.0%) io(96.0%)	ClassLoader	Classloadi...
getResourcePaths(String)		29.02	0.00	cpu(64.0%) io(96.0%)	ApplicationConte...	Servlet
canonicalize(String)		29.02	29.02	cpu(64.0%) io(96.0%)	WinNTFileSystem	Servlet
log(Level, String, String, String)		29.02	0.00	cpu(64.0%) io(96.0%)	Logger	Log
writeBytes(byte[], int, int, boolean)		29.02	29.02	cpu(64.0%) io(96.0%)	FileOutputStream	Log
getNextEntry(long, int)		58.05	58.05	cpu(64.0%) io(96.0%)	ZipFile	JSP

Dynatrace PurePath

Le tout en ayant la possibilité d'avoir une vision globale et visuelle.



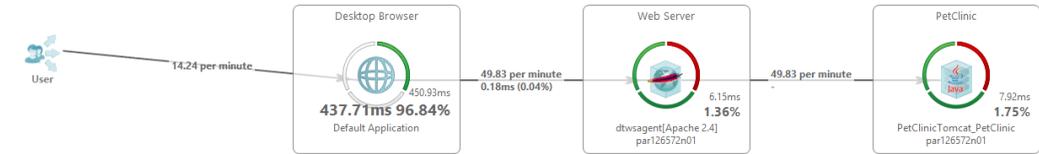
Dynatrace Transaction Flow

Reprenons nos deux critères de sélection.

Comme nous pouvons le voir sur la capture d'écran ci-dessous, plusieurs acteurs

19. <http://www.dynatrace.com>

entrent en jeu (Java, navigateur Web et serveur Apache httpd).



### Dynatrace Transaction Flox multi technologies

De nombreuses technologies sont gérées :

- .Net
- PHP
- C/C++
- Mobile
- zOS
- NodeJS
- etc.

Passons au deuxième point : l'intégration avec JMeter.

L'intégration entre les deux outils se fait par l'ajout d'entêtes HTTP spécifiques à Dynatrace dans les requêtes HTTP de JMeter.

HTTP Header Manager	
Name:	HTTP Header Manager
Comments:	
Headers Stored in the Header Manager	
Name:	Value
dynaTrace	PC=JMETER;NA=Clients_00_HomePage

### Dynatrace entête HTTP spécifique

Une fois cela réalisé, observons dans Dynatrace les PurePath désormais renommés comme les transactions de JMeter.

	Timer Name	Failed %	Count	Total Avg [...]	Total Sum ...
	Clients_00_HomePage	0 %	114	3.52	401.18
	Clients_01_Find_Owners_Page	0 %	109	3.50	381.88
	Clients_02_Owner_Information_Page	0 %	217	7.15	1551.13

### Dynatrace Tagged Web Requests

PurePath	Response Time [ms]	Breakdown	Size	Agent	Application	Start Time	Duration [ms]
Clients_01_Find_Owners_Page	78.00	cpu (85.0%) io	5	dtwsagent[Apache 2.4...	Default Applic...	2015-07-21 22:44:22.640	78.00

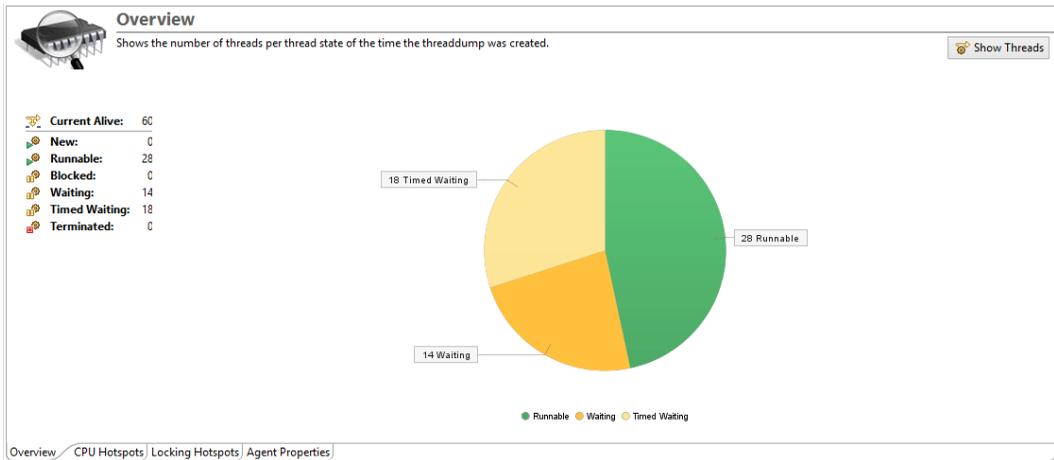
PurePath Tree (showing only relevant nodes)							
Method	Argument	Exec Total [ms]	Exec [ms]	Breakdown	Class	API	
Web request	/petclinic/owners...	78.00	0.44	io (100.0%)	Web server	Web serv	
Synchronous Invocation		-	-				
Synchronous Path (Webserver Call)		-	-				
Web request	http://localhost:8...	77.56	4.05	io (100.0%)	Web server	Web serv	
Synchronous Invocation		-	-				
Synchronous Path (partly asynchronous) (HTTP)		-	-				
doFilter(ServletRequest request, ServletResponse response, FilterCha	/petclinic/owners...	73.51	1.62	cpu (85.0%) io	OncePerRequestFi...	Servlet	
doGet(HttpServletRequest request, HttpServletResponse response	/petclinic/owners...	71.89	0.00	cpu (87.0%) io	FrameworkServlet	Servlet	
service(HttpServletRequest request, HttpServletResponse)		71.89	0.00	cpu (87.0%) io	JspServlet	JSP	
getDeclaredConstructors0(boolean)		47.58	15.89	cpu (80.9%) io	Class	JSP	
loadClass(String)		31.72	0.00	cpu (80.9%) io	JasperLoader	Classloac	
defineClass1(String, byte[], int, int, ProtectionDomain		31.72	15.86	cpu (80.9%) io	ClassLoader	Classloac	
defineClass1(String, byte[], int, int, ProtectionDom		15.86	15.86	cpu (80.0%) io	ClassLoader	Classloac	
service(ServletRequest, ServletResponse)		24.31	0.00	cpu (100.0%)	HttpServletRequest	Servlet	
jspService(HttpServletRequest request, HttpServletResponse	/petclinic/WEB-IN...	24.31	0.00	cpu (100.0%)	search_jsp	Servlet	
getClass()		8.10	0.00	cpu (100.0%)	TagHandlerPool	JSP	
loadClass(String)		8.10	0.00	cpu (100.0%)	WebappClassLoa...	Classloac	
defineClass1(String, byte[], int, int, ProtectionD		8.10	8.10	cpu (100.0%)	ClassLoader	Classloac	
loadClass(String)		8.10	0.00	cpu (100.0%)	JasperLoader	Classloac	
defineClass1(String, byte[], int, int, ProtectionDom		8.10	8.10	cpu (100.0%)	ClassLoader	Classloac	
toString()		8.10	8.10	cpu (100.0%)	StringBuilder	Servlet	

### Dynatrace PurePath

Dès lors, nous avons accès à toute la puissance de Dynatrace pour la supervision et le diagnostic.

