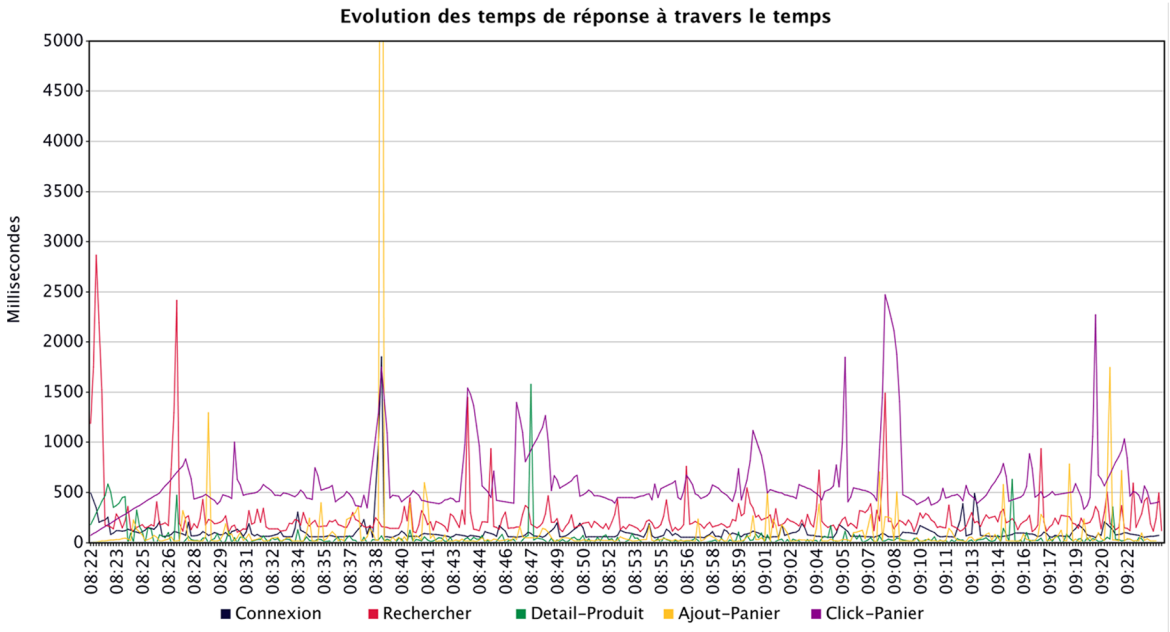
Maîtriser Apache JMeter

Du test de charge à Devops



Antonio Gomes Rodrigues
Bruno Demion (Milamber)
Philippe Mouawad

APACHE JMeter™

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

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

Droits

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[1] et lors de relectures techniques de livre.

Il est actuellement 'committer' et membre PMC du projet JMeter[2] au sein de la fondation Apache Software - ASF[3].

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[4] au sein de la fondation Apache Software - ASF[5]. Il est également ASF member[6] officiel. Son ID Apache est milamber[7].

Milamber a également un blog[8] 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 patchs 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[9] au sein de la fondation Apache Software - ASF[10]. Son ID Apache est pmouawad[11].

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[12].

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[13] est un ensemble de plugins gratuits et open source améliorant certaines parties de JMeter.

Pour cela il va ajouter un certain nombre de :

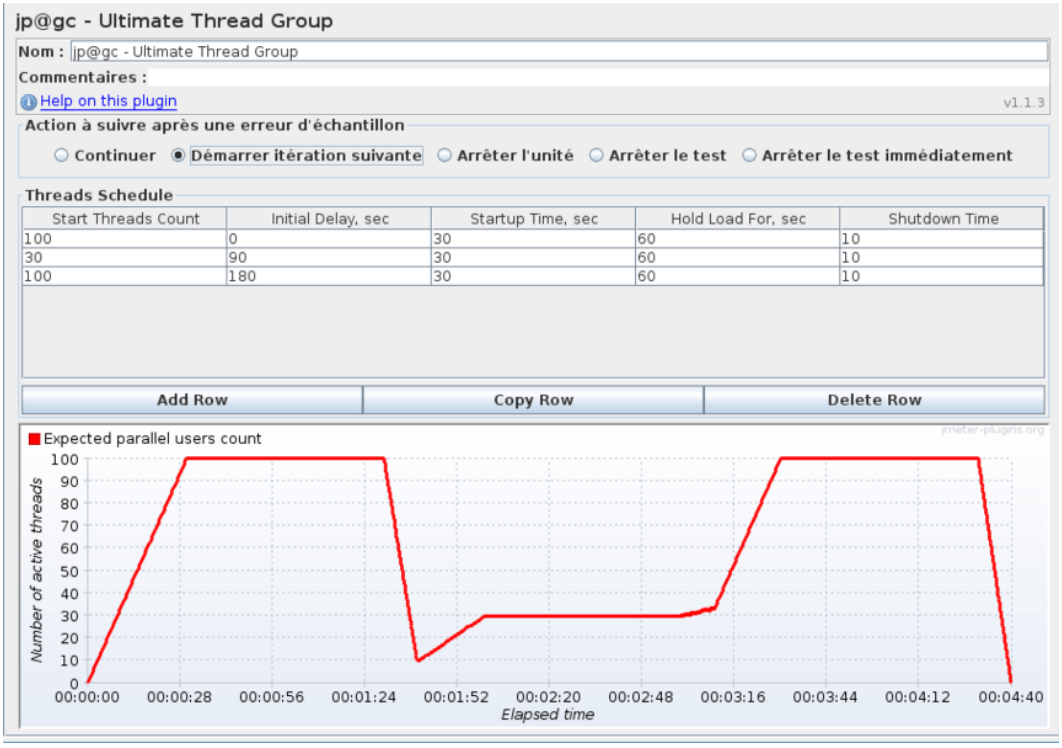
- récepteurs

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.

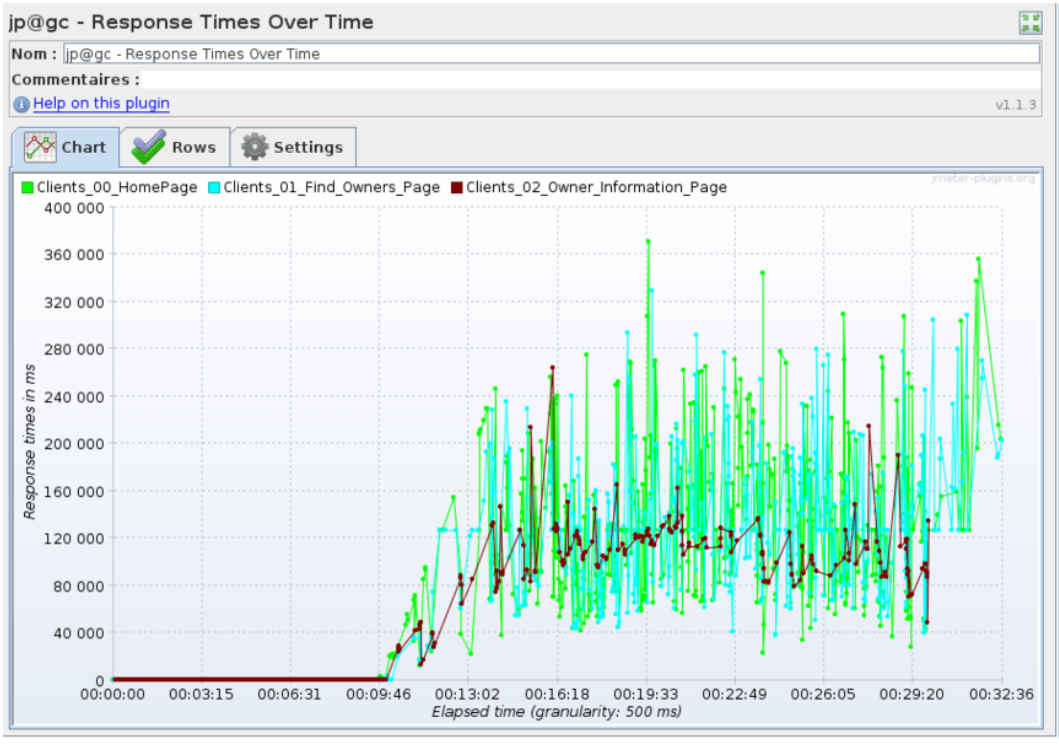
Commençons par le groupe d'unités **Ultimate Thread Group**. Ce plugin va nous permettre de contrôler facilement et visuellement notre injection.



Ultimate Thread Group

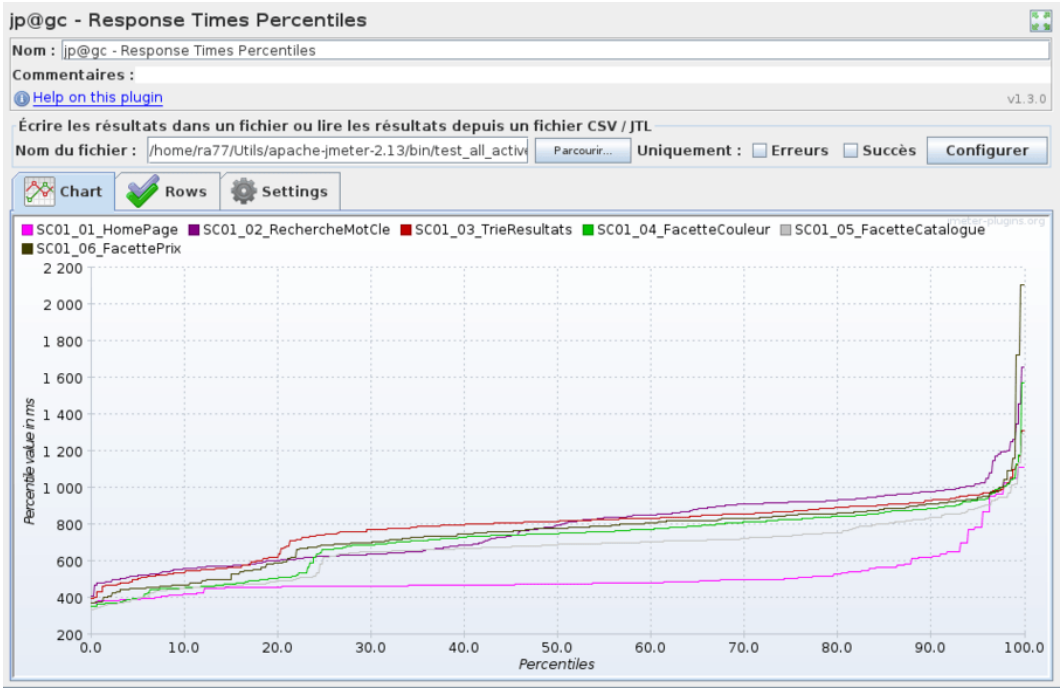
Regardons d'un peu plus près le point fort de JMeter Plugins : les graphiques.

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.

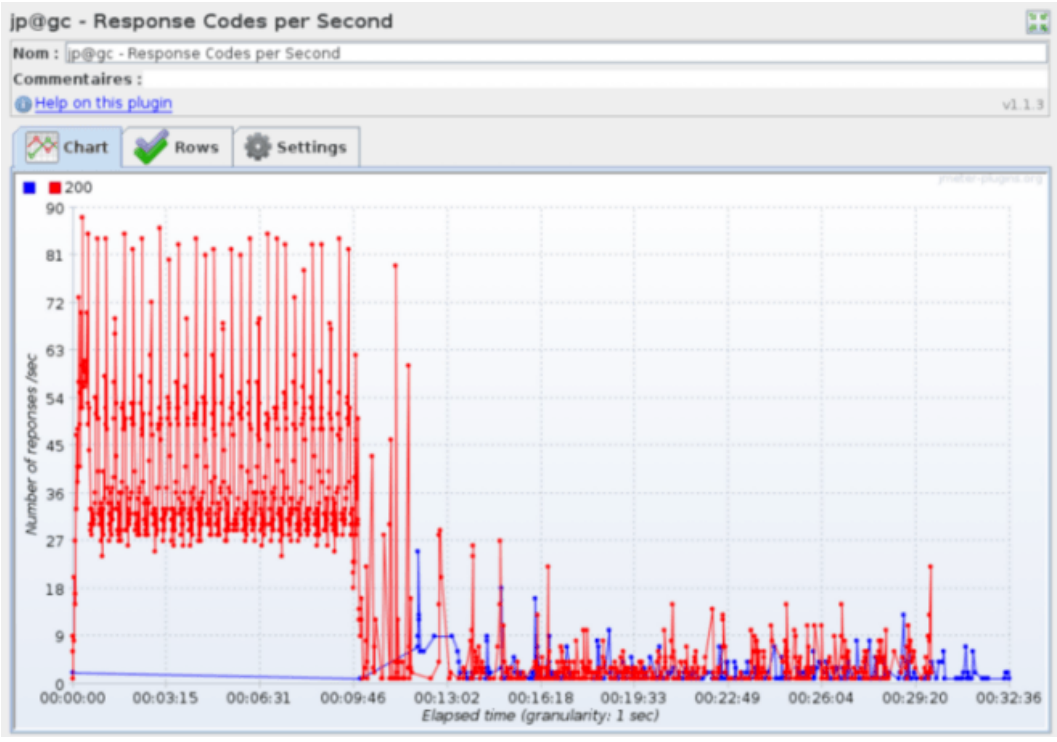


Response Times Percentiles

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

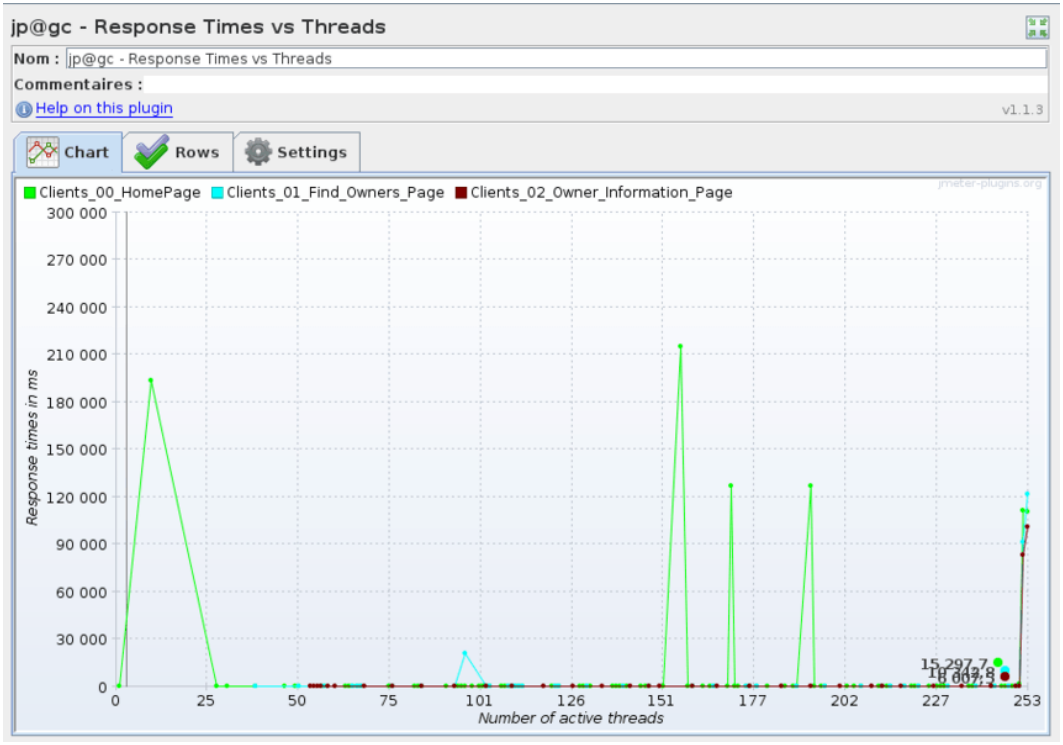
Active Threads Over Time

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



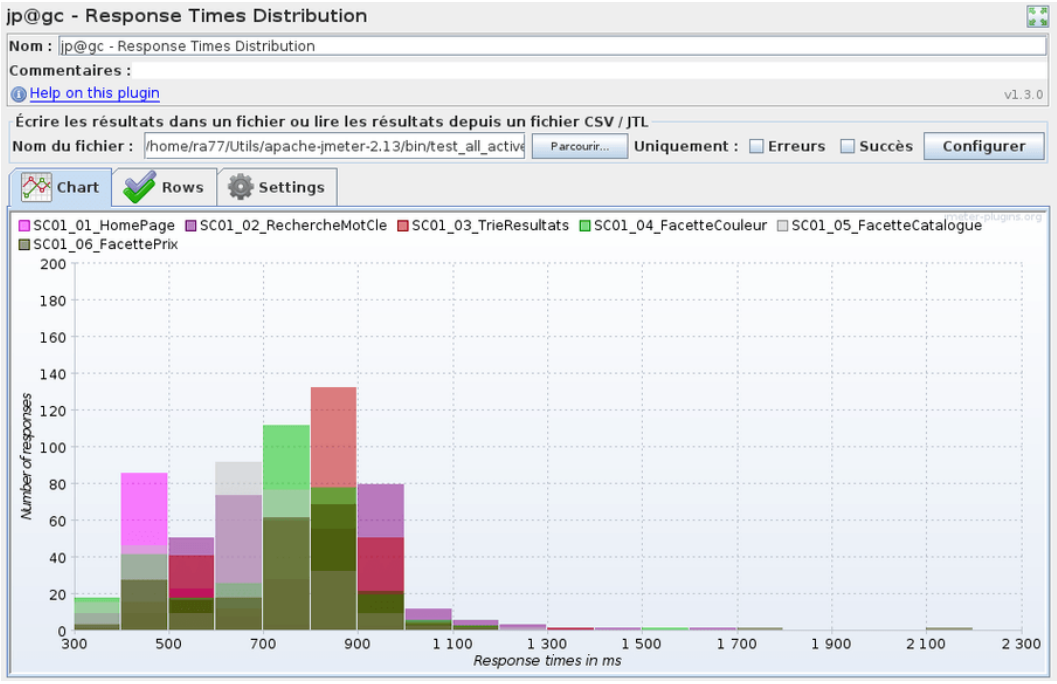
Response Codes per Second

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



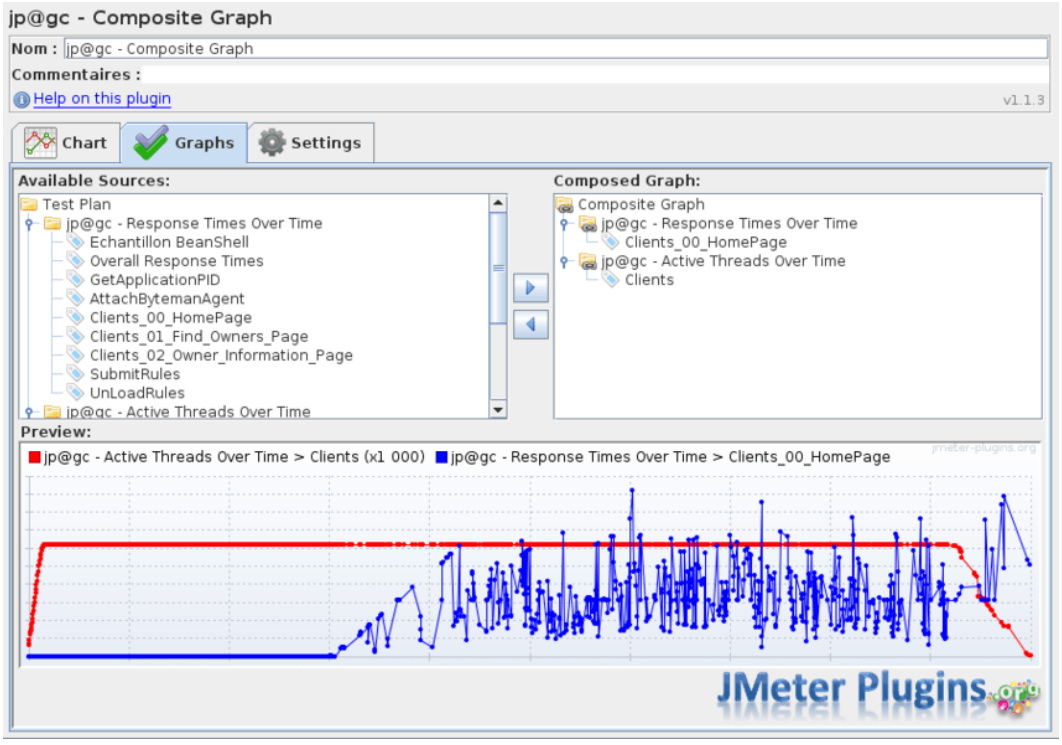
Response Times vs Threads

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



Response Times Distribution

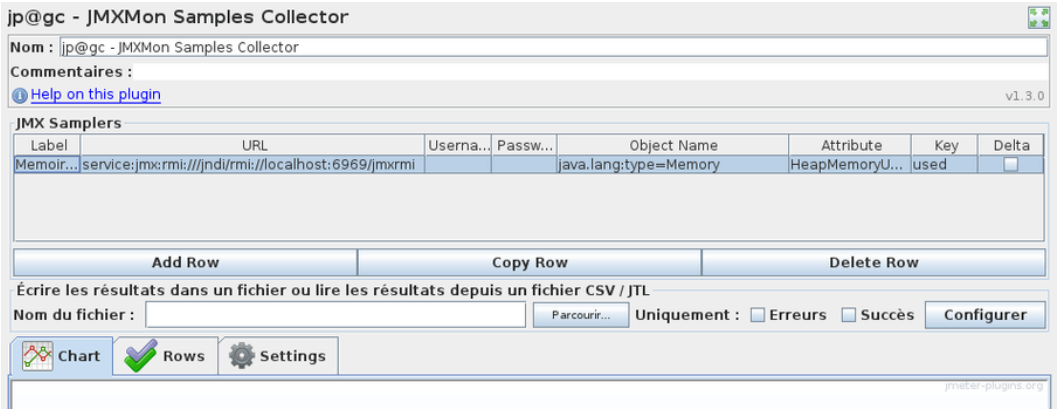
Si cela ne suffit toujours pas, il est possible de superposer les graphes à l'aide des données de tous les autres graphes en utilisant **Composite Graph**.



