

Day 5 - Jmeter Plugins and Blazemeter Integration

Pada sesi materi ini, kita akan mempelajari berbagai jenis skenario pengujian lainnya menggunakan Ultimate Thread Group dan integrasi dengan BlazeMeter. Kita akan implementasi bagaimana Ultimate Thread Group dapat mensimulasikan pola beban pengguna secara fleksibel, serta cara menggunakan BlazeMeter untuk menjalankan performance testing di cloud.

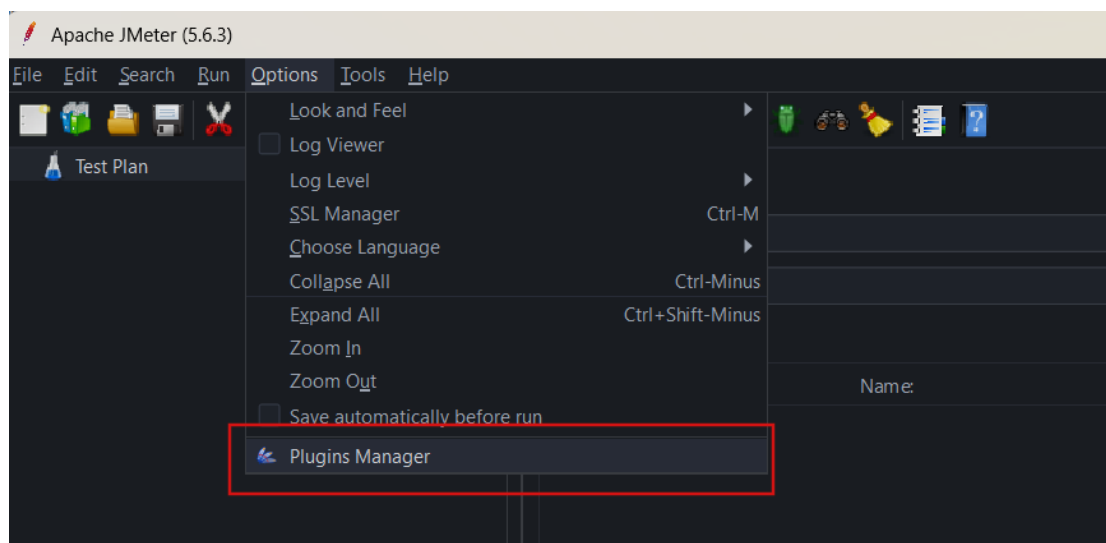
Jmeter Plugin

JMeter Plugin adalah modul tambahan yang dapat diinstal pada Apache JMeter untuk meningkatkan kemampuannya. Plugin ini memungkinkan pengguna untuk menyesuaikan dan memperluas fungsionalitas JMeter, sehingga dapat memenuhi kebutuhan pengujian yang lebih spesifik dan kompleks. Dengan menggunakan plugin, pengguna dapat menambahkan fitur baru, alat analisis, dan berbagai jenis pengujian yang tidak tersedia dalam instalasi JMeter standar.

Menginstal Plugins Manager

Langkah-langkah untuk instalasi Plugins Manager adalah:

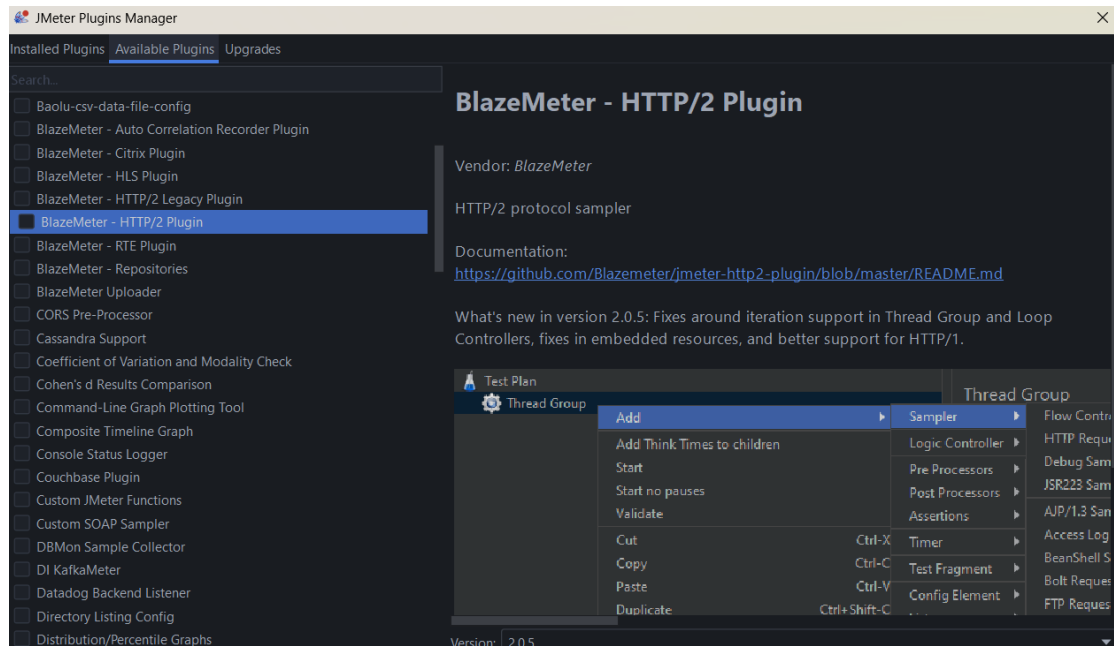
1. Download file JAR plugin manager [disini](#)
2. Simpan file tersebut di folder `JMETER_HOME/lib/ext` agar JMeter dapat mengenali elemen baru ini.
3. Kemudian restart Jmeter, dan kita bisa melihat elemen baru ini



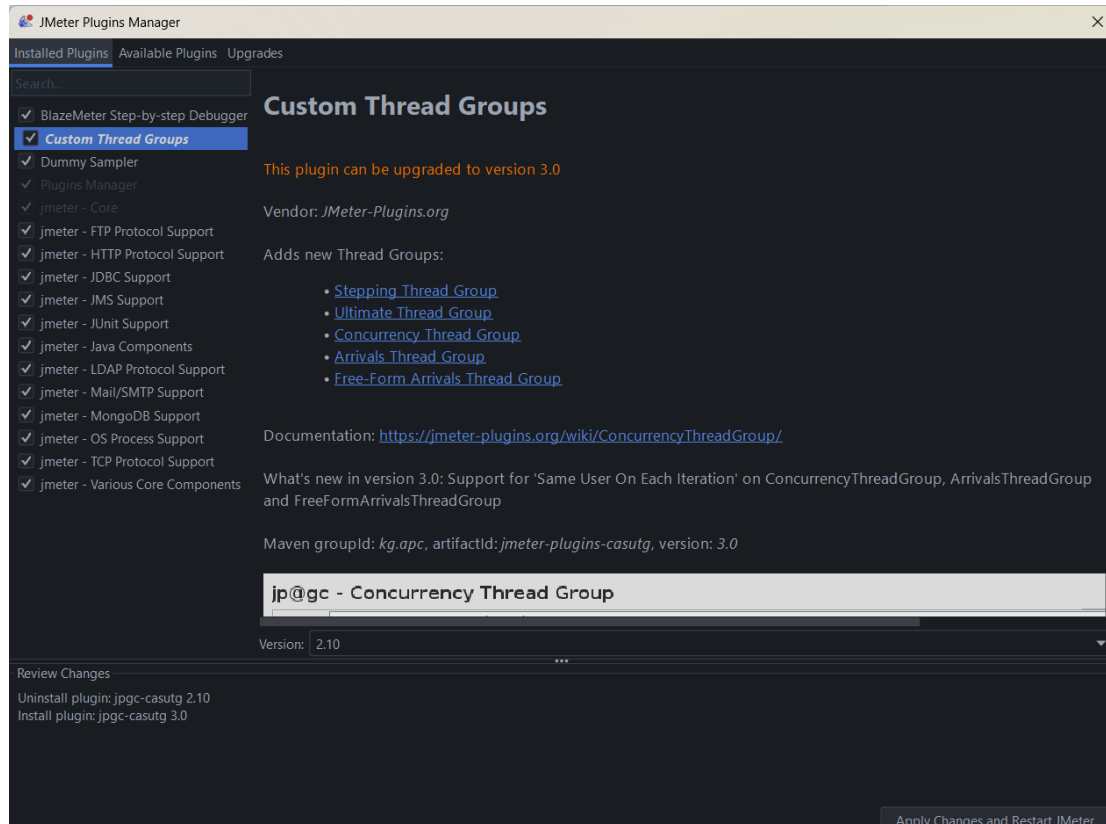
Instalasi/Hapus Plugin Secara Manual

Plugins Manager dapat melakukan tugas-tugas berikut:

