



SÃO  
PAULO  
TECH  
SCHOOL

# Computação e sistemas distribuídos

## Testes de performance

**Eduardo Verri, Diego Brito**

[eduardo.verri@sptech.school](mailto:eduardo.verri@sptech.school)

[diego.brito@sptech.school](mailto:diego.brito@sptech.school)

**Sobre os testes**

# Uma introdução

- ❑ Tem como objetivo avaliar a capacidade de um sistema ou componente em atender determinados requisitos de desempenho, como tempo de resposta, capacidade de processamento, uso de memória, entre outros.
- ❑ Ele tem um papel importante em determinar níveis de qualidade adequados de desempenho para uma boa experiência do usuário, independentemente da plataforma.
- ❑ Não se limita ao domínio baseado na Web, onde temos o usuário final como foco, sendo também relevante para aplicações com uma variedade de arquiteturas de sistemas, como o cliente-servidor, distribuído e embarcado.
- ❑ São um conjunto de testes não funcionais realizados para determinar como um sistema funciona em termos de capacidade de resposta e estabilidade sob uma determinado carga de trabalho simulada

# Alguns tipos de teste de performance

- ❑ **Load Testing:** envolve submeter um sistema de software a vários níveis de carga do usuário para avaliar seus tempos de resposta, utilização de recursos e desempenho geral sob diferentes cenários de uso. Ele ajuda a identificar gargalos de desempenho, como tempos de resposta lentos ou travamentos do servidor, que podem surgir à medida que o tráfego do usuário aumenta.



- ❑ **Stress Testing:** levam o aplicativo ao seu limite ou além. Ao aplicar cargas extremas, o teste de estresse revela como o sistema se comporta quando os recursos são escassos, as conexões de banco de dados são saturados ou os componentes de hardware estão sob tensão. Essa metodologia revela potenciais pontos de falha e ajuda a avaliar mecanismos de recuperação do sistema.



# Alguns tipos de teste de performance

- ❑ **Volume Testing:** concentra-se na avaliação do desempenho do sistema ao lidar com grandes volumes de dados. Isto ajuda a identificar problemas de desempenho do banco de dados, manipulação de dados gargalos e possíveis problemas de corrupção de dados que podem surgir à medida que os dados aumentam.
- ❑ **Endurance Testing:** o teste de resistência, também conhecido como teste de absorção, envolve sujeitar uma aplicação a uma carga sustentada por um período prolongado. Esta metodologia ajuda a descobrir vazamentos de memória, esgotamento de recursos e outras degradações de desempenho ou problemas que só podem surgir após uso prolongado.



# Alguns tipos de teste de performance

- ❑ **Spike Testing:** o teste de pico examina a resposta do aplicativo a aumentos extremos na carga de usuários. Simula cenários onde o tráfego do usuário aumenta inesperadamente, como pode ocorrer durante o lançamento de um produto ou evento de notícias de última hora. Essa metodologia avalia quão bem o sistema lida com mudanças rápidas e flutuações na demanda.



- ❑ **Concurrency Testing:** o teste de simultaneidade avalia a capacidade do sistema de lidar com vários usuários ou transações simultâneas de forma eficaz. Isto ajuda a identificar problemas relacionados à integridade de dados, recursos, contenção e sincronização que podem surgir em ambientes de vários usuários.



# Bottleneck – Gargalo



Um gargalo é um fenômeno onde o desempenho ou capacidade de um sistema inteiro é limitado por um único ou número limitado de componentes ou recursos.

Sua aplicação pode consistir em vários módulos usados para processar a solicitação. Se um deles tem limitação técnica, limita o desempenho de todo o sistema.

O gargalo no aplicativo pode ser identificado realizando o teste de carga com a carga de usuário simultânea definida para vários cenários



**Planejando os testes**

# Recomendações para testes de desempenho

## ❑ Preparar o teste

- Definir critérios de aceitação
- Selecionar o tipo de teste
- Selecionar ferramentas de teste
- Criar cenários de teste
- Configurar o ambiente de teste

## ❑ Executar os testes

## ❑ Analisar os resultados

## ❑ Estabelecer linhas de base

## ❑ Testar continuamente

[Recomendações para teste de desempenho - Microsoft Azure Well-Architected Framework | Microsoft Learn](#)

# Planejamento básico

Para determinar a ferramenta e a configuração corretas para o teste de carga, deixe claro por que você está executando o teste. As seguintes perguntas devem ser respondidas com o tipo certo de teste:

- ☐ Quanta carga minha aplicação pode comportar?
- ☐ Minha aplicação pode lidar com a carga X?
- ☐ A minha aplicação pode aumentar ou reduzir a escala verticalmente?
- ☐ O comportamento da minha aplicação se degrada com o tempo com uma quantidade X de carga?
- ☐ Minha aplicação está funcionando?

[Planejamento básico para testes de carga - AWS Orientação prescritiva \(amazon.com\)](#)

## Cuidado!

- ❑ Executar testes de carga na Amazon Web Services (AWS) pode iniciar mecanismos de segurança.
- ❑ O teste de penetração só pode ser executado em serviços da AWS permitidos.
- ❑ O teste de negação de serviços distribuída (DDoS) deve ser realizado por um AWS Partner pré-aprovado.
- ❑ Considere os custos que os testes de carga gerarão e as cotas de serviço para seus serviços. Seus requisitos de teste podem exceder o limite configurado pela AWS para cada serviço

[AWS Orientação prescritiva - Aplicativos de teste de carga \(amazon.com\)](#)

[Ferramentas que podem ser usadas - AWS Orientação prescritiva \(amazon.com\)](#)

[Amazon EC2 Testing Policy](#)

# Algumas ferramentas de Load Testing

- ❑ **Apache JMeter: open-source**
- ❑ Gatling: open-source
- ❑ LoadRunner: comercial
- ❑ Locust: open-source baseado em python
- ❑ K6: open-source, voltado para UI
- ❑ WebLOAD: comercial
- ❑ NeoLoad: comercial
- ❑ BlazeMeter: comercial, voltado para Mobile
- ❑ LoadView: comercial, voltado para streaming
- ❑ Flood: open-source http load test
- ❑ Zoho Qengine: comercial
- ❑ The Grinder: open-source
- ❑ WebServer Stress Tool: open-source

# Benefícios do Apache JMeter



JMeter é open-source e oferece uma ampla gama de recursos de teste de desempenho. É um aplicativo de desktop baseado em Java que permite testar aplicativos cliente-servidor, como bancos de dados, servidores FTP, sites, serviços da web, etc. Alguns dos tipos comuns de aplicativos que podem ser testados usando JMeter incluem:

- ❑ Database Servers
- ❑ FTP Servers
- ❑ LDAP Servers
- ❑ Mail Servers – SMTP, POP3, IMAP
- ❑ Shell Scripts
- ❑ TCP Servers
- ❑ Websites – HTTP and HTTPS
- ❑ Web Services – REST and SOAP

# Benefícios do Apache JMeter



- ❑ Design de GUI amigável ao usuário em comparação com outras ferramentas
- ❑ A estrutura multithreading completa permite a amostragem simultânea por muitos threads e amostragem simultânea de funções diferentes por grupos de threads separados
- ❑ Cache e Cookies podem ser habilitados e impactam como estão sendo executados no navegador
- ❑ Os controladores são configuráveis e podem monitorar o desempenho do servidor
- ❑ Os resultados dos testes são mais confiáveis em comparação com outras ferramentas de código aberto
- ❑ Os resultados do teste podem ser capturados em vários formatos, como relatório de resumo, gráfico, agregado relatório, gráfico agregado, resultados na árvore e resultados na tabela

# Benefícios do Apache JMeter



Para a elaboração dos planos de teste, o JMeter pode te ajudar em:

- ❑ Configurar diversos tipos de requisições
- ❑ Criar loops e condições lógicas para cada requisição
- ❑ Importar dados para o plano através de arquivos csv (usuários, senhas)
- ❑ Configurar paralelismo através do número de threads, a quantidade de execução de cada thread e o intervalo entre cada uma
- ❑ Criar testes mais eficientes simulando múltiplos usuários e requisições independentes
- ❑ Simular um ataque ao seu servidor
- ❑ Apresentar os resultados do teste de várias formas (em árvore, tabela, gráficos, etc.)







