

# **Performance Test menggunakan JMeter**

Oleh : Suhendar

## Table of Contents

1	Jmeter .....	1
1.1	Membuka Jmeter .....	1
1.2	Elemen Utama di jmeter .....	2
1.2.1	Thread group.....	3
1.2.2	Sampler .....	5
1.2.3	Configuration .....	7
1.2.4	Listener.....	10
1.2.5	Lain lain .....	11
1.3	Scripting di Jmeter .....	13
1.4	Distributed Testing di Jmeter .....	13
1.5	Running di Jmeter .....	15
1.5.1	Running Jmeter di GUI .....	15
1.5.2	Running jmeter dari command prompt .....	16
1.6	Analisis Hasil di Jmeter.....	18

# 1 Jmeter

Jmeter adalah aplikasi open source berbasis java yang dapat digunakan untuk perofomance test.

Selain itu jmeter juga dapat diinstall di computer pribadi dengan menginstall java dan mengunduh jmeter dari web resmi jmeter.

## 1.1 Membuka Jmeter

1. Masuk ke salah satu server jmeter dan buka directory  
C:\Users\Administrator\Documents\apache-jmeter-5.3\bin
2. Untuk membuka jmeter terdapat beberapa cara, yaitu
  - Membuka file jmeter.bat yang ada di folder tadi .

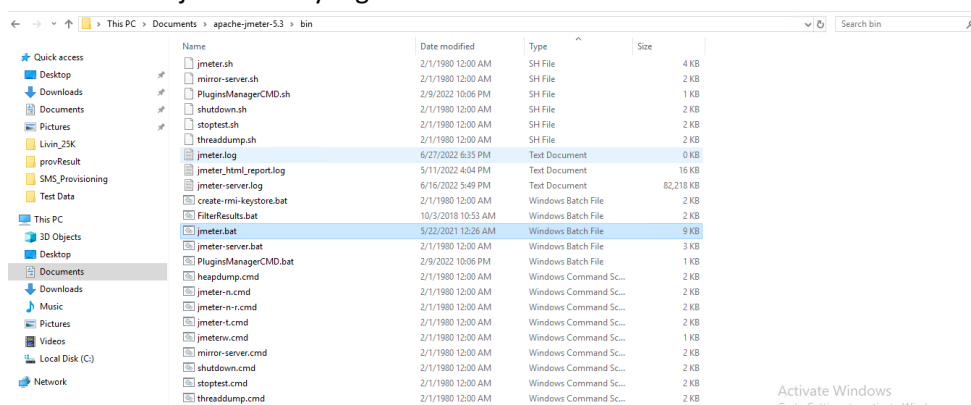


Figure 1 Jmeter Directory

- Menggunakan command prompt di directory tadi dengan command “jmeter”

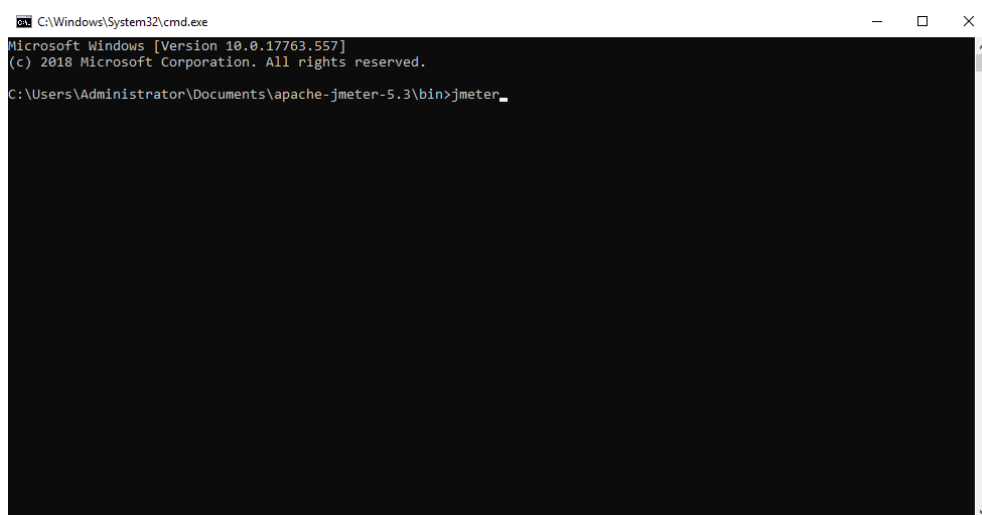


Figure 2 Command to open jmeter from cmd

3. Akan muncul tampilan GUI jmeter seperti pada Figure 3. Setiap membuka jmeter, akan muncul komponen Test Plan pada Test Plan pane. Semua script dan scenario running di jmeter akan disimpan di dalam tiap Test Plan.

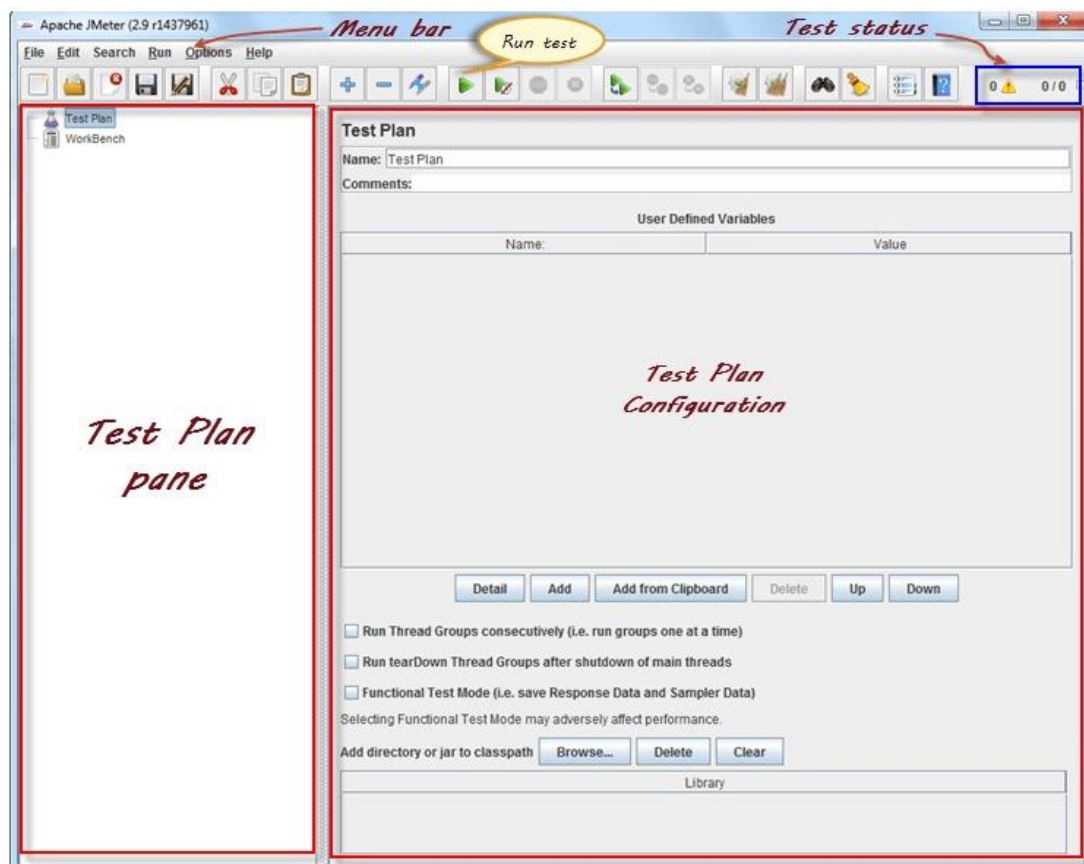


Figure 3 Jmeter GUI

## 1.2 Elemen Utama di jmeter

Terdapat sangat banyak elemen di jmeter, untuk menambah elemen di jmeter dapat dengan klik kanan di test plan (atau komponen lain tempat komponen baru ingin diletakkan) dan hover mouse ke "add"

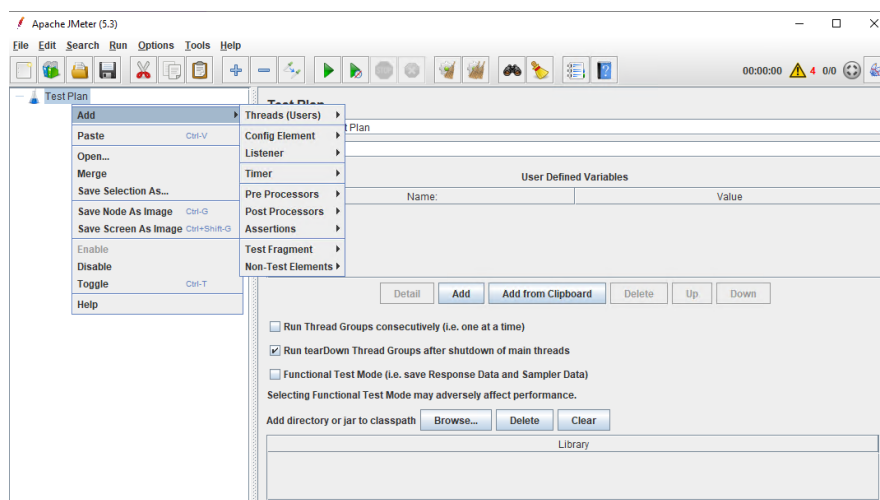


Figure 4 Add component to Test Plane Panel

Terdapat 4 kelompok elemen utama yang sering digunakan untuk membuat script di jmeter, antara lain thread group, sampler, configuration, listener, dan beberapa elemen lain

### 1.2.1 Thread group

Thread group adalah kumpulan dari user (atau disebut thread di jmeter) yang akan melakukan running. Figure 5 menunjukkan user-user yang akan mensimulasikan load testing digenerate dari thread group.