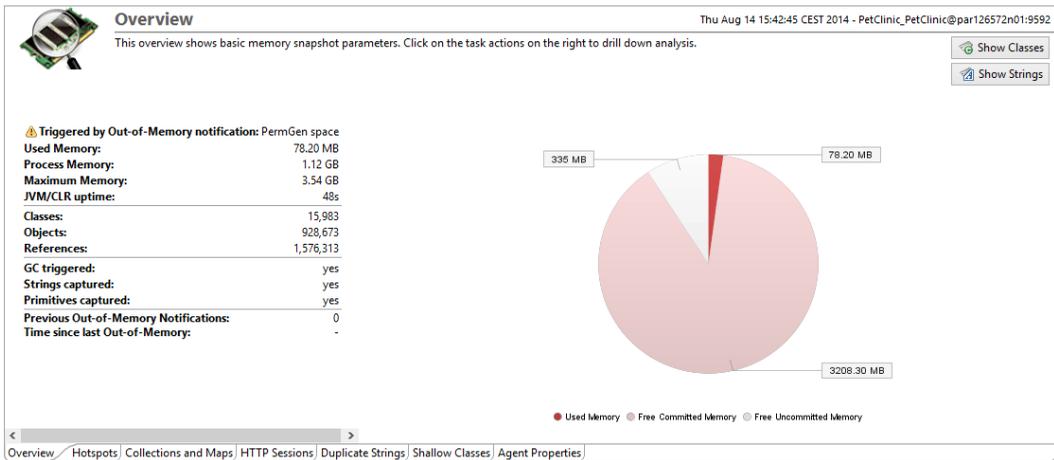
Ci-dessous quelques-unes de ses possibilités (non exhaustives) pour le diagnostic.

Réaliser des threads dump.



### Threads Dump

Réaliser des dumps mémoire.



### Memory Dump

Connaître les méthodes qui prennent le plus de temps et avoir leur arbre d'appel.

### Method Breakdown by Execution Time

Select a method to find out where it is called from

Method	Exec Sum	Breakdown	Class	APIs
Proxy Request()	1.24s	cpu (89.0%)	Web server	Web server
doGet(HttpServletRequest, HttpServletResponse)	1.09s	io (71.0%)	org.springframework.web.servlet.FrameworkServlet	Servlet
doFilter(ServletRequest, ServletResponse, FilterChain)	281ms	io (71.0%)	org.springframework.web.filter.OncePerRequestFilter	Servlet
park(boolean, long)	213ms	wait (81.0%)	sun.misc.Unsafe	Tangosol
_jspService(HttpServletRequest, HttpServletResponse)	167ms	io (84.0%)	org.apache.jsp.WEB_002dINF.jsp.owners.show_jsp	Servlet
_jspService(HttpServletRequest, HttpServletResponse)	87ms	io (98.0%)	org.apache.jsp.WEB_002dINF.jsp.welcome_jsp	Servlet
_jspService(HttpServletRequest, HttpServletResponse)	80ms	io (96.0%)	org.apache.jsp.WEB_002dINF.jsp.owners.search_jsp	Servlet
clone()	20ms	cpu (47.0%) io (53.0%)	java.lang.Object	JSP, Servlet
resolveHandlerArguments(Method, Object, ...)	17ms	io (79.0%)	org.springframework.web.bind.annotation.support.HandlerMethodInvoker	Servlet
getLastModifiedTime(File)	17ms	io (100.0%)	java.io.WinNTFileSystem	JSP, Servlet
isAssignableFrom(Class)	15ms	cpu (79.0%)	java.lang.Class	JSP, Servlet

### Caller Breakdown of 'doGet(HttpServletRequest, HttpServletResponse)'

Find out from what components the method is called and which call path has the biggest performance impact

Method	Contribution	APIs	Package
FrameworkServlet.doGet(HttpServletRequest, HttpServletResponse)		Servlet	org.springframework
OncePerRequestFilter.doFilter(ServletRequest, ServletResponse, FilterChain)	88.0%	Servlet	org.springframework
ApplicationFilterChain.internalDoFilter(ServletRequest, ServletResponse)		Servlet	org.apache.c
ApplicationFilterChain.doFilter(ServletRequest, ServletResponse)		Servlet	org.apache.c
StandardWrapperValve.invoke(Request, Response)		Servlet	org.apache.c
StandardContextValve.invoke(Request, Response)		Servlet	org.apache.c
AuthenticatorBase.invoke(Request, Response)		Servlet	org.apache.c
StandardHostValve.invoke(Request, Response)		Servlet	org.apache.c
ErrorReportValve.invoke(Request, Response)		Servlet	org.apache.c
AccessLogValve.invoke(Request, Response)		Servlet	org.apache.c
StandardEngineValve.invoke(Request, Response)		Servlet	org.apache.c
CoyoteAdapter.service(Request, Response)		Servlet	org.apache.c
AbstractHttp11Processor.process(SocketWrapper)		Servlet	org.apache.c
AbstractProtocol\$AbstractConnectionHandler.process(SocketWrapper, SocketStatus)		Servlet	org.apache.c
JioEndpoint\$SocketProcessor.run()		Servlet	org.apache.c
ThreadPoolExecutor.runWorker(ThreadPoolExecutor\$Worker)		Servlet	java.util.conc
ThreadPoolExecutor\$Worker.run()		Servlet	java.util.conc
Thread.run()	88.0%	Servlet	java.lang
HttpServlet.service(HttpServletRequest, HttpServletResponse)		Servlet	javax.servet

### Dynatrace Method Hotspots

Avoir la répartition des temps de réponse.