Composite Graph

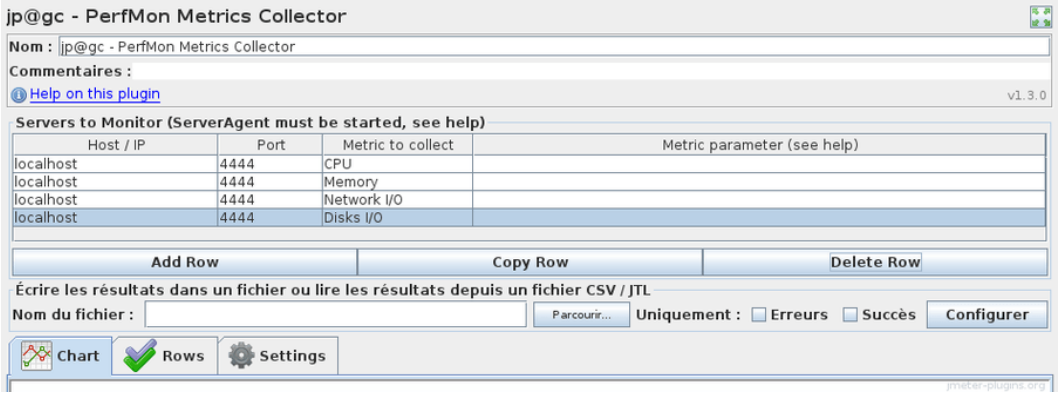
Pour l'instant, nous avons utilisé des graphiques se basant sur les données du test, mais il est possible de récupérer des données externes.

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



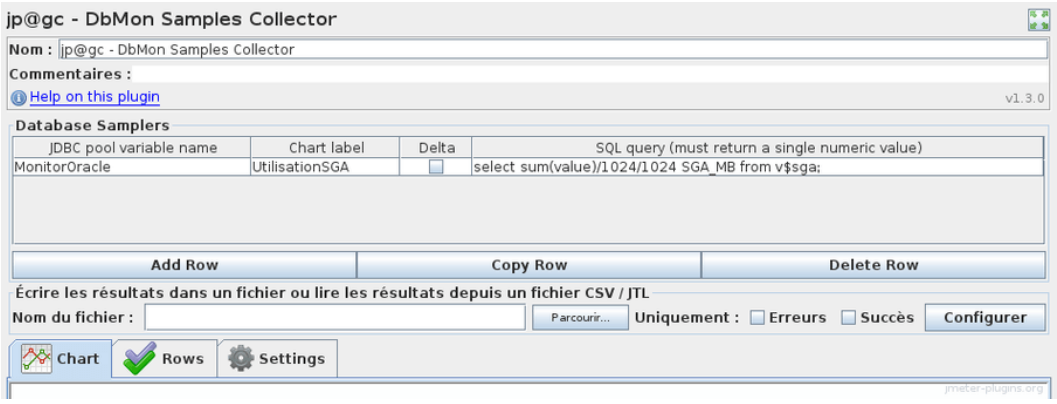
JMXMon Samples Collector

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



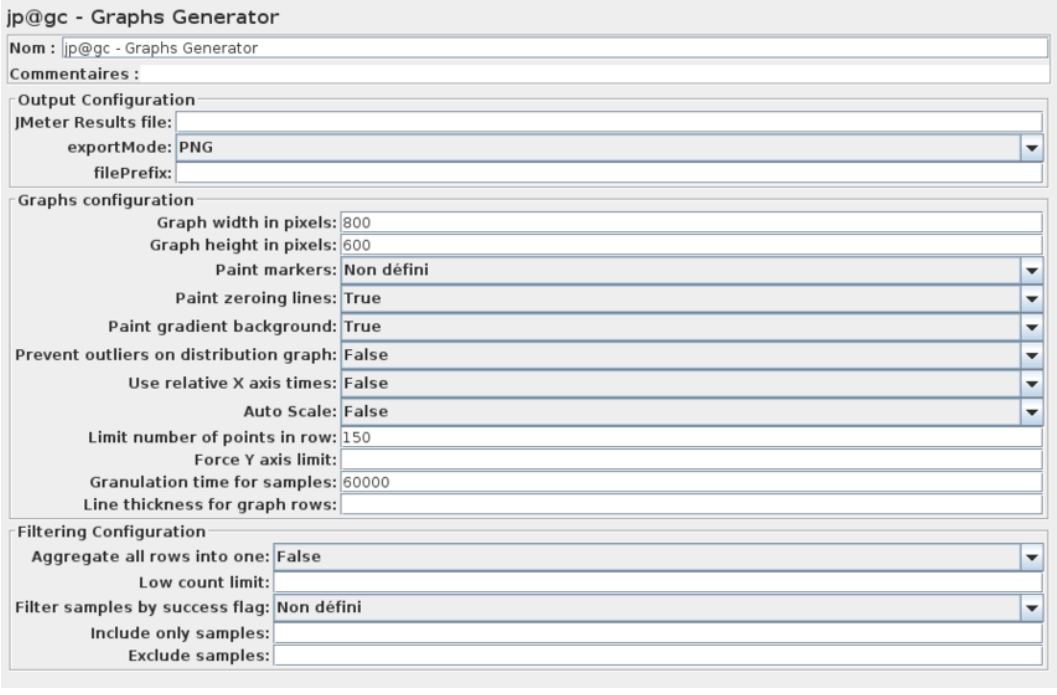
PerfMon Metrics Collector

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



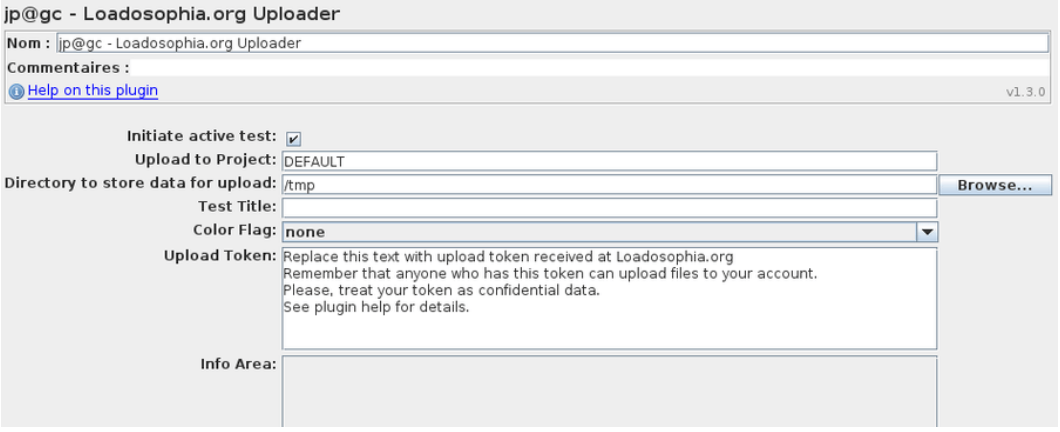
DbMon Samples Collecto

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

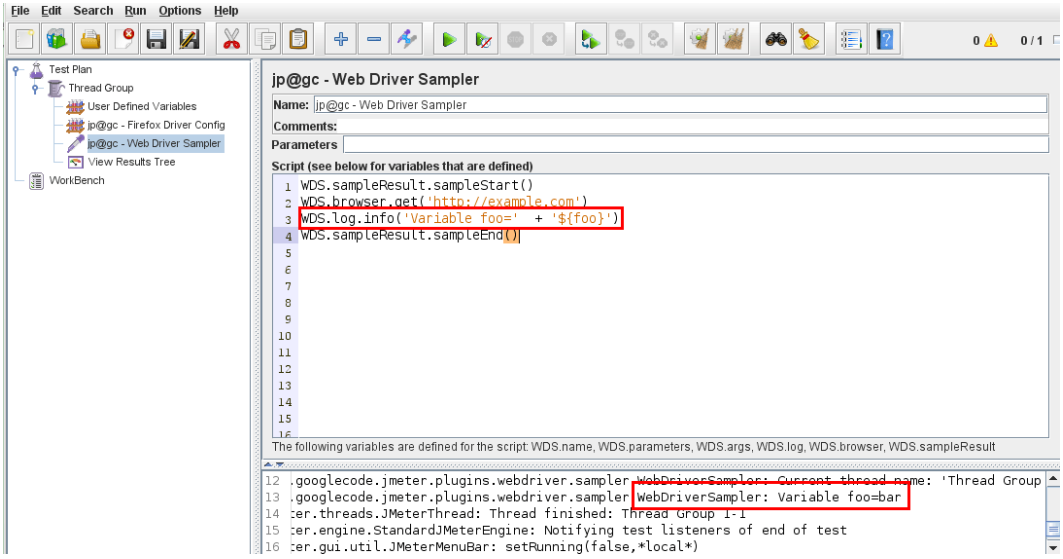
Une autre option consiste à uploader ses résultats sur Loadosophia.org à l'aide de **Loadosophia.org Uploader**.



Loadosophia.org Uploader

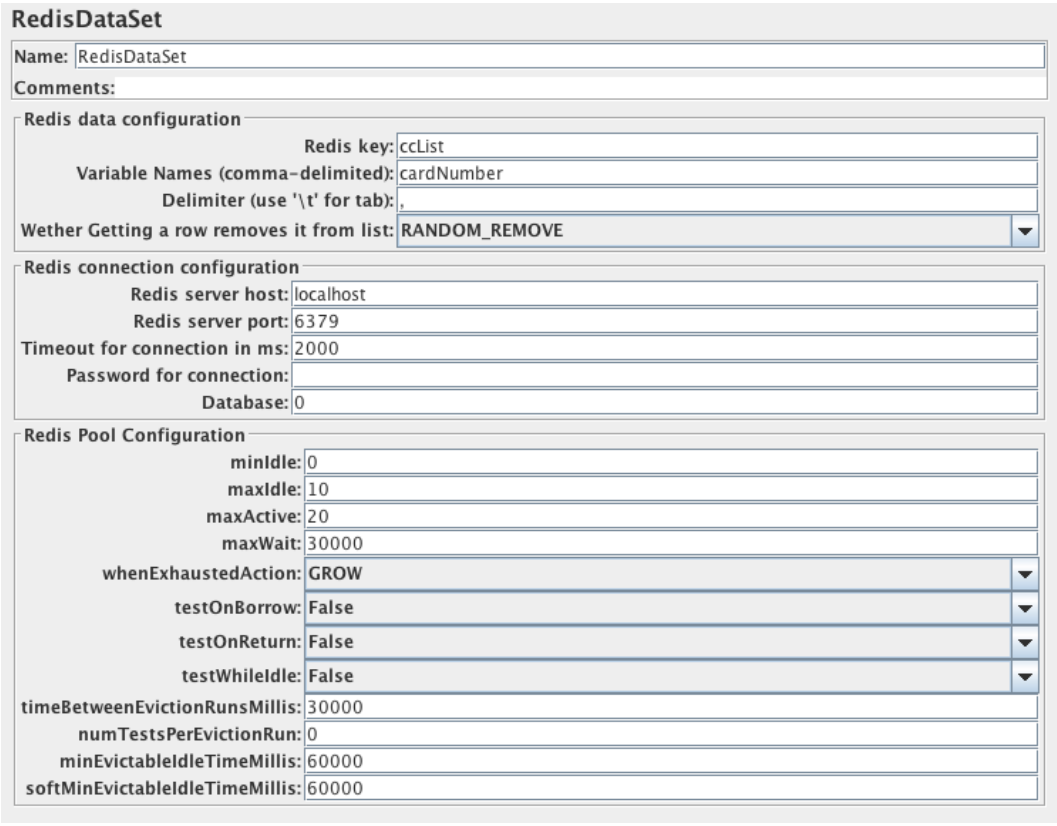
Si nous souhaitons connaître le ressenti utilisateur dans le browser (les temps de réponse JMeter n'intègrent pas le rendu dans le browser), nous pourrons utiliser dans notre test :

- 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.



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.

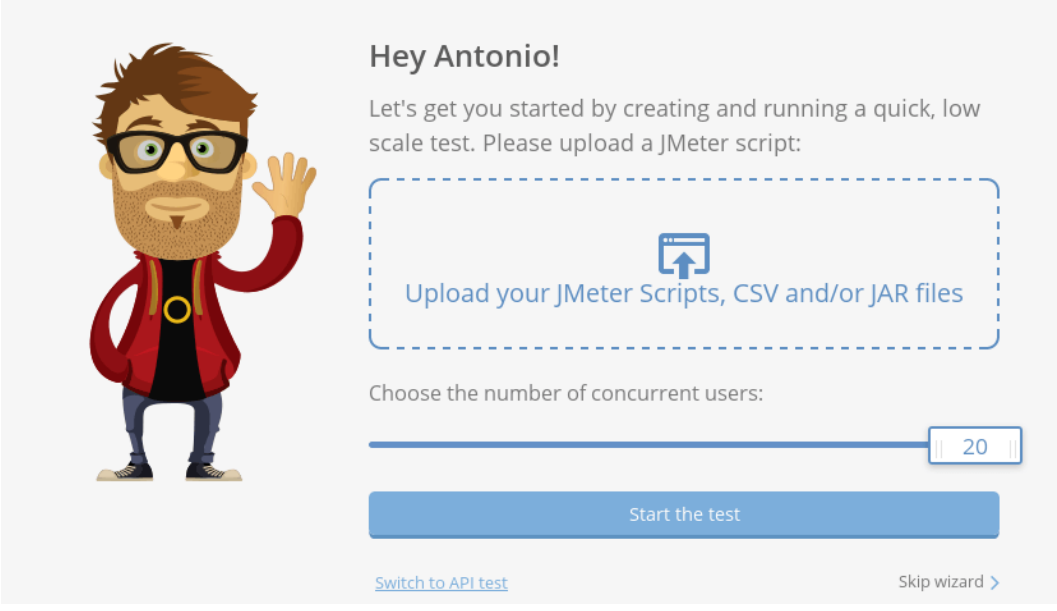
Pour cela, une des solutions possibles est l'utilisation de services commerciaux dans le cloud.

BlazeMeter

À noter que BlazeMeter[14] 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.

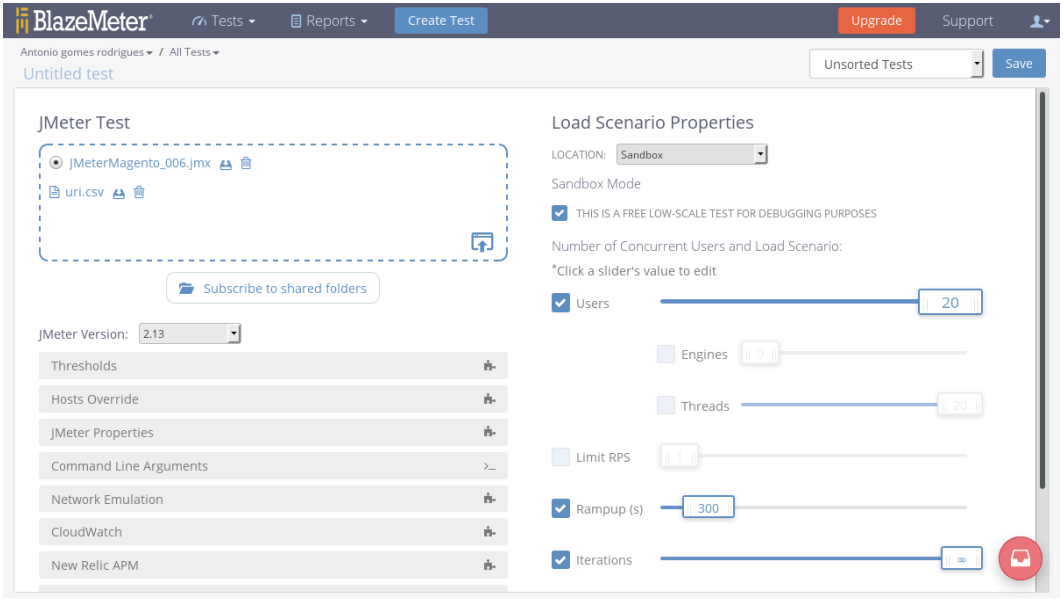
Lorsque nous nous connectons sur le site pour créer un test, un assistant apparaît.