**Hands on JMeter [GUI]**

# Instalando o JMeter

- ❑ Instale o Java (Necessita Java 8+)
- ❑ Download JMeter: [Apache JMeter - Download Apache JMeter](#)
- ❑ Baixe o arquivo de binários (zip ou tgz)
- ❑ Extrair pasta zipada
- ❑ Acessar `/bin`
- ❑ Executar `java -jar ApacheJMeter.jar`

[Apache JMeter - User's Manual: Getting Started](#)





## Download Apache JMeter

We recommend you use a mirror to download our release builds, but you **must** *verify the integrity* of the downloaded files using signatures downloaded from our main distribution directories. Recent releases (48 hours) may not yet be available from all the mirrors.

You are currently using <https://dlcdn.apache.org/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup* mirrors (at the end of the mirrors list) that should be available.

Other mirrors:

The **KEYS** link links to the code signing keys used to sign the product. The **PGP** link downloads the OpenPGP compatible signature from our main site. The **SHA-512** link downloads the sha512 checksum from the main site. Please *verify the integrity* of the downloaded file.

For more information concerning Apache JMeter, see the [Apache JMeter](#) site.

**KEYS**

## Apache JMeter 5.6.3 (Requires Java 8+)

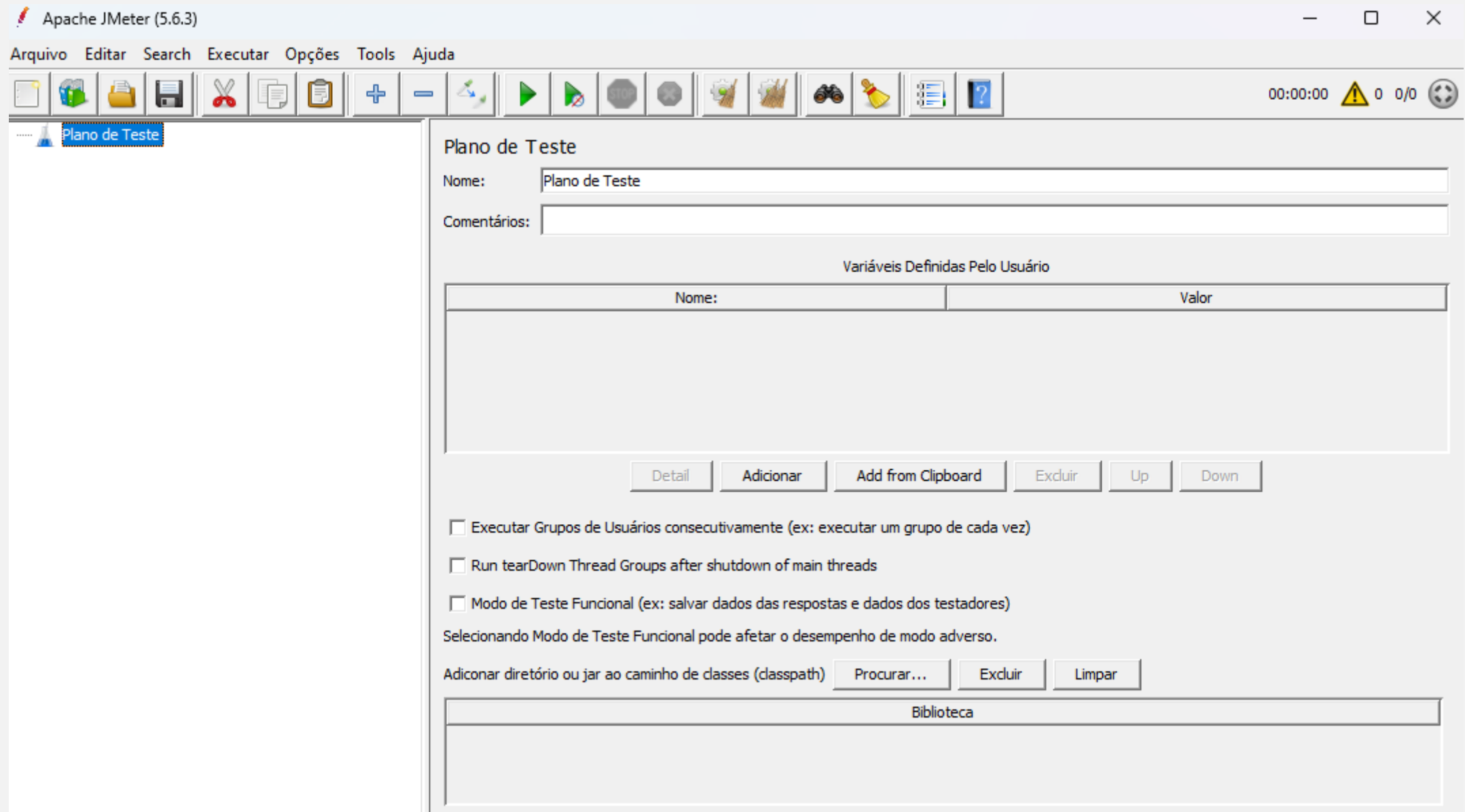
### Binaries

<a href="#">apache-jmeter-5.6.3.tgz sha512 pgp</a>
<a href="#">apache-jmeter-5.6.3.zip sha512 pgp</a>

# Interface JMeter

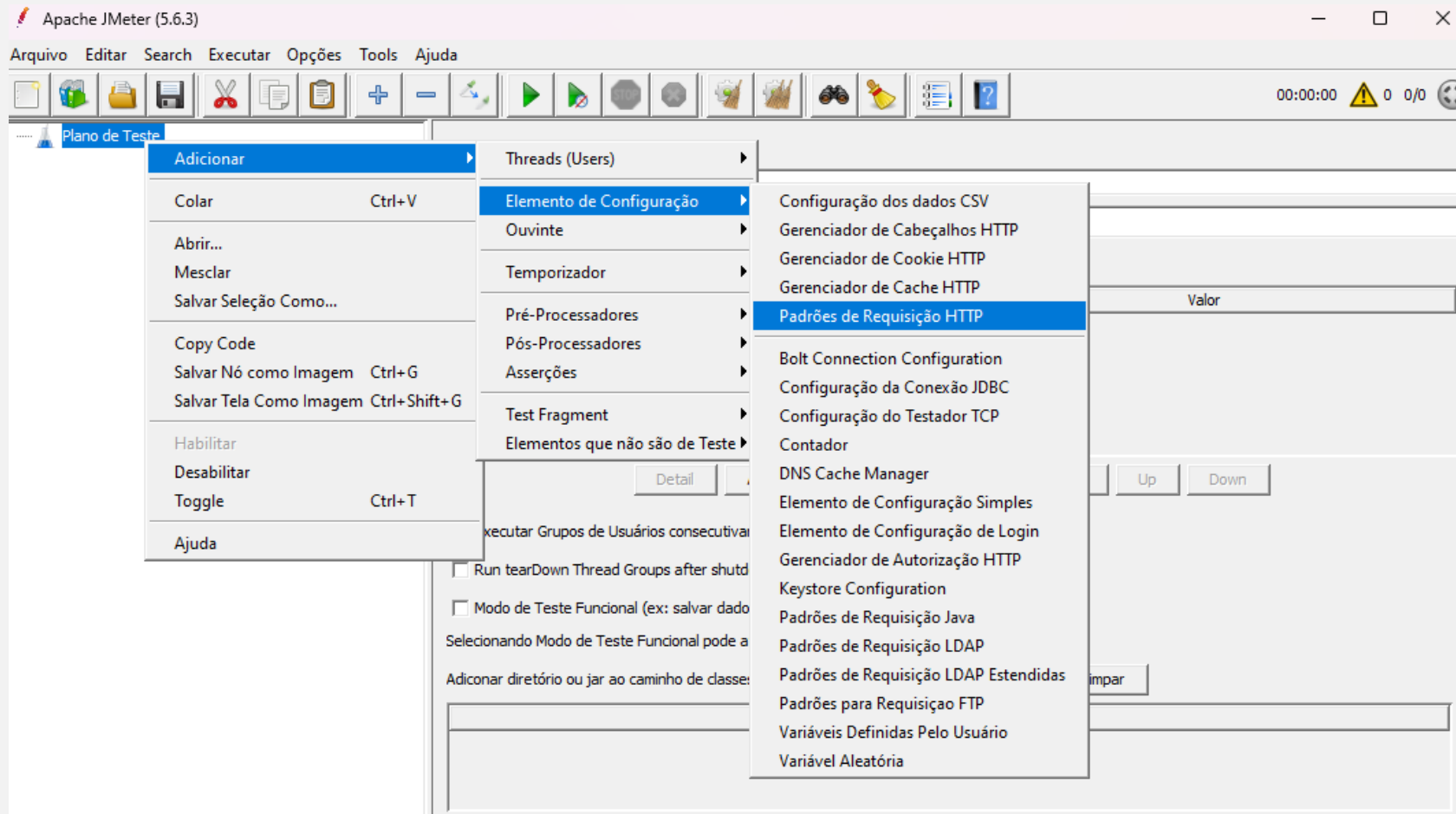
```
Windows PowerShell
PS C:\Users\Eduardo Verri\Desktop\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin
> java -jar .\ApacheJMeter.jar
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
=====
===
Don't use GUI mode for load testing !, only for Test creation and Test debugging.
For load testing, use CLI Mode (was NON GUI):
    jmeter -n -t [jmx file] -l [results file] -e -o [Path to web report folder]
& increase Java Heap to meet your test requirements:
    Modify current env variable HEAP="-Xms1g -Xmx1g -XX:MaxMetaspaceSize=256m"
    in the jmeter batch file
Check : https://jmeter.apache.org/usermanual/best-practices.html
=====
===
```