### PetClinic

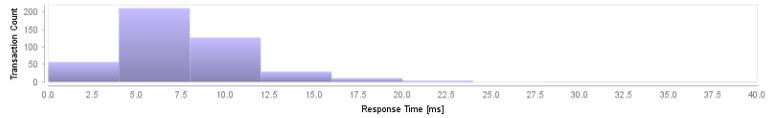
Percentile Filter

Analyze the transaction response time and its distribution across tiers and APIs to isolate performance problems

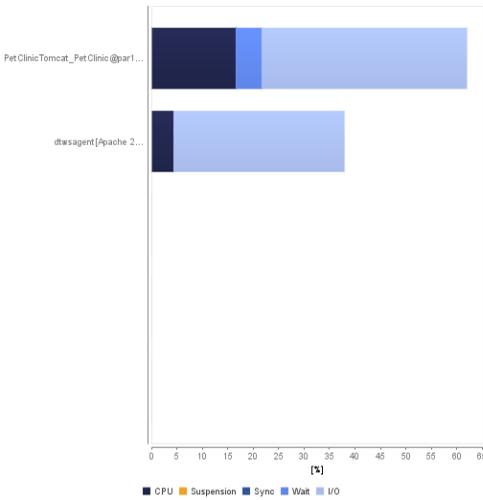
#### Server Transaction Statistics

[Show Method Hotspots](#)

Transaction Count: **443**  
Average Execution Time: **7.89 ms**  
Failed Transactions: **0 (0%)**

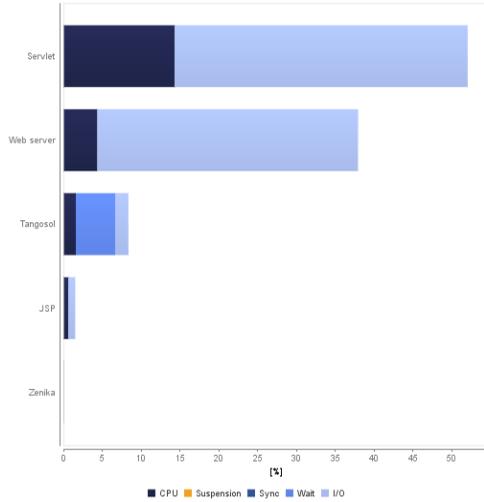


#### Hotspots by Tier



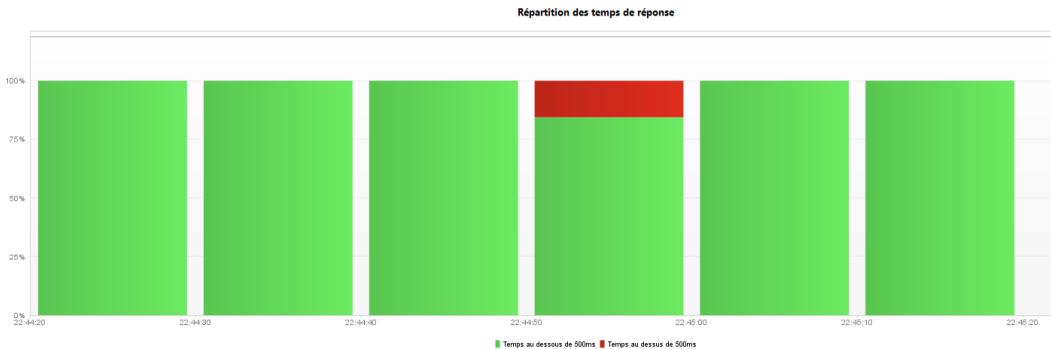
#### Hotspots by API

[Show API Breakdown](#)



### Dynatrace Response Time Hotspots

Ou encore la possibilité de créer ses propres tableaux de bord.



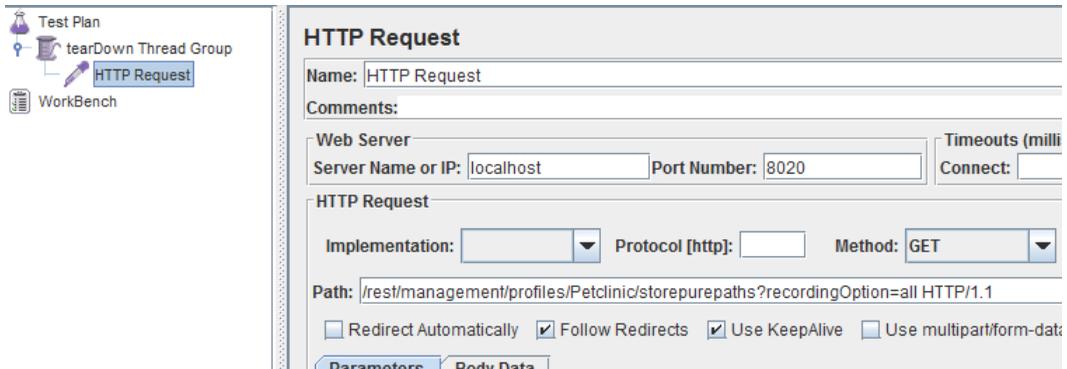
### Dynatrace Custom Dashboard

Une fois notre test de charge réalisé, il est possible de sauvegarder une session

Dynatrace contenant toutes les informations (les erreurs, les transactions, les temps de réponse, les PurePath, etc.) afin de nous donner la possibilité :

- de comparer plusieurs tirs
- d'analyser les résultats plus tard
- d'envoyer les informations à une autre équipe (par exemple les développeurs pour qu'ils puissent approfondir l'analyse).

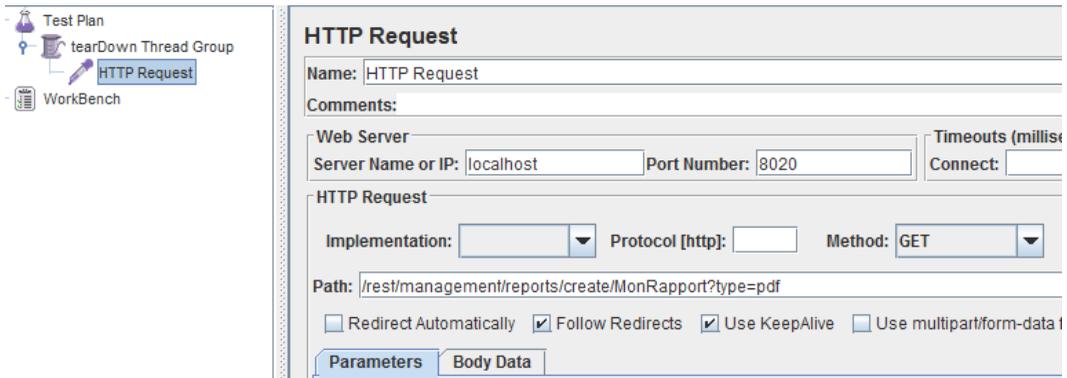
Il est même possible de réaliser cette action automatiquement à l'aide de l'API REST de Dynatrace.