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/

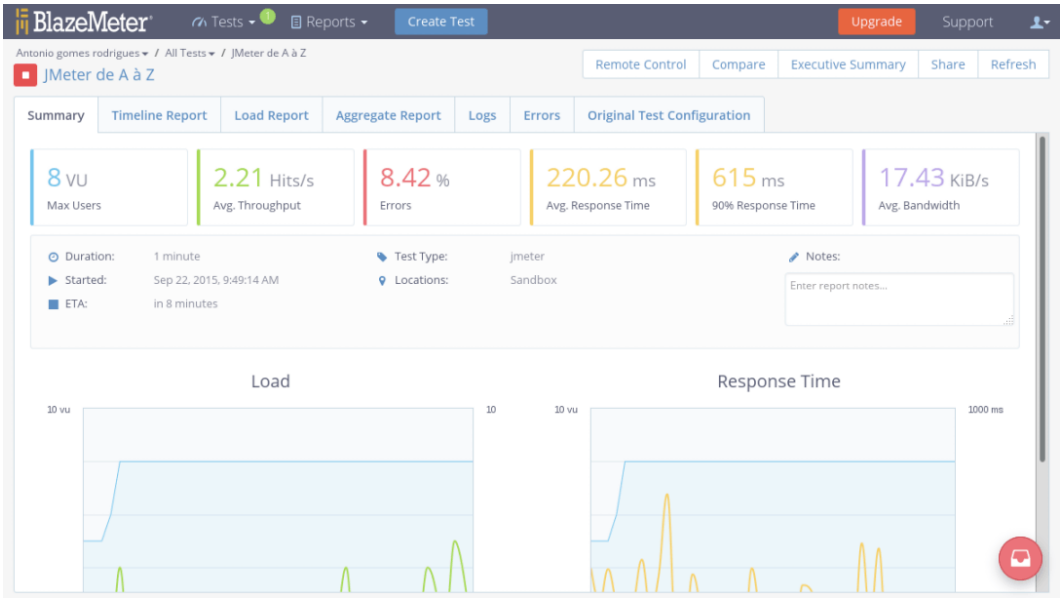


BlazeMeter - Création d'un test

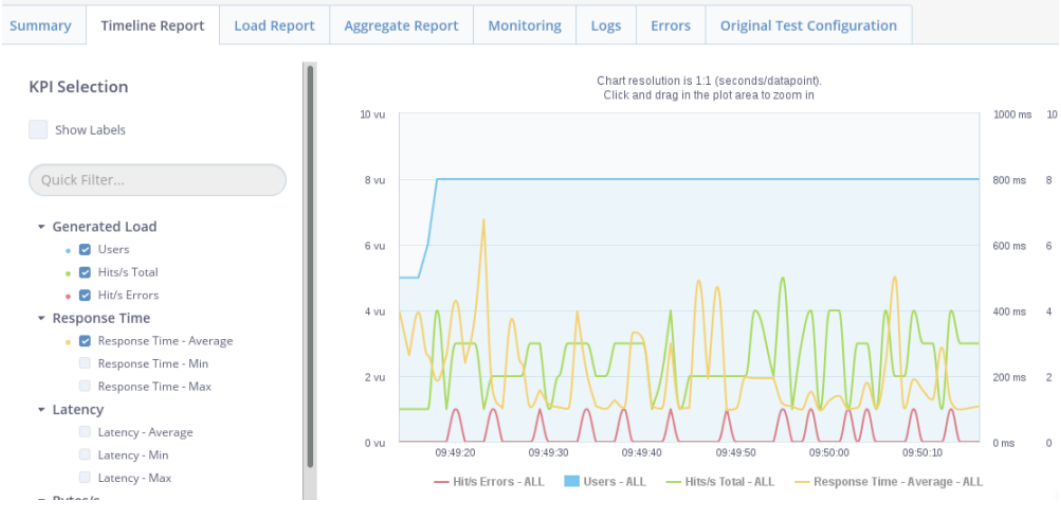
Nous pourrons :

- 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 interfacer avec des outils d'APM (Newrelic, Dynatrace)
- Surcharger des valeurs de propriétés
- Etc.

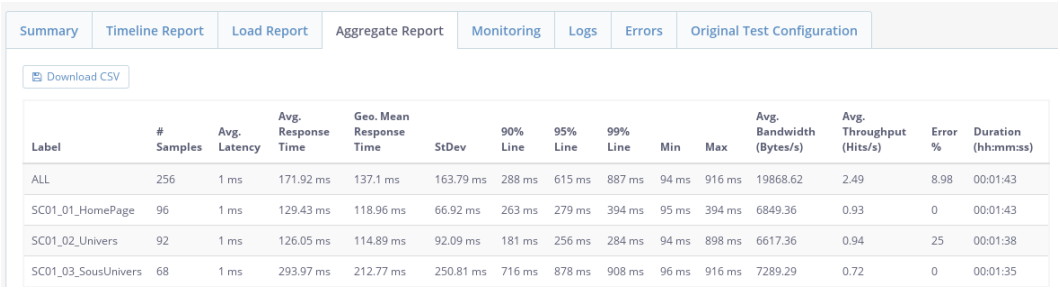
Lançons notre test et suivons son évolution en temps réel.



BlazeMeter - Suivi d'un test



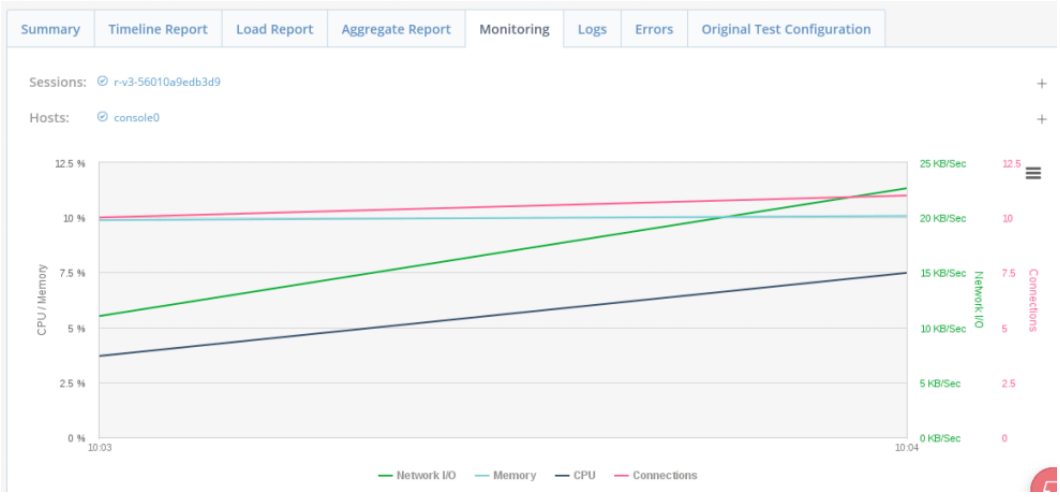
BlazeMeter - Suivi d'un test



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

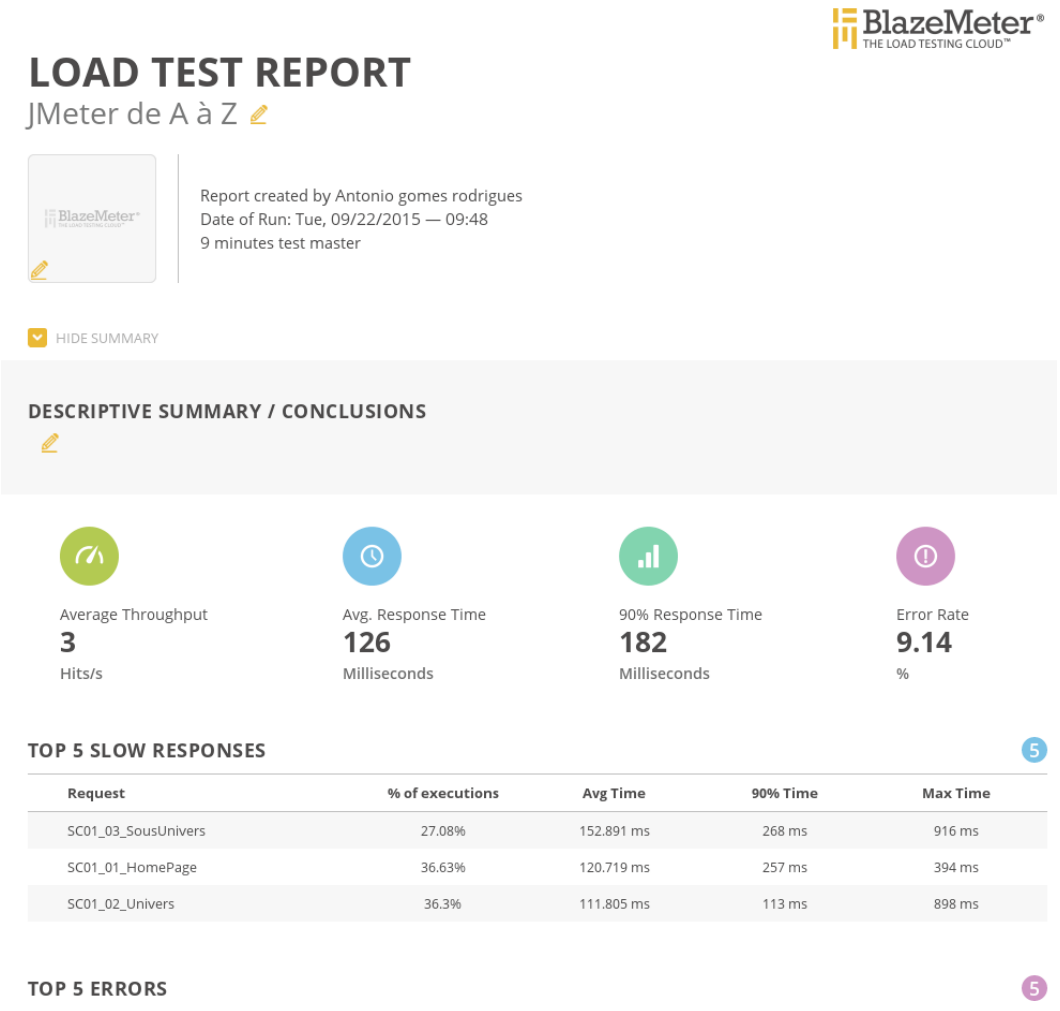
BlazeMeter - Suivi d'un test

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.



BlazeMeter - Rapport de test

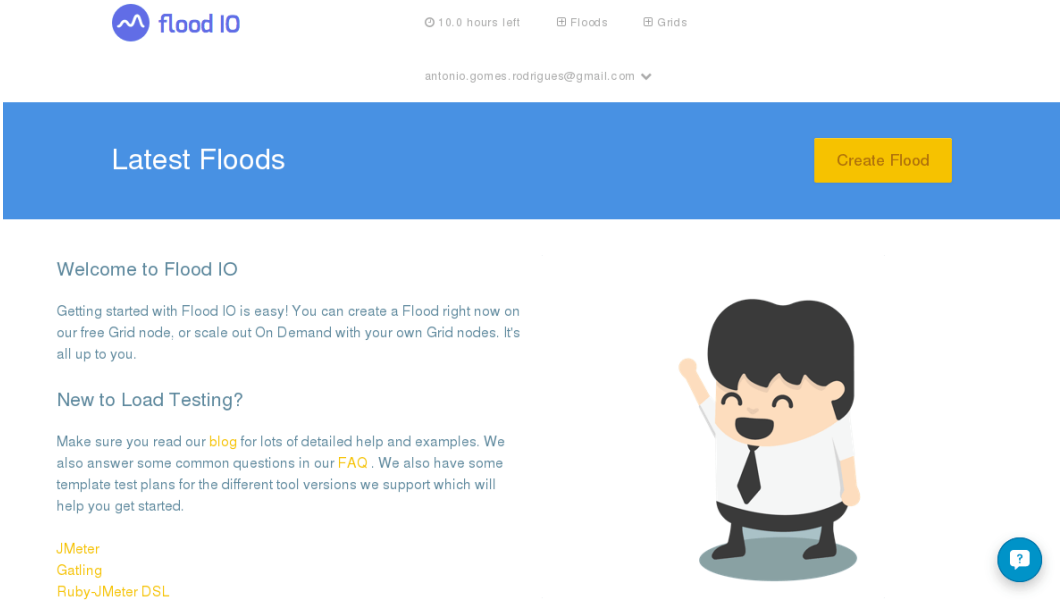
Tricentis Flood

Tricentis Flood[15] 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/

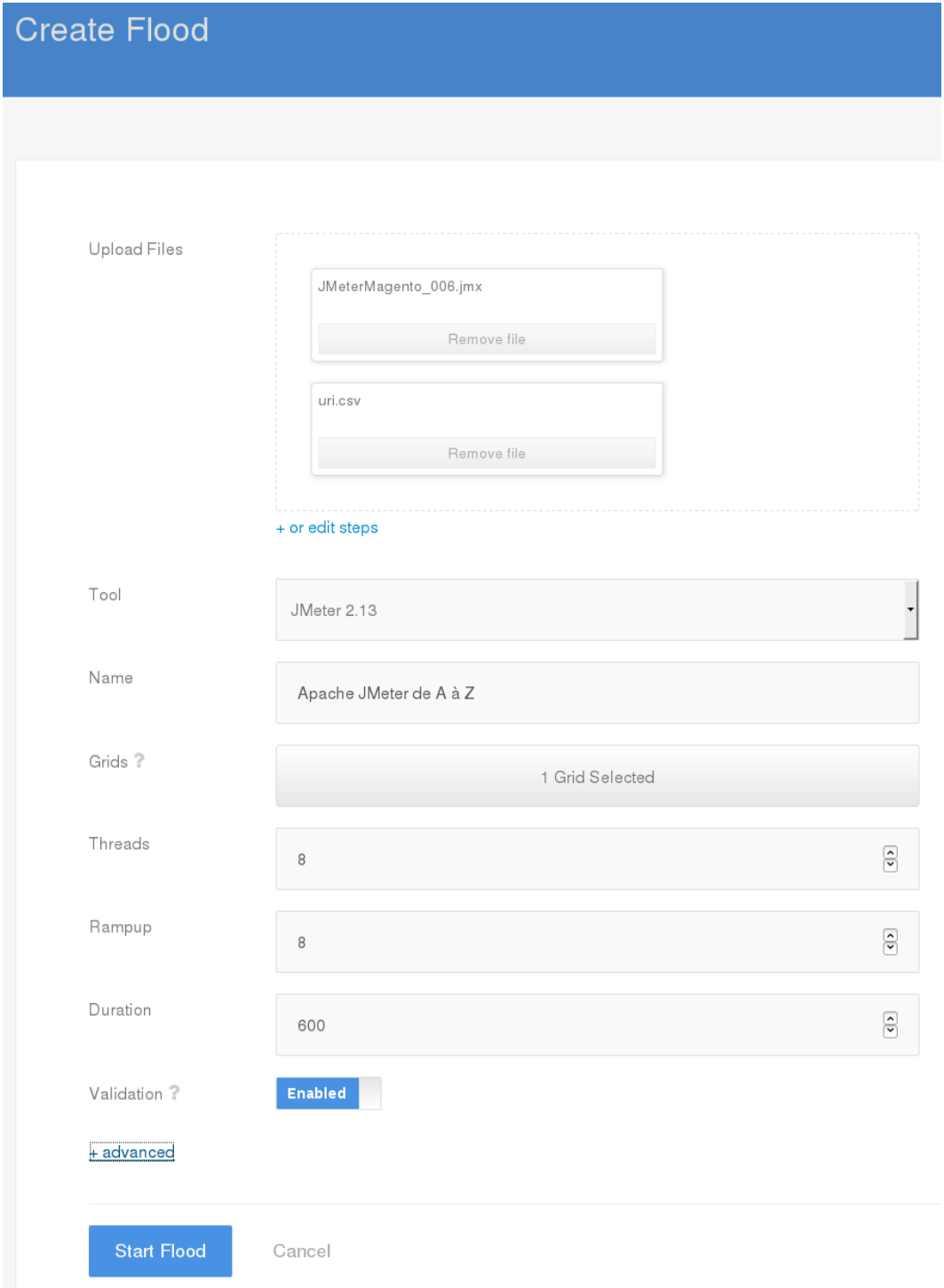
Son interface est très simple d’utilisation.



Tricentis Flood

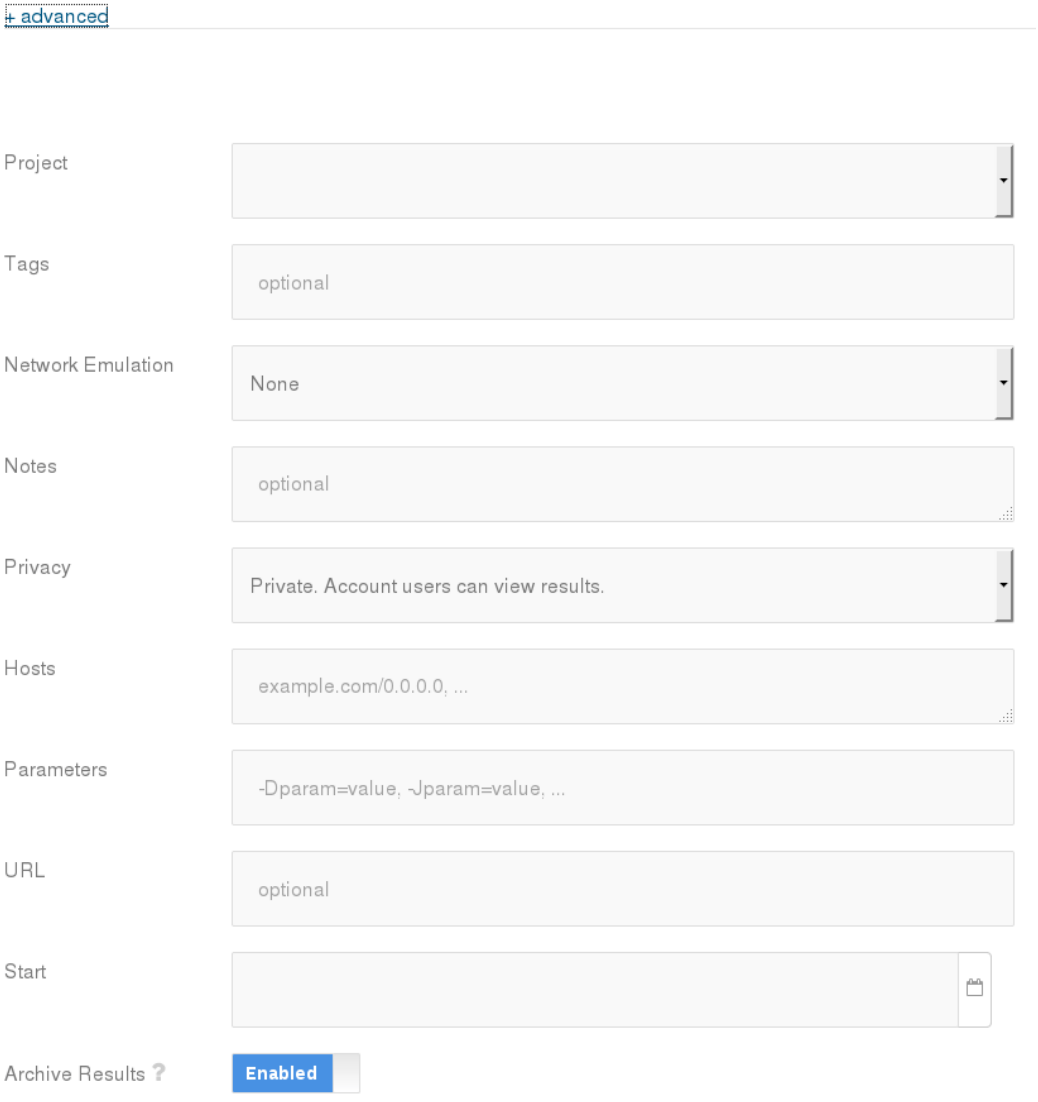
Nous importons d’abord notre script accompagné de son jeu de données.

Puis nous définissons les paramètres de notre test.