1. Menginstal plugin baru dari plugin yang tersedia.



2. Menghapus plugin lama dari daftar plugin yang terinstal.



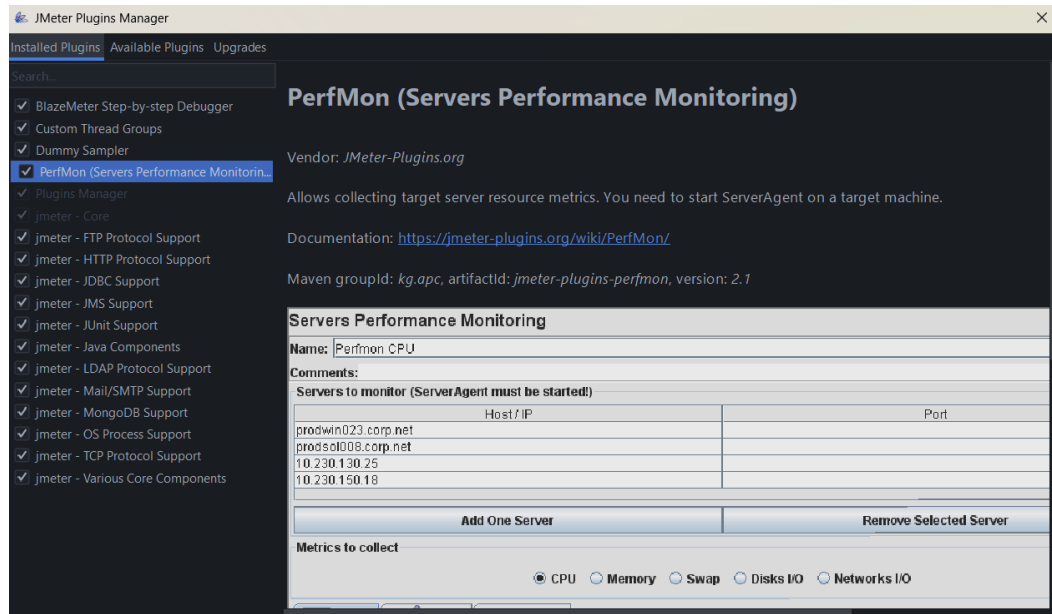
3. Jika ada pembaruan yang tersedia, memperbarui plugin yang ada.

Top 5 JMeter Plugins

Berikut ini adalah 5 plugin yang paling umum digunakan di JMeter:

1. PerfMon Servers Performance Monitoring

Plugin ini extends JMeter dengan listener PerfMon Servers Performance Monitoring. Listener ini memungkinkan kita untuk memantau CPU, Memori, Swap, Disk I/O, dan Networks I/O dari server yang dimuat.



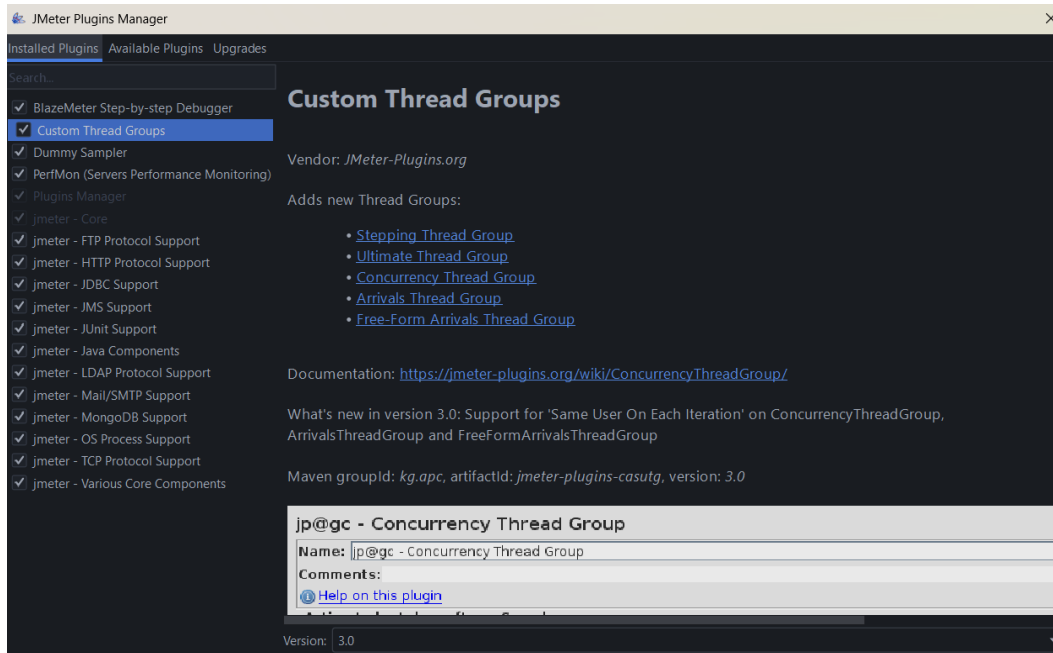
Untuk menggunakannya, klik: **Test plan -> Add -> Listener -> jp@gc — PerfMon Metrics Collector**

2. Custom Thread Groups

Plugin Custom Thread Groups akan menambahkan lima jenis grup thread:

- Stepping Thread Group
- Ultimate Thread Group
- Concurrency Thread Group
- Arrivals Thread Group
- Free-Form Arrivals Thread Group

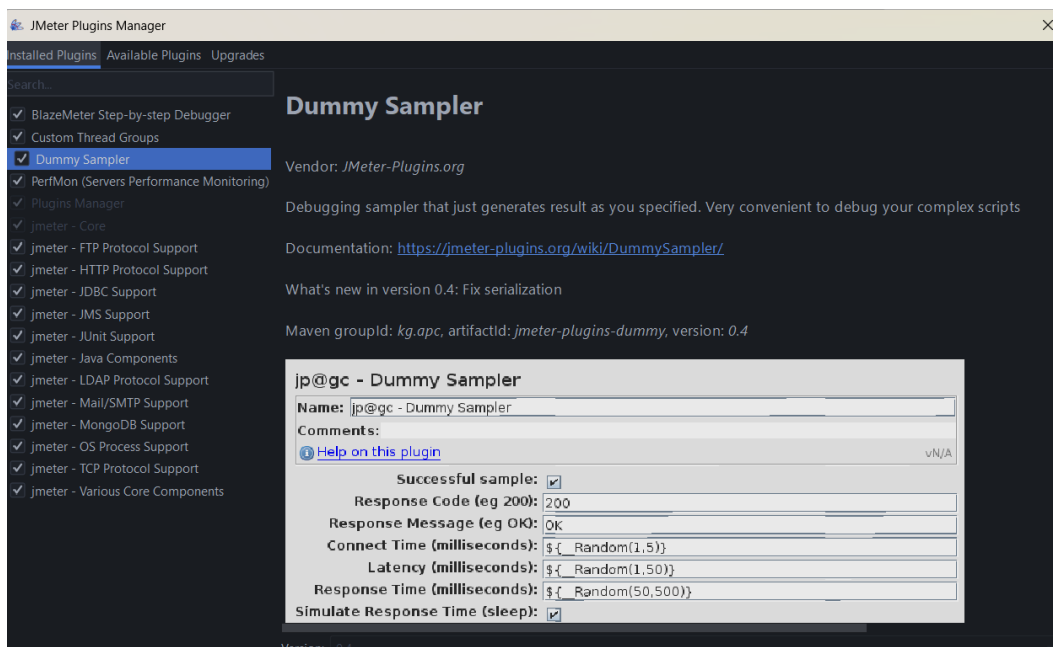
Kelima grup thread ini membuka kemungkinan besar untuk membuat schedules yang diperlukan untuk pengujian. Untuk menggunakannya, klik: **Test plan -> Add -> Threads (Users) -> jp@gc — Ultimate Thread Group**



3. Dummy Sampler

Dummy Sampler meniru kerja requests dan response tanpa benar-benar menjalankan request. Data request dan response didefinisikan di sampler's fields. Ini adalah cara yang sangat mudah untuk melakukan debug pada post-processor dan extractor.