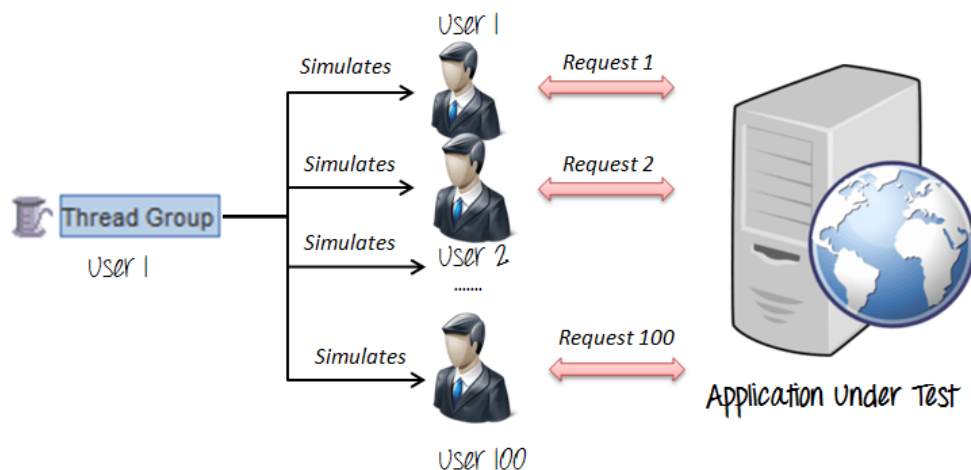


Figure 5 Visualization of thread and thread group

Terdapat banyak jenis thread group di jmeter, namun terdapat 2 thread group yang sering digunakan, yaitu Thread Group yang merupakan bawaan dari jmeter, dan jp@gc – Stepping Thread Group yang merupakan plugin yang dapat diinstall ke jmeter

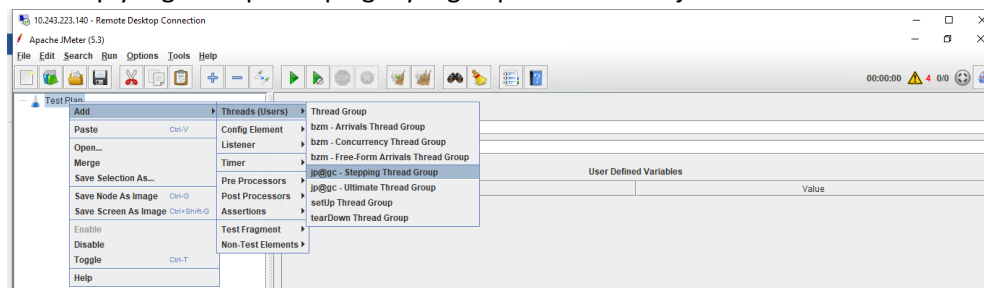


Figure 6 Thread Group in Jmeter

#### 1.2.1.1 Thread Group (elemen)

Thread group mampu mensimulasikan sejumlah user, dengan tiap user melakukan sejumlah request sesuai yang diset di thread group.

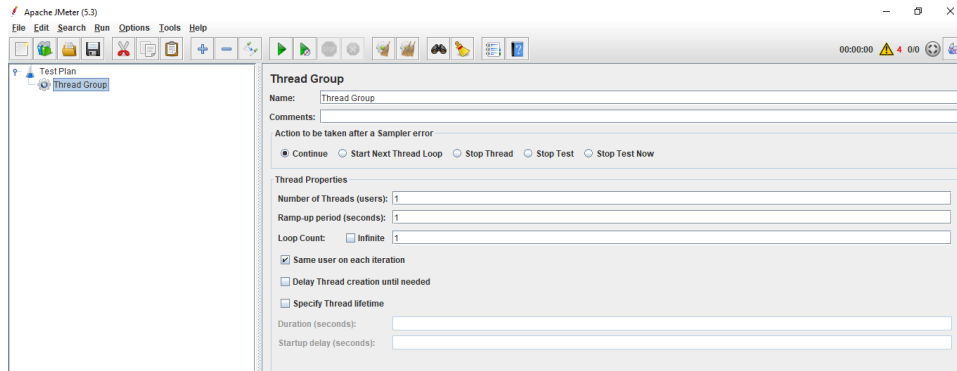


Figure 7 Thread Group (Element)

- Number of Threads (users) : Jumlah user yang ingin disimulasikan
- Ramp-up period : Menyatakan waktu kenaikan user secara linear dari 0 sampai dengan jumlah yang ditetapkan di number of threads
- Loop Count : Menyatakan berapa kali tiap user akan melakukan loop request, apabila checkbox infinite dicek, maka running akan terus berjalan dan akan berhenti apabila waktu running ditentukan atau running dihentikan secara paksa
- Same user each iteration : Untuk mensimulasikan untuk user yang sama pada iterasi yang berbeda, hal ini bertujuan untuk mensimulasikan cookies dan cache agar digunakan pada iterasi berikutnya
- Specify Thread lifetime dapat dicek untuk menentukan durasi run dalam detik.

**Thread Group lebih sering digunakan ketika scripting agar dapat mensimulasikan 1 request saja.**

#### 1.2.1.2 jp@gc – Stepping Thread Group

Stepping thread group dapat mensimulasikan sejumlah user dengan behavior yang lebih kompleks misalnya dengan jumlah user yang naik secara berkala (ramp-up).

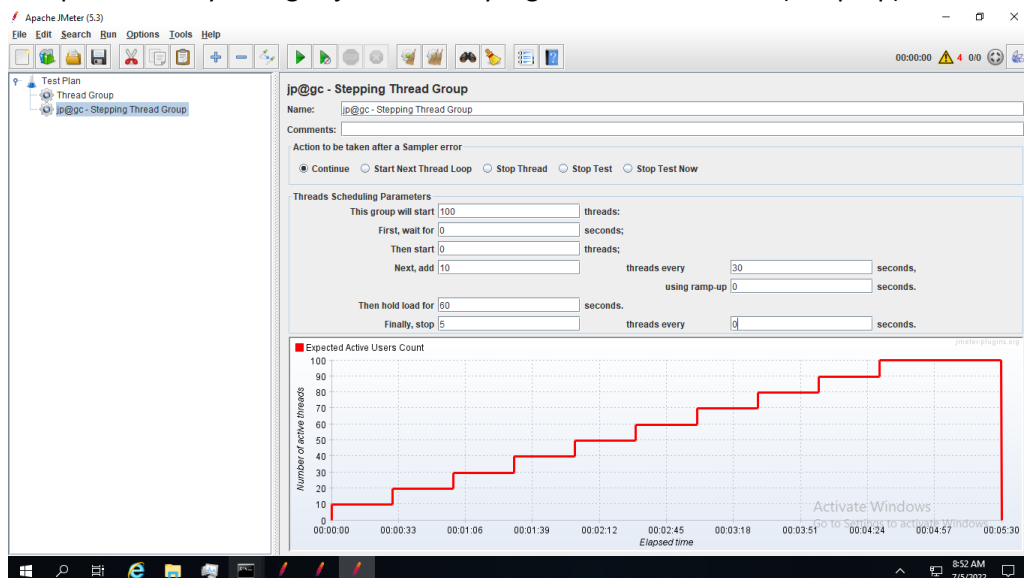


Figure 8 jp@gc - Stepping Thread Group

- This group will start : Jumlah user yang ingin disimulasikan
- First, wait for : Delay sebelum user naik
- Then start : Jumlah user di awal running

- Next, add.... : Jumlah user yang ingin ditambah (ramp-up) per waktu yang ditentukan
- ... thread, every : Durasi per ramp up
- Then hold load for : Durasi running pada jumlah user maximum yang ditentukan di "This group will start"
- Finally, stop... : Jumlah user yang ingin dikurangi (ramp down) per waktu yang ditentukan
- ... thread, every : Durasi ramp down

**jp@gc – Stepping Thread Group digunakan untuk Stress Test, Load Test, maupun Endurance Test untuk mensimulasikan ramp-up scenario testing.**

### 1.2.2 Sampler

Sampler adalah elemen request yang akan dilakukan oleh tiap user. Figure 9 menunjukkan semua sampler yang ada di jmeter, dari semua sampler tersebut, yang paling sering digunakan adalah HTTP Request dan JDBC Request.