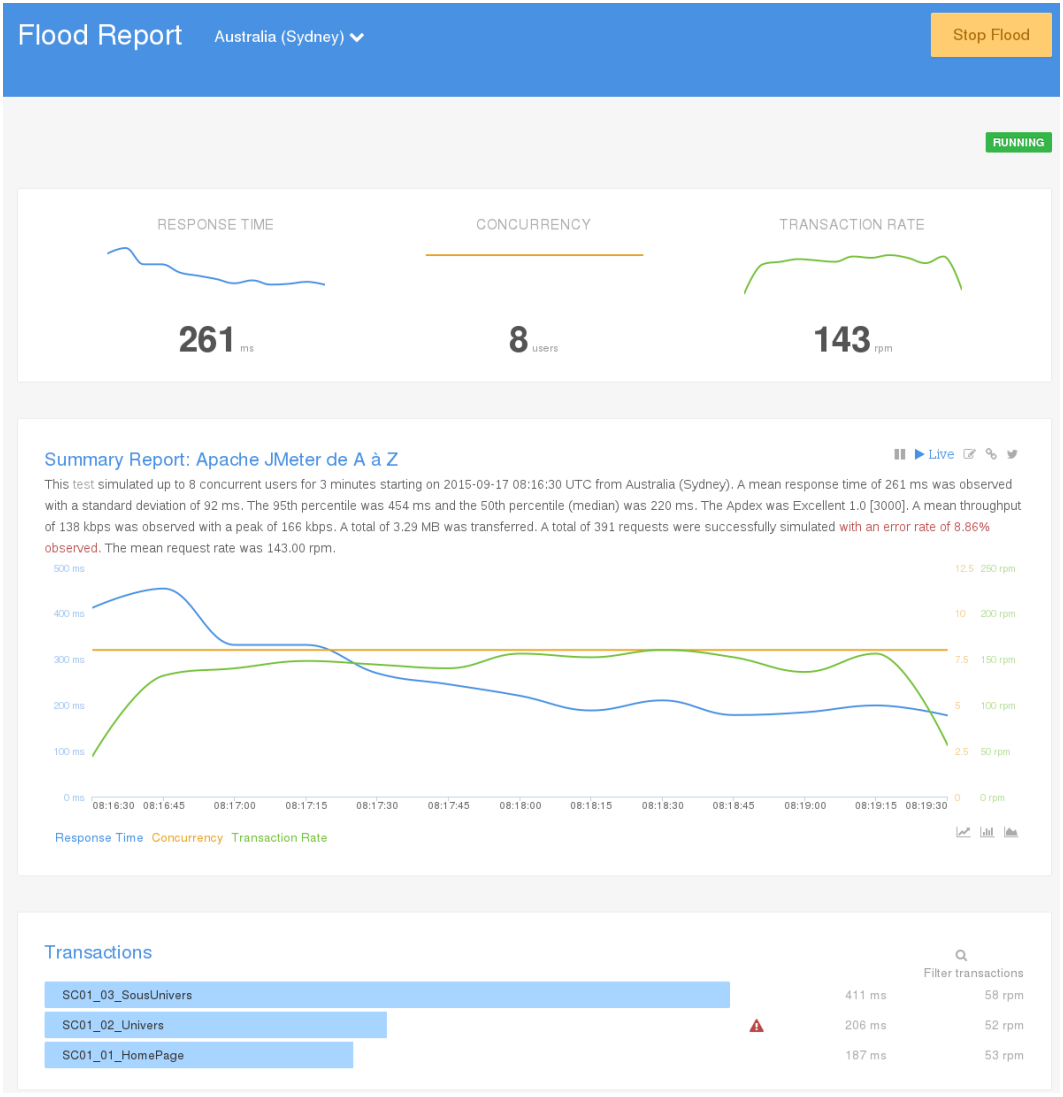
Tricentis Flood - Création d'un test

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



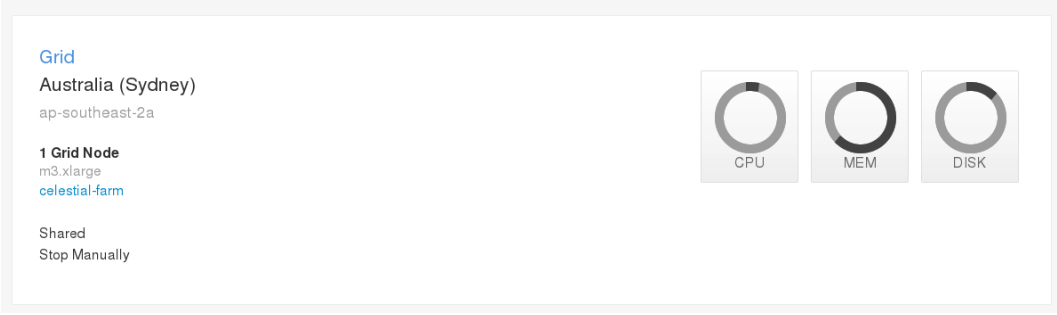
Tricentis Flood - Création d'un test : advanced

Lançons notre test et suivons son exécution en temps réel.



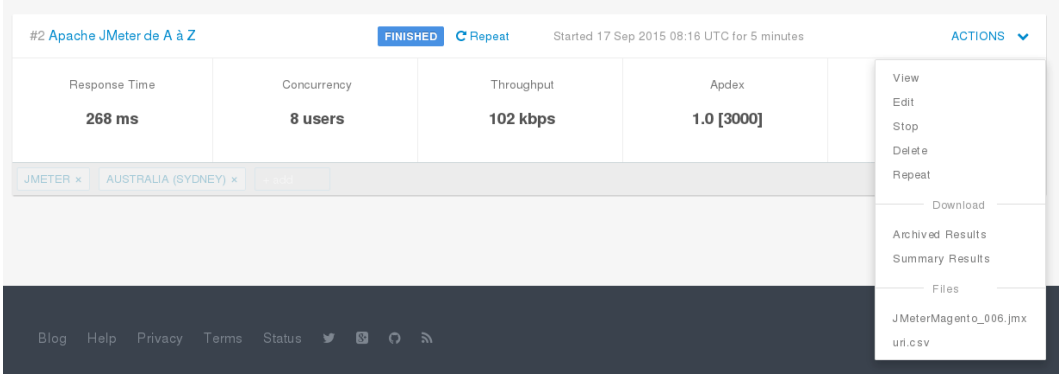
Tricentis Flood - Suivi d'un test

Une supervision des injecteurs est proposée.



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.



Tricentis Flood - Rapport de test

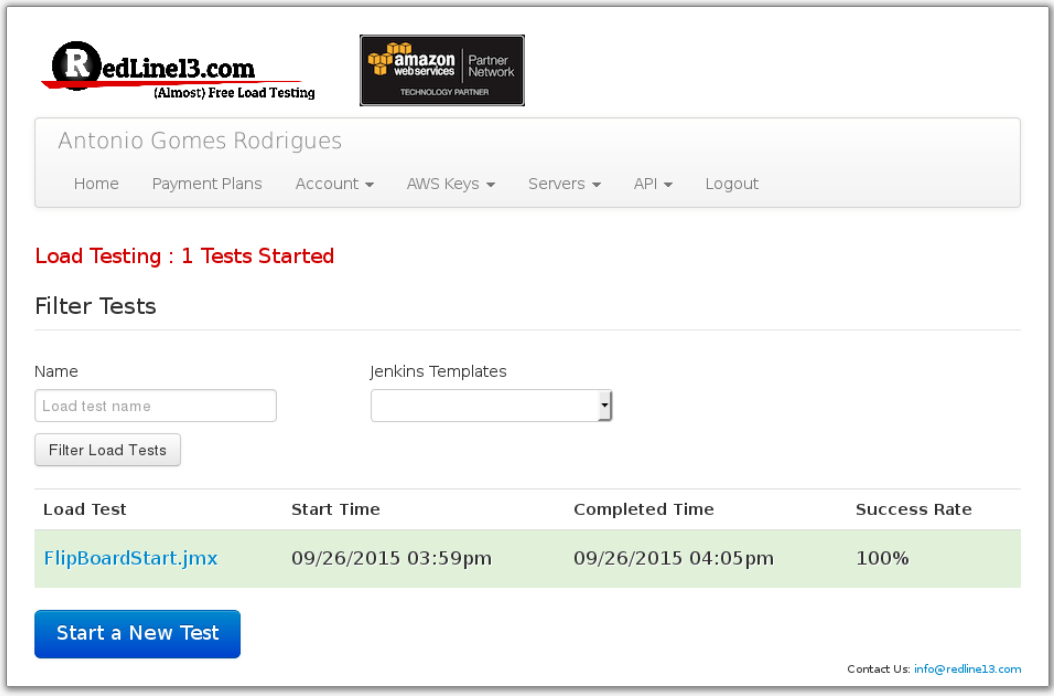
Redline 13

Redline 13[16] 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.

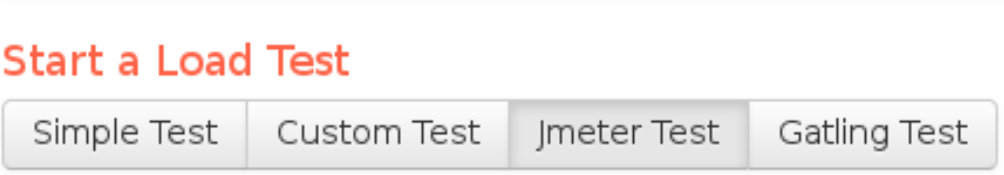


Redline 13 - Page d'accueil

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

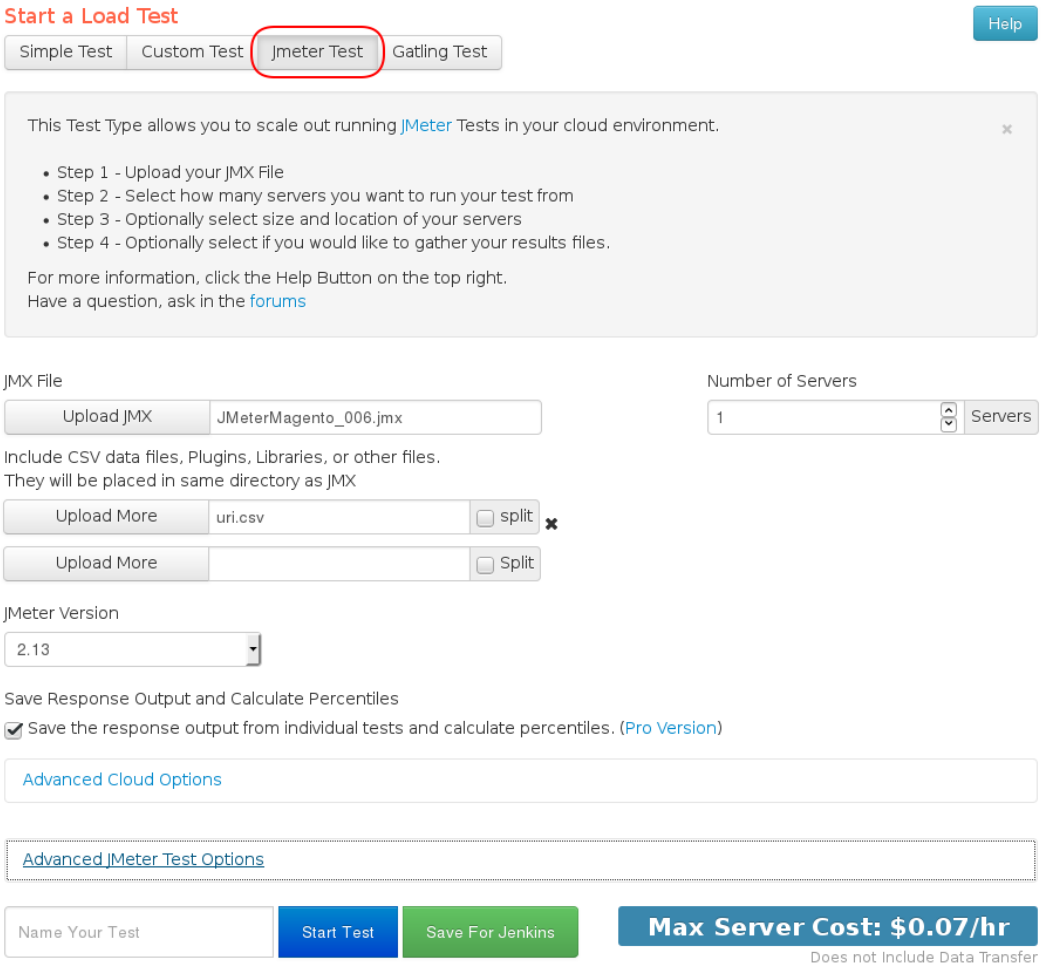


Redline 13 - Type de test

Choisissons JMeter comme type de tests.

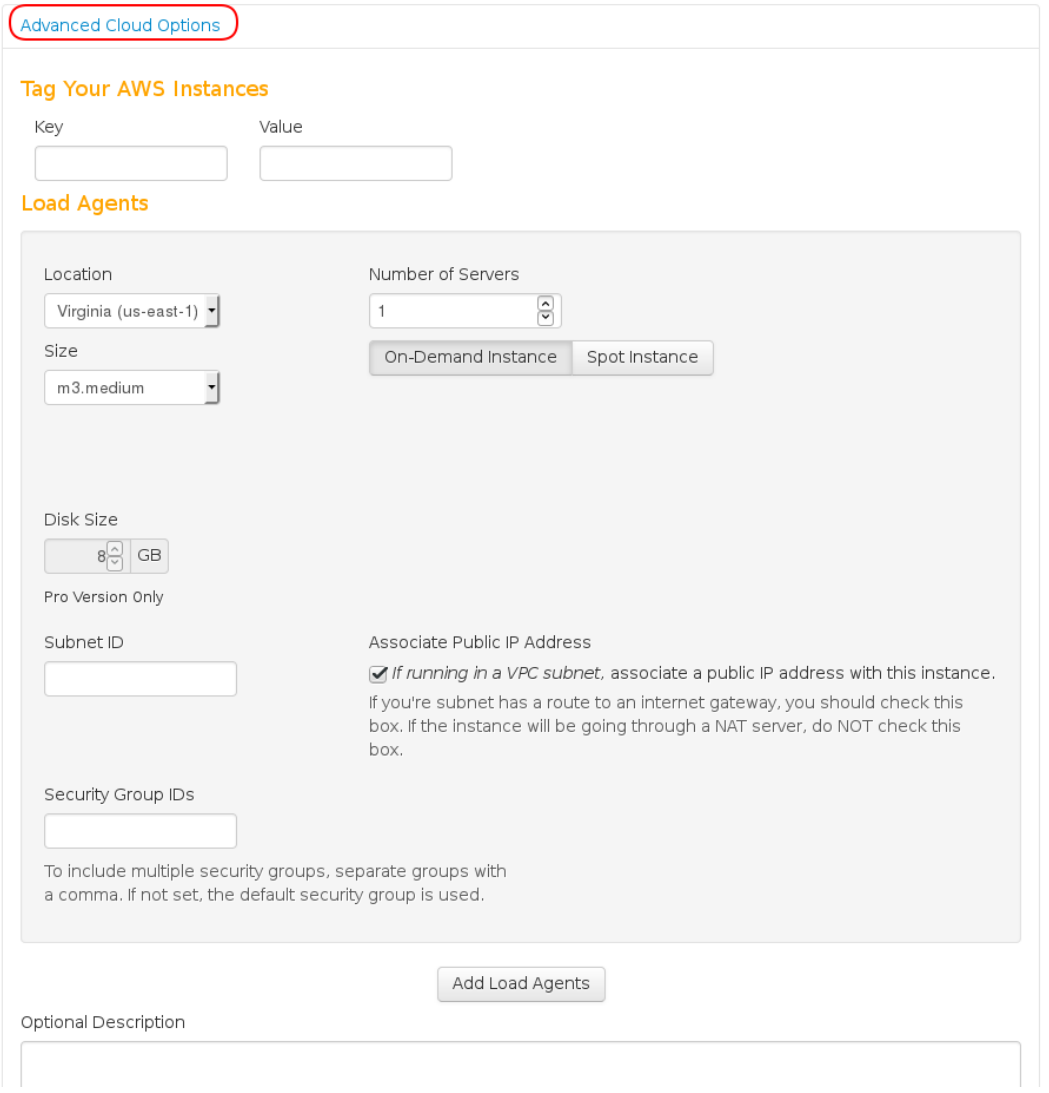
Ici, nous pouvons choisir :

- 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



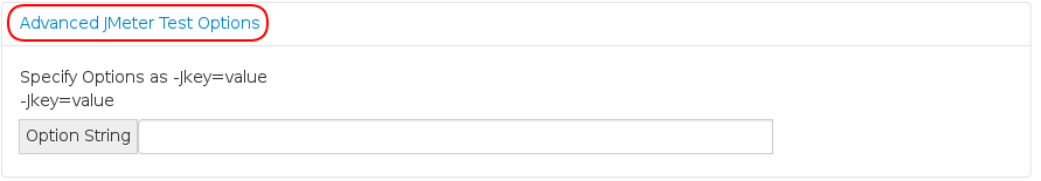
Redline 13 - Création de notre plan de test

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.



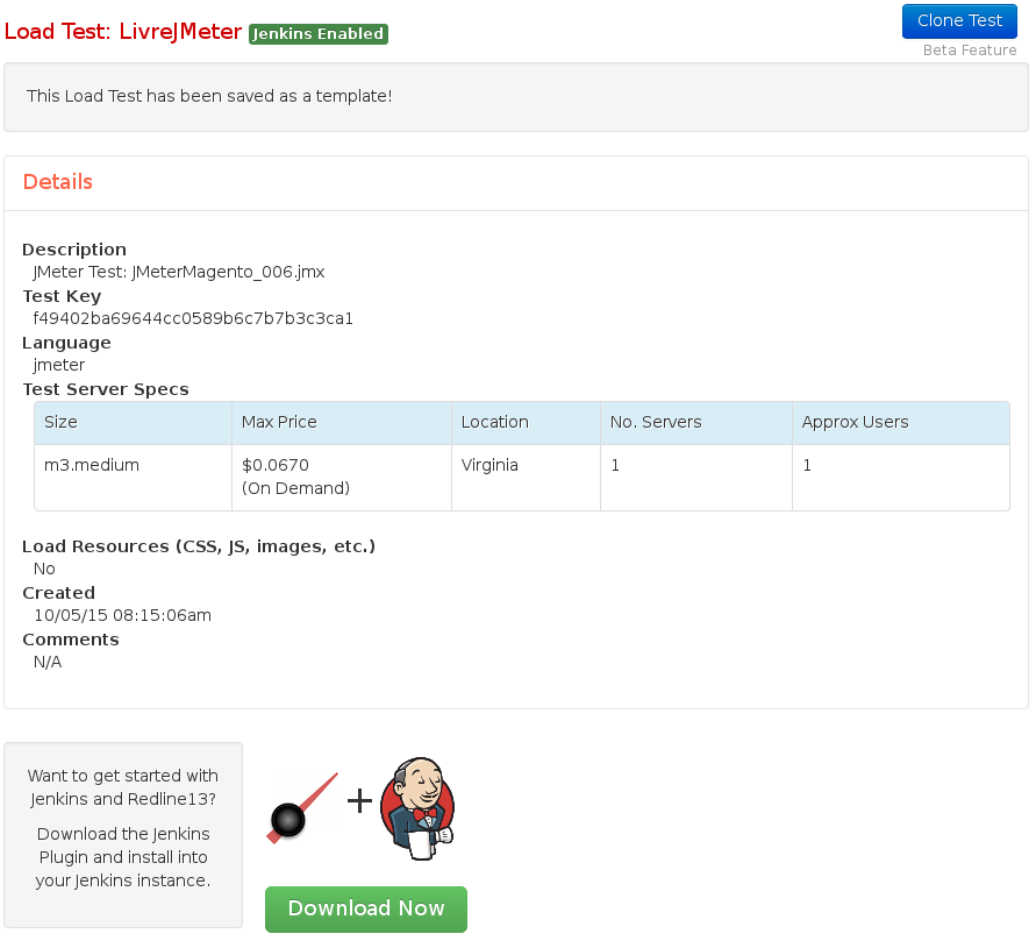
Redline 13 - Création de notre plan de test - options avancées