Untuk menggunakannya, klik: **Thread Group -> Add -> Sampler -> jp@gc — Dummy Sampler**



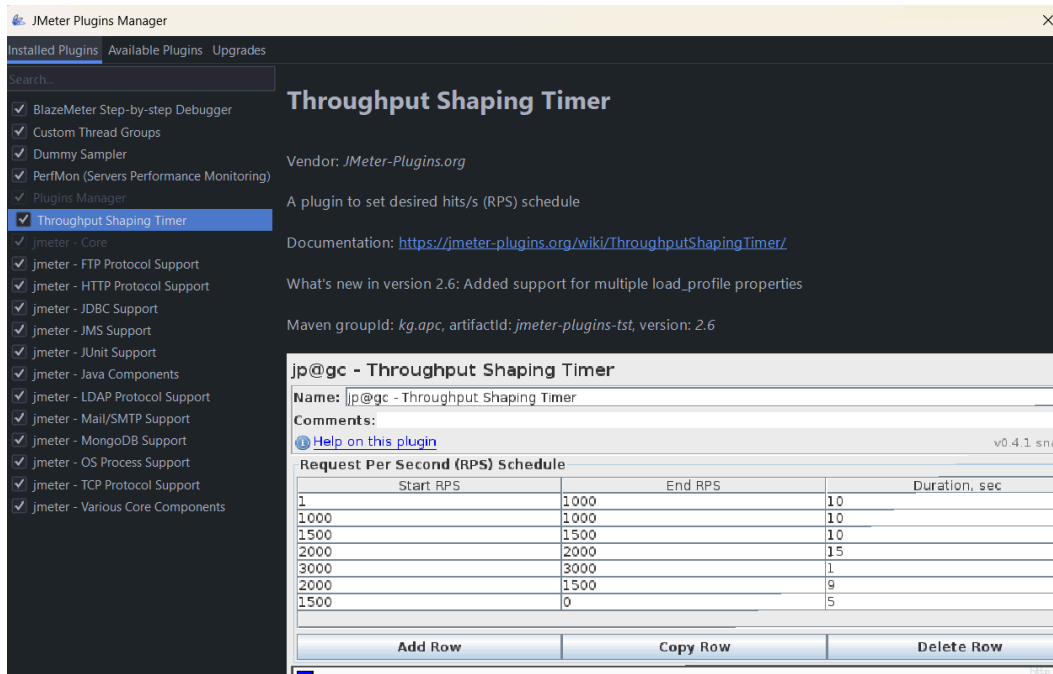
4. Throughput Shaping Timer

Plugin ini menambahkan fungsi berikut ke JMeter: **Throughput Shaping Timer**, **Special Property Processing**, dan **Schedule Feedback Function**. Elemen-elemen

ini memungkinkan kita untuk membatasi throughput pengujian, untuk memastikan kita tidak melebihi nilai throughput yang diperlukan.

Timer ini dirancang untuk mengontrol permintaan per detik ke server selama pengujian.

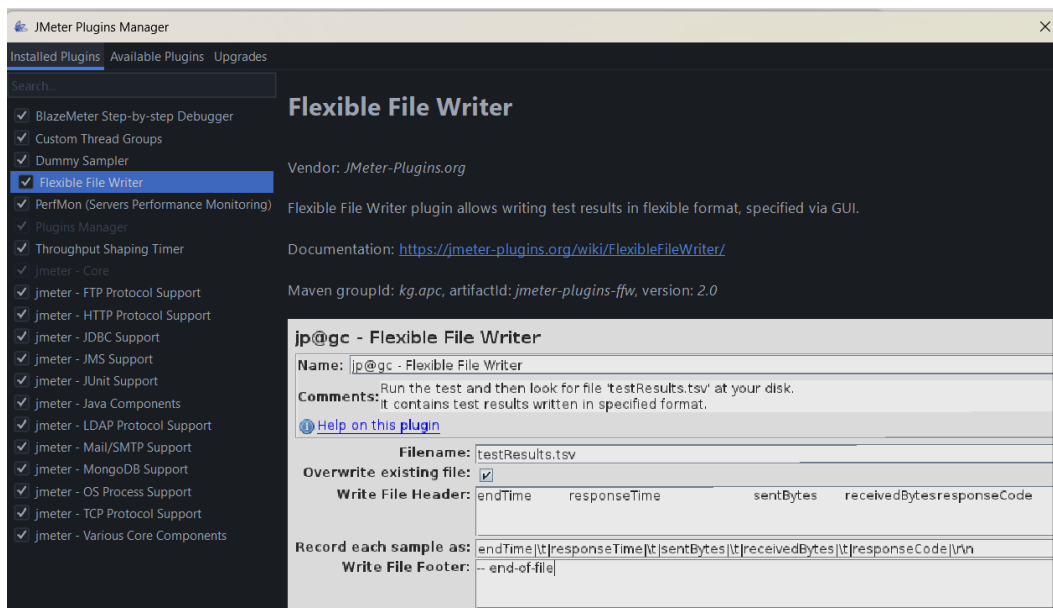
Untuk menggunakannya, klik: **Thread Group -> Add -> Timer -> jp@gc — Throughput Shaping Timer**



5. Flexible File Writer

Plugin ini extends JMeter dengan listener Flexible File Writer. Listener ini dirancang untuk menulis hasil pengujian ke dalam file dalam format yang fleksibel, yang dapat ditentukan melalui GUI JMeter.

Untuk menggunakannya, klik: **Test Plan -> Add -> Listener -> jp@gc — Flexible File Writer**



Advance Performance Testing Scenario with Custom Thread Group

Ketika berbicara tentang Performance Testing, faktor yang sangat penting adalah mencoba untuk mensimulasikan perilaku pengguna yang sebenarnya seakurat mungkin.

Thread group adalah elemen dasar (induk) yang diperlukan dalam setiap rencana pengujian. Ini sebenarnya adalah kumpulan thread yang menjalankan skenario yang sama. Terdapat beberapa thread group yang tersedia yang kita konfigurasi dengan berbagai cara untuk secara akurat mensimulasikan bagaimana pengguna berinteraksi dengan aplikasi, bagaimana beban dipertahankan, dan dalam periode waktu berapa.

Pengaturan ini berasal dari apa yang dibutuhkan klien aplikasi dari pengujian beban, data dari analitik, dan hal-hal pengujian yang masuk akal.

Pembaruan dan plugin JMeter terbaru datang dengan beberapa thread group baru. Semua jenis thread group memiliki opsi umum yaitu '**Action to be taken after a Sampler error**' yang cukup jelas — ini bisa berupa melanjutkan thread, memulai iterasi berikutnya, menghentikan thread saat ini, menghentikan seluruh pengujian, atau memaksa menghentikan pengujian dan semua thread yang sedang berjalan.

Disini kita akan eksplorasi setiap jenis thread group beserta opsi konfigurasi spesifiknya dan mencoba untuk memahami kapan sebaiknya menggunakan masing-masing dari thread group tersebut.

Thread Group (the classic)

Ini adalah thread group paling dasar yang ada. Ini memiliki beberapa pengaturan standar dan dapat disesuaikan untuk memenuhi sebagian besar skenario load testing.

Thread group capabilities and configuration options :

- **Number of threads:** mewakili total jumlah pengguna virtual yang menjalankan *script* pengujian kita per eksekusi.
- **Ramp-up period:** adalah jangka waktu yang dibutuhkan untuk semua pengguna virtual mulai menjalankan *script* mereka, misalnya, jika kita memiliki 30 pengguna dengan periode ramp-up 30 detik, setiap detik, satu pengguna virtual baru akan mulai menjalankan "*user flow*" yang diwakili oleh *script* yang dibuat.
- **Loop count:** adalah jumlah eksekusi untuk *script* kita. Misalnya, jika loop count diatur ke 1 dan kita telah mengatur 20 thread, maka *script* akan dijalankan 20 kali dengan penundaan eksekusi thread yang ditentukan oleh periode ramp-up. Jika loop count diatur ke '*Forever*', maka thread baru akan terus mulai hingga pengujian

dihentikan. Kondisi untuk skenario ini adalah bahwa kita akan memiliki maksimum thread aktif yang sama dengan nilai yang diatur pada '*Number of threads*'.