The screenshot shows the JMeter configuration for an HTTP Request. The 'Name' field is 'HTTP Request'. The 'Web Server' section has 'Server Name or IP' set to 'localhost' and 'Port Number' set to '8020'. The 'HTTP Request' section has 'Implementation' set to 'Default', 'Protocol [http:]' set to 'http', and 'Method' set to 'GET'. The 'Path' field contains the URL: '/rest/management/profiles/Petclinic/storepurepaths?recordingOption=all HTTP/1.1'. The 'Follow Redirects' and 'Use KeepAlive' checkboxes are checked. The 'Parameters' and 'Body Data' tabs are visible at the bottom.

Dynatrace REST API

Et toujours avec l'API REST, il est possible de générer un rapport Dynatrace à la fin du test.

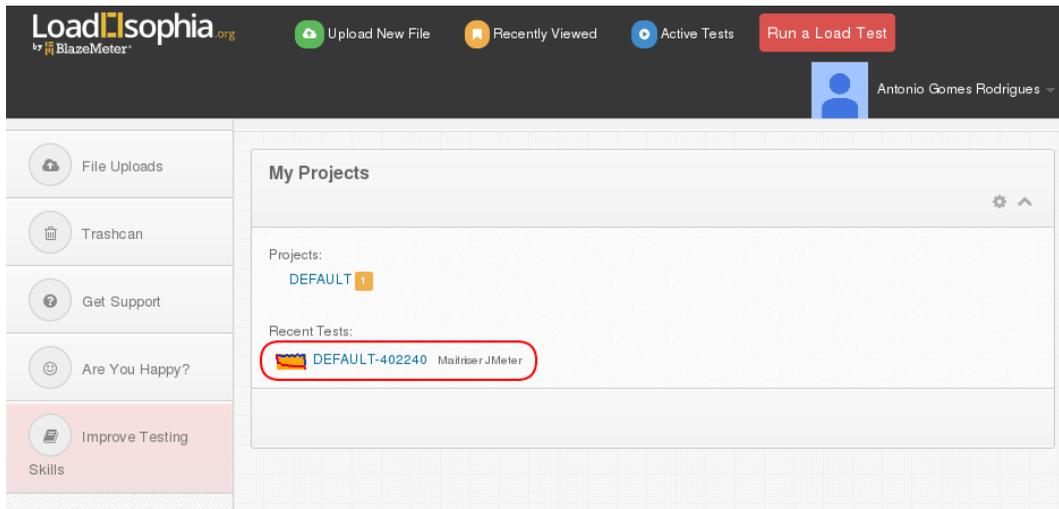


The screenshot shows the JMeter configuration for an HTTP Request. The 'Name' field is 'HTTP Request'. The 'Web Server' section has 'Server Name or IP' set to 'localhost' and 'Port Number' set to '8020'. The 'HTTP Request' section has 'Implementation' set to 'Default', 'Protocol [http:]' set to 'http', and 'Method' set to 'GET'. The 'Path' field contains the URL: '/rest/management/reports/create/MonRapport?type=pdf'. The 'Follow Redirects' and 'Use KeepAlive' checkboxes are checked. The 'Parameters' and 'Body Data' tabs are visible at the bottom.

Dynatrace REST API







Nos résultats uploadés dans Loadosophia.org

Regardons d'un peu plus près nos résultats. Un résumé du test apparaît.

The screenshot shows the Maitriser JMeter interface for a test named 'DEFAULT-402240'. The interface includes a navigation bar with tabs for Summary, Distributions, Timelines, Pivots, Target Monitoring, Composite Timeline Analysis, and Insights. The 'Summary' tab is active, displaying 'Test Info' and 'Comments' sections.

Test Info	
Uploaded File:	Loadosophia_2668258756130689569.jtl.gz
Started at:	22/10/2015, 10:15:26
Test Duration:	00:07:29
Transactions Count:	1264
HTTP Codes Presence:	2xx
Minimum Response Time, ms:	1
Average Response Time, ms:	7
Maximum Response Time, ms:	305
Average Throughput (TPS):	2.81514
Average Virtual Users:	4.97897
Max Virtual Users:	5
Classification:	VU-Driven Load-Test (more on Insights tab)
Storage Status:	primary storage

Comments: No comments yet. Add Comment

Download PDF Public Access: OFF

### Résumé de notre test

Comme nous le voyons, de nombreux graphiques sont générés automatiquement.