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



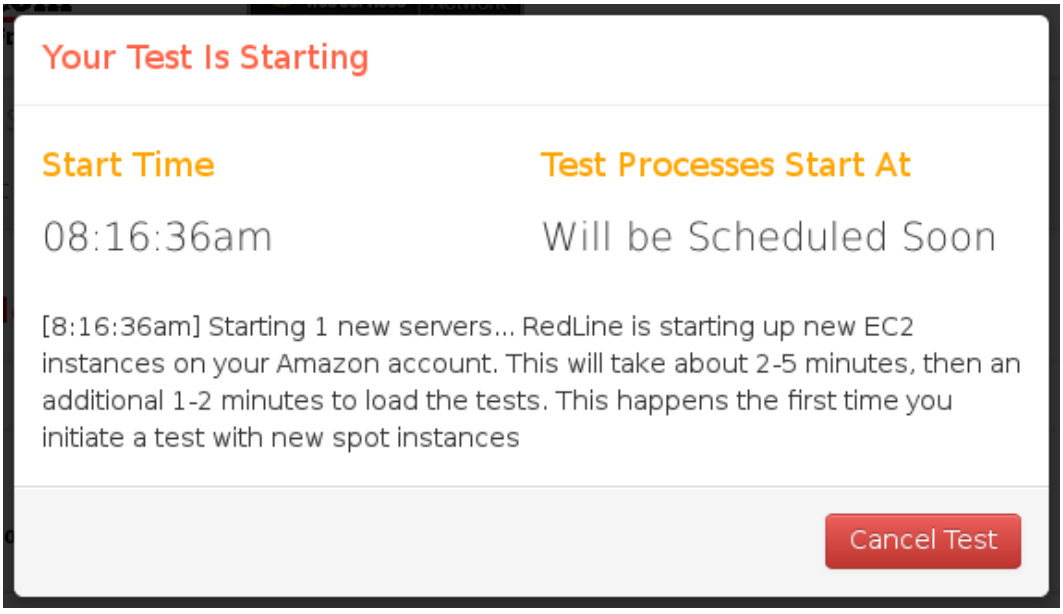
Redline 13 - Création de notre plan de test - options avancées JMeter

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



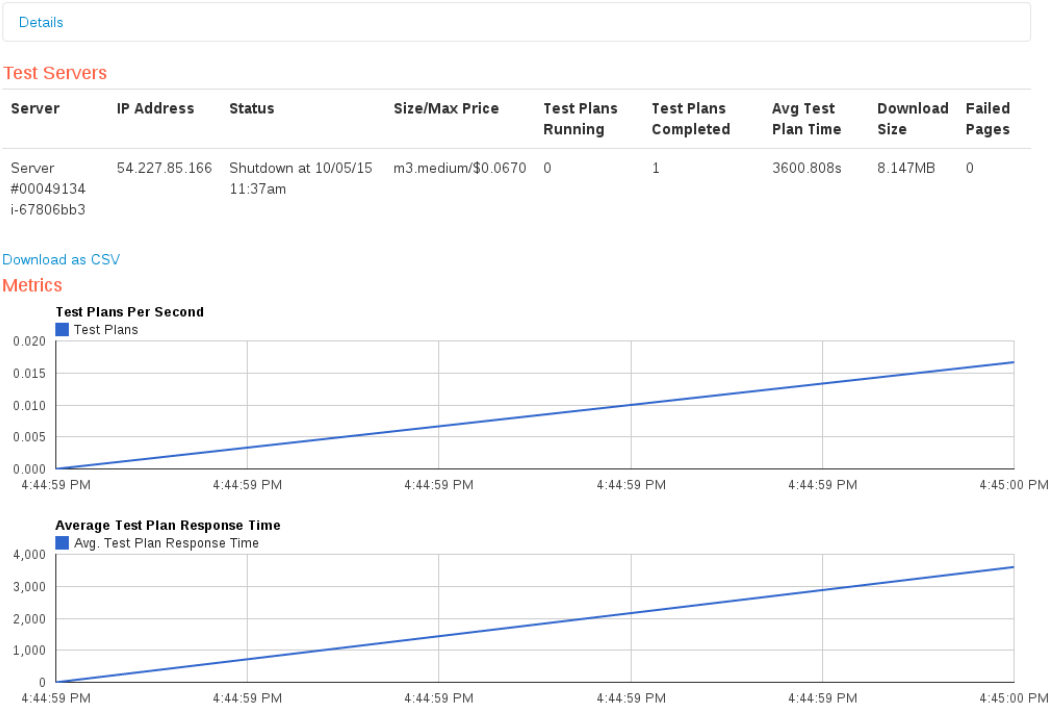
Redline 13 - Modèle personnalisé pour Jenkins

Exécutons notre test.



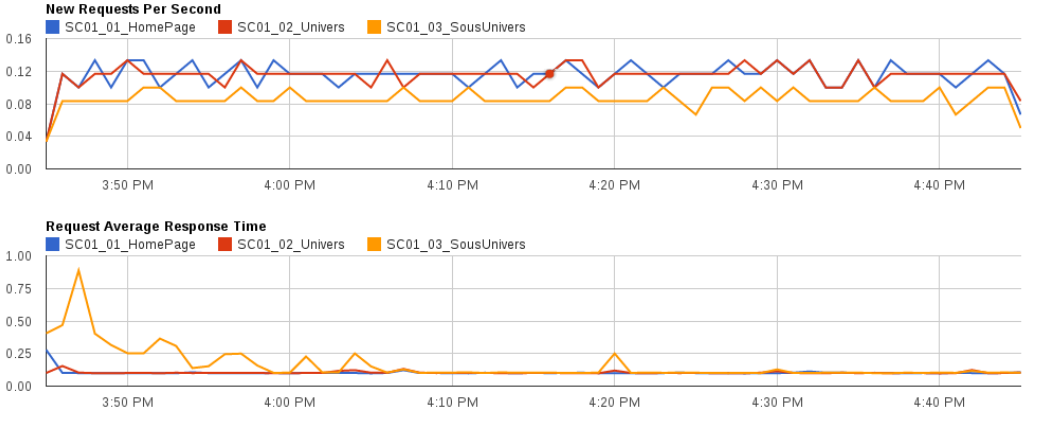
Redline 13 - Démarrage de notre test

Et suivons-le.



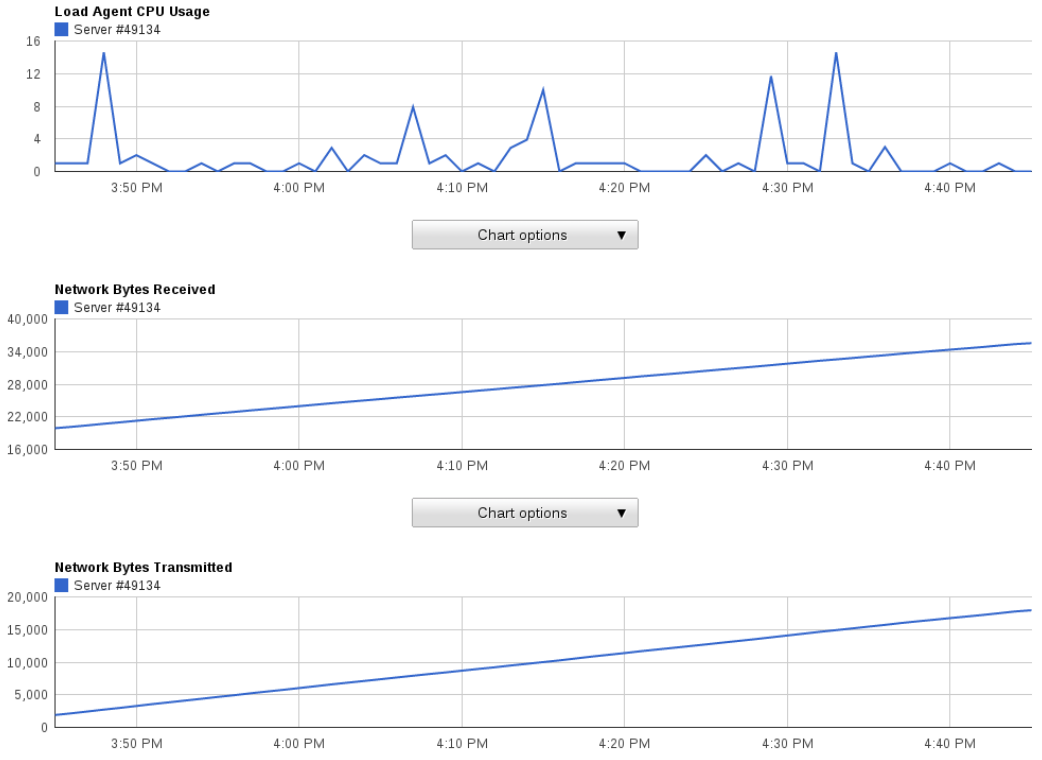
Redline 13 - Suivi de notre test

Les métriques de notre scénario sont disponibles.



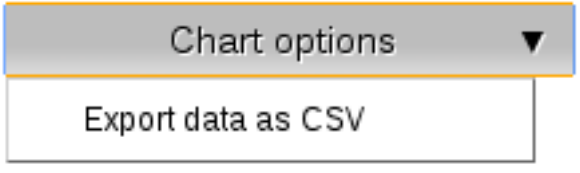
Redline 13 - Suivi de notre test : métriques de notre scénario

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.

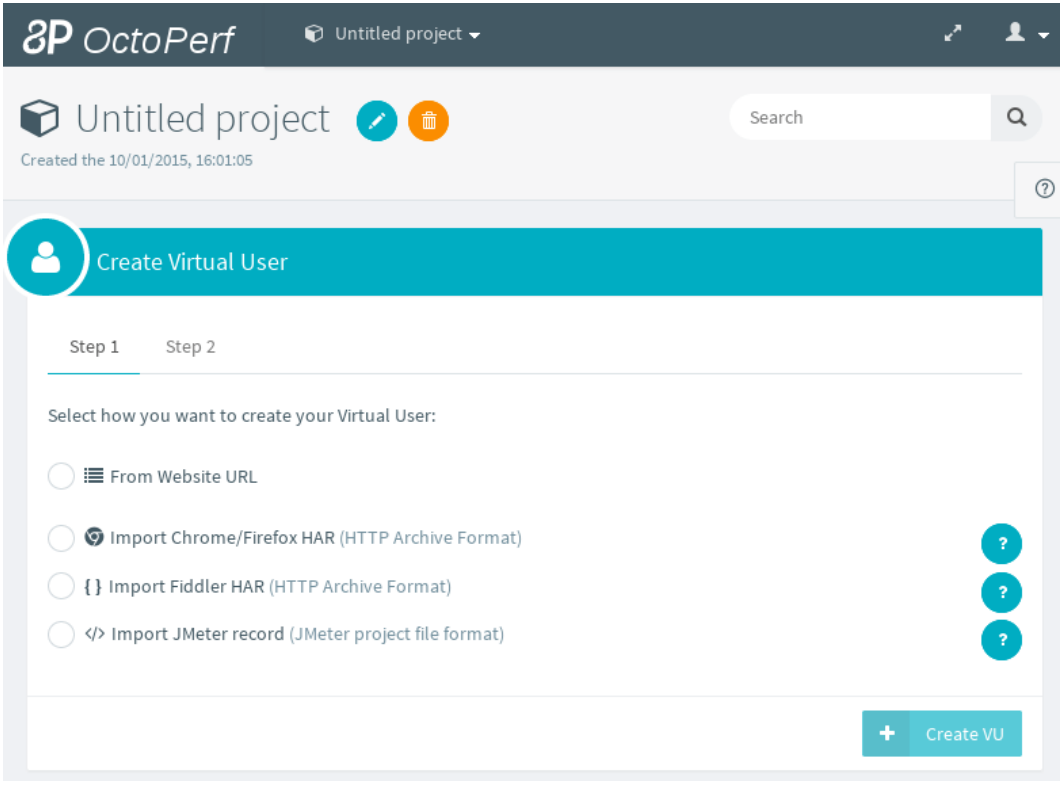
OctoPerf (anciennement Jellly.IO)

Dernière solution qui a vu le jour, OctoPerf[17] 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 :

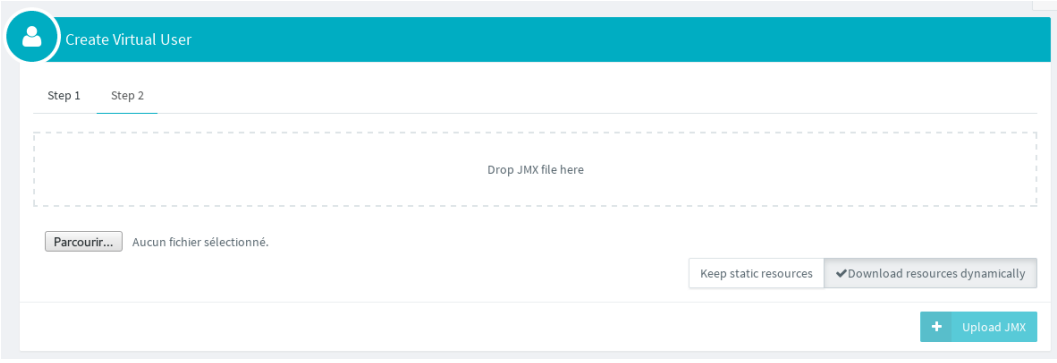
- Tester une URL
- Utiliser un fichier HAR (HTTP Archive) comme script
- Importer un script JMeter



OctoPerf

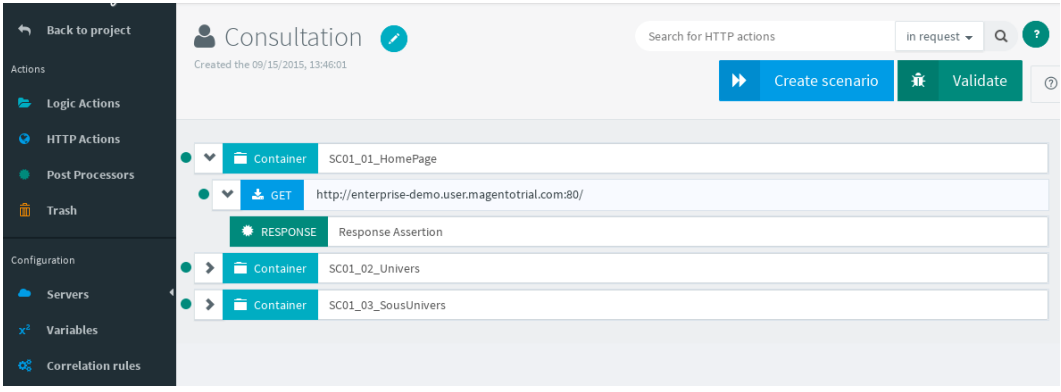
17. http://octoperf.com/

Nous choisissons d'importer notre script JMeter.



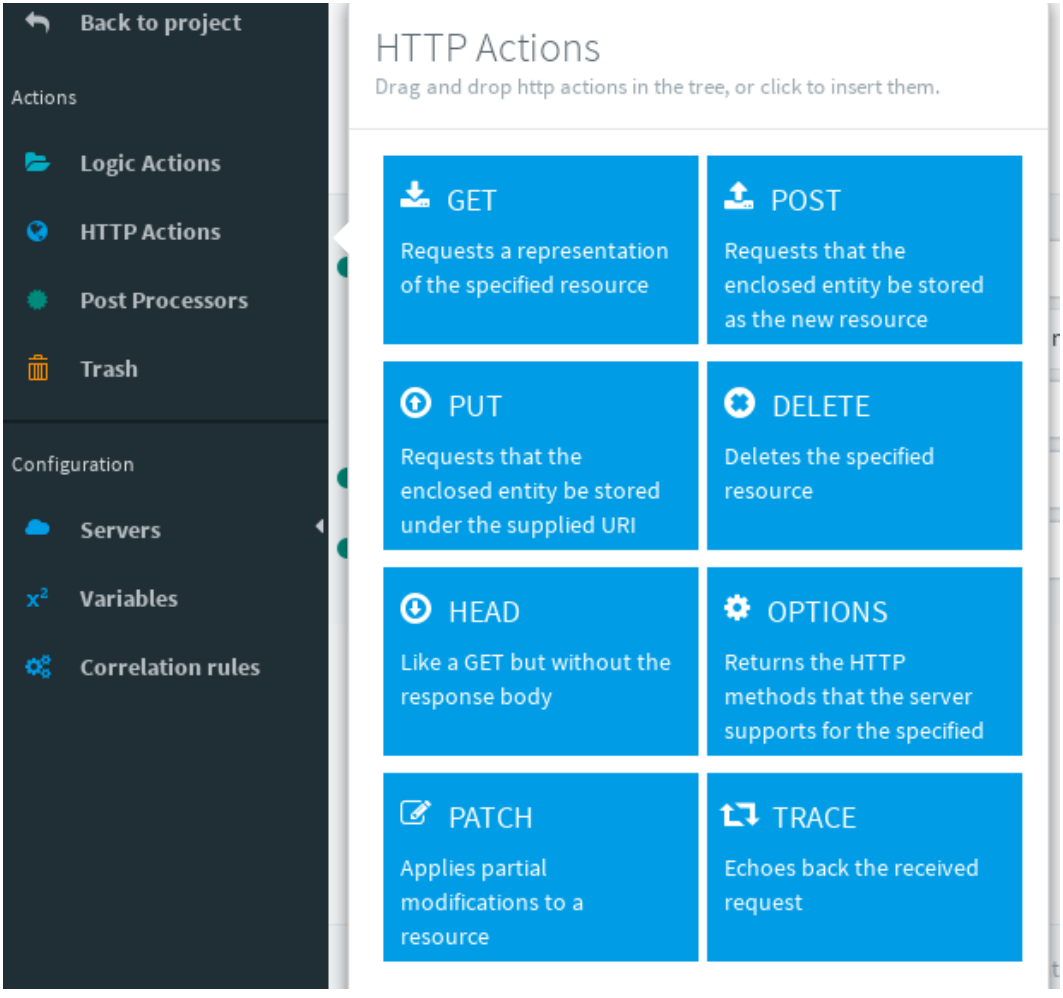
OctoPerf - Import d'un script JMeter

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

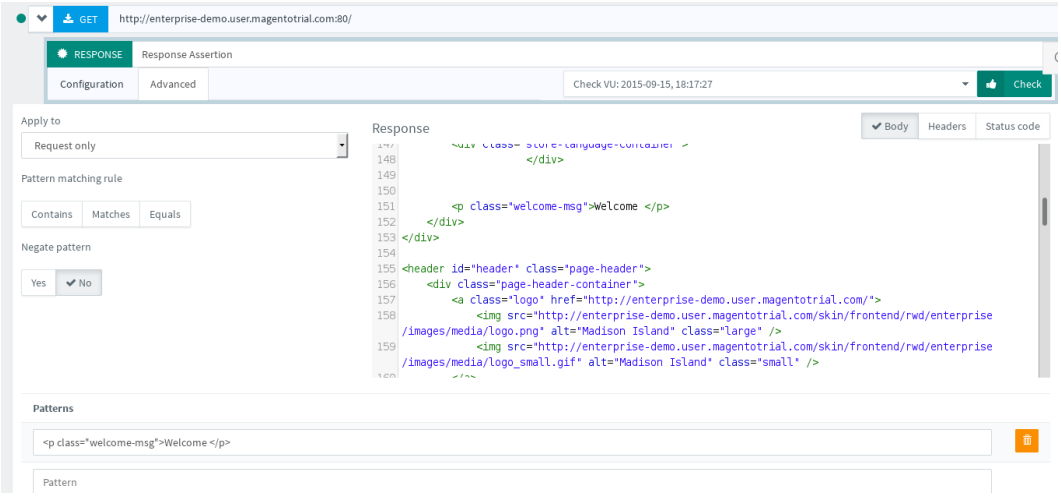


OctoPerf - Notre script

Si nécessaire nous pouvons modifier notre script.