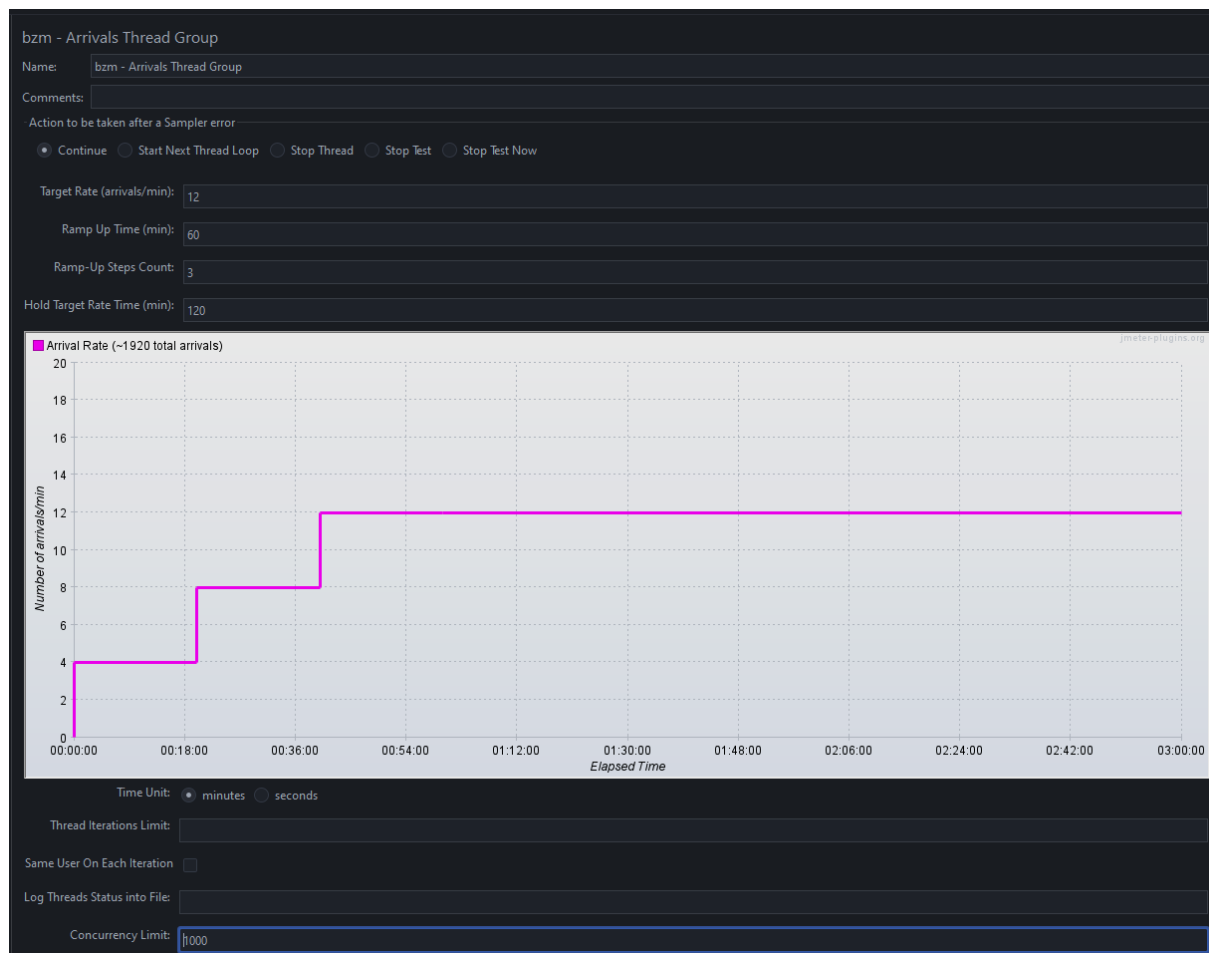
- **Delay Thread creation until needed:** jika opsi ini dicentang, penundaan ramp-up dan penundaan startup dilakukan sebelum data thread dibuat. Jika tidak, semua data yang diperlukan untuk semua thread dibuat sebelum memulai eksekusi aktual di awal pengujian.
- **Scheduler:** kita dapat mengatur waktu mulai dan akhir yang telah ditentukan untuk pengujian kita tetapi juga durasi kustom, untuk situasi di mana kita ingin memberikan beban konstan ke server selama n menit. Perlu dicatat bahwa opsi durasi akan menimpa waktu mulai dan akhir.

Skenario yang mungkin digunakan untuk:

- **Skenario sederhana (load testing)** — di mana kita mengatur jumlah pengguna yang telah ditentukan dan periode ramp-up dan kita mulai menjalankan *script* kita n kali.
- **Skenario Stress Testing** — menentukan batas atas jumlah pengguna bersamaan untuk aplikasi kita sebelum mulai gagal.
- **Skenario Soak Testing** — di mana kita dapat mengatur beban yang diharapkan untuk pengujian kita dan mempertahankan beban tersebut selama periode waktu yang telah ditentukan.

Arrivals Thread Group

Ini adalah custom plugin open source yang diimplementasikan oleh tim dari BlazeMeter. Thread group ini beroperasi dengan jadwal "arrivals" sebagai cara untuk mengekspresikan beban. "Arrival" berarti awal iterasi thread. Ini akan membuat thread baru jika semua thread yang ada sedang sibuk di tengah iterasi.



Thread group capabilities and configuration options:

- **Time Unit:** ini cukup jelas dan dapat diatur ke menit atau detik.
- **Target rate:** di mana kita mendefinisikan jumlah 'kedatangan' baru per unit waktu.
- **Ramp Up Time:** periode waktu yang diperlukan untuk meningkatkan jumlah pengguna virtual yang aktif dalam pengujian.
- **Ramp Up Steps Count:** berkorelasi dengan ramp-up time. Ini adalah jumlah langkah yang digunakan untuk meningkatkan jumlah pengguna virtual dalam pengujian kinerja secara bertahap. Ini menentukan seberapa halus atau kasar transisi dari jumlah pengguna yang lebih sedikit ke jumlah pengguna yang lebih banyak selama periode ramp-up
- **Hold target rate:** durasi di mana kita ingin menjalankan pengujian kita (berapa lama kita ingin terus menambahkan pengguna baru setiap menit).
- **Thread Iterations Limit:** mewakili jumlah loop. Sejauh ini belum ada kasus penggunaan untuk fitur khusus ini, jadi biasanya kita biarkan kosong yang berarti diatur ke 'tak terbatas' hingga batas waktu tercapai.
- **Concurrency Limit:** adalah batas maksimum jumlah pengguna virtual (threads) yang dapat aktif secara bersamaan dalam pengujian kinerja. Ini membantu

mencegah kelebihan beban pada sistem yang diuji dan memastikan pengujian dilakukan dalam batas yang aman.

Skenario yang mungkin digunakan untuk:

- **Skenario Soak Testing** — di mana kita dapat mengatur beban yang diharapkan untuk pengujian kita dan mempertahankan beban tersebut selama periode waktu yang telah ditentukan. Kita dapat melakukan ini dengan mengatur batas konkuren sehingga memastikan bahwa jumlah thread aktif tidak akan melebihi itu.
- **Conditioned Scenarios** di mana klien mungkin mengatakan: *‘Tujuannya adalah untuk melihat bagaimana aplikasi berperilaku jika 50 pengguna baru mengaksesnya setiap menit’*. Ini dapat dimasukkan dalam skenario **scalability testing** dengan syarat bahwa waktu eksekusi untuk sebuah thread lebih dari 1 menit. Jika waktu eksekusi sebuah thread lebih lama dari 1 menit, atau permintaan akan secara bertahap memakan waktu lebih lama untuk merespons, maka jumlah pengguna bersamaan akan meningkat secara bertahap sehingga menghasilkan bentuk **scalability testing**. Untuk environment yang lebih terkontrol untuk scalability testing, kita akan melihat thread group berikutnya:

Free Form Arrivals Thread Group

Ini pada dasarnya sama persis dengan Arrivals thread group, minus elemen yang kurang digunakan seperti **ramp-up time** dan **Ramp Up Steps Count**.

Thread group capabilities and configuration options:

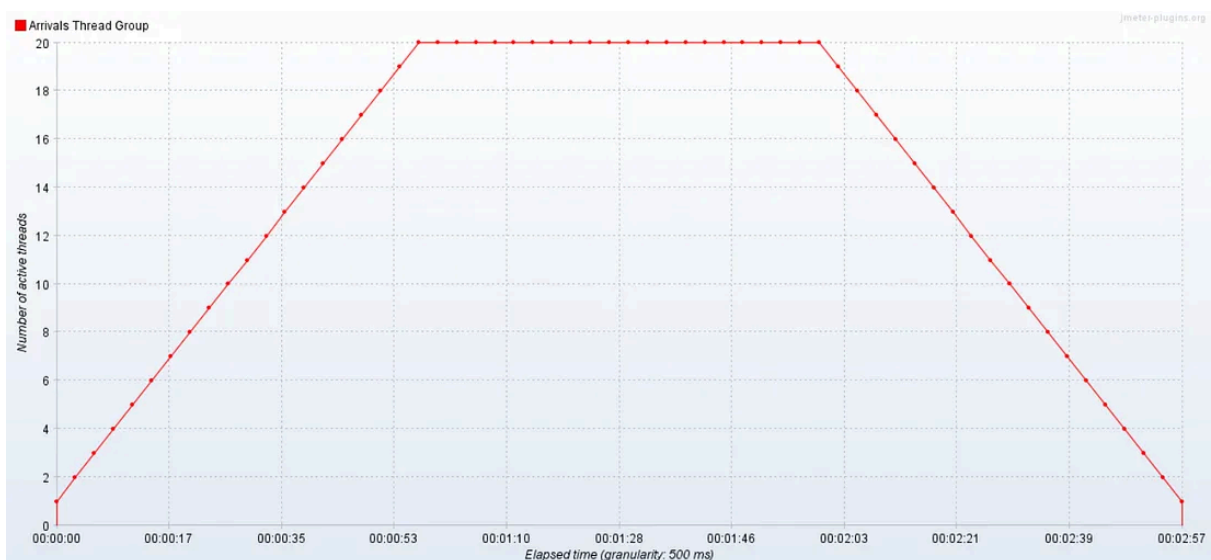
- **Time Unit:** ini cukup jelas dan dapat diatur ke menit atau detik.
- **Start Value dan End Value:** ini setara dengan ‘target rate’ dari **Concurrency Thread Group**. Jika nilai mulai sama dengan nilai akhir dan misalnya 20, dan kita mengatur durasi 2 menit, ini berarti bahwa setiap menit, kita mendapatkan 20 kedatangan baru dengan ramp-up konstan 3 detik antara thread. Jika waktu mulai dan akhir berbeda, saya belum menemukan dengan tepat bagaimana ramping dilakukan tetapi dalam hal apapun, itu tidak terlalu seragam.
- **Duration:** sama dengan **hold target rate** dari thread group sebelumnya.
- **Concurrency Limit:** adalah batas maksimum jumlah pengguna virtual (threads) yang dapat aktif secara bersamaan dalam pengujian kinerja. Ini membantu mencegah kelebihan beban pada sistem yang diuji dan memastikan pengujian dilakukan dalam batas yang aman.
- **Threads Schedule Table:** di mana kita dapat menambahkan beberapa baris, masing-masing dieksekusi secara berurutan setelah durasi baris sebelumnya berakhir.