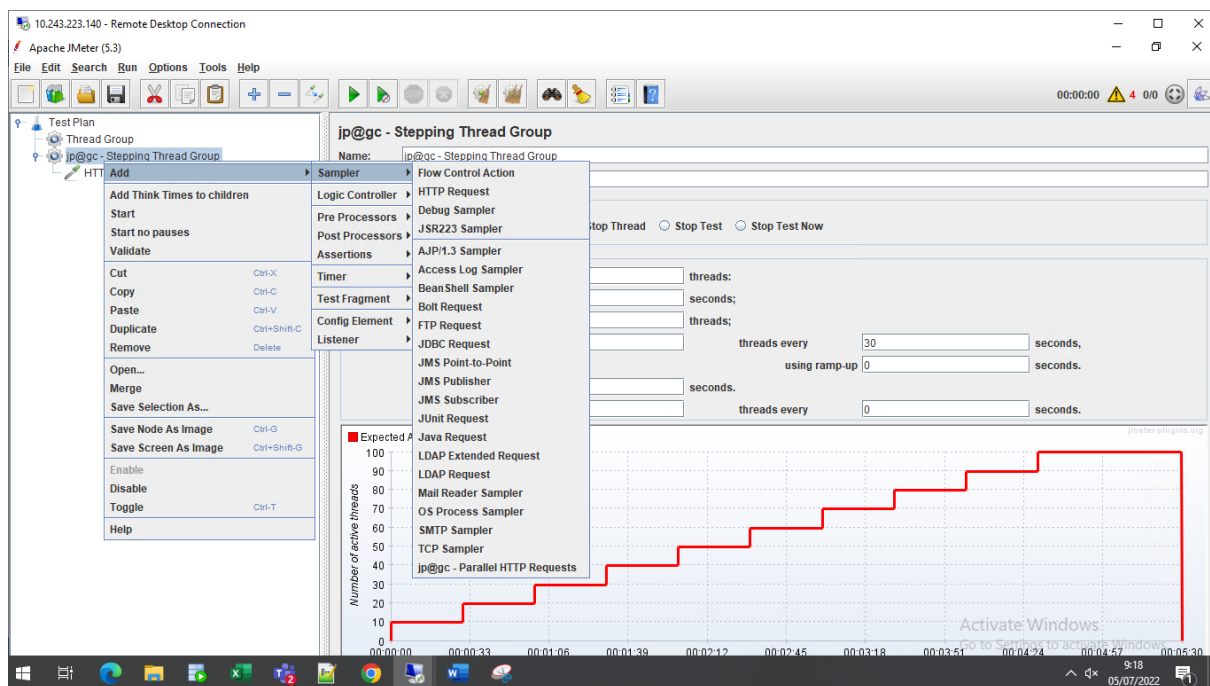


Figure 9 Sampler

#### 1.2.2.1 HTTP Request

HTTP Request digunakan untuk membuat sample request ke url yang ingin dilakukan testing. UI dari HTTP request ditunjukkan pada Figure 10.

- Protocol [http] : Menentukan protocol dari url yang akan disimulasikan (http atau https)
- Server Name or IP : Server atau IP yang ingin disimulasikan (tanpa http:// E.g. google.com)
- Port Number : Port dari ip yang ingin disimulasikan, apabila tidak ada, bisa dikosongkan

- HTTP Request : drop down untuk memilih method request yang ingin disimulasikan (POST, PUT, GET, dan sebagainya)
- Path : API endpoints yang ingin disimulasikan tanpa "/" setelah url. Apabila api yang ingin disimulasikan adalah

*"http://api-pt.apps.ocp.everest.supporting.devmandiri.co.id/identity-service/identity/v1/login",*

maka, HTTP request diset seperti pada Figure 10

- Body : Request Body dari api yang ingin disimulasikan dengan format json
- Parameters : Query param untuk api yang ingin disimulasikan. **Untuk dapat mengakses tab Parameters, Body harus kosong.** Untuk menambah query parameters, tekan tombol add di bagian bawah, akan muncul 1 baris data kosong yang dapat diisi. Misalnya ingin membuat api dengan query param "\${url}?page=1, pada kolom "name" diisi "page" dan kolom "value" diisi "1" seperti yang terlihat pada Figure 11.
- Pada tab advanced, bisa ditentukan timeout dan proxy untuk HTTP Request

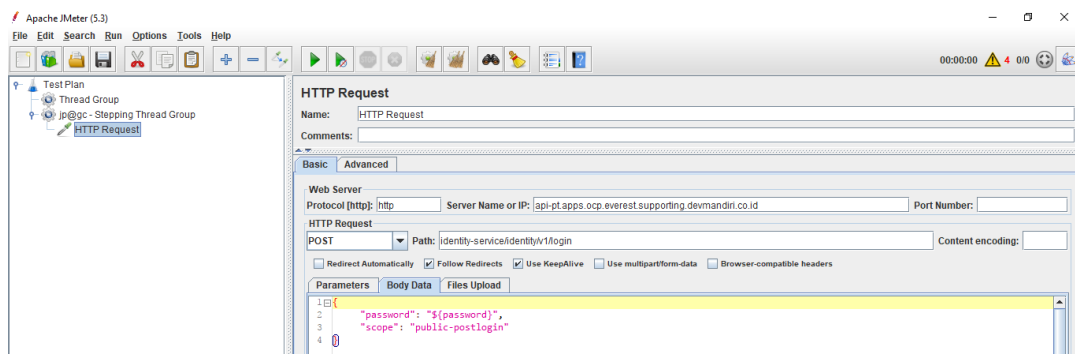


Figure 10 HTTP Request

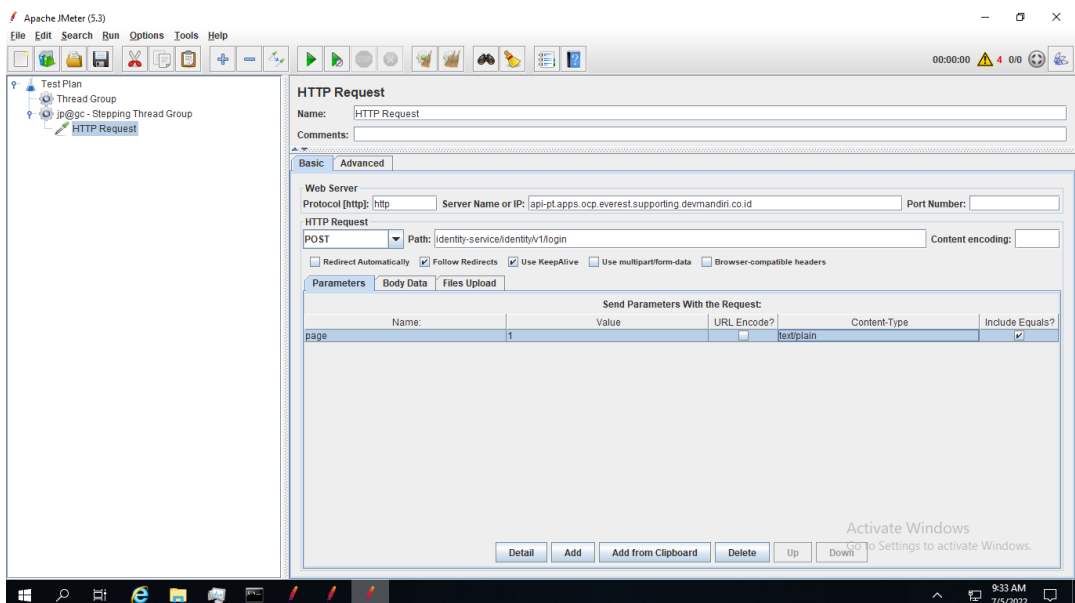


Figure 11 Parameters in HTTP Request



### 1.2.2.2 JDBC Request

JDBC request digunakan untuk melakukan query langsung ke database. UI JDBC Request ditunjukkan pada Figure 12.

- Variable Name of Pool Declared in JDBC Connection Configuration diisi dengan variable pool yang dinyatakan di JDBC configuration (1.2.4)
- SQL Query diisi dengan query yang akan disimulasikan.

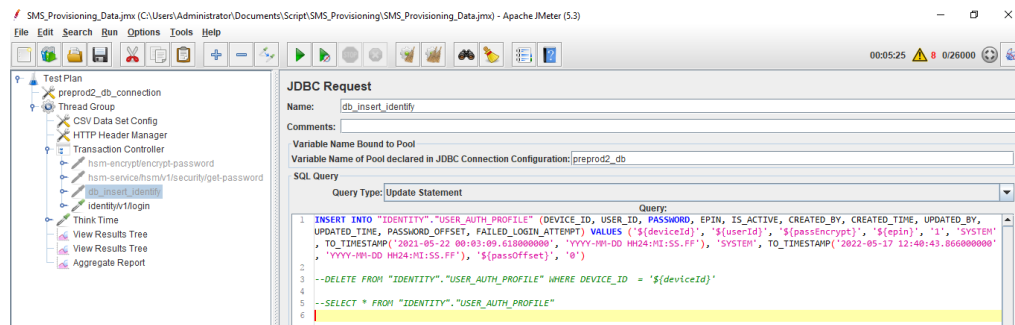


Figure 12 JDBC Request

### 1.2.3 Configuration

Configuration berisi set up variable yang nantinya akan digunakan di sampler, terdapat beberapa configuration yang sering digunakan di dalam scripting, antara lain csv data set config, HTTP header manager, HTTP request default, dan JDBC Connection Configuration