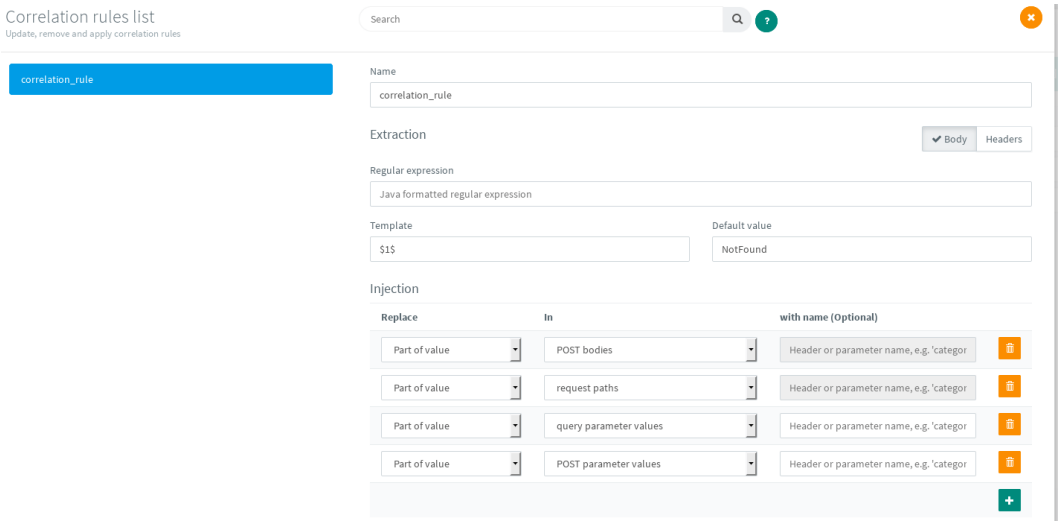


OctoPerf - Modification d'un script



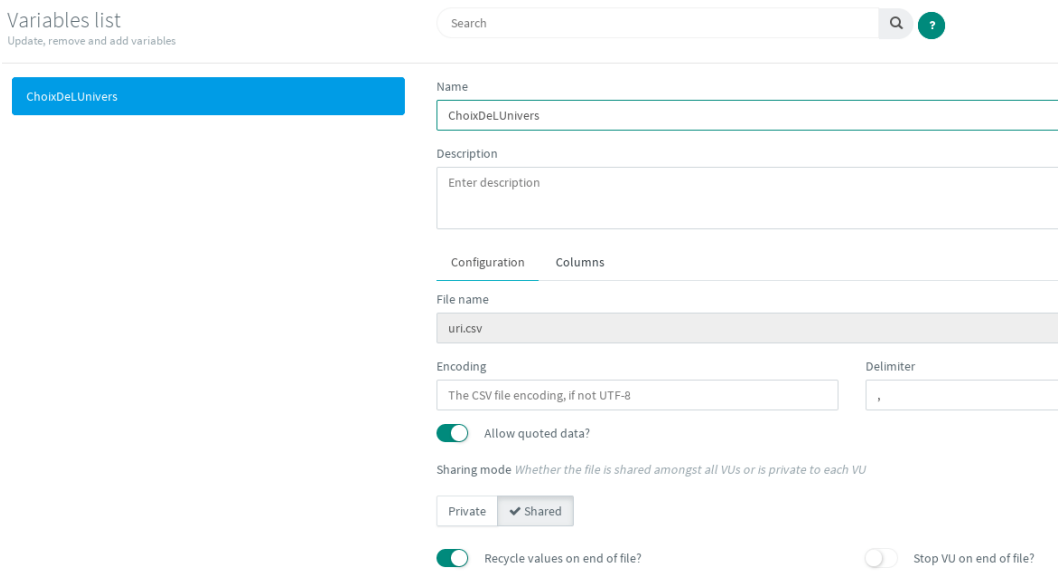
OctoPerf - Modification d'un script

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).



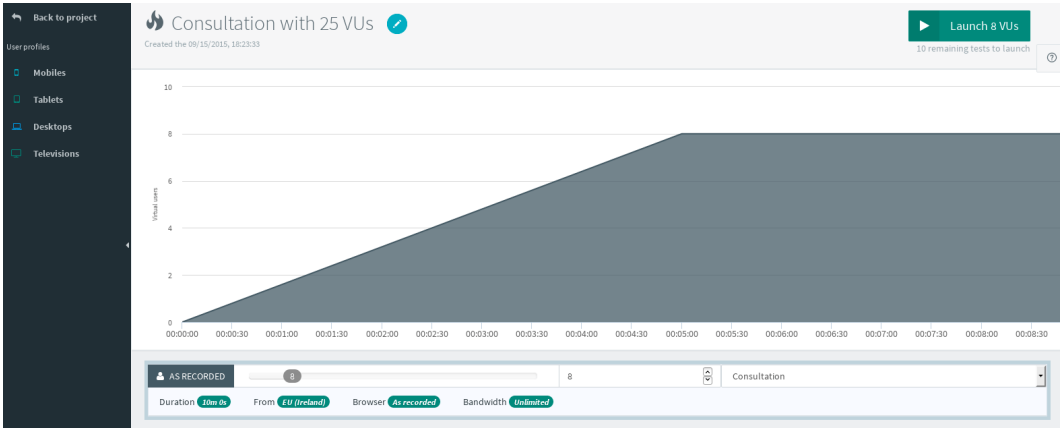
OctoPerf - Règles de corrélation

Définissons et importons notre jeu de données.



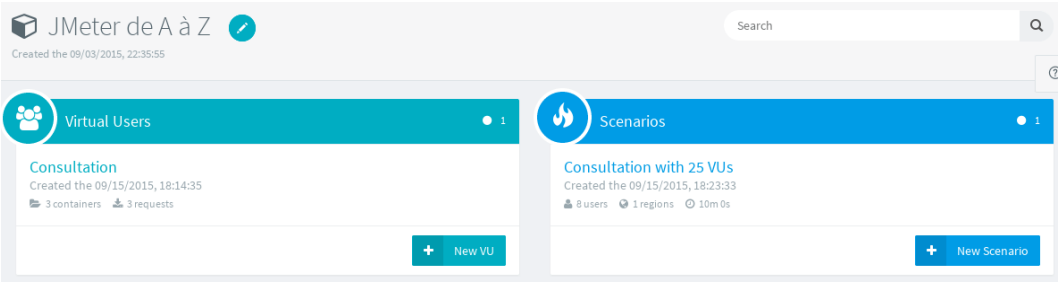
OctoPerf - Importation du jeu de données

Créons notre plan de tir.



OctoPerf - Création de notre plan de test

Notre projet de test de charge est prêt.

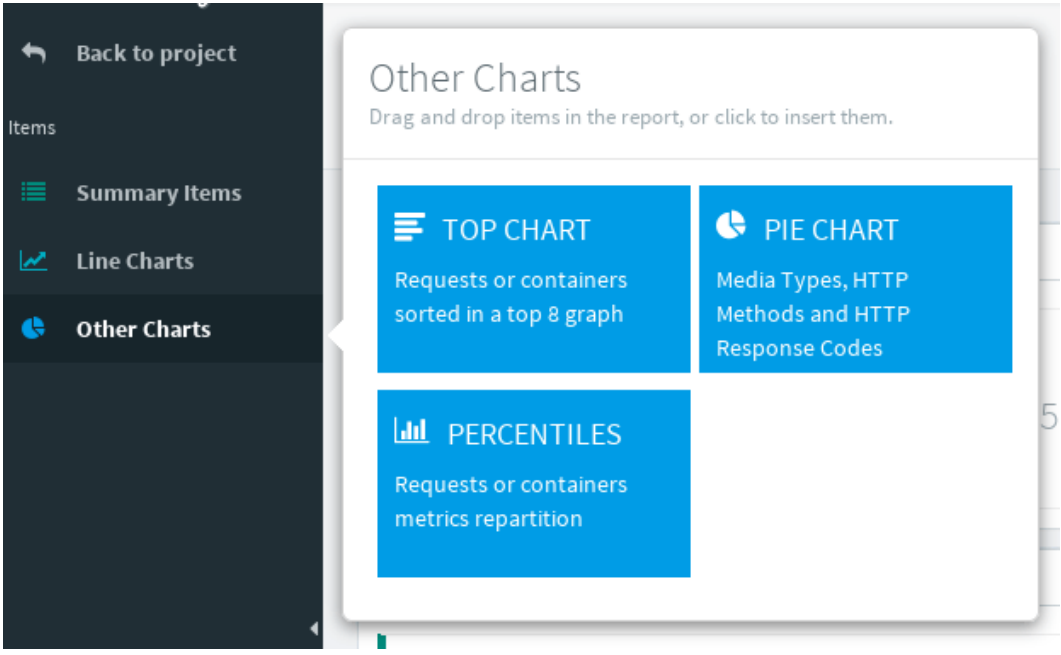


OctoPerf - Projet de test de charge

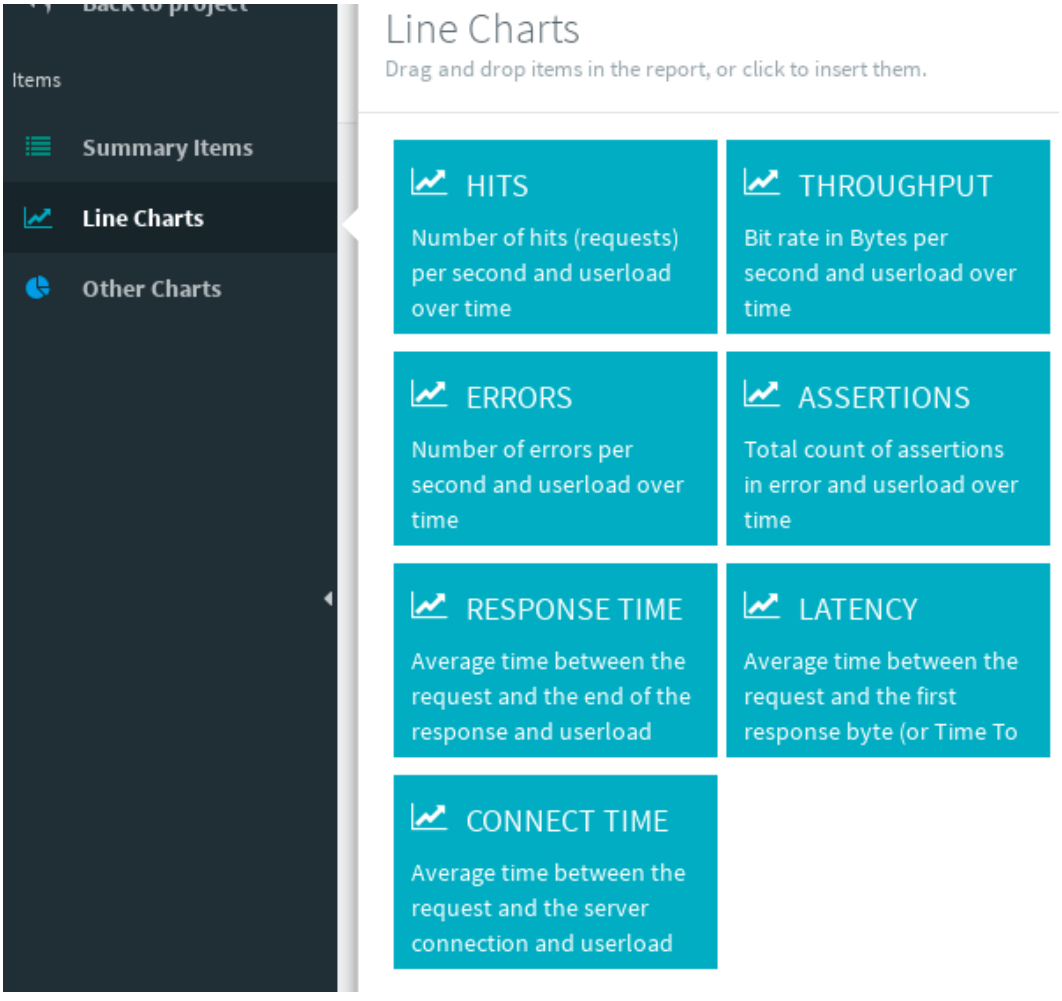
Exécutons notre test pour le suivre en temps réel.

OctoPerf - Suivi d'un test

De nombreux graphiques sont disponibles.



OctoPerf - Rapport et graphiques



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[18] 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.

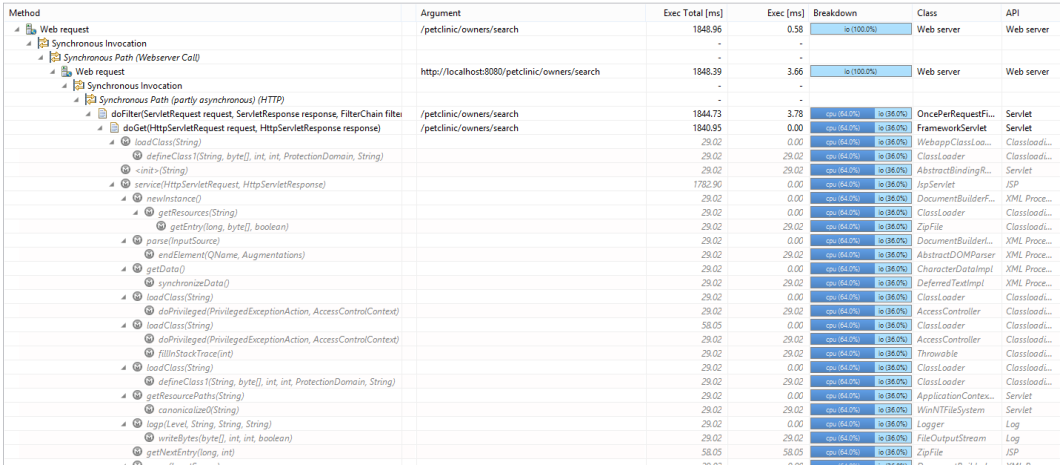
Il existe une solution répondant à ces deux critères.

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

Dynatrace APM

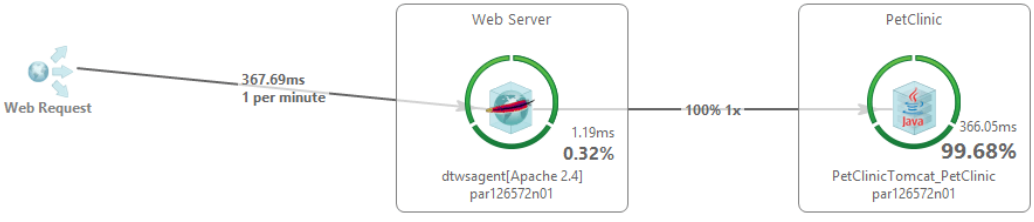
Cette solution est l'APM Dynatrace[19].

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.



Dynatrace PurePath

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



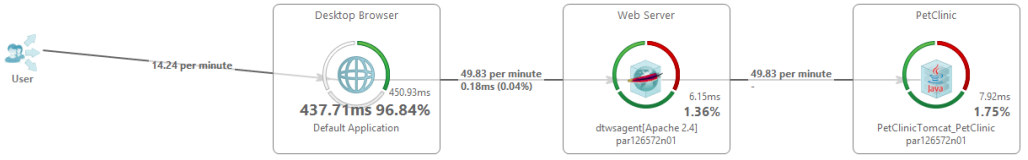
Dynatrace Transaction Flow

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).



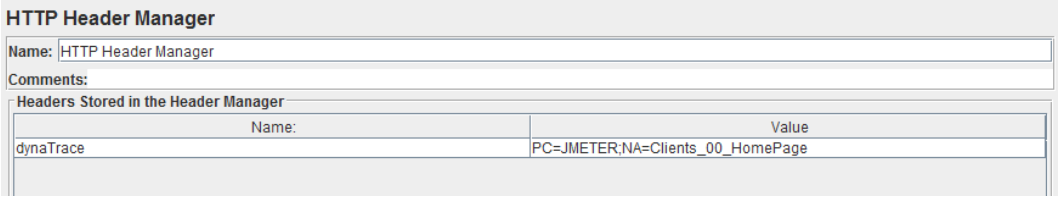
Dynatrace Transaction Flox multi technologies

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.

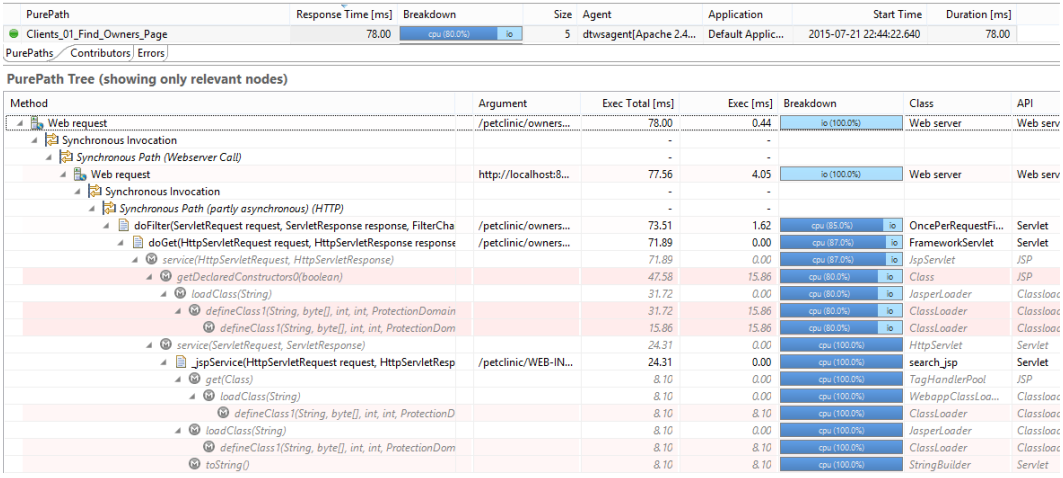


Dynatrace entête HTTP spécifique

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

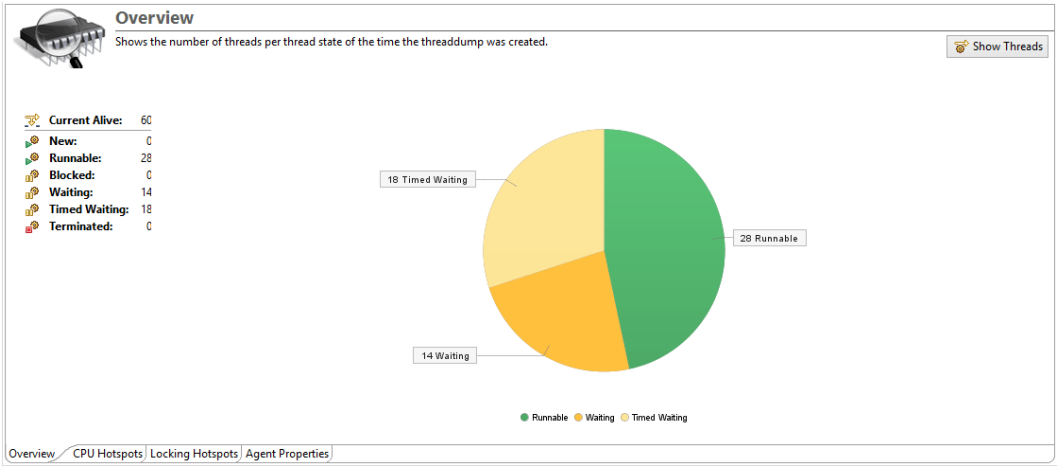


Dynatrace PurePath

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

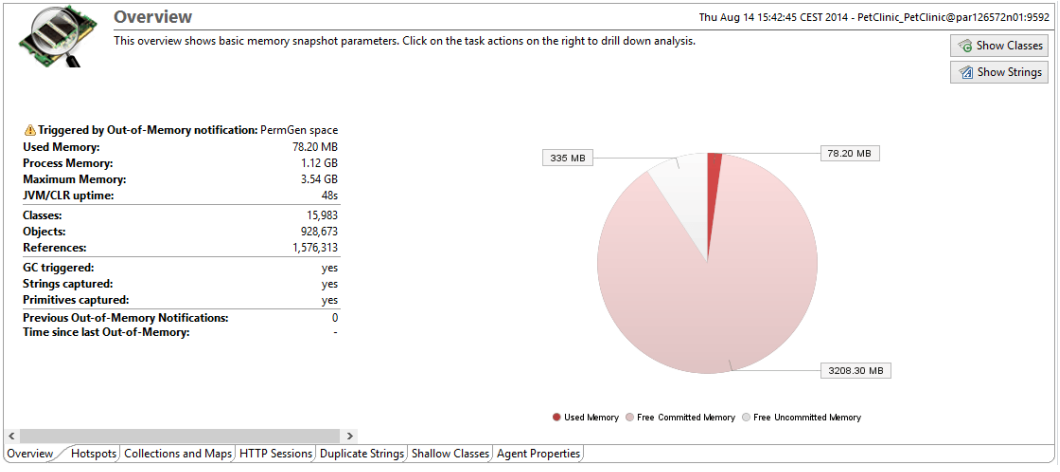
Ci-dessous quelques-unes de ses possibilités (non exhaustives) pour le diagnostic.