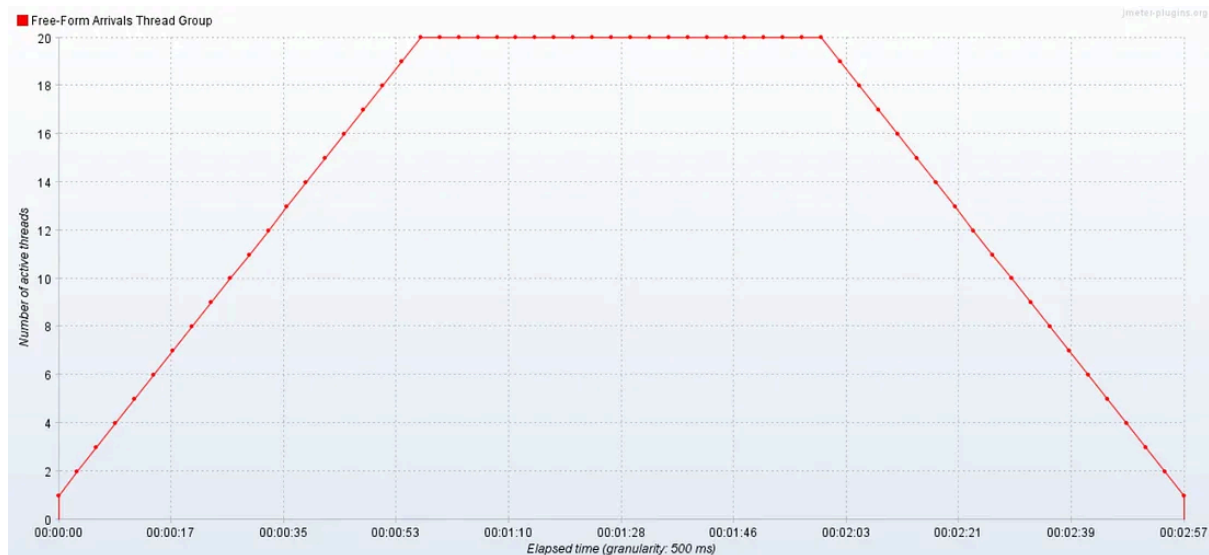


Dalam contoh di atas, eksekusi kedatangan baru akan sebagai berikut:

- Menit 1: kita akan memiliki 10 kedatangan baru (satu kedatangan baru setiap 6 detik).
- Menit 2: 10 kedatangan baru lainnya (satu kedatangan setiap 6 detik). Ini karena durasi untuk baris pertama diatur ke 2 menit.
- Menit 3: 20 kedatangan baru (satu setiap 3 detik).
- Menit 4: 10 kedatangan baru (satu setiap 6 detik).

Ini berarti bahwa jika kita hanya memiliki satu baris di bagian '**Threads Schedule**', thread group ini berperilaku persis seperti **Arrivals Thread Group**.





Skenario yang mungkin digunakan untuk:

- Segala sesuatu yang akan kita gunakan untuk Arrivals Thread Group.
- **A more controlled form of scalability testing** di mana kita selalu memiliki visibilitas tentang berapa banyak kedatangan baru yang kita miliki setiap menit.
- **Spike testing scenarios** — di mana kita dapat tiba-tiba meningkatkan jumlah pengguna per menit, kemudian menurunkannya lagi dan melihat bagaimana aplikasi berperilaku. Dengan menggunakan teknik ini, kita juga dapat menguji kemampuan pemulihan dari sistem tertentu.

Concurrency Thread Group

Jika tujuan kita adalah untuk mengontrol jumlah pengguna bersamaan di aplikasi kita selama periode waktu tertentu, maka ini adalah jenis thread group yang paling tepat untuk membantu kita melakukannya.

Thread group capabilities and configuration options:

- **Time Unit:** ini cukup jelas dan dapat diatur ke menit atau detik.
- **Target Concurrency:** jumlah pengguna bersamaan yang harus dipertahankan setelah ramp-up.
- **Ramp Up Time:** periode waktu yang diperlukan untuk meningkatkan jumlah pengguna virtual yang aktif dalam pengujian.
- **Ramp Up Steps Count:** berkorelasi dengan ramp-up time. Ini adalah jumlah langkah yang digunakan untuk meningkatkan jumlah pengguna virtual dalam pengujian kinerja secara bertahap. Ini menentukan seberapa halus atau kasar transisi dari jumlah pengguna yang lebih sedikit ke jumlah pengguna yang lebih banyak selama periode ramp-up

- **Hold target rate:** durasi untuk mempertahankan target concurrency sebelum mulai secara bertahap mematikan semua thread.

Skenario yang mungkin digunakan untuk:

- **Soak testing scenarios** — di mana kita dapat mengatur beban yang diharapkan untuk pengujian kita dan mempertahankan beban tersebut selama periode waktu yang telah ditentukan.

Perlu dicatat bahwa semua yang dilakukan thread group ini, juga dapat dilakukan menggunakan Thread Group dasar jika kita mengatur jumlah thread yang konstan, mengatur loop count ke infinity dan menjadwalkan durasi.

Jadi mengapa menggunakan Concurrency Thread Group?

- Lebih mudah untuk mengkonfigurasi semua pengaturan ini dengan thread group baru ini.
- Tidak seperti thread group dasar, yang ini tidak membuat semua thread di awal, tetapi hanya membuatnya saat diperlukan, yang berarti bahwa memori tambahan akan dihemat.
- Tidak seperti menggunakan thread group sederhana, Concurrency Thread Group dengan anggun mematikan semua thread, sementara Thread Group 'kills' them ketika 'end time' tiba, tidak ada ramp-down.
- Kita mendapatkan pratinjau tentang bagaimana concurrency kita dari waktu ke waktu akan terlihat hanya dengan mengubah pengaturan awal. Pratinjau ini akurat tetapi tidak mencakup ramp-down dari pengguna virtual karena thread tidak dibunuh dan durasi eksekusi *script* berbeda untuk setiap pengujian dan pelaksanaan pengujian.

Stepping Thread Group

Ini adalah versi lama dari Concurrency Thread Group dan memerlukan sedikit lebih banyak konfigurasi. Pada dasarnya, hasil akhirnya sama, satu-satunya hal yang berubah adalah cara ramp-up dan ramp-down dilakukan secara berbeda.

Bidang-bidangnya cukup jelas dan mengisi bidang input seperti menyelesaikan kalimat yang tidak lengkap: Grup ini akan memulai **xx thread** (konkurensi yang ditargetkan). Pertama tunggu **xx** detik, kemudian mulai **xx** thread, selanjutnya tambahkan **xx** thread setiap **xx** detik menggunakan ramp-up **xx** detik. Kemudian tahan beban selama **xx** detik (setara dengan waktu hold rate). Akhirnya, hentikan **xx** thread setiap **xx** detik.