# Interface JMeter



# Adicionando elementos de configuração

O primeiro elemento que iremos adicionar é o Padrões de Requisição HTTP (HTTP Request Defaults), que define valores padrões para as requisições HTTP. Para adicionarmos o elemento, basta clicar com o botão direito do mouse em cima do Plano de Teste em seguida em **Adicionar > Elementos de configuração > Padrões de Requisição HTTP**.



# Padrões de Requisição HTTP [HTTP Request Defaults]

Neste elemento de configuração podemos definir os valores padrões das requisições HTTP que serão adicionadas posteriormente ao teste, como o endereço e a porta. Com este elemento, não é preciso repetir as informações que são padrão em todos os elementos de requisição HTTP

## Padrões de Requisição HTTP

Nome:

Comentários:

Basic | **Advanced**

### Servidor Web

Protocolo [http]:  Nome do Servidor ou IP:  Número da Porta:

### Requisição HTTP

Caminho:  Codificação do conteúdo:

Parameters | **Body Data**

# Grupo de usuários [Thread Group]

O thread group é o início de qualquer plano de teste, é abaixo dele que ficarão todos os controladores e testadores do plano. Nele podemos estabelecer configurações como: número de usuários (threads), tempo de intervalo para cada usuário (ramp-up period) e o número de vezes que o teste será executado (loop count)

## Grupo de Usuários

Nome:

Comentários:

### Ação a ser tomada depois de erro do testador

☒ Continuar   ☐ Start Next Thread Loop   ☐ Interromper Usuário Virtual   ☐ Interromper Teste   ☐ Interrompe Teste Agora

### Propriedades do Usuário Virtual

Número de Usuários Virtuais (threads):

Tempo de inicialização (em segundos)

Contador de Iteração   ☐ Infinito

# O que é Ramping Up no JMeter

**Ramping Up:** é a quantidade de tempo que o Apache JMeter™ levará para adicionar todos os usuários de teste (threads) à execução do teste – ou em outras palavras, quanto tempo levará para o JMeter iniciar a execução de todos os threads. Por exemplo:

- ❑ 1.000 threads de destino com 1.000 segundos de aceleração: o JMeter adicionará um usuário a cada segundo.
- ❑ 1.000 threads de destino com aceleração de 100 segundos: o JMeter adicionará 10 usuários a cada segundo.
- ❑ 1.000 threads de destino com aceleração de 50 segundos: o JMeter adicionará 20 usuários a cada segundo.

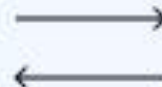
[Apache JMeter - User's Manual: Elements of a Test Plan](#)

[JMeter Ramp-Up: The Ultimate Guide - DZone](#)



# O que é Ramping Up no JMeter

**Principal:** sistema rodando JMeter GUI/CLI que controla o teste



User 1

User 2

User N

SECUNDÁRIO 1

Send Request  
HTTP Response

SECUNDÁRIO 2

Send Request  
HTTP Response

SECUNDÁRIO 3

Send Request  
HTTP Response

**Secundário:** sistema rodando JMeter-server que recebe comandos da GUI/CLI e envia as requisições para o sistema alvo

Return

Return

Return



Server

**Alvo:** o servidor Web planejado para o teste de carga

# Configuração de Conjunto de Dados CSV [CSV Data Set Config]

Este elemento de configuração permite a leitura de arquivos CSV e a atribuição desses dados em variáveis do JMeter que podem ser utilizadas durante o teste, como dados de login de usuários. O arquivo txt deve estar salvo no mesmo local que o arquivo do plano de teste

## Configuração dos dados CSV

Nome: Configuração dos dados CSV

Comentários:

### Configurar fonte de dados CSV

Nome do arquivo: accounts.txt

Codificação do arquivo (encoding):

Nomes das variáveis (separados por vírgula): USER,PASSWORD

Separador (usar '\t' para tabulações): ,

Permitir dados com citações?: False

Reciclar no final do arquivo (EOF)? True

Finalizar usuário virtual no final do arquivo?: False

Modo de compartilhamento: Todos os usuários virtuais

# Requisição HTTP [HTTP Request]

A requisição HTTP é o elemento principal de um plano de teste para um sistema web, podendo existir um ou vários no mesmo plano. Nele podemos definir o verbo que será utilizado (POST, GET, PUT, etc.), enviar parâmetros com a requisição

## Requisição HTTP

Nome:	<input type="text" value="Requisição HTTP"/>
Comentários:	<input type="text"/>

Basic | Advanced

### Servidor Web

Protocolo [http]:	<input type="text" value="http"/>	Nome do Servidor ou IP:	<input type="text" value="100.25.104.130"/>	Número da Porta:	<input type="text" value="80"/>
-------------------	-----------------------------------	-------------------------	---	------------------	---------------------------------

### Requisição HTTP

<input type="text" value="GET"/>	Caminho:	<input type="text" value="/"/>	Codificação do conteúdo:	<input type="text" value="UTF-8"/>
----------------------------------	----------	--------------------------------	--------------------------	------------------------------------

☒ Redirecionar automaticamente   ☐ Seguir redireções   ☒ Usar Manter Ativo (KeepAlive)   ☐ Usar multipart/form-data para HTTP POST   ☐ Browser-compatible headers

# Atributos para Configuração

- ❑ **Nome (Name):** Use o nome para lembrar qual a responsabilidade do elemento. Exemplo: “Pedir novo email de senha” ou “fazer login”.
- ❑ **Nome do Servidor ou IP (Server Name or IP):** Endereço ou IP do site que será testado, por exemplo: meusite.com.br. Caso você não tenha configurado o valor default anteriormente nos Padrões de Requisição HTTP, o HTTP Request irá utilizar a informação deste elemento.
- ❑ **Número da Porta (Port Number):** Aqui deve ser especificado a porta que o servidor responde, caso não seja a porta 80, como por exemplo quando rodamos o server de uma api a porta poderia ser alterada para 3000.
- ❑ **Protocolo (Protocol):** O protocolo default é o HTTP, mas pode ser sobreescrito caso seja necessário o uso do protocolo HTTPS.