Réaliser des threads dump.



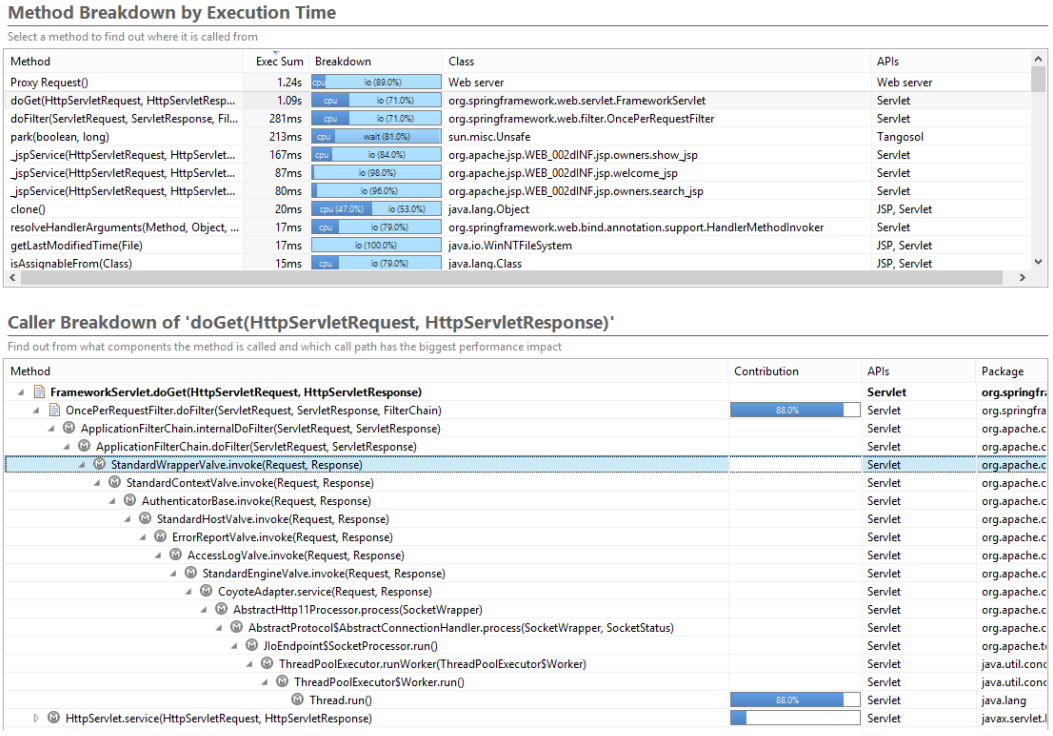
Theads Dump

Réaliser des dumps mémoire.



Memory Dump

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

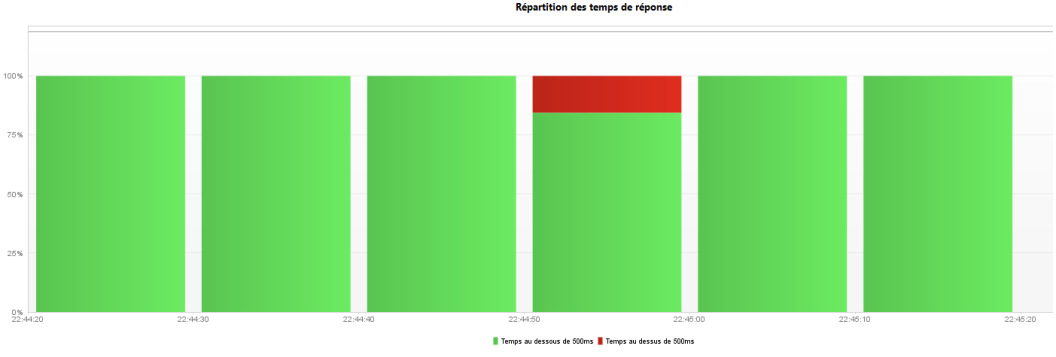


Dynatrace Method Hotspots

Avoir la répartition des temps de réponse.

Dynatrace Response Time Hotspots

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



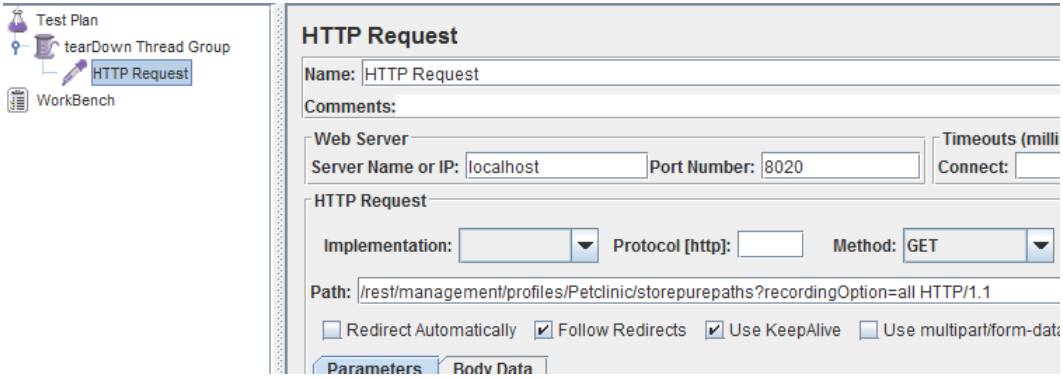
Dynatrace Custom Dashboard

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 :

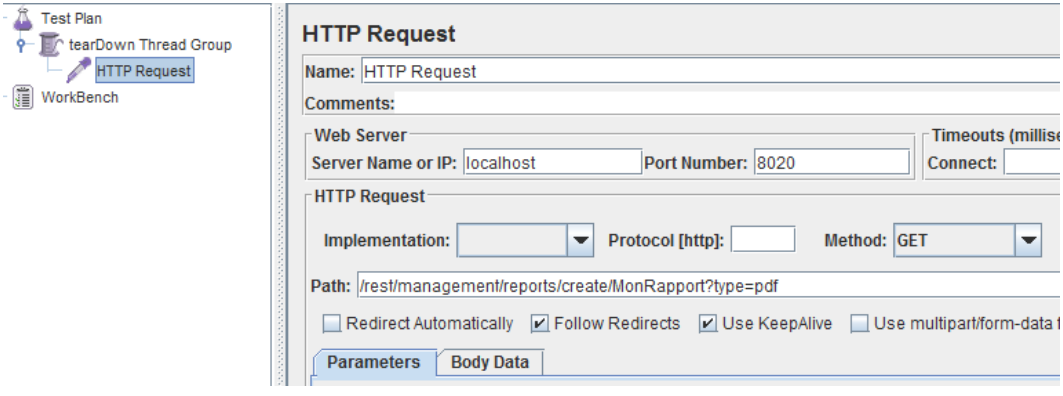
- de comparer plusieurs tirs
- d'analyser les résultats plus tard
- d'envoyer les informations à une autre équipe (par exemple les développeurs pour qu'ils puissent approfondir l'analyse).

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



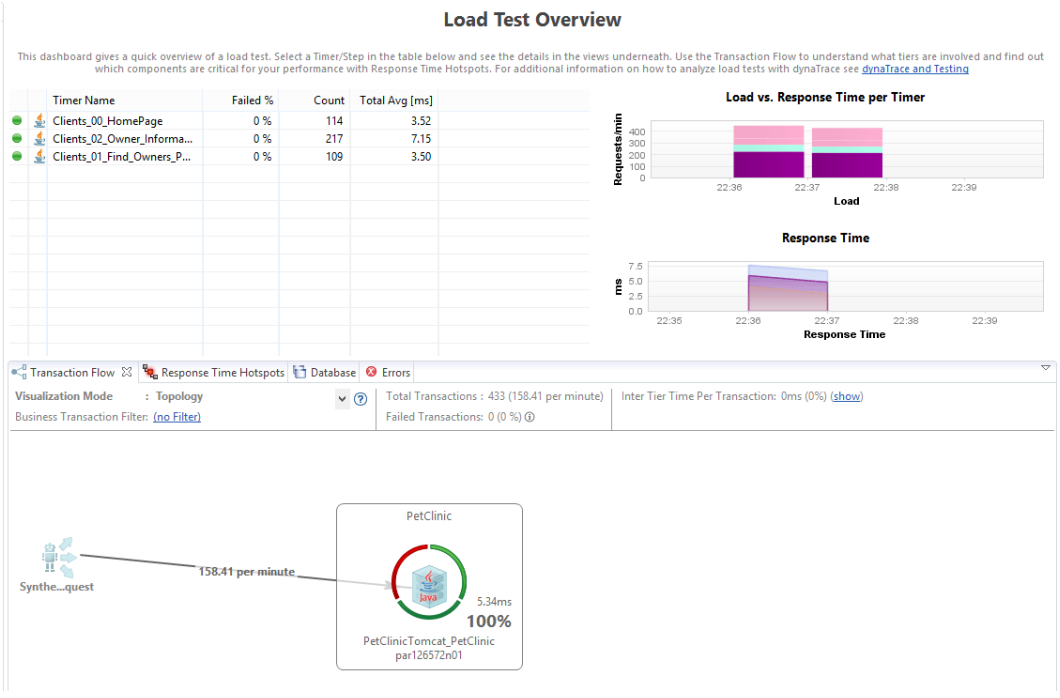
Dynatrace REST API

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



Dynatrace REST API

Pour les impatients, il existe un tableau de bord Dynatrace dédié aux tests de charge.



Dynatrace Load Test Overview

Nous venons de voir un aperçu des possibilités de Dynatrace qui montrent son ROI en termes d'analyse lors d'une campagne de tests de charge.

Loadosophia.org

Loadosophia.org[20] est un service en mode SAAS permettant de stocker et d'analyser le résultat de tests réalisés avec JMeter (à noter qu'Apache Benchmark(ab) et Yandex Tank sont également supportés).

Loadsophia est un service de Blazemeter.

Une fois nos résultats de tests uploadés, un grand nombre de graphiques est généré afin de faciliter l'analyse des résultats.

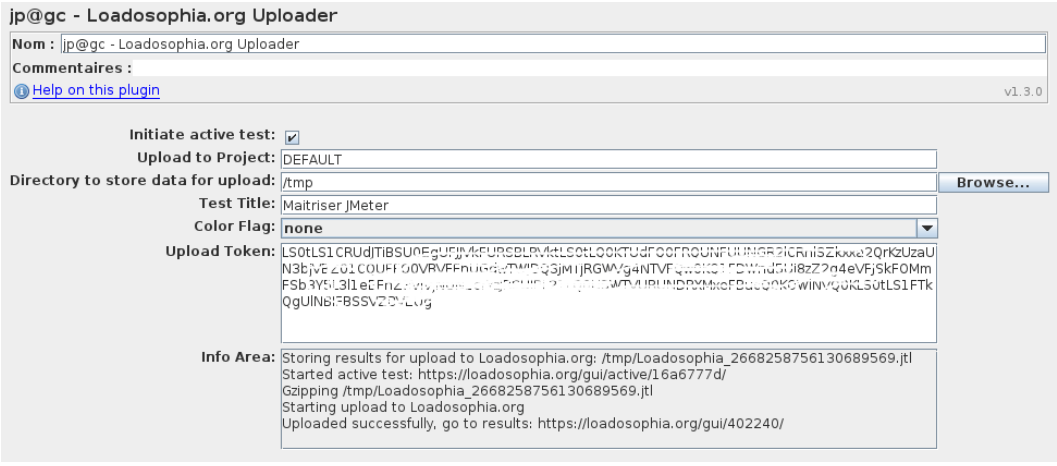
Pour uploader nos résultats, deux moyens existent :

20. https://loadosophia.org/

- le faire par l'intermédiaire du site Web
- utiliser le récepteur **jp@gc - Loadosophia.org Uploader** livré avec JMeter-Plugins

Nous allons utiliser la deuxième solution.

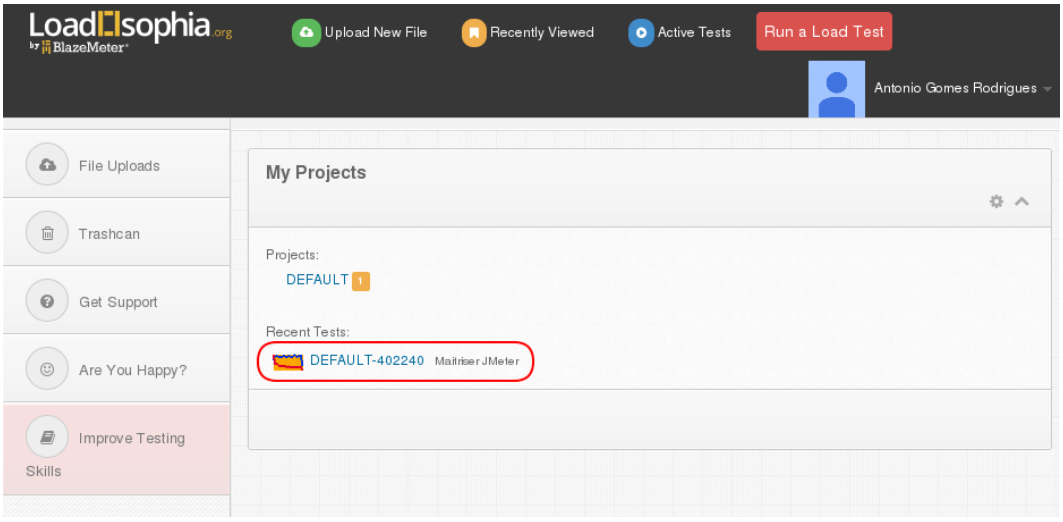
Pour cela il suffit d'ajouter **jp@gc - Loadosophia.org Uploader** à notre plan de test et de le configurer.



jp@gc - Loadosophia.org Uploader

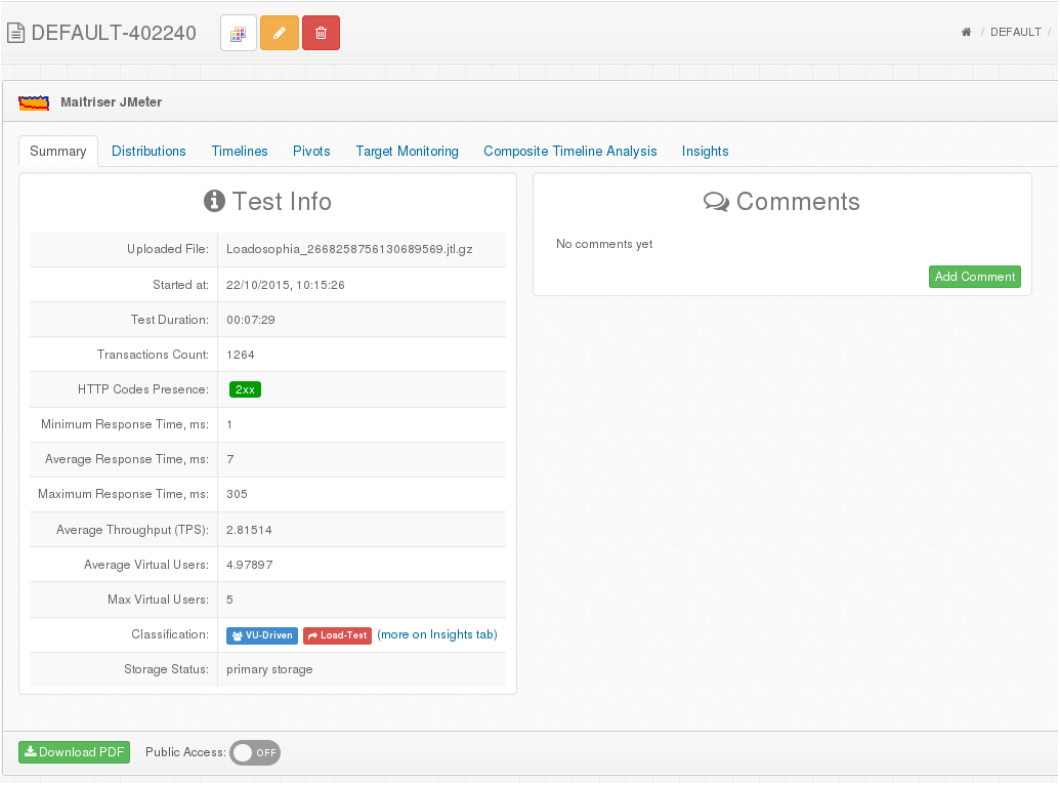
Pour avoir son Upload Token, il suffit de se connecter sur Loadosophia.org et de le récupérer dans la partie Settings de son compte.

Exécutons notre test pour que l'upload des résultats se fasse.