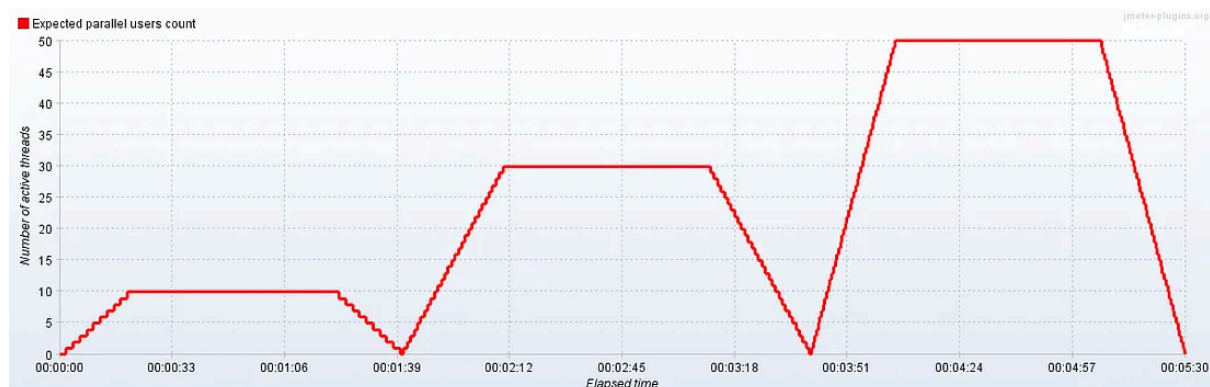
Dalam hal apapun, untuk skenario di mana kita harus mempertahankan sejumlah pengguna bersamaan selama periode waktu tertentu, saya sarankan menggunakan Concurrency Thread Group karena kita tidak perlu repot-repot dengan banyak konfigurasi.

Ultimate Thread Group

Ultimate Thread Group **highly customizable** dan tidak seperti Arrivals, atau Free Form dan Concurrency Thread Groups, yang ini kills active thread setelah waktu yang ditentukan berakhir. Ini akan menerjemahkan ke perilaku pengguna yang sebenarnya karena pengguna hanya menutup tab aplikasi atau browser. Ini juga berarti bahwa grafik pratinjau yang menunjukkan jumlah pengguna paralel yang diharapkan akan sangat akurat karena ramp-down dikendalikan oleh thread group, bukan ditentukan oleh waktu eksekusi setiap thread. Thread group ini juga memulai thread baru untuk mempertahankan jumlah thread dalam situasi di mana satu thread telah selesai dieksekusi.

Lihat contoh untuk ini di bawah ini:

Grafik pratinjau yang dihasilkan setelah mengkonfigurasi thread group:



Grafik yang dihasilkan setelah benar-benar menjalankan *script* dengan konfigurasi di atas:



Kemampuan dan opsi konfigurasi thread group:

- **Threads schedule table:** di mana kita dapat menambahkan beberapa baris, masing-masing dieksekusi secara berurutan, masing-masing dimulai sesuai dengan pengaturan 'Initial Delay'. Ini berarti bahwa baris (grup pengguna virtual) akan dieksekusi secara paralel jika pengaturan 'Initial Delay' sama untuk semua bidang.
- **Start threads count field:** di mana kita mengatur target konkurensi thread.
- **Initial delay:** di sini kita mengatur berapa lama kita menunggu sebelum mulai mengeksekusi thread dari baris tertentu itu.
- **Startup time:** mewakili waktu ramp-up untuk semua thread yang diatur dalam jumlah thread. Misalnya, untuk 20 thread dan waktu mulai 20 detik, kita akan memiliki 1 thread baru yang dimulai setiap 1 detik.
- **Hold load for field:** di sinilah kita mengatur durasi untuk mempertahankan jumlah pengguna bersamaan yang ditentukan dan sepenuhnya independen dari waktu ramp-up.
- **Shutdown time:** interval waktu untuk membunuh semua thread aktif.

Kita dapat menambahkan beberapa baris yang berisi pengaturan ini sehingga menghasilkan pola kompleks yang dapat mensimulasikan jumlah pengguna yang berfluktuasi yang mengakses server kita.

Skenario yang mungkin digunakan untuk:

- **Complex spike testing scenarios** — di mana kita dapat tiba-tiba meningkatkan jumlah pengguna per menit, kemudian menurunkannya lagi dan melihat bagaimana aplikasi berperilaku. Dengan menggunakan teknik ini, kita juga dapat menguji kemampuan pemulihan dari sistem tertentu.

Blazemeter Introduction

BlazeMeter adalah platform berbasis cloud yang memungkinkan pengujian kinerja dengan skala yang lebih besar dan integrasi yang lebih mudah. BlazeMeter mendukung JMeter dan menawarkan fitur tambahan seperti analisis real-time dan pengujian dari berbagai lokasi.

Perbedaan Antara JMeter dan BlazeMeter

1. Tipe Alat:

- JMeter:
 - JMeter adalah alat open-source yang digunakan untuk pengujian kinerja dan beban aplikasi. Ini dapat diunduh dan dijalankan secara lokal di komputer pengguna.
- BlazeMeter:
 - BlazeMeter adalah platform berbasis cloud yang dirancang untuk pengujian kinerja. Ini memungkinkan pengguna untuk menjalankan pengujian dari server cloud, yang dapat mensimulasikan ribuan pengguna dari berbagai lokasi.

2. Skalabilitas:

- JMeter:
 - JMeter dapat digunakan untuk pengujian skala kecil hingga menengah, tetapi ketika melakukan pengujian dengan jumlah pengguna yang sangat besar, mungkin memerlukan konfigurasi tambahan dan sumber daya yang lebih besar.
- BlazeMeter:
 - BlazeMeter dirancang untuk pengujian skala besar. Ini dapat dengan mudah mensimulasikan ribuan hingga jutaan pengguna virtual tanpa memerlukan pengaturan infrastruktur yang rumit.

3. Integrasi dan Fitur Tambahan:

- JMeter:
 - JMeter memiliki banyak plugin dan fitur, tetapi semua pengujian dilakukan secara lokal. Pengguna harus mengelola semua aspek pengujian, termasuk pengaturan server dan analisis hasil.
- BlazeMeter:
 - BlazeMeter menawarkan integrasi yang lebih baik dengan alat CI/CD dan menyediakan fitur tambahan seperti analisis real-time, laporan yang lebih

mendetail, dan kemampuan untuk menjalankan pengujian dari berbagai lokasi di seluruh dunia.

4. Antarmuka Pengguna:

- JMeter:
 - JMeter memiliki antarmuka pengguna berbasis desktop yang mungkin terasa kurang intuitif bagi beberapa pengguna baru. Pengguna harus menginstal dan mengkonfigurasi JMeter di komputer mereka.
- BlazeMeter:
 - BlazeMeter memiliki antarmuka pengguna berbasis web yang lebih modern dan ramah pengguna, memungkinkan pengguna untuk mengelola pengujian dengan lebih mudah dan cepat.

5. Biaya:

- JMeter:
 - JMeter adalah perangkat lunak open-source dan gratis untuk digunakan. Pengguna tidak perlu membayar untuk menggunakan JMeter, meskipun mereka mungkin perlu mengeluarkan biaya untuk infrastruktur jika melakukan pengujian besar.
- BlazeMeter:
 - BlazeMeter menawarkan model berlangganan dengan berbagai tingkatan harga, tergantung pada fitur dan kapasitas yang dibutuhkan. Meskipun ada versi gratis, fitur lengkap biasanya memerlukan biaya.

Automate Your Performance Testing using Jmeter and Blazemeter

Untuk melakukan automate testing di blazemeter, kita perlu memiliki akun yang telah dikonfigurasi di BlazeMeter <https://www.blazemeter.com>. Silahkan daftarkan akun baru di situs blazemeter.

Setelah itu, kita perlu mendapatkan API Keys kita untuk mengakses sistem melalui JMeter. Untuk melakukannya, buka **Configurations > Personal > API Keys**.

Settings

 Account >

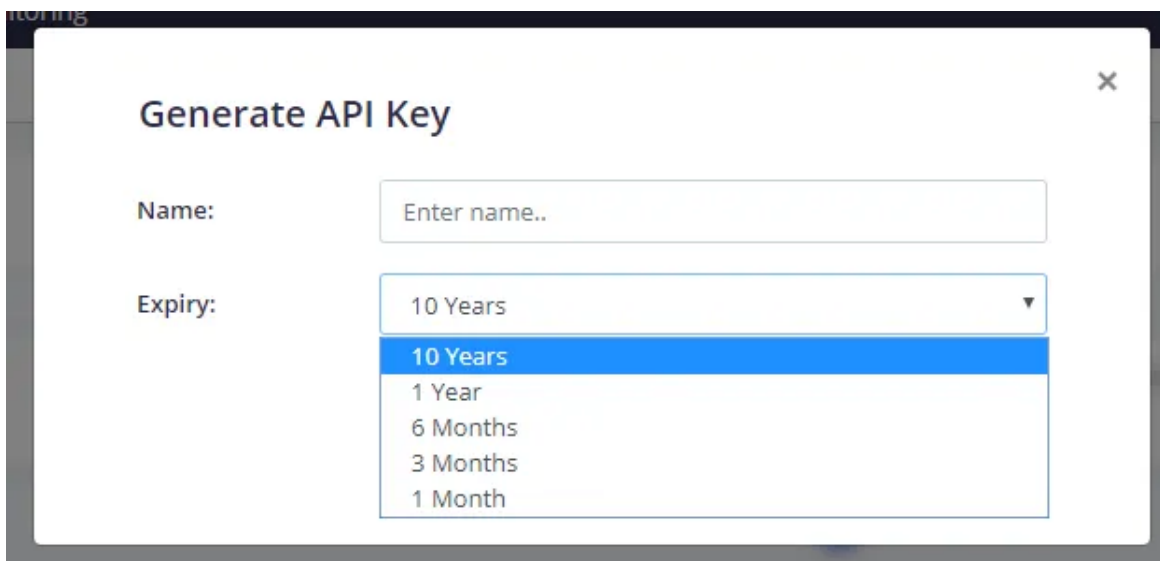
 Workspace >

 Personal v

Personal Settings

API Keys

Klik pada “+” untuk menambahkan **API Keys** baru dan pilih nama serta periode kedaluwarsa untuk API Keys yang akan kita buat.