# Atributos para Configuração

- ❑ **Método (Method):** Caso a requisição esteja realizando alguma solicitação, o protocolo GET deve ser mantido, mas se você estiver enviando dados possivelmente vai precisar de um POST ou PUT, ou se estiver apagando, um DELETE.
- ❑ **Codificação do Conteúdo (Content Encoding):** Esse atributo estabelece o formato de codificação dos dados que são enviados junto da requisição, por default esse formato é o UTF-8, mas pode ser alterado conforme suas necessidades.
- ❑ **Caminho (Path):** O caminho é a página que será acessada, complementando o nome do servidor ou IP. Por exemplo em uma requisição de POST para efetuar um login, poderíamos estabelecer o caminho para /login

# Atributos para Configuração

- ❑ **Enviar Parâmetros com a Requisição (Send parameters with request):** Aqui podem ser adicionados os parâmetros que devem ser enviados juntos do request, especificando o nome e seu respectivo valor, por exemplo onde a requisição efetua o login de um usuário através de um POST enviando o email e senha, além do token para autenticação do request. Os valores nessa requisição são de variáveis do JMeter criadas durante a execução do plano, como USER e PASSWORD que foram criadas naquele elemento de configuração para Dados CSV, que adicionamos anteriormente para fazer a leitura de um arquivo txt.

Requisição HTTP

Implementação:  Protocolo [http]:  Método: POST Codificação do conteúdo:

Caminho: /login

☐ Redirecionar automaticamente ☒ Seguir redireções ☒ Usar Manter Ativo (KeepAlive) ☐ Usar multipart/form-data para H

**Parameters** Post Body

Enviar Parâmetros Com a Requisição

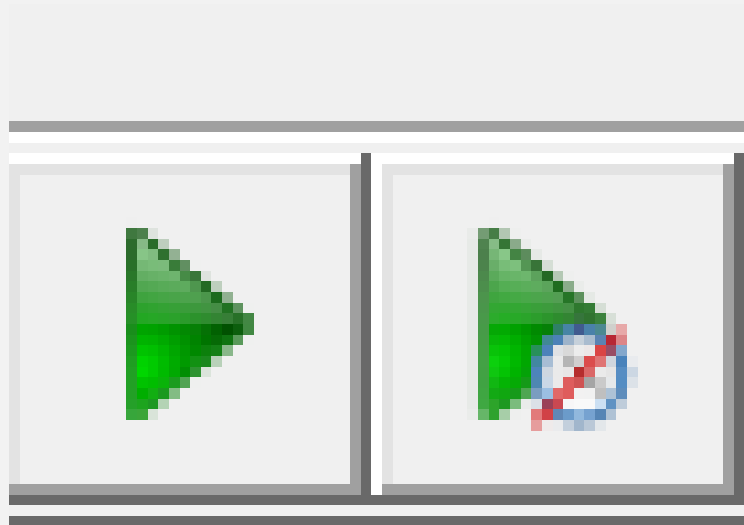
Nome:	Valor
user[email]	\${USER}
user[password]	\${PASSWORD}
authenticity_token	\${TOKEN}

Detail Adicionar Add from Clipboard Excluir Up Down

# Formato para visualização de Resultados

O elemento para visualização dos resultados pode ser adicionado clicando com o botão direito no Plano de Teste em seguida Adicionar > Ouvinte > Ver Resultados em Tabela ou Ver Árvore de Resultados. As informações em cada elemento podem ser apresentadas como Tabela ou Árvore

Agora só salvar o arquivo e mandar rodar!



# Árvore de Resultado

teste\_basico.jmx (C:\Users\Eduardo Verri\Desktop\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\teste\_basico.jmx) - Apache JMeter (5.6.3)

Arquivo Editar Search Executar Opções Tools Ajuda

00:00:20 0 0/200

teste\_basico

- Padrões de Requisição HTTP
- Grupo de Usuários
  - Requisição HTTP
  - Ver Árvore de Resultados**
  - Relatório de Sumário
  - Relatório Agregado
  - Ver Resultados em Tabela

### Ver Árvore de Resultados

Nome: Ver Árvore de Resultados

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo  Procurar... Apenas Logar/Exibir ☐ Erros ☐ Sucessos Configurar

Search:  ☐ Case sensitive ☐ Regular exp. Search Reset

Exibir Texto

	Resultados do testador	Requisição	Dados da resposta
✓ Requisição HTTP	Nome do Usuário Virtual: Grupo de Usuários 1-1		
✓ Requisição HTTP	Início da Amostra: 2024-05-08 01:14:51 BRT		
✓ Requisição HTTP	Tempo de Carga: 295		
✓ Requisição HTTP	Connect Time: 149		
✓ Requisição HTTP	Latência: 295		
✓ Requisição HTTP	Tamanho em bytes: 548		
✓ Requisição HTTP	Sent bytes: 119		
✓ Requisição HTTP	Headers size in bytes: 238		
✓ Requisição HTTP	Body size in bytes: 310		
✓ Requisição HTTP	Contagem de amostras: 1		
✓ Requisição HTTP	Contador de Erros: 0		
✓ Requisição HTTP	Data type ("text" "bin" ""): text		
✓ Requisição HTTP	Código de Resposta: 200		
✓ Requisição HTTP	Mensagem de resposta: OK		
✓ Requisição HTTP	HTTPSampleResult campos:		
✓ Requisição HTTP	ContentType: text/html		
✓ Requisição HTTP	DataEncoding: null		



# Resultado em tabela

teste\_basico.jmx (C:\Users\Eduardo Verr\Deskto\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\teste\_basico.jmx) - Apache JMeter (5.6.3)

Arquivo Editar Search Executar Opções Tools Ajuda



00:00:20 0 0/200

- teste\_basico
  - Padrões de Requisição HTTP
  - Grupo de Usuários
    - Requisição HTTP
    - Ver Árvore de Resultados
    - Relatório de Sumário
    - Ver Resultados em Tabela

## Ver Resultados em Tabela

Nome: Ver Resultados em Tabela

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo

Procurar...