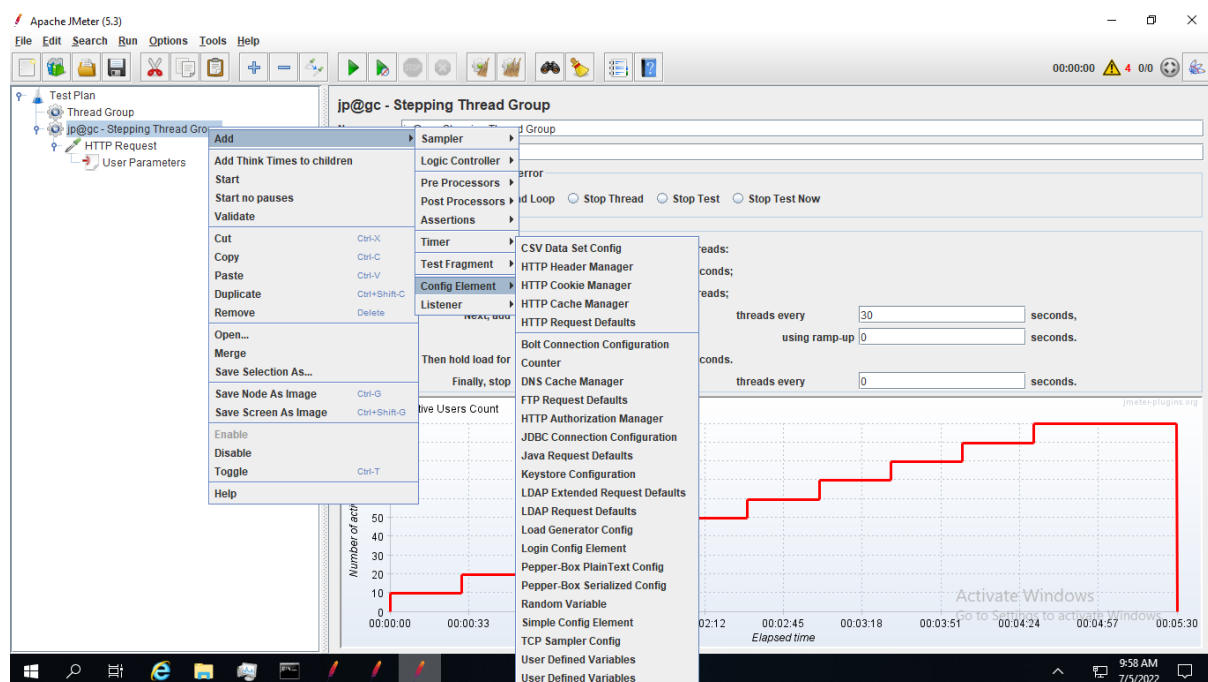


Figure 13 Config Element

#### 1.2.3.1 CSV data set config fungsi untuk mengambil data user atau data lain yang dibutuhkan untuk testing dari file csv.

- Filename : directory dari file yang ingin dimuat
- Ignore first line: Untuk mengabaikan data line 1 apabila terdapat header pada csv
- Delimiter : Menentukan pemisah tiap data yang akan dimuat

- Allow quoted data : Untuk menentukan apakah jmeter akan membaca data di antara petik dua (“...”) atau tidak.
- Recycle on EOF : Recycle on End of Cycle, menentukan apakah apabila semua data telah terpakai, maka request akan menggunakan data pertama lagi atau tidak.
- Stop thread of EOF : Menentukan apakah running dihentikan ketika semua data telah terpakai atau tidak
- Sharing Mode : Menentukan distribusi data untuk thread atau bahkan thread group yang berbeda. Apabila dipilih All thread, maka data akan terdistribusi ke semua thread, artinya dalam 1 iterasi, setiap thread pada semua thread group dalam 1 test plan.

Data yang dimuat di csv data set config dapat diakses sesuai dengan nama headernya dengan format `${nama_header}` misal terdapat header `userId`, apabila terdapat `requestBody` `userId`, dapat ditulis

```
{
    "userId": "${userId}"
}
```

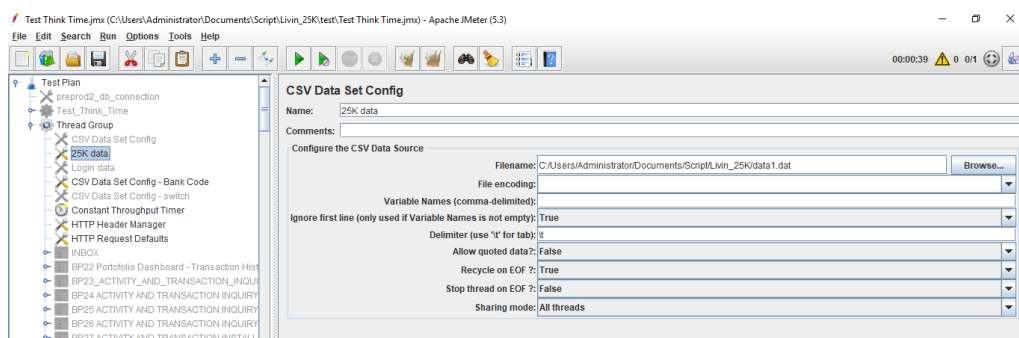


Figure 14 CSV Data Set Config

### 1.2.3.2 HTTP Header Manager

HTTP Header Manager digunakan untuk menambah header pada request. Untuk menambah header, dilakukan dengan menekan tombol add pada Figure 16. HTTP header manager dapat berlaku secara khusus untuk 1 request atau ke banyak request tergantung dari peletakan HTTP Header Manager

Pada Figure 15 HTTP Header Manager 1 akan berlaku pada HTTP Request 1 dan 2, sementara HTTP Header Manager 2 hanya akan berlaku ke HTTP Request 1.

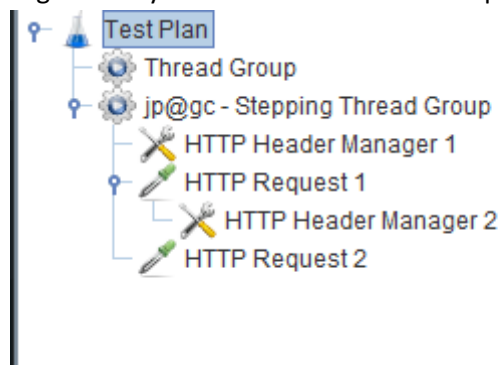


Figure 15 HTTP Header Manager Level

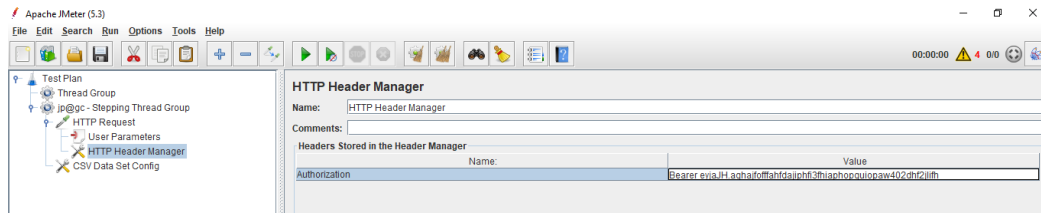


Figure 16 HTTP Header Manager

### 1.2.3.3 HTTP Request Default

HTTP Request Default berfungsi untuk menghilangkan perulangan penulisan protocol, url, port, maupun input lain di HTTP Request. Dalam testing, url pada banyak HTTP request adalah sama, untuk itu dapat diletakkan HTTP Request Default sebelum HTTP Request agar tidak perlu mengisi url dan protocol tiap HTTP Request secara manual.

Misalnya diinginkan testing untuk 2 api

- `http://api-pt.devqa.co.id/identity-service/identity/v1/public-limited`
- `http://api-pt.devqa.co.id/identity-service/identity/v1/login`

Maka, dapat buat HTTP Request Default sebelum 2 HTTP Request dibuat HTTP Request Default dengan protocol dan url yang sudah diisi seperti pada Figure 17

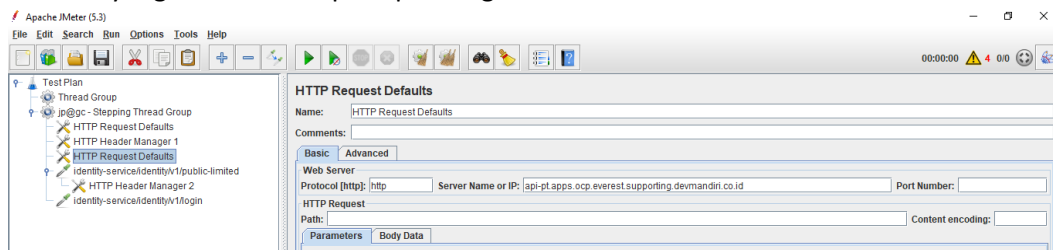


Figure 17 HTTP Request Default

Maka, 2 HTTP Request di bawah HTTP Header manager tidak perlu diisi di bagian protocol dan server Name or IP seperti pada Figure 18.