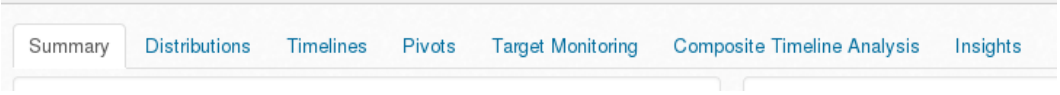
Nos résultats uploadés dans Loadosophia.org

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

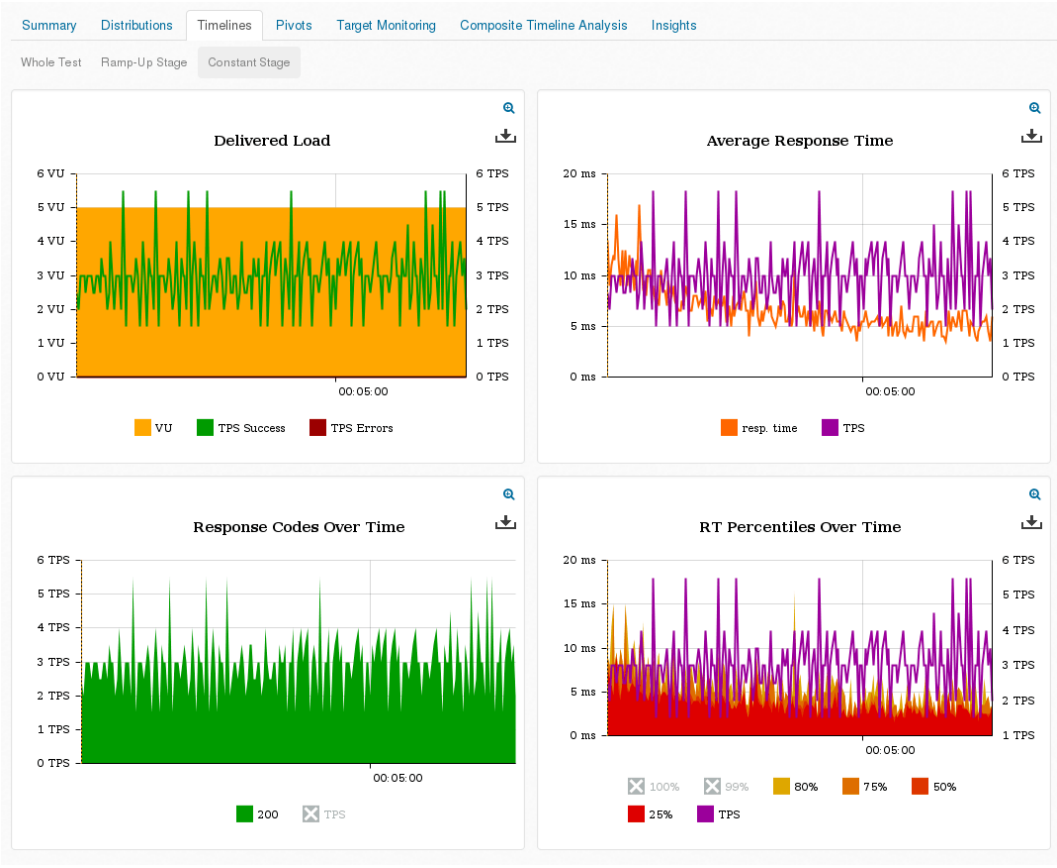


Résumé de notre test

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

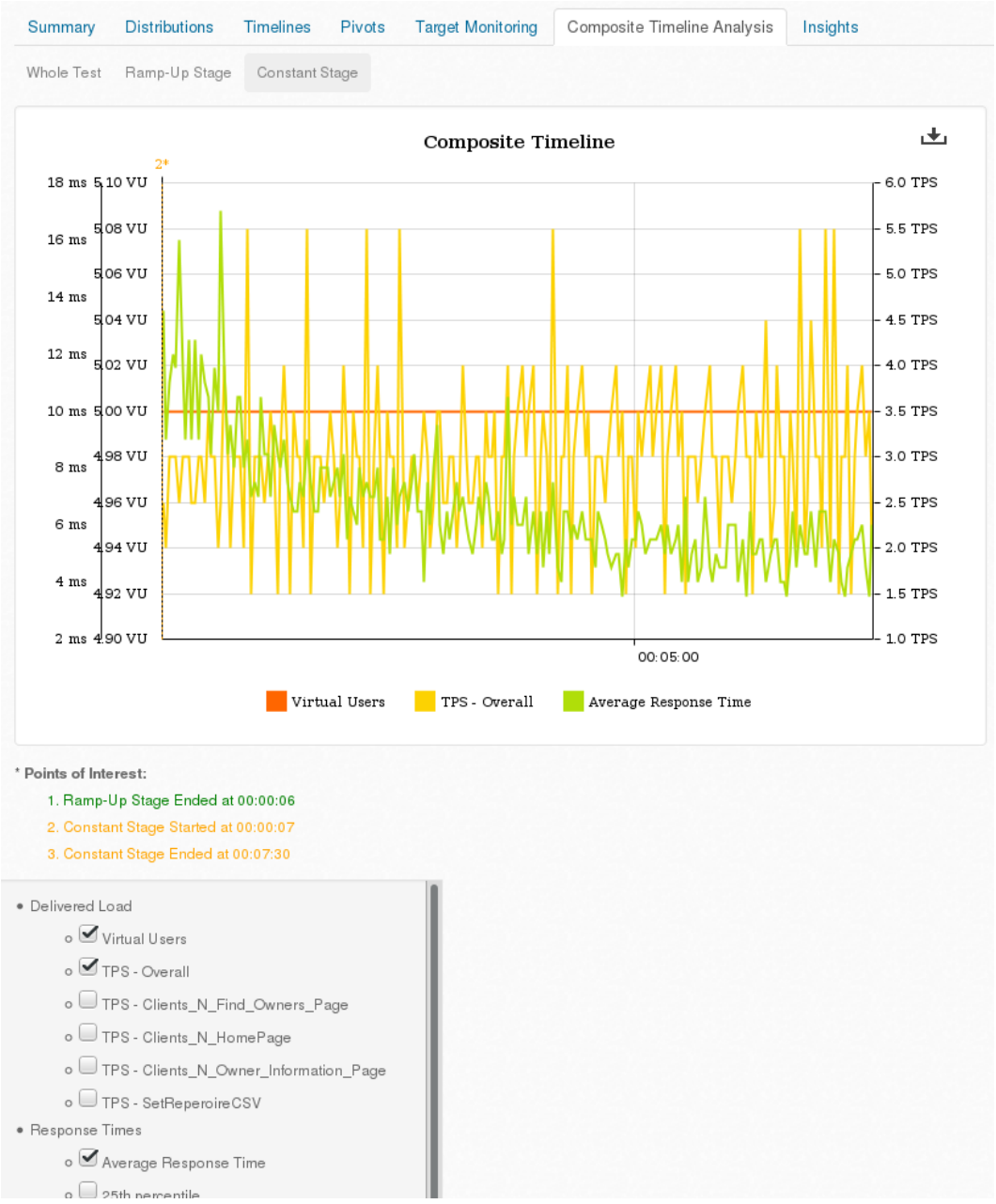


Type de graphiques possibles



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

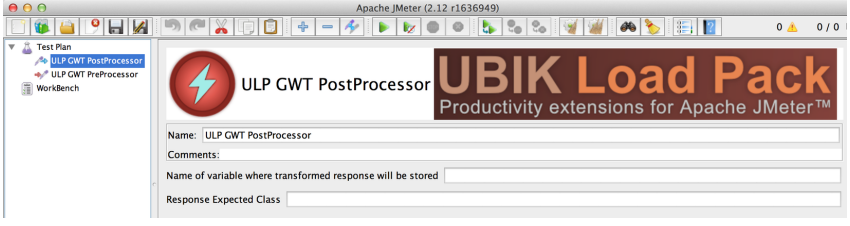
Malgré le nombre élevé de protocoles supportés par Apache JMeter, il reste tout de même plusieurs protocoles non supportés nativement.

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

UbikLoadPack

UbikLoadPack[21] est une solution d' Ubik-Ingénierie[22] offrant des plugins pour les protocoles suivants :

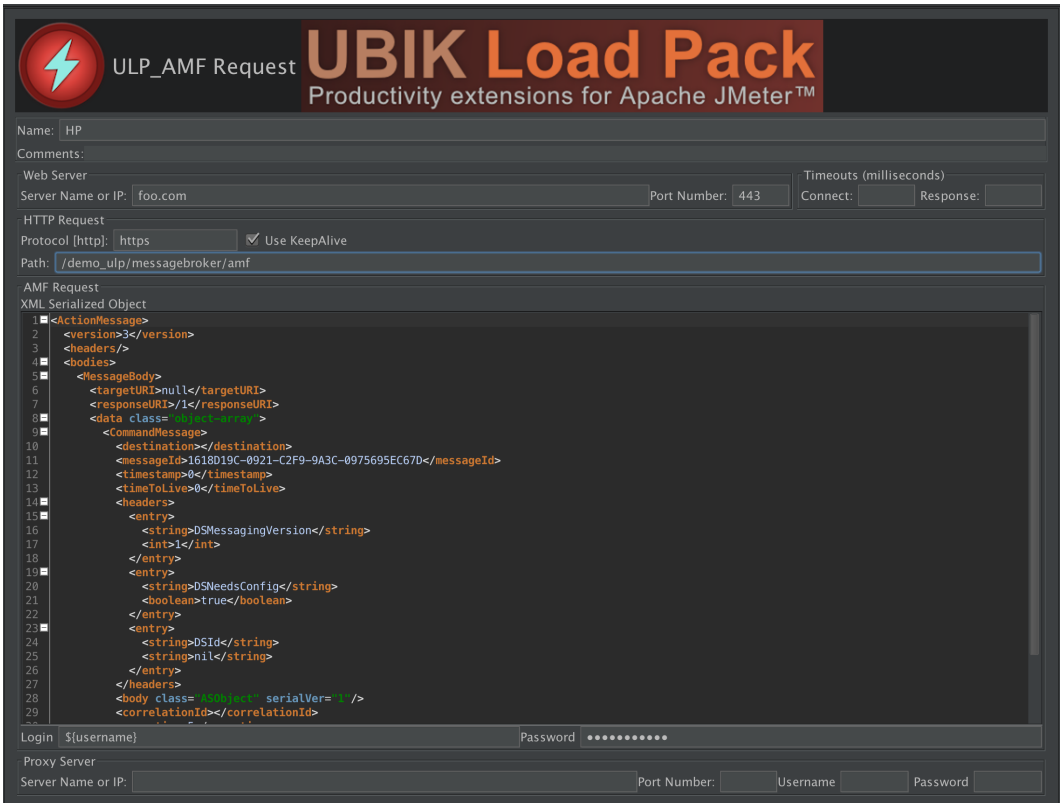
- GWT-RPC du Framework [Google Web Toolkit] (http ://www.gwtproject.org) versions 1.5 à 2.8 (au moment de l'édition de ce livre)



GWTRPC

- FLEX/AMF gérant Adobe Flex et Apache Flex

21. https://ubikloadpack.com/
22. https://www.ubik-ingenierie.com



FLEX/AMF

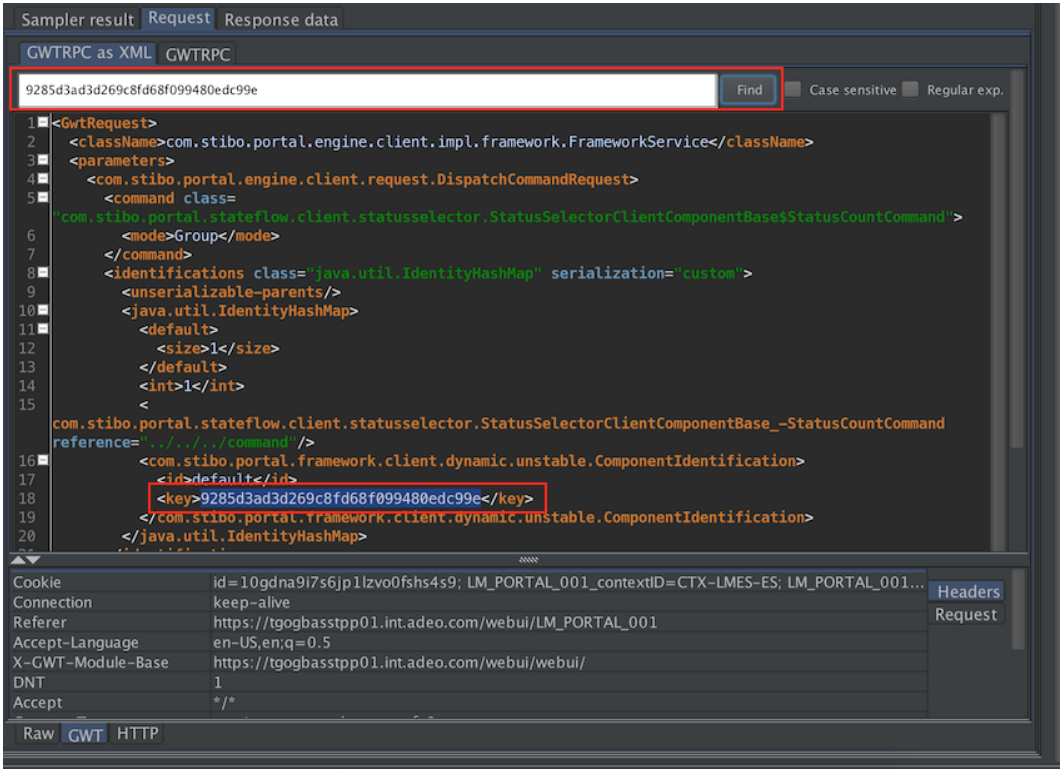
- Java Serialization permettant de simuler des applets ou des applications utilisant Spring Remoting

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 :

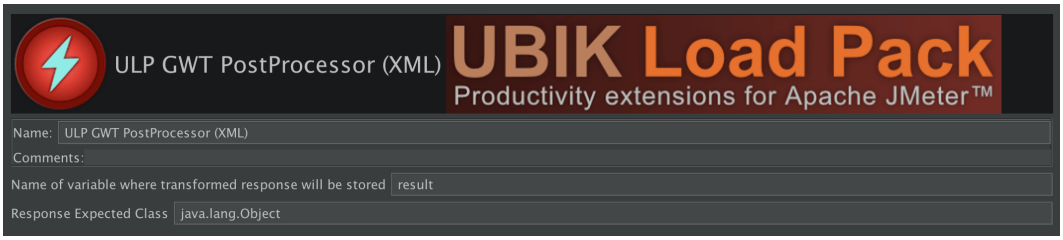
- Avec **Enregistreur script de test HTTP(S)**, vous enregistrez votre navigation sur l'application et créez en quelques minutes le test JMeter.
- À l'aide de samplers dédiés ou de préprocesseurs, le plugin transforme les requêtes « illisibles » du protocole en XML que vous pouvez ainsi facilement

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



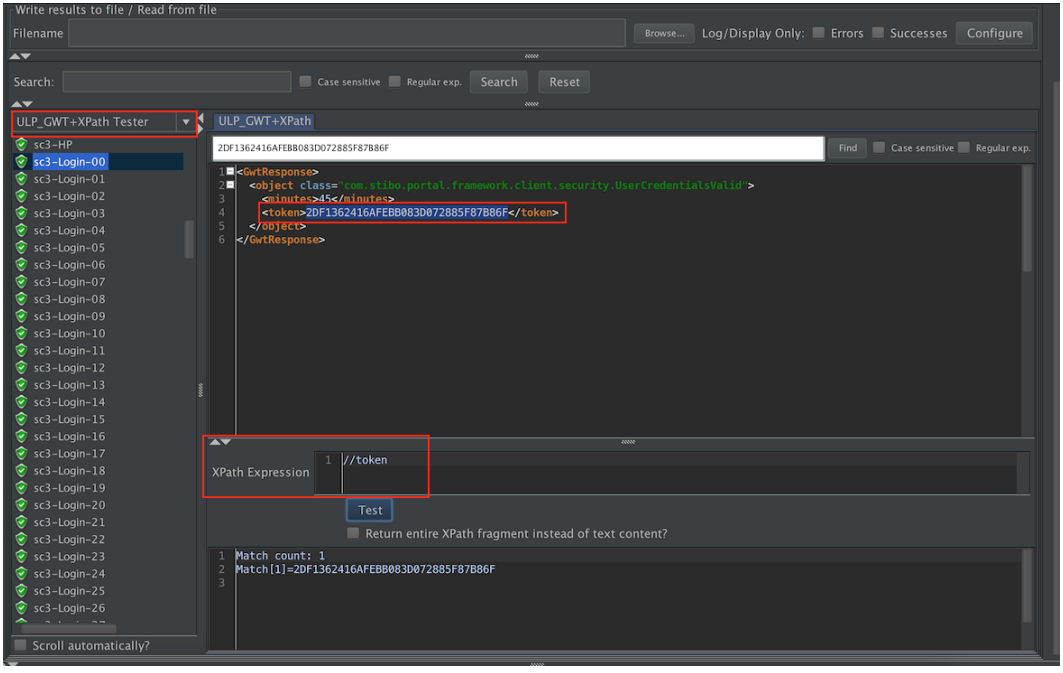
Requête GWT représentée en XML

- 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 Renderer/Visualiseur 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

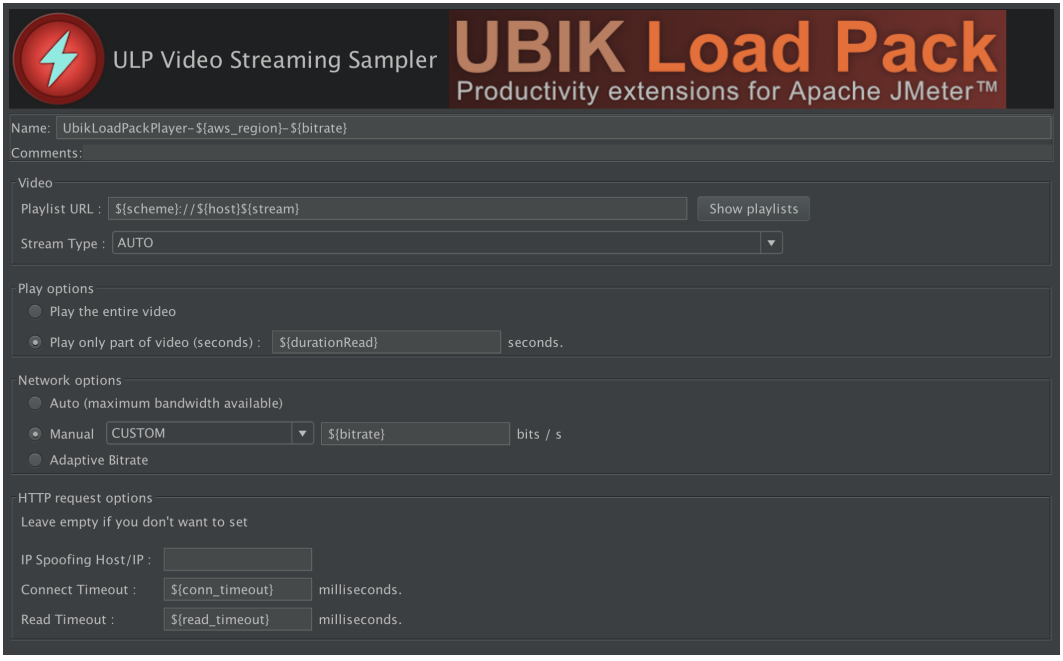
Une fois votre script prêt, vous utiliserez alors la démarche du chapitre 3 pour variabiliser votre script, vérifier les réponses et exécuter votre tir comme expliqué dans le chapitre 8.

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

charge la correlation 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)
  - Format [Mpeg-Dash] (https ://en.wikipedia.org/wiki/Dynamic_Adaptive_-Streaming_over_HTTP)
  - Format Microsoft Smooth Streaming
  - Format Adobe HDS



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 quelque 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 video
- 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[23]. 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

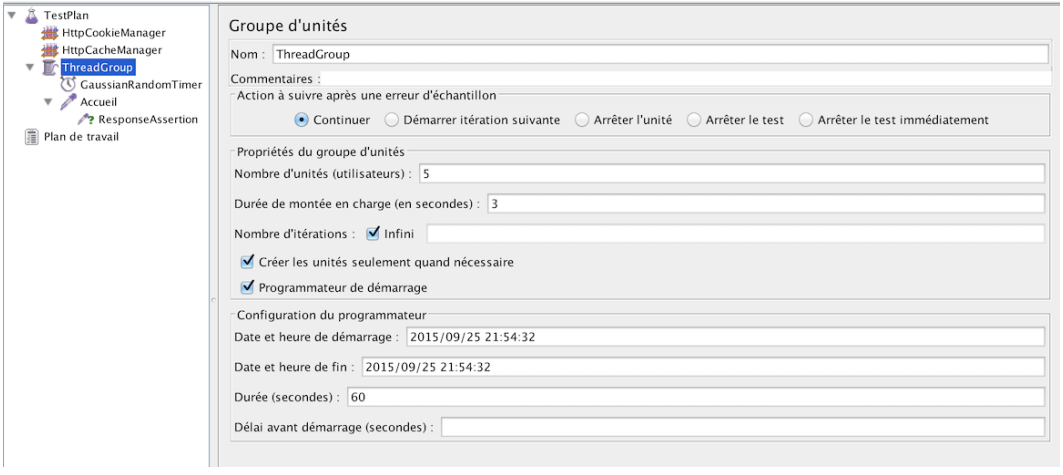
5. Chaque unité navigue sur la page d'accueil de JMeter (http ://jmeter.apache.org), la requête est nommée Accueil
6. Nous vérifions la présence du texte "Further Information About JMeter" (en bas de la page si possible) dans la réponse principale de l'échantillon, l'option utilisée est "Contient (texte 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.