The dialog box titled "Generate API Key" contains two input fields. The "Name:" field is a text box with the placeholder "Enter name..". The "Expiry:" field is a dropdown menu with "10 Years" selected. The dropdown menu is open, showing options: "10 Years", "1 Year", "6 Months", "3 Months", and "1 Month".

Setelah itu, kita akan mendapatkan dua parameter yang sangat penting: **API Key ID** dan **API Key Secret**. Simpan nilai-nilai tersebut dengan baik, karena kita akan membutuhkannya.

Generate API Key

Name: teste

Expiry: 1 Month

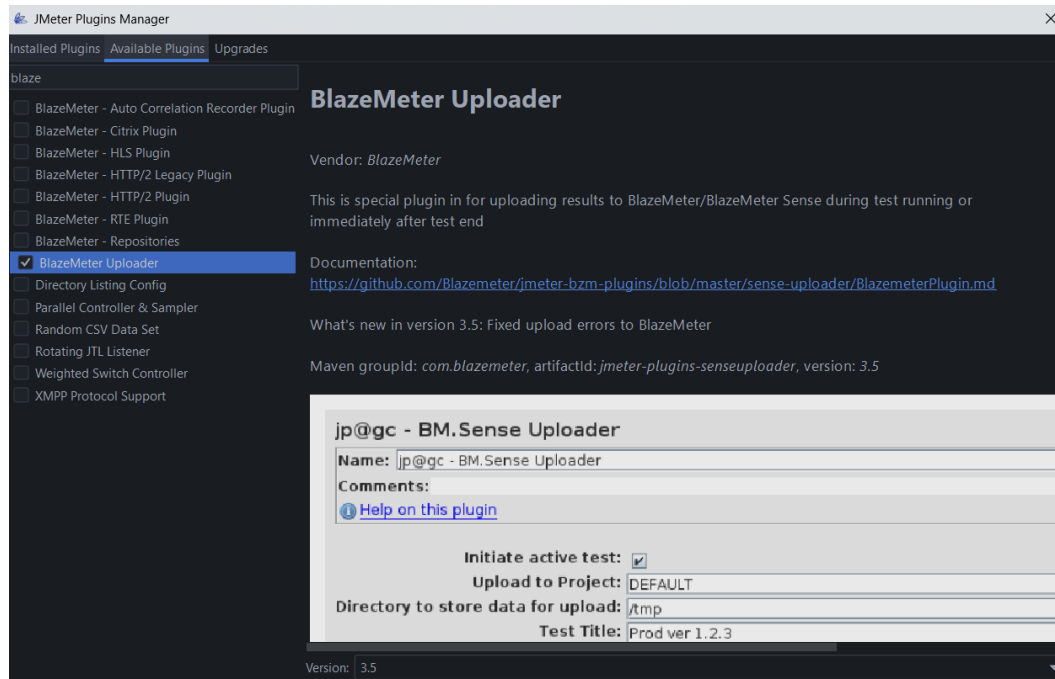
API Key Id:  

API Key Secret:  

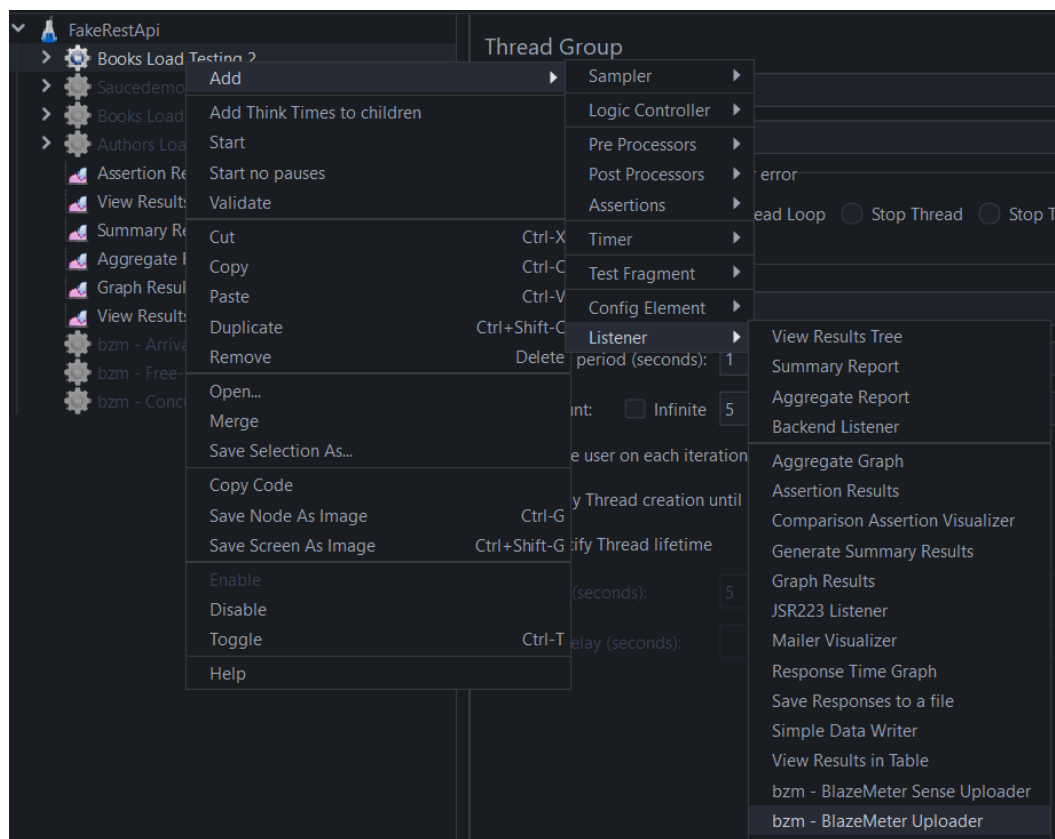
[Download as JSON](#)

Mengintegrasikan Plugin BlazeMeter dengan JMeter

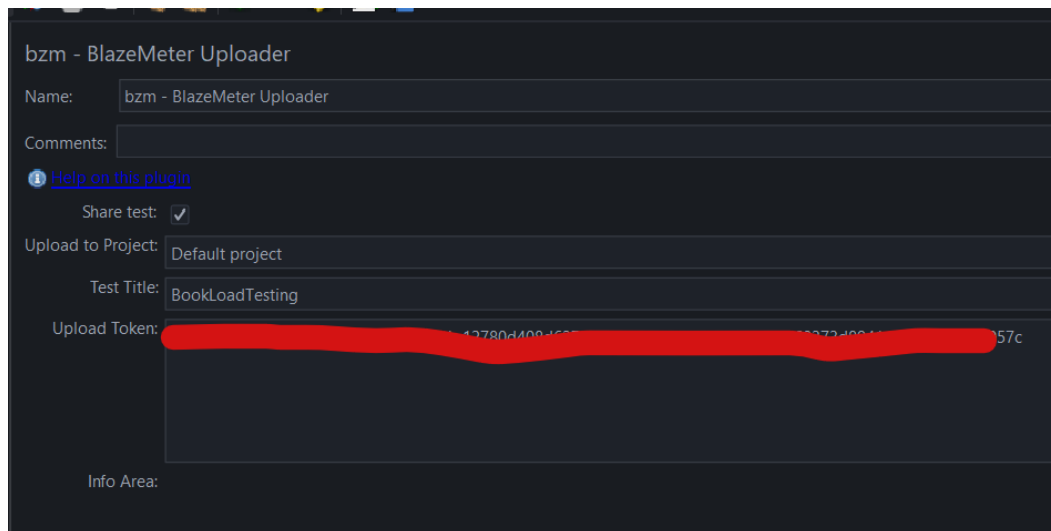
Sebelum memulai, kita perlu memastikan bahwa kita telah menginstal Plugin Manager di JMeter kita. Jika belum, silahkan install dulu Blazemeter Uploader plugin.



Sekarang, buka project yang ingin kita jalankan menggunakan platform BlazeMeter dan tambahkan **BlazeMeterUploader Listener** ke project kita.