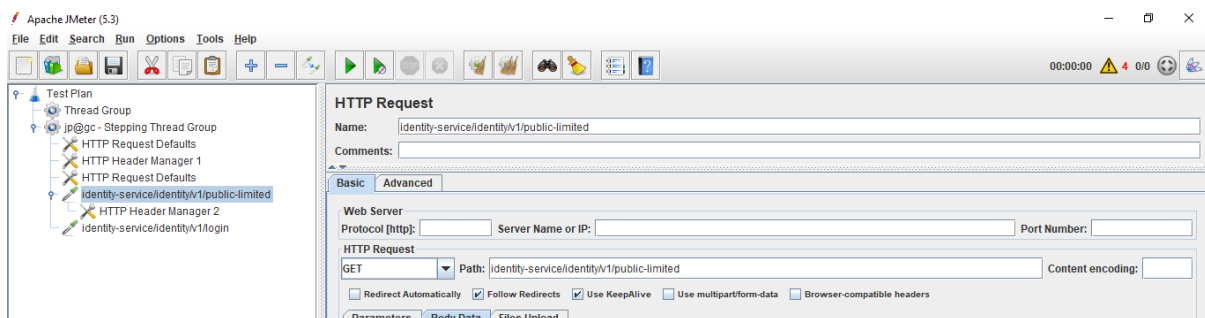


Figure 18 HTTP Request with HTTP Request Default

### 1.2.3.4 JDBC Connection Configuration

JDBC Connection Configuration digunakan untuk menghubungkan jmeter terhadap database sehingga dapat dilakukan query langsung dari jmeter ke database di dalam JDBC Request.

- Variable name for created pool adalah nama variable yang nantinya akan digunakan di JDBC Request
- Max number of connection adalah jumlah koneksi maksimal ke DB
- Database connection configuration perlu data credential dari database

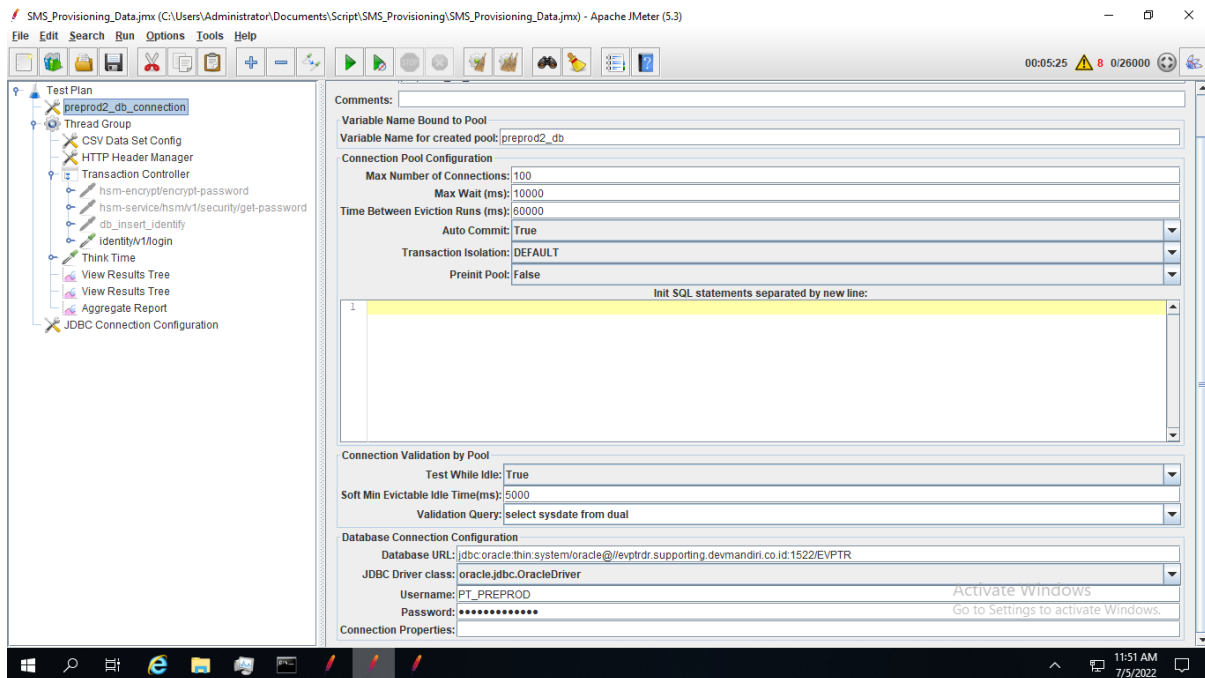


Figure 19 JDBC Connection Configuration

#### 1.2.4 Listener

Listener adalah elemen jmeter untuk memantau hasil dari running terkait nilai response time, error rate, TPS, grafik-grafik yang dibutuhkan maupun sample request dan response dari simulasi yang dilakukan. Listener yang sering digunakan antara lain adalah aggregate report, result tree, dan beberapa listener dari plugin jp@gc.

##### 1.2.4.1 Aggregate report

Aggregate report berisi ringkasan dari hasil running seperti total hit yang dilakukan, nilai TPS (throughput), error rate, response time, dan sebagainya. Hasil running dapat disimpan dengan cara memberi directory dan nama pada box Filename.

Aggregate Report

Name:Aggregate Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec
1.1.1. identity/v1/...	121	9007	9015	9250	9268	9306	8400	9310	100.00%	12.9/sec	30.83
02.1.1. inbox/v1/...	9	7484	7506	7853	7919	7919	6946	7919	100.00%	1.1/sec	2.71
transfer/v1/user/...	13	9053	9084	9245	9245	9253	8796	9253	100.00%	1.4/sec	3.32
03.1.1. bi-fast/v1/...	7	9024	9021	9205	9315	9315	8750	9315	100.00%	44.9/min	1.79
BP03_01_WEL...	7	9024	9021	9205	9315	9315	8750	9315	100.00%	44.4/min	1.77
TOTAL	157	8925	8993	9245	9268	9310	6946	9315	100.00%	16.6/sec	39.54

Figure 20 Aggregate report

#### 1.2.4.2 View Result Tree

Result tree menunjukkan semua sample request yang disimulasikan. Result tree dapat memfilter untuk hanya mengambil request yang berhasil atau gagal dengan men-cek checkbox Log/Display only di bagian kana atas.

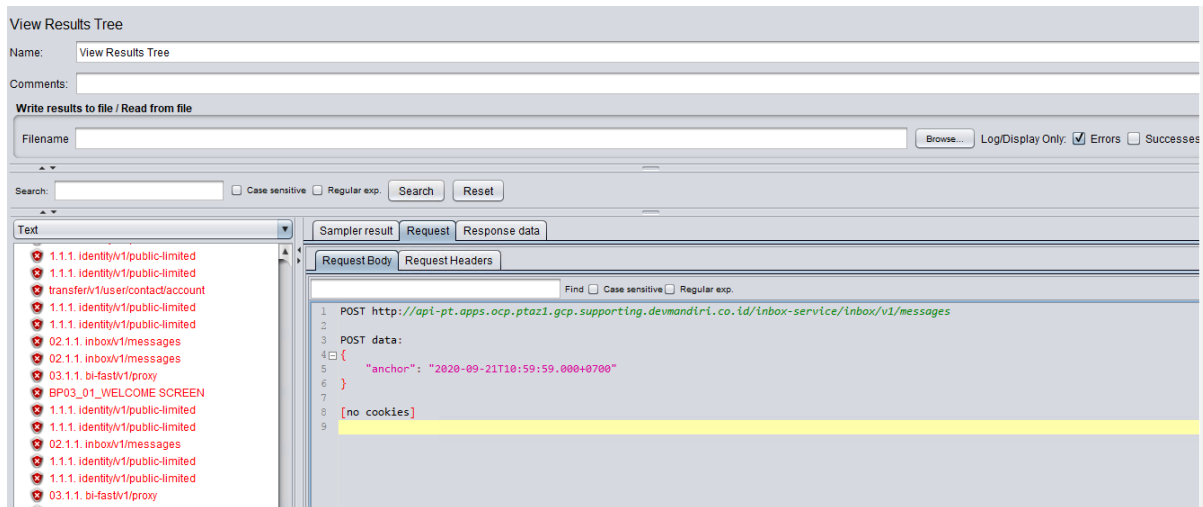


Figure 21 View Result Tree

#### 1.2.5 Lain lain

Selain elemen tadi, ada juga beberapa elemen yang sering digunakan ketika scripting, antara lain

##### 1.2.5.1 Transaction Controller

Transaction controller digunakan untuk membuat transaksi atau biasa disebut screen transition. 1 screen transition biasanya terdiri dari beberapa request, sehingga dapat dimunculkan result average response time, error rate, dan sebagainya dalam 1 transaction yang berisi banyak request. Figure 22 menunjukkan contoh script dengan transaction controller. BP03\_02\_SELECT ACCOUNT AND PROXY adalah sebuah transaction controller dengan 2 request di dalamnya, yaitu casa/v1/account/cached dan bi-fast/v1/add/proxy/verification.