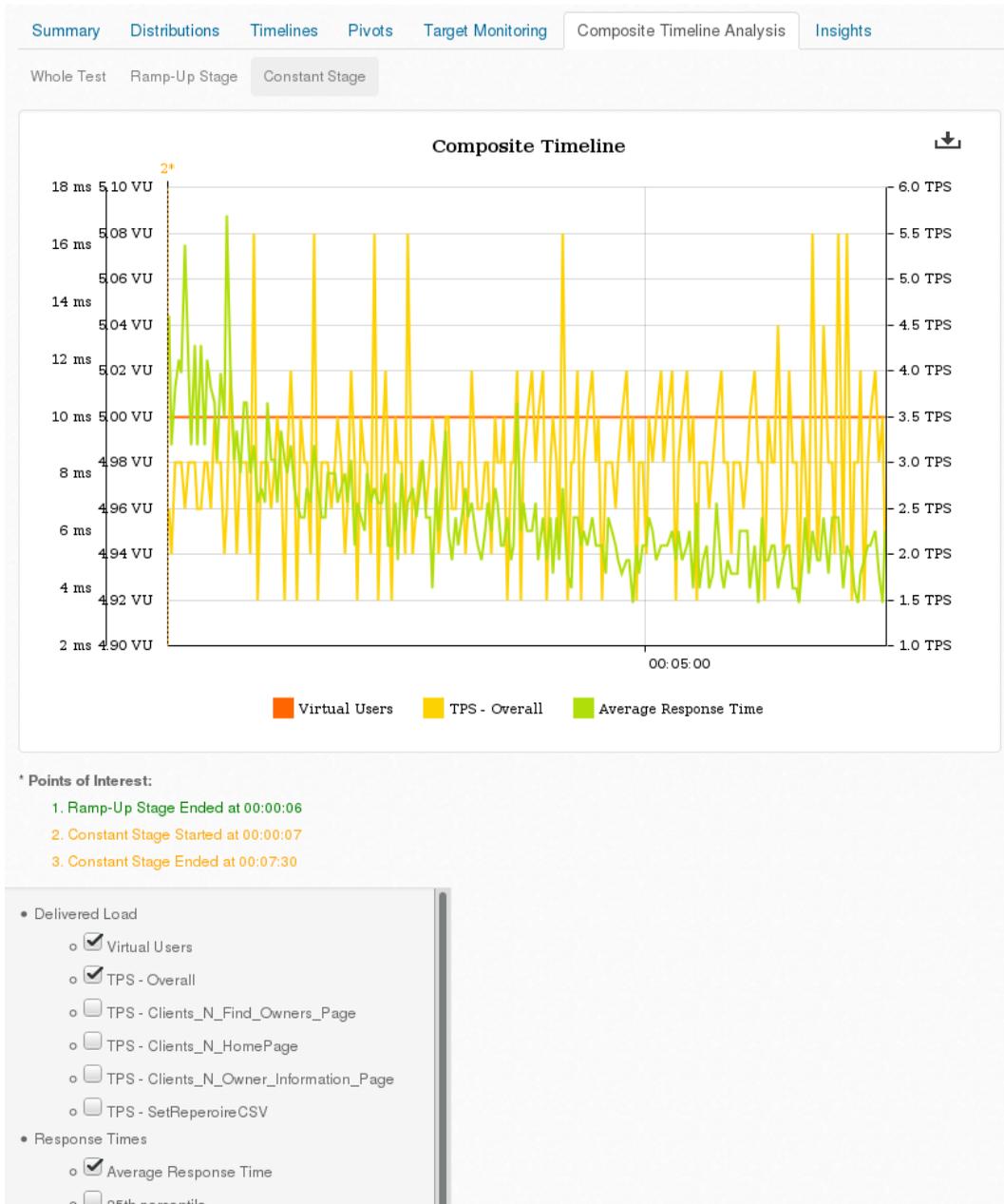
The screenshot shows the navigation bar of the Maitriser JMeter interface, with tabs for Summary, Distributions, Timelines, Pivots, Target Monitoring, Composite Timeline Analysis, and Insights.

### Type de graphiques possibles



Graphiques de type Timelines

Nous avons la possibilité de créer un graphique personnalisé.



Graphique personnalisé

Le tout peut être téléchargé au format PDF.

Loadosophia.org permet de gagner du temps en générant automatiquement les principaux graphiques nécessaires à notre analyse. De plus, il permet également de stocker les résultats pour comparaison ultérieure.

## D'autres protocoles

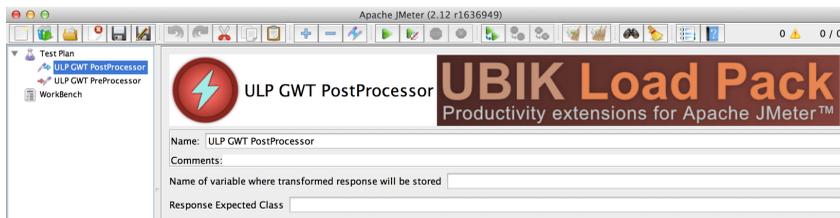
Malgré le nombre élevé de protocoles supportés par Apache JMeter, il reste tout de même plusieurs protocoles non supportés nativement.

Heureusement, de nombreux plugins (en plus des ceux de JMeter plugins vus précédemment) existent.

## UbikLoadPack

UbikLoadPack<sup>21</sup> est une solution d' Ubik-Ingénierie<sup>22</sup> offrant des plugins pour les protocoles suivants :

- GWT-RPC du Framework [Google Web Toolkit] (<http://www.gwtproject.org>) versions 1.5 à 2.8 (au moment de l'édition de ce livre)