Apenas Logar/Exibir

☐ Erros

☐ Sucessos

Configurar

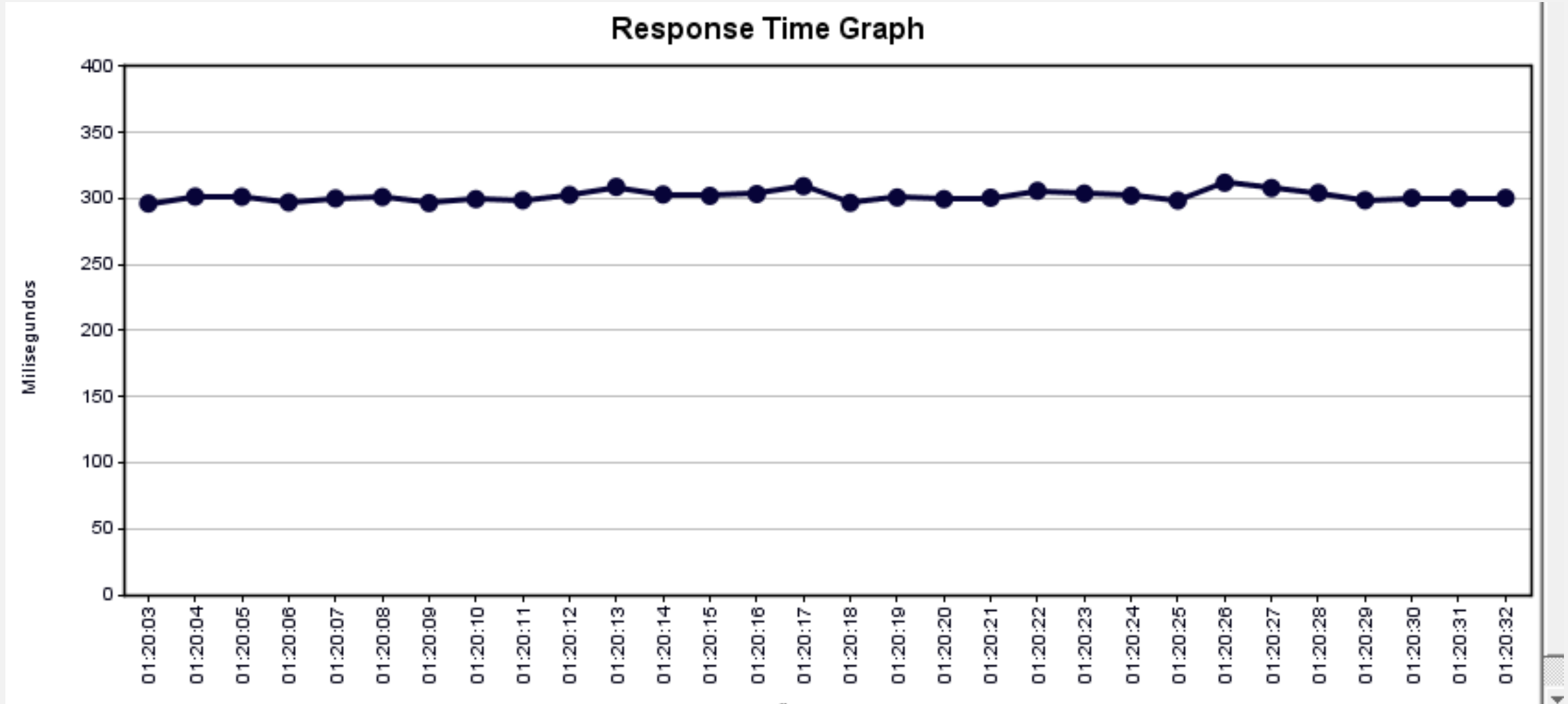
Amostra #	Tempo de início	Nome do Usu...	Rótulo	Tempo da am...	Estado	Bytes	Sent Bytes	Latency	Connect Tim...
1	01:14:51.566	Grupo de Usu...	Requisição H...	295	✓	548	119	295	149
2	01:14:51.670	Grupo de Usu...	Requisição H...	313	✓	543	119	312	163
3	01:14:51.766	Grupo de Usu...	Requisição H...	306	✓	548	119	306	149
4	01:14:51.876	Grupo de Usu...	Requisição H...	294	✓	543	119	294	146
5	01:14:51.967	Grupo de Usu...	Requisição H...	302	✓	543	119	302	151
6	01:14:52.074	Grupo de Usu...	Requisição H...	308	✓	548	119	308	153
7	01:14:52.180	Grupo de Usu...	Requisição H...	300	✓	543	119	300	154
8	01:14:52.272	Grupo de Usu...	Requisição H...	305	✓	548	119	305	152
9	01:14:52.382	Grupo de Usu...	Requisição H...	296	✓	543	119	296	149
10	01:14:52.473	Grupo de Usu...	Requisição H...	305	✓	548	119	305	152
11	01:14:52.582	Grupo de Usu...	Requisição H...	304	✓	543	119	304	150
12	01:14:52.675	Grupo de Usu...	Requisição H...	306	✓	548	119	306	152
13	01:14:52.769	Grupo de Usu...	Requisição H...	299	✓	543	119	299	151
14	01:14:52.874	Grupo de Usu...	Requisição H...	297	✓	548	119	297	150
15	01:14:52.967	Grupo de Usu...	Requisição H...	301	✓	543	119	301	154
16	01:14:53.074	Grupo de Usu...	Requisição H...	296	✓	548	119	296	149
17	01:14:53.165	Grupo de Usu...	Requisição H...	303	✓	543	119	303	149
18	01:14:53.259	Grupo de Usu...	Requisição H...	293	✓	548	119	293	146
19	01:14:53.368	Grupo de Usu...	Requisição H...	297	✓	543	119	297	149
20	01:14:53.477	Grupo de Usu...	Requisição H...	293	✓	548	119	293	144
21	01:14:53.569	Grupo de Usu...	Requisição H...	300	✓	548	119	300	154

# Acompanhamento Docker compose

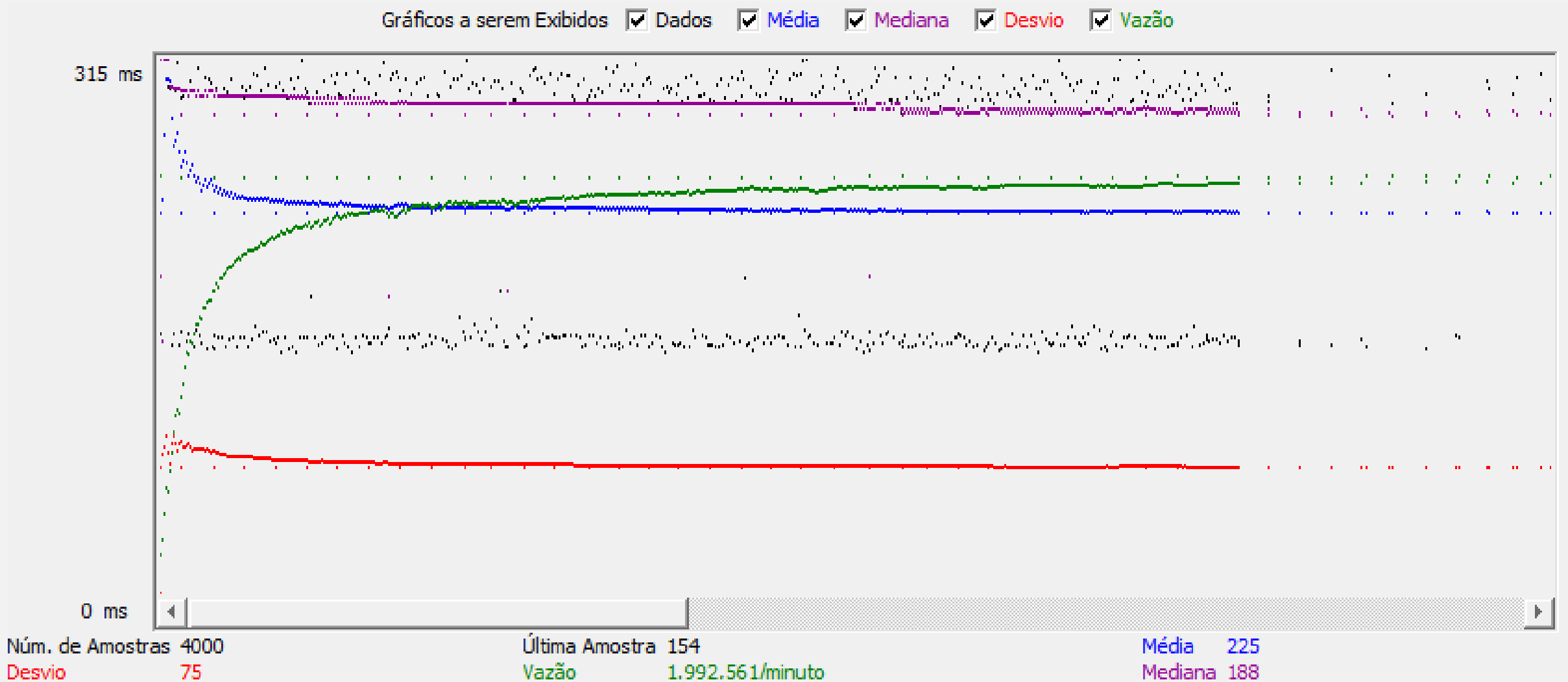
Podemos acompanhar os containers com `docker compose run -f`

```
load_balancer-1 | 179.117.69.55 - - [08/May/2024:04:15:10 +0000] "GET / HTTP/1.1" 200 305 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)"
site1-1         | 172.18.0.4 - - [08/May/2024:04:15:10 +0000] "GET / HTTP/1.0" 200 305 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)" "179.117.69.55"
load_balancer-1 | 179.117.69.55 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.1" 200 310 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)"
site2-1         | 172.18.0.4 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.0" 200 310 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)" "179.117.69.55"
load_balancer-1 | 179.117.69.55 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.1" 200 305 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)"
site1-1         | 172.18.0.4 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.0" 200 305 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)" "179.117.69.55"
site1-1         | 172.18.0.4 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.0" 200 305 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)" "179.117.69.55"
load_balancer-1 | 179.117.69.55 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.1" 200 305 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)"
site2-1         | 172.18.0.4 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.0" 200 310 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)" "179.117.69.55"
load_balancer-1 | 179.117.69.55 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.1" 200 310 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)"
site1-1         | 172.18.0.4 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.0" 200 305 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)" "179.117.69.55"
load_balancer-1 | 179.117.69.55 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.1" 200 305 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)"
site2-1         | 172.18.0.4 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.0" 200 310 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)" "179.117.69.55"
load_balancer-1 | 179.117.69.55 - - [08/May/2024:04:15:11 +0000] "GET / HTTP/1.1" 200 310 "-" "Apache-HttpClient/4.5.14 (Java/1.8.0_411)"
```