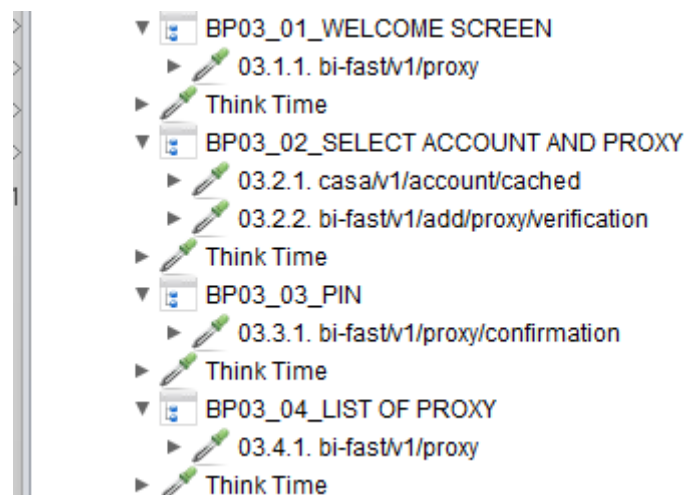


Figure 22 Transaction Controller

##### 1.2.5.2 Throughput Controller

Throughput controller digunakan untuk membagi presentase user yang akan mensimulasikan testing pada request atau transaction tertentu. Dalam scenario testing, seringkali terdapat beberapa

business scenario (BP) dengan load distribution yang berbeda beda, throughput controller digunakan untuk membagi load tiap BP. Figure 23 menunjukkan contoh throughput controller untuk Business Proses Add proxy dengan load distribution sebesar 3%.

- Based on menentukan metode pembagian load berdasarkan prosentase atau total sample yang disimulasikan
- Throughput menyatakan nilai prosentasi atau total sample yang diinginkan

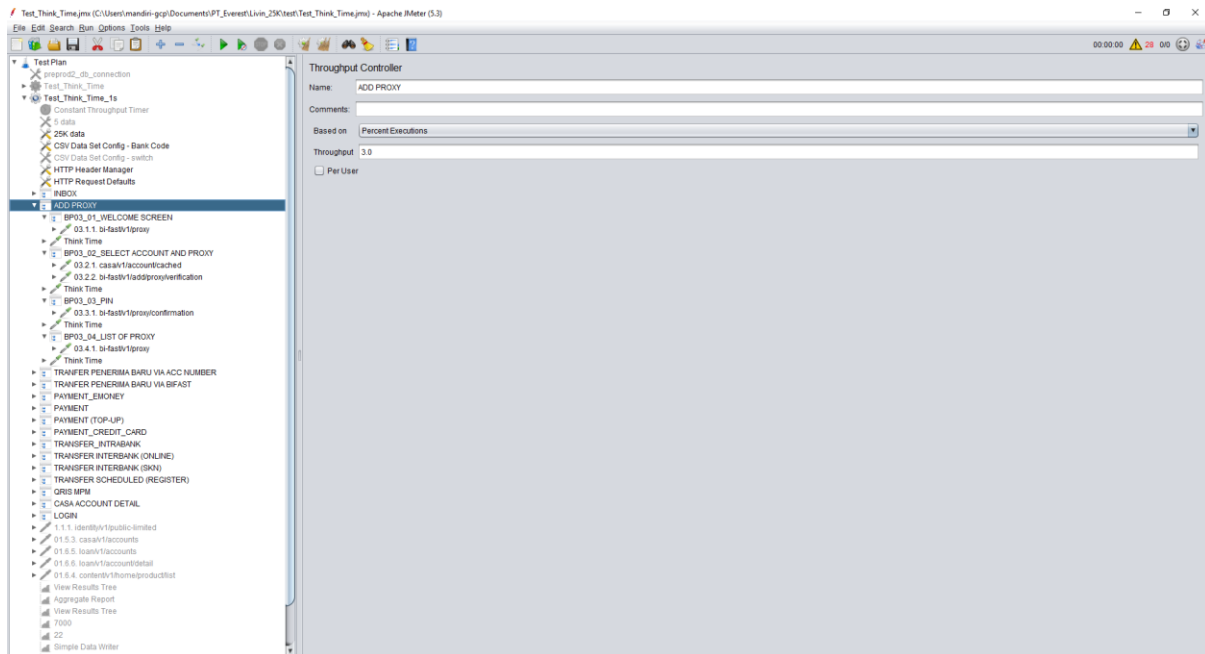


Figure 23 Throughput Controller

### 1.2.5.3 Boundary Extractor

Boundary extractor digunakan untuk mengambil value dari response sebuah request. Dalam testing sering kali terdapat scenario yang mengharuskan sebuah body sebuah request memiliki value yang merupakan response dari request sebelumnya. Untuk itu perlu dipasang boundary extractor pada request yang menghasilkan value yang dibutuhkan.

Figure 24 adalah contoh boundary extractor. Pada scenario tersebut, terdapat api *payment/v1/credit-card/preparation* dengan response body memiliki value untuk key "transactionId". TransactionId ini nantinya akan digunakan di request selanjutnya yaitu *verification/v1/provide* dan *payment/v1/credit-card/execution*. Sehingga dipasang boundary extractor pada *payment/v1/credit-card/preparation* untuk mengambil transactionId.

- Name of created variable : nama variable di mana value tadi disimpan, value dari variable ini dapat dipanggil dengan \${nama variable}
- Left boundary : Batas kiri di mana value diambil dari response
- Right boundary : Batas kanan di mana value diambil dari response

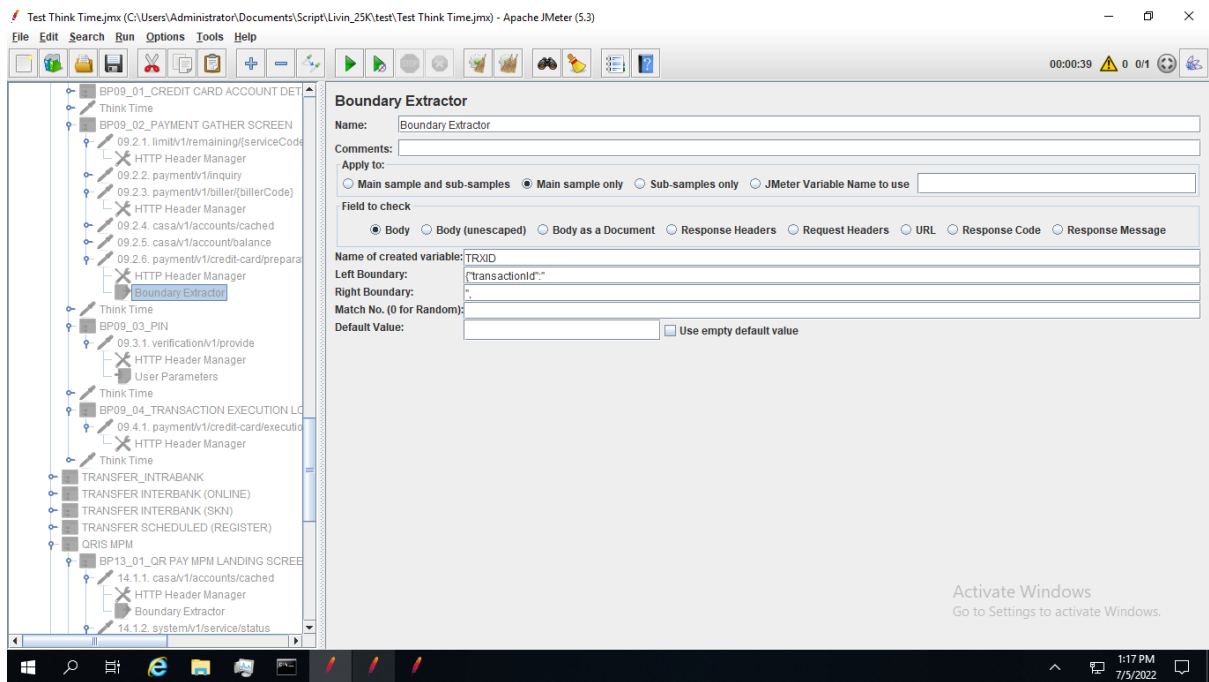


Figure 24 Boundary Extractor

### 1.3 Scripting di Jmeter

Scripting di jmeter dilakukan dengan membuat rangkaian elemen seperti yang terlihat di Figure 23. Setiap BP dibungkus di dalam 1 throughput controller, setiap transaction di dalam BP dibungkus di dalam transaction controller. **Ketika melakukan scripting, ada baiknya menggunakan throughput controller dengan 1 thread (user) untuk 1 iterasi** agar hanya dilakukan satu kali hit untuk setiap api. Running script jmeter dapat dilakukan dengan menekan tombol run yang dilingkari dengan warna merah pada Figure 25. Hasil running dapat dianalisa di aggregate report untuk melihat semua request yang disimulasikan dan result tree untuk melihat request dan response dari setiap HTTP request.