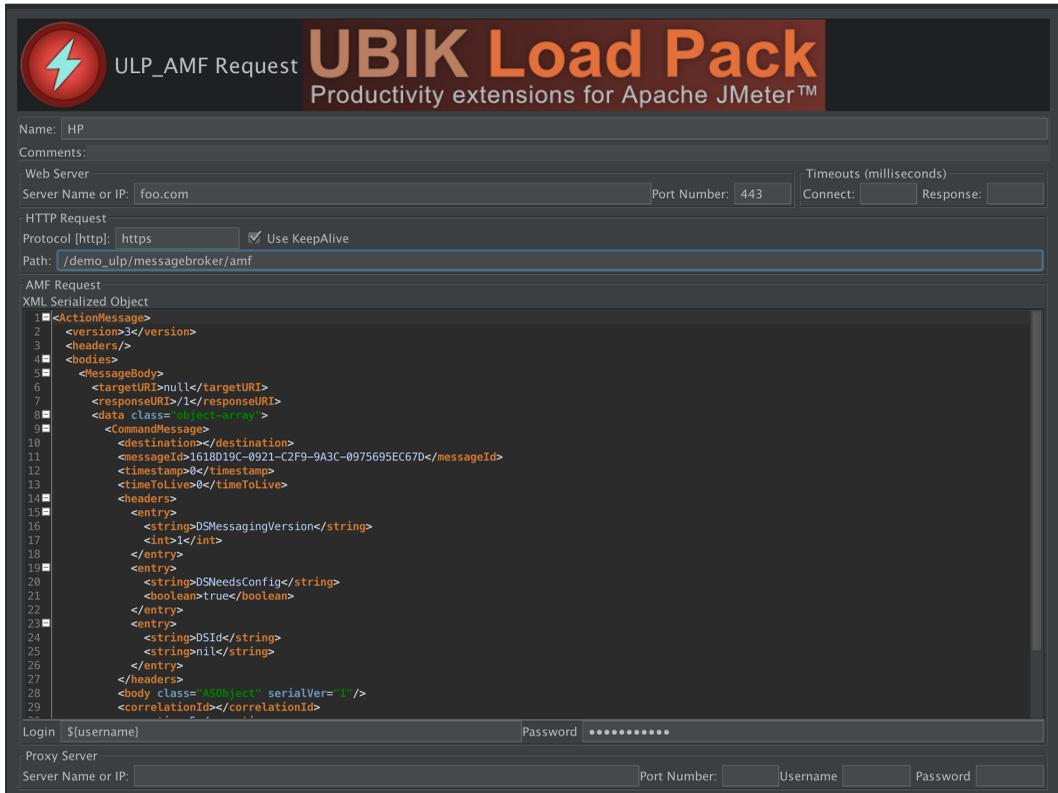
GWT-RPC

- FLEX/AMF gérant Adobe Flex et Apache Flex

---

21. <https://ubikloadpack.com/>

22. <https://www.ubik-ingenierie.com>



### FLEX/AMF

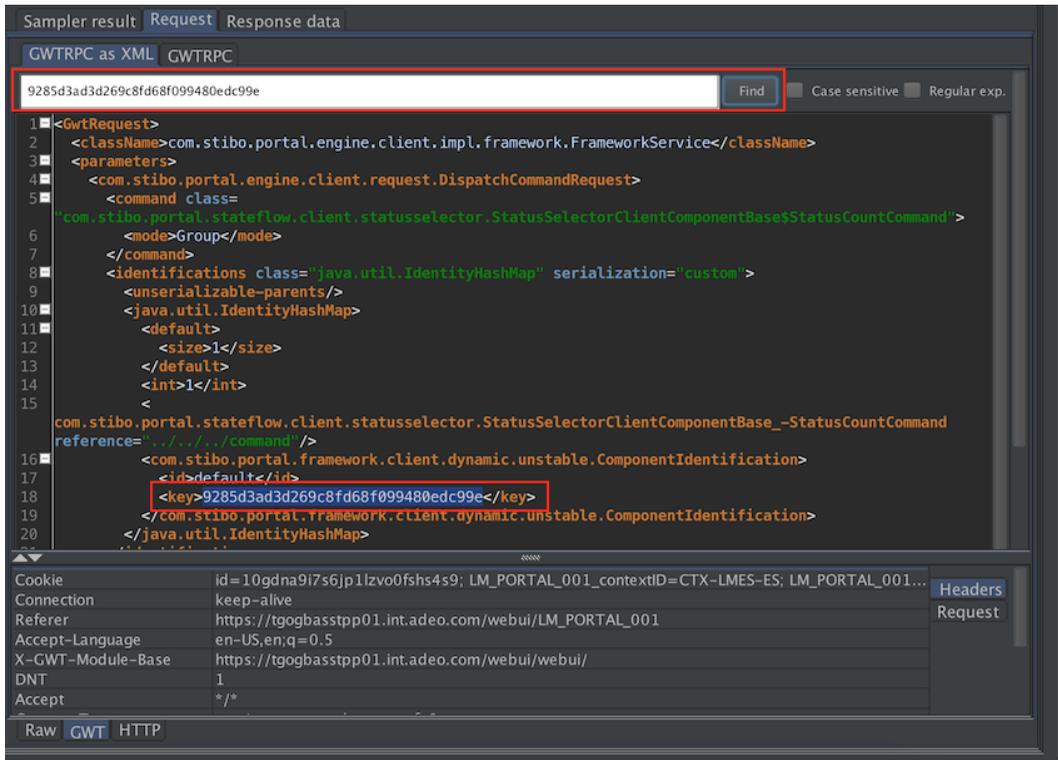
- Java Serialization permettant de simuler des applets ou des applications utilisant Spring Remoting

La solution permet de tester de façon réaliste des applications basées sur ces protocoles, c'est à dire en permettant la corrélation par la mise à disposition d'extracteurs et de transformateurs.

Le fonctionnement global des plugins est le suivant :

- Avec **Enregistreur script de test HTTP(S)**, vous enregistrez votre navigation sur l'application et créez en quelques minutes le test JMeter.
- À l'aide de samplers dédiés ou de préprocesseurs, le plugin transforme les requêtes « illisibles » du protocole en XML que vous pouvez ainsi facilement

variabiliser puisqu'il est possible d'injecter des variables JMeter par la syntaxe `${variable}`



```
Sampler result | Request | Response data
-----|-----|-----
GWTRPC as XML | GWTRPC
9285d3ad3d269c8fd68f099480edc99e | Find | Case sensitive | Regular exp.
1 <GwtRequest>
2 <className>com.stibo.portal.engine.client.impl.framework.FrameworkService</className>
3 <parameters>
4 <com.stibo.portal.engine.client.request.DispatchCommandRequest>
5 <command class=
6 'com.stibo.portal.stateflow.client.statusselector.StatusSelectorClientComponentBase$StatusCountCommand >
7 <mode>Group</mode>
8 </command>
9 <identifications class="java.util.IdentityHashMap" serialization="custom" >
10 <unserializable-parents/>
11 <java.util.IdentityHashMap>
12 <default>
13 <size>1</size>
14 </default>
15 <int>1</int>
16 <com.stibo.portal.stateflow.client.statusselector.StatusSelectorClientComponentBase_-StatusCountCommand
reference=.../>
17 <com.stibo.portal.framework.client.dynamic.unstable.ComponentIdentification>
18 <id>default</id>
19 <key>9285d3ad3d269c8fd68f099480edc99e</key>
20 </com.stibo.portal.framework.client.dynamic.unstable.ComponentIdentification>
21 </java.util.IdentityHashMap>
```

Cookie	id=10gdna9I7s6jp1lzvo0fshs4s9; LM_PORTAL_001_contextID=CTX-LMES-ES; LM_PORTAL_001...	Headers
Connection	keep-alive	Request
Referer	https://tgogbasstpp01.int.adeo.com/webui/LM_PORTAL_001	
Accept-Language	en-US,en;q=0.5	
X-GWT-Module-Base	https://tgogbasstpp01.int.adeo.com/webui/webui/	
DNT	1	
Accept	/*/*	