# Adicionando Response Time Graph

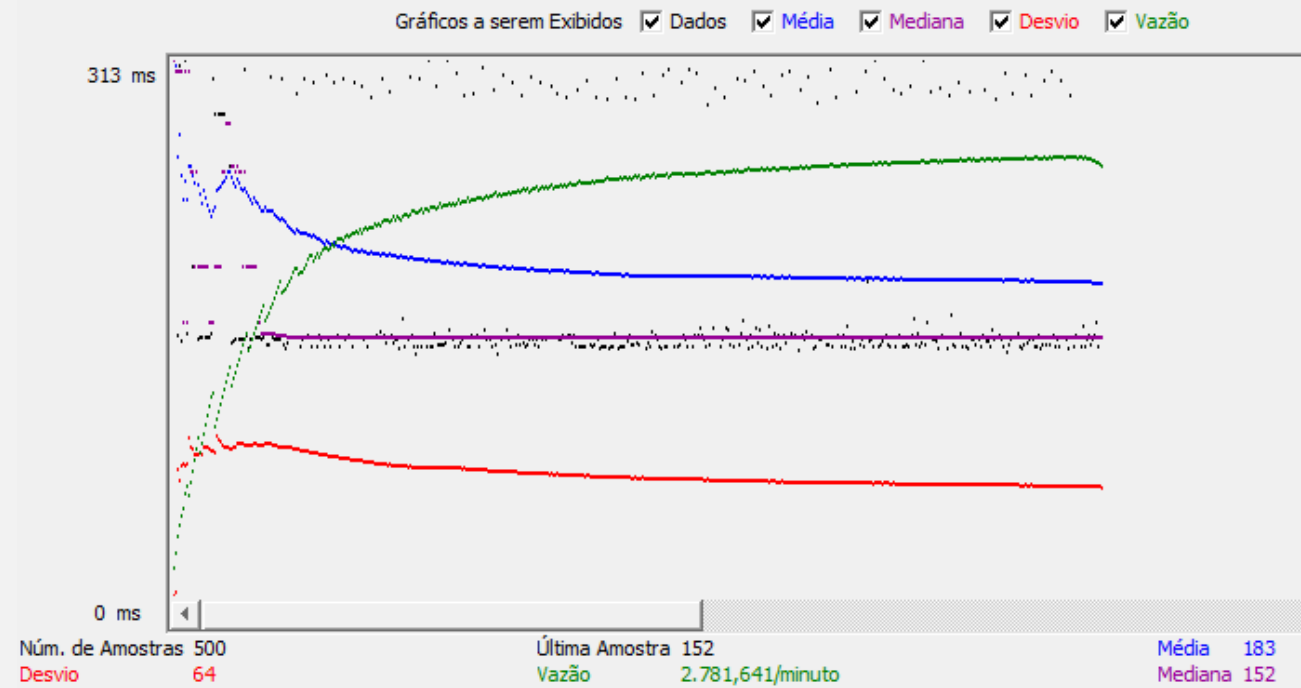


# Adicionando Gráfico de resultados

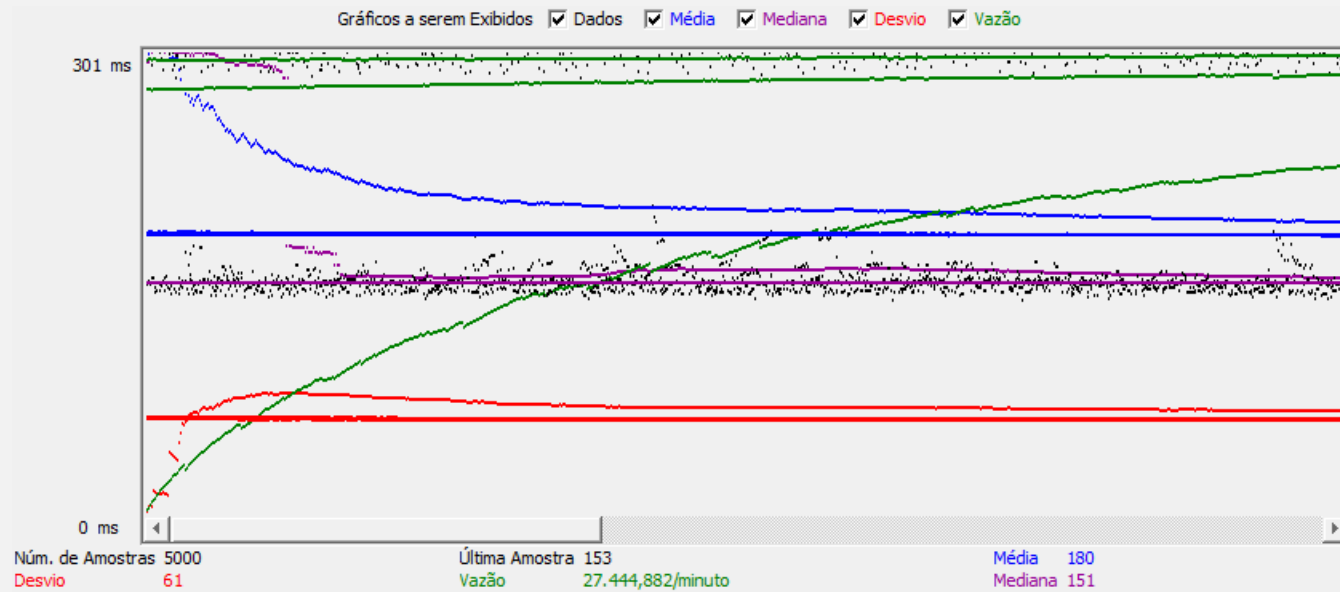


# Comparando resultados

100 usuários

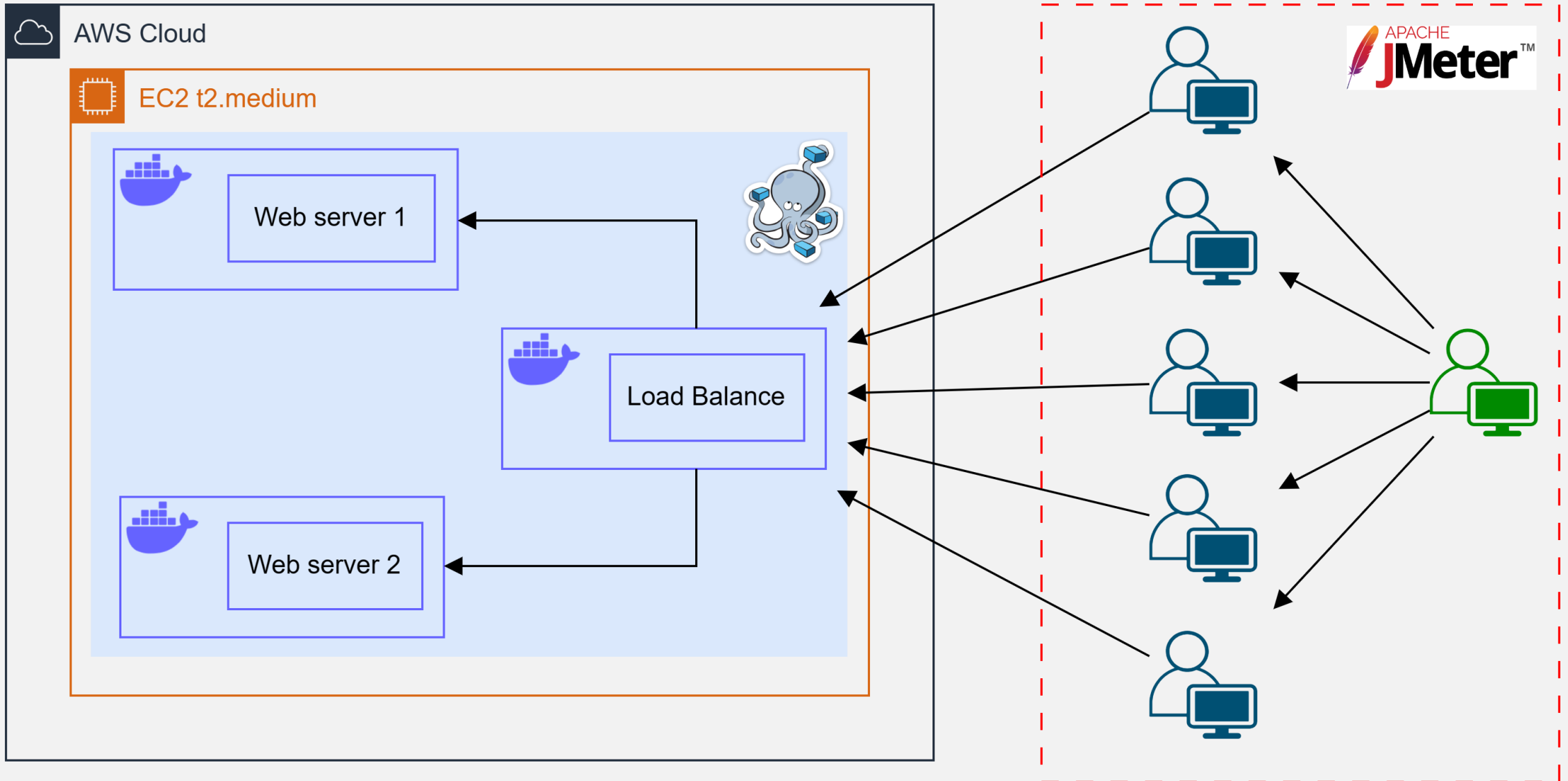


1.000 usuários

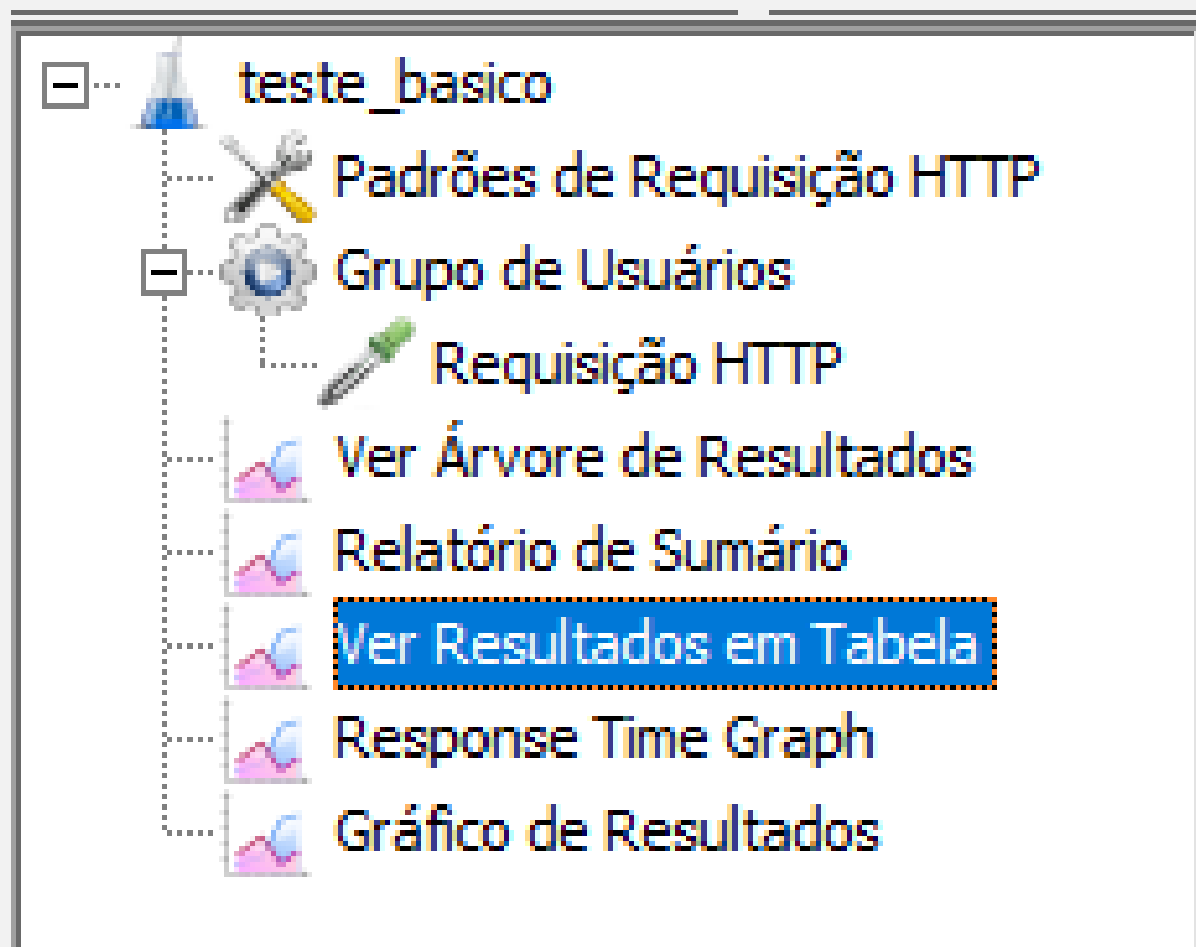


**Cenário exemplo de teste**

# Arquitetura testada



# Disposição dos recursos utilizados





# Disposição dos recursos utilizados

...



teste\_basico

## Plano de Teste

Nome:

teste\_basico

Comentários:

### Variáveis Definidas Pelo Usuário

Nome:	Valor

Detail

Adicionar

Add from Clipboard

Excluir

Up

Down

# Disposição dos recursos utilizados



## Padrões de Requisição HTTP

### Padrões de Requisição HTTP

Nome:

Comentários:

Basic | **Advanced**

#### Servidor Web

Protocolo [http]:  Nome do Servidor ou IP:  Número da Porta:

#### Requisição HTTP

Caminho:  Codificação do conteúdo:

Parameters | **Body Data**

# Disposição dos recursos utilizados



Grupo de Usuários

## Grupo de Usuários

Nome: Grupo de Usuários

Comentários:

Ação a ser tomada depois de erro do testador