Setelah selesai, kita perlu mengkonfigurasi plugin. Cukup isi nama dan judul untuk pengujian kita. Setelah itu, tambahkan API Keys yang kita dapatkan dari BlazeMeter di kolom **Upload Token**, mengikuti format: **API Key ID:API Key Secret**.



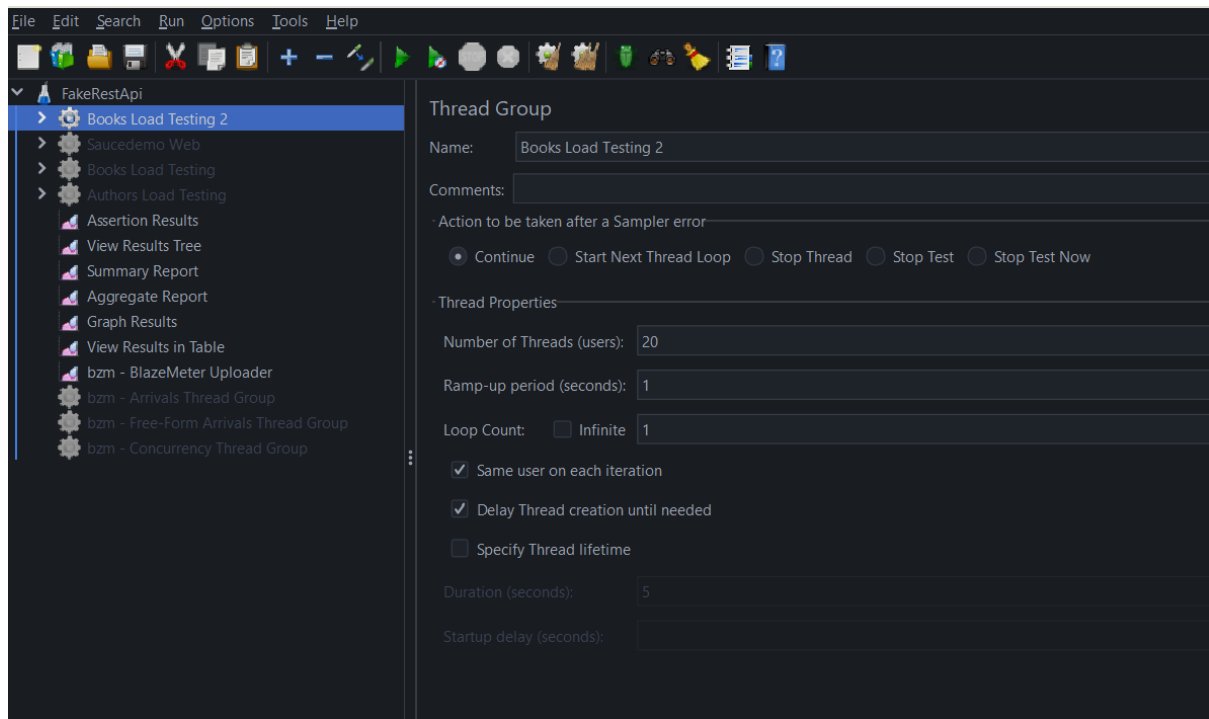
Note: Jangan lupa untuk menempatkan “:” di antara API Keys, jika tidak, itu tidak akan berfungsi.



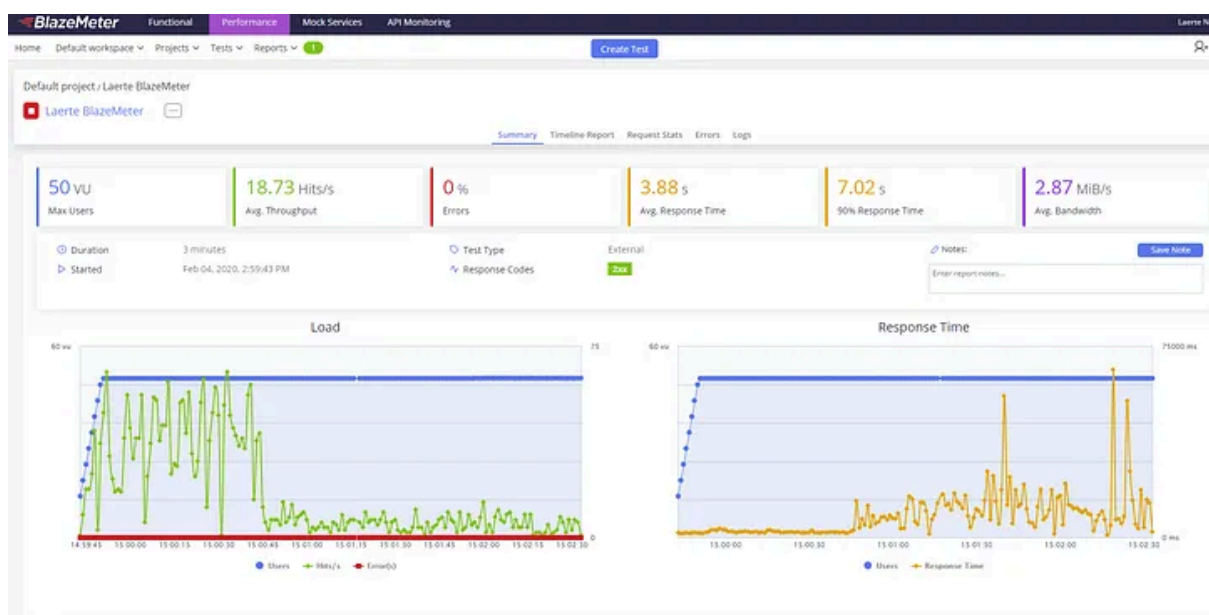
Running Tests and Analyses

Saatnya untuk mulai menjalankan beberapa pengujian. Jadi, mari kita konfigurasi Thread Group kita di JMeter dengan beberapa pengguna.

Jika kita menjalankan flow skenario sederhana untuk melakukan pengujian ini, tempatkan setidaknya 20 concurrent user untuk mendapatkan lebih banyak data untuk dianalisis. Jika kita menjalankan skenario yang kompleks yang memerlukan waktu untuk diselesaikan, pastikan untuk menyesuaikan jumlah pengguna sesuai dengan kebutuhan pengujian kita.



Klik tombol play pada test plan kita di JMeter. Lalu kita akan melihat sebuah halaman web terbuka dengan pengujian kita yang sedang dieksekusi. Ini adalah platform BlazeMeter yang menjalankan pengujian kita di cloud!



Setelah pengujian selesai, kita akan menerima email dengan beberapa data ringkasan yang menarik tentang eksekusi pengujian kita.

Your test results are ready

Hi Laerte Neto,

Your test "Laerte BlazeMeter" is now complete.

[View the full report;](#)

[View the executive summary;](#)

Test summary



51s
Duration



5 vu
Max Users



3.92 Hits/s
Avg. Throughput



0%
Errors

Label	Hits		Response Time	
	Total	Failed	Avg.	90%
ALL	200	0	1,701	3,444
Boston - Rome	50	0	1,929	2,280
http://blazedemo.com/confirmation.php	50	0	773	791
http://blazedemo.com/purchase.php	50	0	700	888
Passagem de avião THINK TIME	50	0	3,403	4,228



www.blazemeter.com



Kita juga dapat menjelajahi semua tab yang tersedia di BlazeMeter dan mengumpulkan data yang relevan bagi kita.

Berikut adalah beberapa tips tentang data menarik yang dapat kita kumpulkan dari setiap tab:

- **Summary:** Tampilkan gambaran eksekusi yang baik dengan data relevan tentang response server dan beban. Kita dapat menggunakan data ini sebagai gambaran umum project.
- **Timeline Report:** kita dapat menggabungkan permintaan apapun dengan kesalahan, jumlah server yang diakses, rata-rata, beban, dan lainnya. Grafiknya interaktif. Grafik yang bagus untuk dimiliki dalam analisis kita adalah throughput vs pengguna aktif. Kita dapat melihat bagaimana performa aplikasi Anda seiring dengan jumlah pengguna yang aktif.
- **Request Stats:** Kita dapat memeriksa statistik untuk setiap permintaan, seperti rata-rata, throughput, garis 90%, persentase kesalahan, dan banyak lagi. Ini mirip dengan Aggregate Report yang sudah kita miliki di JMeter. Dengan informasi ini, kita dapat menulis laporan terperinci, mencakup dan membandingkan setiap permintaan secara terpisah.
- **Errors:** Rincian tentang kesalahan eksekusi.
- **Logs:** Log eksekusi.

Pada titik ini, kita memiliki proyek yang sepenuhnya dikonfigurasi untuk dijalankan di cloud menggunakan platform BlazeMeter. Kita hanya perlu menyalin dan menempelkan BlazeMeter Uploader ke dalam project kita.