Figure 25 Running scenario in jmeter GUI

### 1.4 Distributed Testing di Jmeter

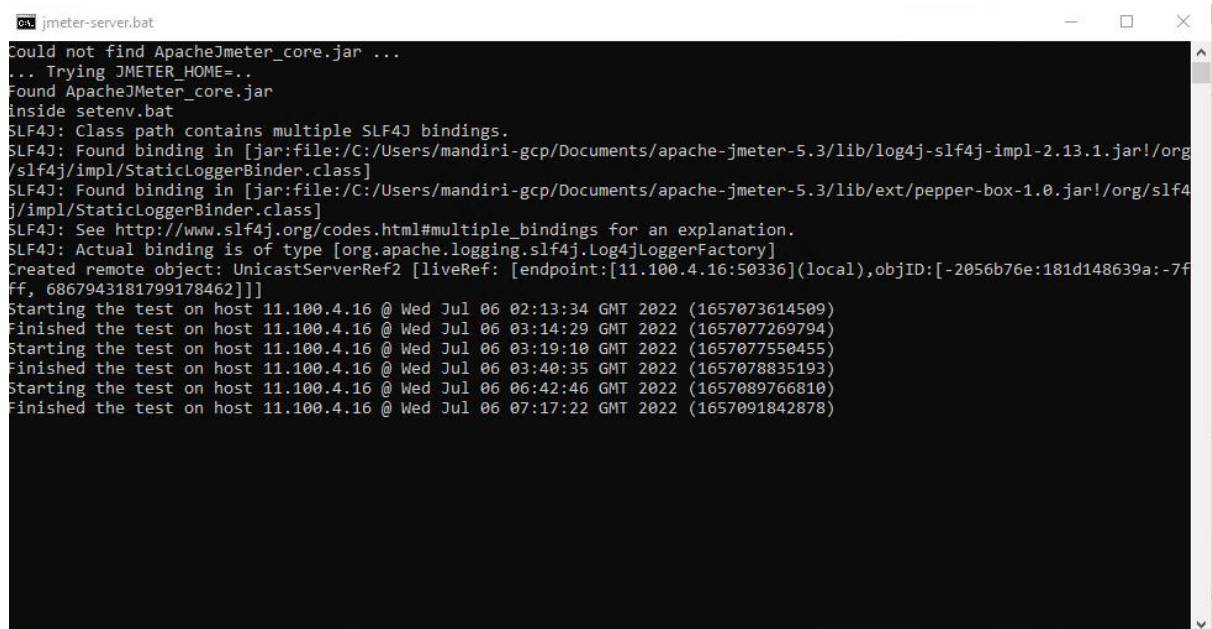
Scenario performance testing di Everest sering kali menggunakan sampai dengan 2000 users sementara 1 jmeter hanya mampu menghandle 300-600 user. Untuk itu dapat digunakan banyak jmeter di server lain untuk membagi beban dari testing agar mampu dicapai testing dengan banyak user. Terdapat 2 istilah utama dalam distributed testing di jmeter

- **Jmeter Master:** adalah jmeter di mana user melakukan perintah untuk running yang nantinya akan memanggil jmeter slave untuk running
- **Jmeter Salve:** adalah server jmeter yang akan melakukan running sesuai perintah dari jmeter master



Untuk melakukan distributed testing, ada konfigurasi yang harus dilakukan baik di server jmeter master maupun jmeter slave

1. Setiap server jmeter slave harus merun file jmeter-server.bat di directory jmeter/bin hingga muncul command prompt seperti pada Figure 26
2. Menambah server slave di file “jmeter.properties” di directory jmeter/bin pada line “remote\_host) seperti pada Figure 27



```
Could not find ApacheJmeter_core.jar ...
... Trying JMeter_HOME=..
Found ApacheJMeter_core.jar
inside setenv.bat
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/Users/mandiri-gcp/Documents/apache-jmeter-5.3/lib/log4j-slf4j-impl-2.13.1.jar!/org
/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/Users/mandiri-gcp/Documents/apache-jmeter-5.3/lib/ext/pepper-box-1.0.jar!/org/slf4
j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Created remote object: UnicastServerRef2 [liveRef: [endpoint:[11.100.4.16:50336](local),objID:[-2056b76e:181d148639a:-7f
ff, 6867943181799178462]]]
Starting the test on host 11.100.4.16 @ Wed Jul 06 02:13:34 GMT 2022 (1657073614509)
Finished the test on host 11.100.4.16 @ Wed Jul 06 03:14:29 GMT 2022 (1657077269794)
Starting the test on host 11.100.4.16 @ Wed Jul 06 03:19:10 GMT 2022 (1657077550455)
Finished the test on host 11.100.4.16 @ Wed Jul 06 03:40:35 GMT 2022 (1657078835193)
Starting the test on host 11.100.4.16 @ Wed Jul 06 06:42:46 GMT 2022 (1657089766810)
Finished the test on host 11.100.4.16 @ Wed Jul 06 07:17:22 GMT 2022 (1657091842878)
```

Figure 26 Jmeter-server.bat while running



## 1.5 Running di Jmeter

Terdapat 2 cara running di jmeter, yaitu melalui GUI dan melalui command prompt.

### 1.5.1 Running Jmeter di GUI

#### 1.5.1.1 Run Jmeter di GUI di master saja

Running di jmeter dapat dilakukan dengan menekan tombol run di Figure 1 yang akan **melakukan running hanya di server master saja**. Waktu running berjalan dan jumlah thread (user) yang sedang melakukan running dapat dilihat di bagian kanan atas GUI jmeter yang ditunjukkan dengan lingkaran merah di Figure 28.

Untuk menghentikan running dapat dilakukan dengan menekan tombol stop yang ditunjukkan dengan lingkaran warna biru di Figure 28.

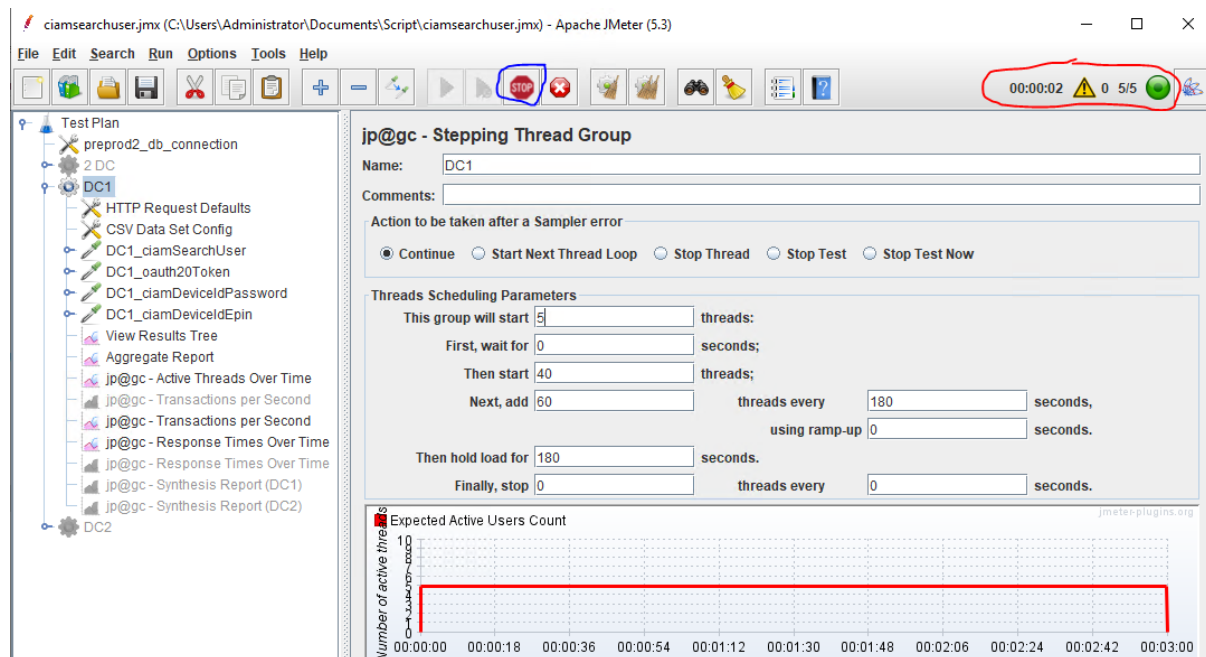


Figure 28 Running Jmeter di GUI

#### 1.5.1.2 Run Jmeter di GUI menggunakan slave

Running di slaves yang sudah diset di jmeter.properties di GUI dilakukan dengan menekan pilihan run pada menu bar, kemudian pilih *remote start all* seperti yang terlihat pada Figure 29. Jmeter akan melakukan running layaknya sebelumnya, namun dengan jumlah thread yang berjalan sesuai dengan jumlah thread yang diinput di thread group dikalikan jumlah slave yang melakukan running.

Untuk menghentikan running, buka menu run di menu bar dan pilih *remote stop all*.