☒ Continuar ☐ Start Next Thread Loop ☐ Interromper Usuário Virtual ☐ Interromper Teste ☐ Interrompe Teste Agora

Propriedades do Usuário Virtual

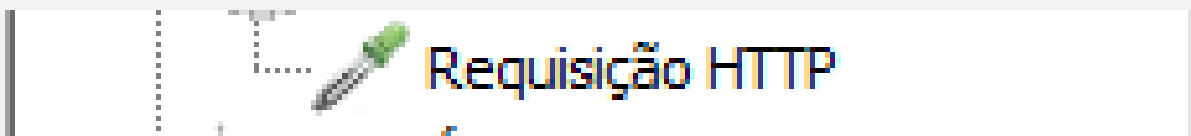
Número de Usuários Virtuais (threads): 1000

Tempo de inicialização (em segundos) 10

Contador de Iteração ☐ Infinito 5

☒ Same user on each iteration

# Disposição dos recursos utilizados



## Requisição HTTP

Nome:

Comentários:

Basic | Advanced

### Servidor Web

Protocolo [http]:

Nome do Servidor ou IP:

Número da Porta:

### Requisição HTTP

Caminho:

Codificação do conteúdo:

☒ Redirecionar automaticamente ☐ Seguir redireções ☒ Usar Manter Ativo (KeepAlive) ☐ Usar multipart/form-data para HTTP POST ☐ Browser-compatible headers

Parameters | Body Data | Files Upload

# Disposição dos recursos utilizados

 Response Time Graph

## Response Time Graph

Nome:

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo   Apenas Logar/Exibir ☐ Erros ☐ Sucessos

Settings | Graph |

### Graph settings

Interval (ms):

☐ Sampler label selection:   ☐ Case sensitive ☒ Regular exp.