Raw | GWT | HTTP

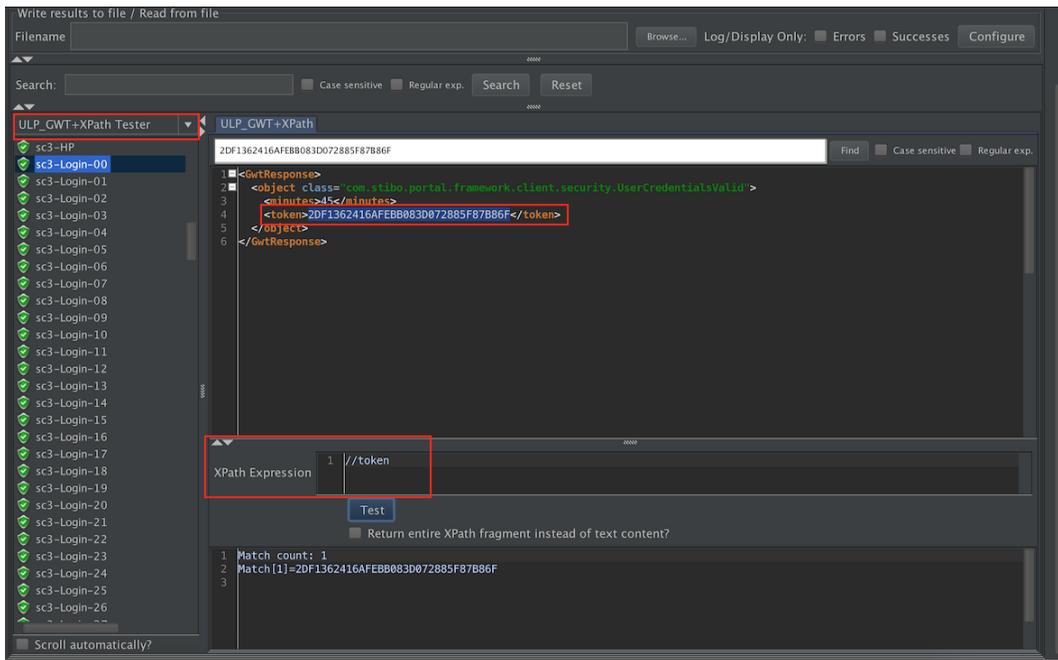
Requête GWT représentée en XML

- Vous pouvez extraire des réponses « illisibles » (transformées en XML par les Post-Processeurs du plugin) n'importe quelle donnée que vous souhaitez vérifier ou injecter dans la requête suivante



Transformer une réponse en XML et la stocker dans la variable « result »

La solution offre également des Renderers/Visualiseurs spéciaux intégrés à **Arbre de résultats** qui vous permettent de déboguer vos scripts en transformant à la volée le format du protocole en XML et de tester vos expressions d'extraction XPath.



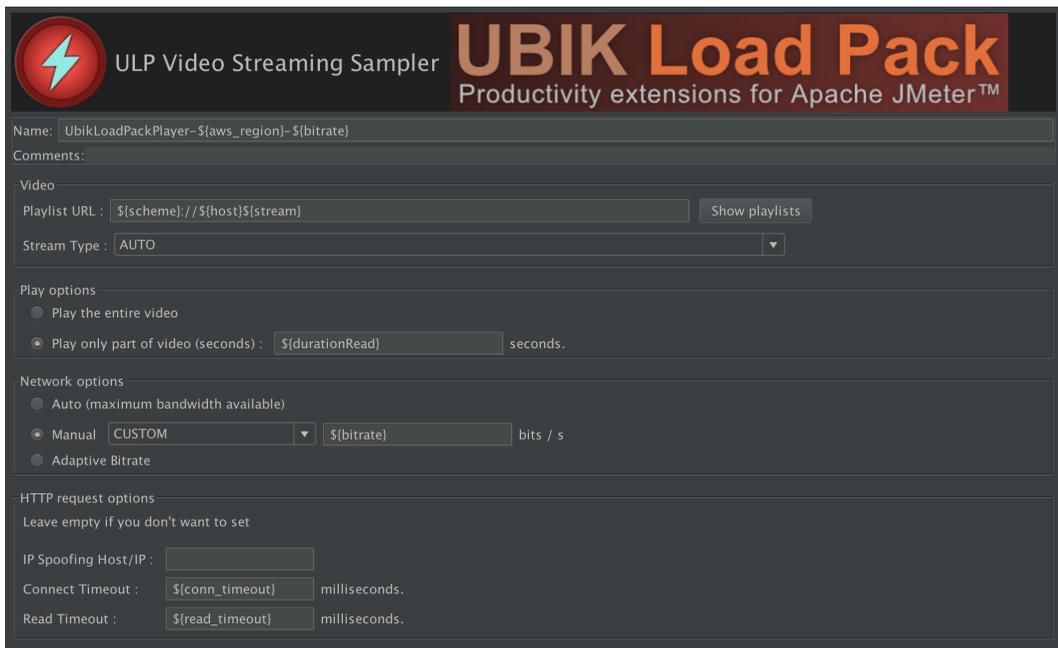
### GWT XPATH TESTER

Une fois votre script prêt, vous utiliserez alors la démarche du chapitre 3 pour variabiliser votre script, vérifier les réponses et exécuter votre tir comme expliqué dans le chapitre 8.

- Auto-correlateur permettant d'enregistrer puis rejouer, le plugin prenant en

charge la corrélation des IDs techniques de la technologie. Ce plugin gère les applications Entreprise avec support de :

- Oracle JD Edwards
- Oracle Hyperion Financial Management
- Oracle PeopleSoft
- Vaadin
- Streaming Video avec support des formats :
  - Format [HTTP Live Streaming] ([https://en.wikipedia.org/wiki/HTTP\\_Live\\_Streaming](https://en.wikipedia.org/wiki/HTTP_Live_Streaming))
  - Format [Mpeg-Dash] ([https://en.wikipedia.org/wiki/Dynamic\\_Adaptive\\_Streaming\\_over\\_HTTP](https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP))
  - Format Microsoft Smooth Streaming
  - Format Adobe HDS



The screenshot shows the configuration window for the 'ULP Video Streaming Sampler' plugin, part of the 'UBIK Load Pack' (Productivity extensions for Apache JMeter™). The interface is dark-themed and contains the following sections:

- Name:** UbiLoadPackPlayer-`#{aws_region}`-`#{bitrate}`
- Comments:** (empty)
- Video:**
  - Playlist URL:** `#{scheme}://#{host}#{stream}` (with a 'Show playlists' button)
  - Stream Type:** AUTO (dropdown menu)
- Play options:**
  - Play the entire video
  - Play only part of video (seconds): `#{durationRead}` seconds.
- Network options:**
  - Auto (maximum bandwidth available)
  - Manual: CUSTOM (dropdown) `#{bitrate}` bits / s
  - Adaptive Bitrate
- HTTP request options:**
  - Leave empty if you don't want to set
  - IP Spoofing Host/IP:** (empty text field)
  - Connect Timeout:** `#{conn_timeout}` milliseconds.
  - Read Timeout:** `#{read_timeout}` milliseconds.

### Video Streaming

Pour le streaming video, la solution permet de simuler le comportement d'un player sans l'impact en performance des players, nous pourrons ainsi facilement tester des

milliers ou centaines de milliers d'utilisateurs avec une infrastructure raisonnable. La solution prend en charge le VOD et le Live sans aucun scripting ce qui permet de créer en quelques minutes des tests réalistes, il suffit de passer l'URL des manifests à tester, la durée de lecture et c'est terminé.

Elle gère l'extraction des « chunks » des vidéos, simule la lecture par un « player », et peut même simuler une bande passante.