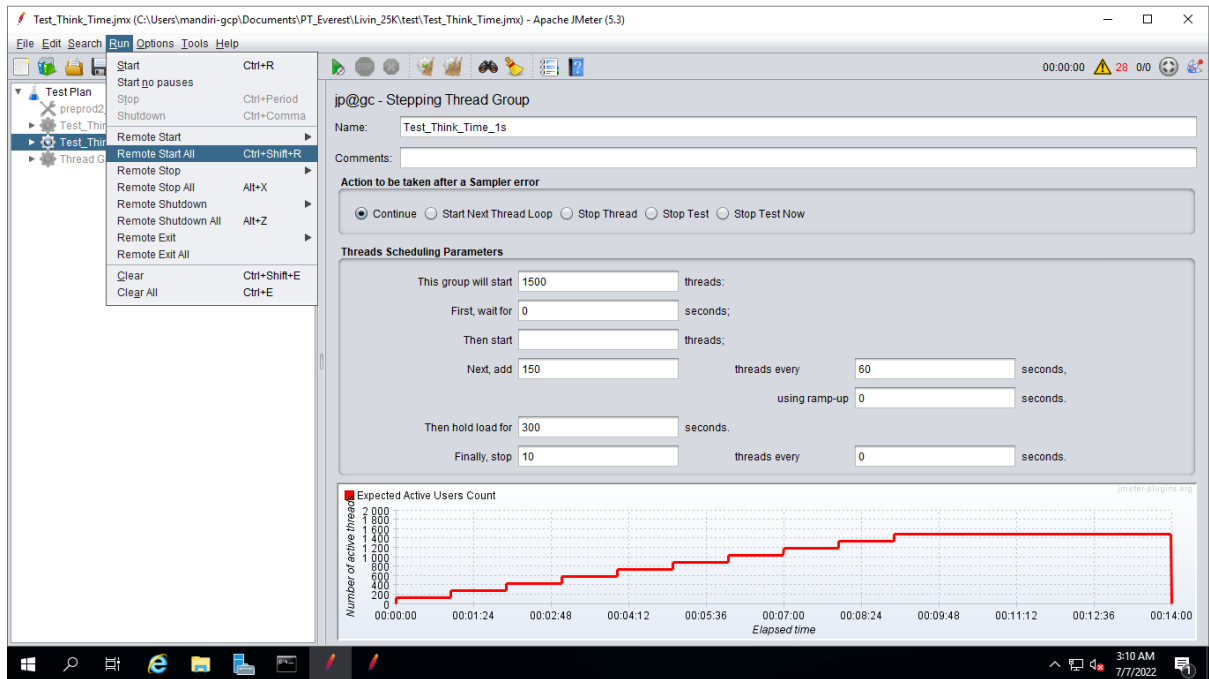


Figure 29 Remote Running Jmeter

### 1.5.2 Running jmeter dari command prompt

Pada umumnya, scenario load test memiliki jumlah user yang cukup besar sehingga tidak memungkinkan dilakukan running pada mode GUI karena jmeter akan crash. Karena itu harus dilakukan running dari command prompt dengan cara:

1. Pastikan sudah tidak ada listener yang menyala di script jmeter, listener bisa dihapus atau di-disable. **Save file jmeter.**
2. Buka command prompt dan masuk ke directory jmeter/bin. Hal ini dapat dilakukan dengan masuk ke file explorer ke directory tadi. Kemudian ketik cmd di box path kemudian enter
3. Masukkan command jmeter berikut:

```
jmeter -n -t {X} -l {Y} -R {Z}
```

- Jmeter : perintah untuk menjalankan aplikasi jmeter
- -n : menyatakan running jmeter di non-GUI mode
- -t : untuk menandai directory file jmeter input (.jmx)
- -l : untuk menandai directory output jmeter (.jtl)
- -R : untuk menandai server slave yang akan digunakan
- {X} : directory file input
- {Y} : directory file output
- {Z} : server slave jmeter yang akan digunakan

Contoh command untuk running jmeter adalah:

```
jmeter -n -t C:\Users\Administrator\Documents\test_payment.jmx -l  
C:\Users\Administrator\Documents\output.jtl -R  
10.243.223.142,10.243.223.143,10.243.223.144
```

4. Ketika running sudah mulai berjalan, di command prompt akan muncul tampilan seperti Figure 30. Selama running akan muncul update status running tiap 30 detik dengan format:

```
summary + 565755 in 00:00:30 = 18856.6/s Avg: 2099 Min: 3 Max: 55627 Err: 5481 (0.97%) Active: 52500 Started: 46249 Finished: 0
summary = 4394581 in 00:04:51 = 15089.0/s Avg: 1619 Min: 3 Max: 59998 Err: 38092 (0.87%)
```

Baris pertama dengan tanda + menyatakan data dalam 30 detik terakhir sementara baris kedua dengan tanda = menyatakan hasil total sampai dengan waktu yang dinyatakan. Data yang ditampilkan di command adalah secara berurutan dari kiri jumlah sample/hits, waktu, throughput (TPS), Average response time, Minimum response time, Maximum Response time, Numbber of error (Error rate), dan jumlah thread yg running.

```
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 19800 in 00:00:21 = 931.9/s Avg: 1529 Min: 9 Max: 15895 Err: 0 (0.00%) Active: 7500 Started: 1249 Finished: 0
summary + 211998 in 00:00:30 = 7065.0/s Avg: 671 Min: 5 Max: 30681 Err: 250 (0.12%) Active: 15000 Started: 8749 Finished: 0
summary + 231798 in 00:00:51 = 4522.4/s Avg: 744 Min: 5 Max: 30681 Err: 250 (0.11%) Active: 15000 Started: 8749 Finished: 0
summary + 173392 in 00:00:30 = 5782.6/s Avg: 160 Min: 4 Max: 54999 Err: 256 (0.15%) Active: 22500 Started: 16249 Finished: 0
summary + 405190 in 00:01:21 = 4987.6/s Avg: 494 Min: 4 Max: 54999 Err: 506 (0.12%) Active: 22500 Started: 16249 Finished: 0
summary + 457201 in 00:00:30 = 15235.0/s Avg: 1875 Min: 4 Max: 55396 Err: 9808 (2.15%) Active: 30000 Started: 23749 Finished: 0
summary + 862391 in 00:01:51 = 7751.8/s Avg: 1226 Min: 4 Max: 55396 Err: 10314 (1.20%) Active: 30000 Started: 23749 Finished: 0
summary + 588924 in 00:00:30 = 19630.8/s Avg: 1195 Min: 4 Max: 55427 Err: 2898 (0.49%) Active: 37500 Started: 31249 Finished: 0
summary + 1451315 in 00:02:21 = 10274.8/s Avg: 1214 Min: 4 Max: 55427 Err: 13212 (0.91%) Active: 37500 Started: 31249 Finished: 0
summary + 602185 in 00:00:30 = 20074.2/s Avg: 1312 Min: 4 Max: 59998 Err: 2301 (0.38%) Active: 45000 Started: 38749 Finished: 0
summary + 2053500 in 00:02:51 = 11991.4/s Avg: 1242 Min: 4 Max: 59998 Err: 15513 (0.76%) Active: 45000 Started: 38749 Finished: 0
summary + 589371 in 00:00:30 = 19647.0/s Avg: 1497 Min: 3 Max: 55719 Err: 3377 (0.57%) Active: 52500 Started: 46249 Finished: 0
summary + 2642871 in 00:03:21 = 13132.5/s Avg: 1299 Min: 3 Max: 59998 Err: 18890 (0.71%) Active: 52500 Started: 46249 Finished: 0
summary + 605534 in 00:00:30 = 20183.8/s Avg: 2078 Min: 3 Max: 55536 Err: 7007 (1.16%) Active: 52500 Started: 46249 Finished: 0
summary + 3248405 in 00:03:51 = 14047.3/s Avg: 1444 Min: 3 Max: 59998 Err: 25897 (0.80%) Active: 52500 Started: 46249 Finished: 0
```

Figure 30 Run Jmeter via Command Propmt

- Running akan berjalan sesuai dengan waktu yang ditentukan di script atau sampai running dihentikan secara paksa. Untuk menghentikan secara paksa, buka file stoptest.cmd yang ada di directory jmeter/bin dan akan muncul layar seperti pada Figure 31.

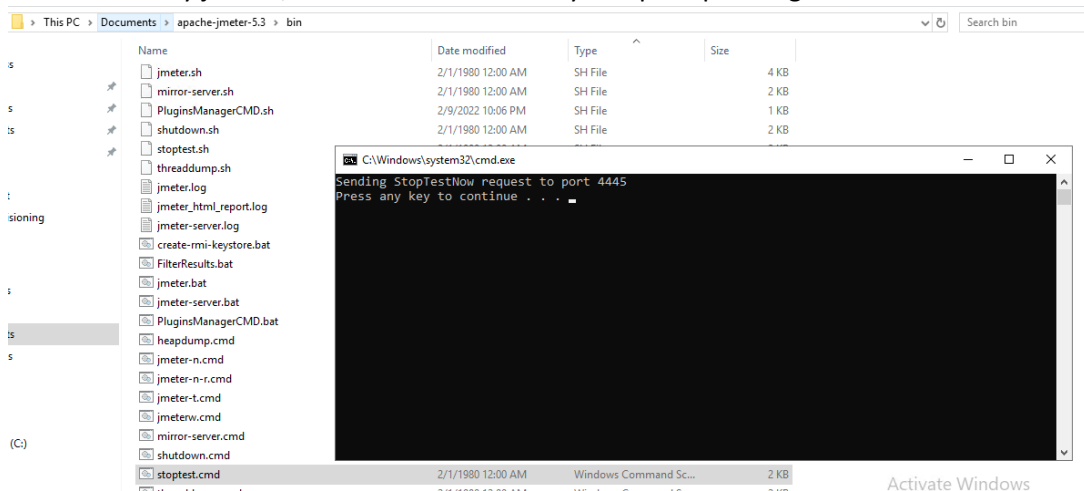


Figure 31 Stoptest.cmd in Jmeter

- Sewaktu running sudah selesai, thread akan tertutup satu per satu sampai dengan semua thread habis dan running selesai. **Apabila waktu sudah lebih dari durasi running namun running belum selesai, sebaiknya running dihentikan secara paksa agar nilai output tidak terbias karena durasi run yang lama.**
- Apabila run selesai akan muncul tulisan ... end of run seperti pada Figure 32, akan muncul file output di directory yang ditentukan sebelumnya.

```
Tidying up remote @ Thu Jul 07 02:29:11 GMT 2022 (1657160951469)
... end of run
```

Figure 32 End of Run in Jmeter non-GUI running