### Title

Graph title:

Font:  Size:  Style:

**Hands on JMeter [CLI] e ApacheBench**

# JMeter via CLI

## **Não execute teste de carga usando o modo GUI!**

Usando o modo CLI, você pode gerar um arquivo CSV (ou XML) contendo resultados e fazer com que o JMeter gere um relatório HTML no final do teste de carga.

Por padrão, o JMeter fornecerá um resumo do teste de carga enquanto estiver em execução. Você também pode obter resultados em tempo real durante o teste usando o Backend Listener.

# Inicie o JMeter no modo de linha de comando

JMeter no modo GUI consome muita memória do computador. Para salvar o recurso, você pode optar por executar o JMeter sem a GUI. Para fazer isso, use as seguintes opções de comando

## JMeter command line

***jmeter -n***

Specified JMeter is  
to run in command  
line mode

***-t testPlan.jmx***

Name of file contains  
the Test Plan

***-l log.jtl***

Log file stores  
test result

***-H proxy\_Server -P 8000***

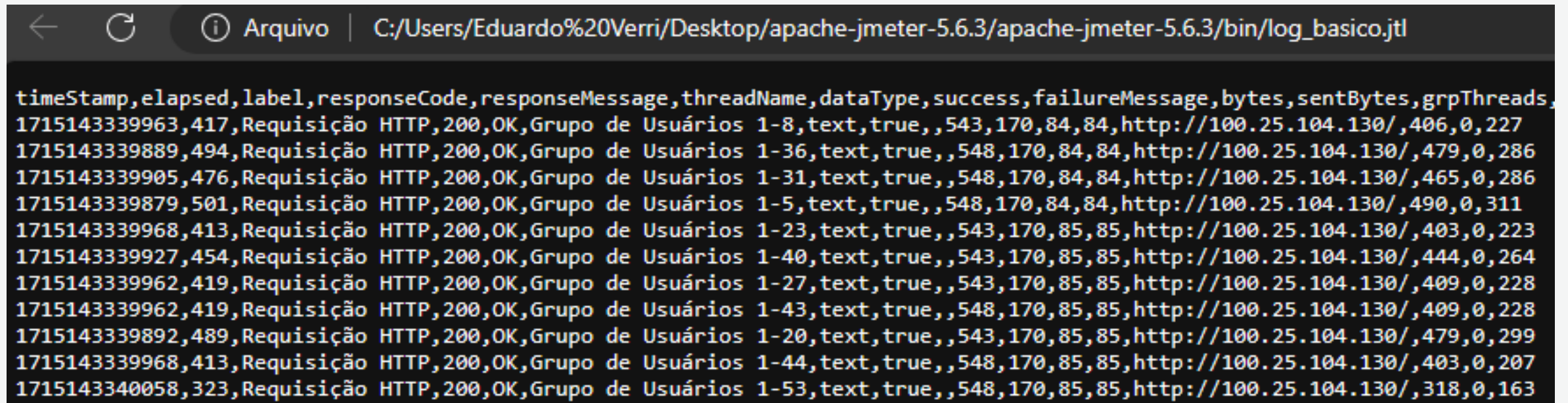
Proxy server host name and port



# Inicie o JMeter no modo de linha de comando

Inicie seu plano de teste com

```
jmeter -n -t .\teste_basico.jmx -l log_basico.jtl -H 100.25.104.130 -P 80
```



The screenshot shows a Windows command prompt window with the title bar "Arquivo | C:/Users/Eduardo%20Verri/Desktop/apache-jmeter-5.6.3/apache-jmeter-5.6.3/bin/log\_basico.jtl". The command prompt displays the output of the JMeter command, which is a CSV log file. The output consists of multiple lines of data, each representing a test result. The first line is a header row with the following fields: timestamp, elapsed time, label, response code, response message, thread name, data type, success status, failure message, bytes sent, bytes received, and group threads. The subsequent lines contain test results for various HTTP requests, all of which were successful (status 200 OK) and returned text data. The test results are grouped by thread name, with each group containing multiple requests. The data is as follows:

timestamp	elapsed	label	responseCode	responseMessage	threadName	dataType	success	failureMessage	bytes	sentBytes	grpThreads
1715143339963	417	Requisição HTTP	200	OK	Grupo de Usuários 1-8	text	true		543	170,84,84	http://100.25.104.130/,406,0,227
1715143339889	494	Requisição HTTP	200	OK	Grupo de Usuários 1-36	text	true		548	170,84,84	http://100.25.104.130/,479,0,286
1715143339905	476	Requisição HTTP	200	OK	Grupo de Usuários 1-31	text	true		548	170,84,84	http://100.25.104.130/,465,0,286
1715143339879	501	Requisição HTTP	200	OK	Grupo de Usuários 1-5	text	true		548	170,84,84	http://100.25.104.130/,490,0,311
1715143339968	413	Requisição HTTP	200	OK	Grupo de Usuários 1-23	text	true		543	170,85,85	http://100.25.104.130/,403,0,223
1715143339927	454	Requisição HTTP	200	OK	Grupo de Usuários 1-40	text	true		543	170,85,85	http://100.25.104.130/,444,0,264
1715143339962	419	Requisição HTTP	200	OK	Grupo de Usuários 1-27	text	true		543	170,85,85	http://100.25.104.130/,409,0,228
1715143339962	419	Requisição HTTP	200	OK	Grupo de Usuários 1-43	text	true		548	170,85,85	http://100.25.104.130/,409,0,228
1715143339892	489	Requisição HTTP	200	OK	Grupo de Usuários 1-20	text	true		543	170,85,85	http://100.25.104.130/,479,0,299
1715143339968	413	Requisição HTTP	200	OK	Grupo de Usuários 1-44	text	true		548	170,85,85	http://100.25.104.130/,403,0,207
1715143340058	323	Requisição HTTP	200	OK	Grupo de Usuários 1-53	text	true		548	170,85,85	http://100.25.104.130/,318,0,163

# ApacheBench [ab]



O ApacheBench (ab) é uma solução gratuita que pode auxiliar na realização de testes de performance em aplicações Web. Um dos componentes do Apache HTTP Server, este utilitário de linha de comando pode ser empregado em cenários simulando múltiplos usuários concorrentes.

Podemos instalá-lo em um SO Ubuntu executando: `sudo apt install apache2-utils`

Por exemplo, se quisermos enviar 100 requisições sendo 10 de forma concorrente/paralela para o site teste de exemplo:

```
ab -n 100 -c 10 http://100.25.104.130/
```

É possível gravar os resultados num arquivo também

```
ab -n 100 -c 10 http://100.25.104.130/ > teste1.txt
```

[ab - Apache HTTP server benchmarking tool - Apache HTTP Server Version 2.4](#)

## **Exercício Testes**

## Com seu projeto de PI...

- ❑ Crie um primeiro roteiro de testes, crie até 3 grupos de usuários (ramping ups diferentes) acessado sua página inicial, utilizando o JMeter
- ❑ Incremente esse cenário de teste com login na aplicação
- ❑ Incremente esse cenário de teste com pelo menos 2 testes em endpoints diferentes se possível
- ❑ Preferencialmente na hora de rodar os testes utilize o JMeter CLI, se não for possível, rode via GUI
- ❑ Gere evidência da criação dos cenários de testes com os prints de tela.

**Agradeço**  
**a sua atenção!**

**Eduardo Verri, Diego Brito**

[eduardo.verri@sptech.school](mailto:eduardo.verri@sptech.school)

diego.brito@sptech.school

SÃO  
PAULO  
TECH  
SCHOOL