Enfin, elle ajoute des métriques spécifiques utiles à l'analyse de l'expérience de lecture vidéo aux résultats JMeter qui vous permettent de connaître :

- Le temps d'attente de l'utilisateur avant que sa vidéo ne commence (Buffer Fill Time)
- Le lag, c'est à dire les pauses pendant la lecture dues aux ralentissements réseau ou du serveur HLS (Lag Time)
- Le temps de lecture (Play Time)
- Le temps de téléchargement (Download Time)
- Le lag ratio, c'est-à-dire le temps de lag divisé par la durée totale de la vidéo (Lag ratio)
- Le nombre total de requêtes par vidéo
- Le temps moyen de téléchargement des manifests de la vidéo
- Le temps moyen de téléchargement des chunks

En synthèse, si vous connaissez JMeter, utiliser UbikLoadPack pour ces protocoles particuliers est très intuitif et ne nécessite pas de formation particulière.

À noter qu'UbikLoadPack/Ubik-Ingénierie est un contributeur très actif des projets Apache JMeter et JMeter-Plugins avec par exemple la donation en octobre 2015 du plugin JSON qui a été intégré à la version 3.0 de JMeter. À noter également la donation d'un générateur de rapports HTML affichant diverses métriques et graphes JavaScript dynamiques intégré à la version 3.0.

## DSL (Domain specific language)

Pour ceux qui n'aiment pas l'interface graphique de JMeter et préfèrent coder directement leurs scripts avec un langage de programmation dans leur IDE (ou pour les habitués de LoadRunner), il existe une solution.

## Ruby based DSL for JMeter

Les créateurs du service Tricentis Flood ont pensé à ces personnes en créant un DSL nommé [Ruby based DSL for JMeter](#)<sup>23</sup>. Comme son nom l'indique, il est basé sur du Ruby et nous permet d'écrire des scripts JMeter dans notre IDE préféré.

Installer le plugin est aussi simple que :

```
gem install ruby-jmeter
```

Prenons cet exemple (contenu du fichier "montest.rb") :

```
require 'rubygems'
require 'ruby-jmeter'
  test do
    cookies
    cache clear_each_iteration : true
      threads count : 5, duration : 60, continue_forever : true do
        random_timer 3000
        visit name : 'Accueil', url : 'http://jmeter.apache.org' do
          assert substring : "Further Information About JMeter", s\
cope : 'main'
        end
      end
    end.jmx(file : "./montest.jmx")
```

Ici, nous positionnons :

1. Le **Gestionnaire de cookies HTTP**
2. Le **Gestionnaire de cache HTTP**, le cache est nettoyé à chaque itération
3. Un **Groupe d'unités** composé de 5 utilisateurs, sa durée est de 60 secondes, et le nombre d'itérations infini
4. Un **Compteur de temps aléatoire gaussien** est positionné, son *Délai de décalage basé gaussien* est positionné à 3 secondes, nous allons donc faire une pause variable de 3s avant chaque appel d'Accueil (point suivant)

---

23. <https://github.com/flood-io/ruby-jmeter>

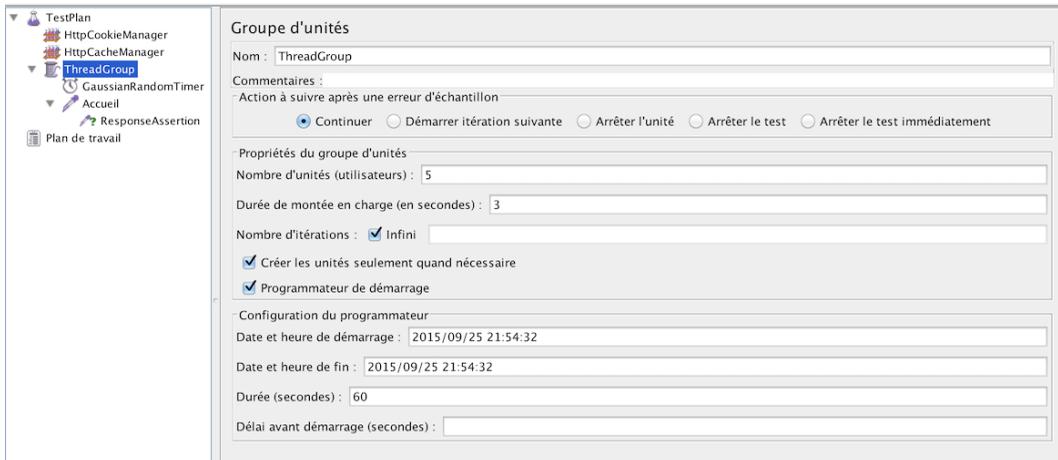
5. Chaque unité navigue sur la page d'accueil de JMeter (<http://jmeter.apache.org>), la requête est nommée Accueil
6. Nous vérifions la présence du texte "Further Information About JMeter" (en bas de la page si possible) dans la réponse principale de l'échantillon, l'option utilisée est "Contient (texte brut)"
7. Enfin, le fichier jmx qui sera généré par ruby s'appellera "montest.jmx"

Lançons :

```
ruby montest.rb
```

```
I, [2015-09-25T21:54:32.437507 #11545] INFO -- : Test plan saved t\  
o : ./montest.jmx
```

Le plan suivant est généré :



Plan de test généré par Ruby

Ce plugin a plusieurs avantages :

- Il offre une vue synthétique du test JMeter
- Il permet de versionner plus facilement le plan de test dans un gestionnaire de source. En effet, comparer le DSL est plus facile que de le faire avec le format XML de JMeter.

Il sera donc assez adapté à des tests de type Webservice où l'enregistrement a peu d'intérêt.

À noter que le projet github est assez actif avec 14 contributeurs (Pull Requests), 54 releases (octobre 2015) et le parrainage de Tricentis Flood. La gestion de versions du projet suit celle de JMeter en ajoutant un numéro de patch, ainsi pour JMeter 2.13, la version de ruby-jmeter correspondante est la 2.13.X. Le DSL gère un sous-ensemble des éléments de JMeter liés aux tests de type Web ainsi qu'une partie des éléments de JMeter-Plugins.



De nombreux exemples sont disponibles sur le site à l'URL <https://github.com/flood-io/ruby-jmeter/tree/master/examples>

## Conclusion

Cette liste déjà très riche de plugins n'est qu'un échantillon, elle nous montre déjà toutes les possibilités de JMeter et de son écosystème. Cette richesse le rend ainsi comparable à n'importe quel outil commercial, et dans certains domaines sa flexibilité et son extensibilité le rendent supérieur.