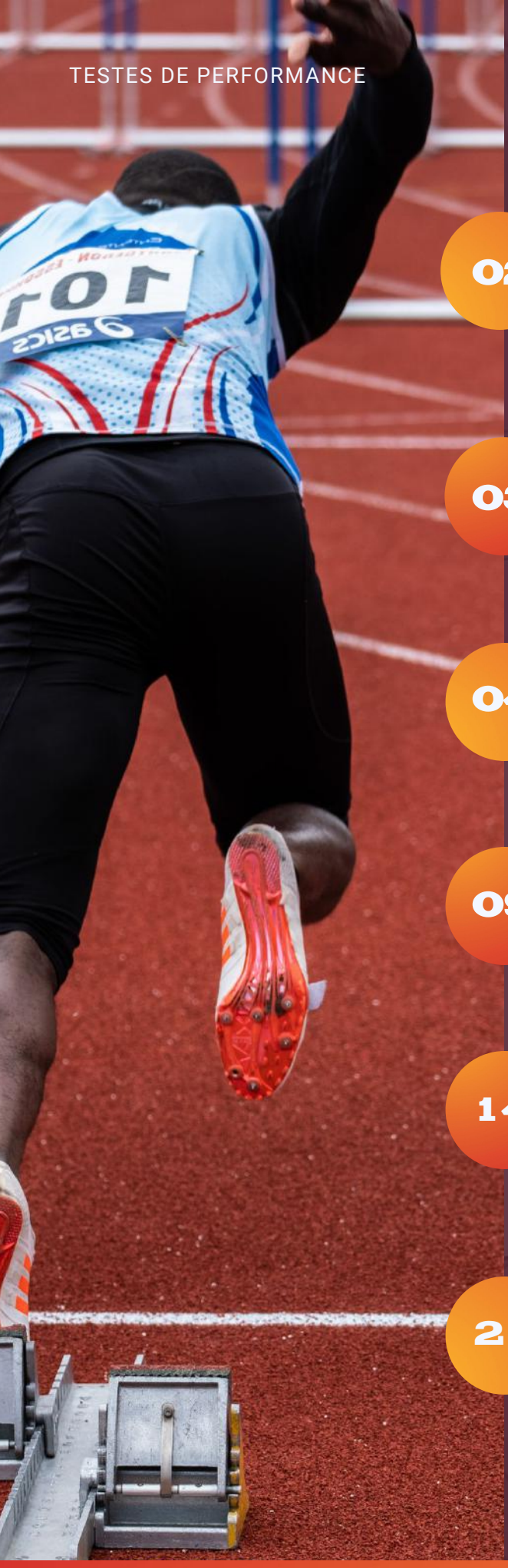
UM POUCO MAIS SOBRE
TESTES DE PERFORMANCE

LOURENA OHARA



CONTEÚDO

02 DEFINIÇÃO

03 PRINCÍPIOS

04 TIPOS DE TESTES

09 CONFIGURANDO TESTES DE PERFORMANCE

14 FERRAMENTAS E EXEMPLOS DE TESTE

21 CONCLUSÃO

O QUE SAO OS TESTES DE PERFORMANCE?

Os teste de performance são essenciais para que, quando os usuários utilizem aplicativos em suas diferentes plataformas, a experiência seja a melhor possível.
Os níveis de qualidade são estabelecidos por este tipo de teste, de maneira crítica integrando papéis como usabilidade e engenharia de performance

Testes de Performance são relevantes em diversas arquiteturas de sistemas como cliente-servidor, sistemas distribuídos e embarcados. A definição dos testes a serem realizados dependem do foco e avaliação de riscos realizadas pelos stakeholders.

PRINCÍPIOS

COMPORTAMENTO DO TEMPO

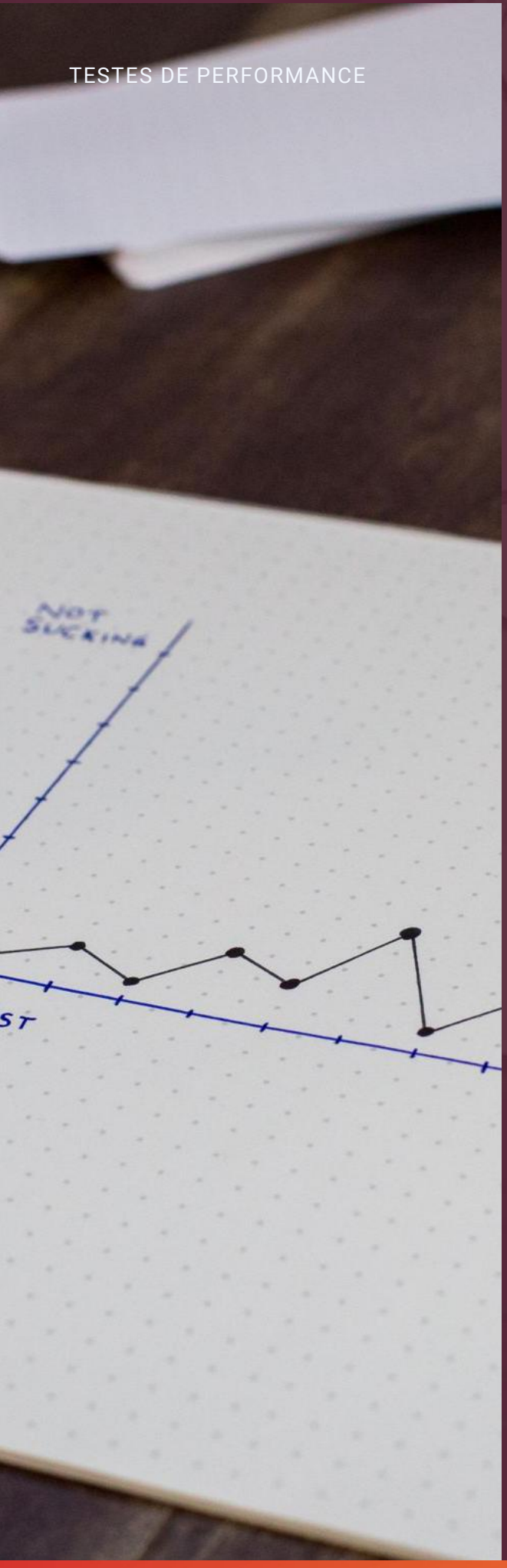
Objetivo mais comum nesse tipo de teste. É examinado, dentro de um tempo e condições específicas, estipulado a capacidade do sistema em responder às entradas do usuário ou dele mesmo. Variações das medições de comportamento podem ser percebidas no período de ponta a ponta ou em números de ciclos de CPU.

UTILIZAÇÃO DE RECURSOS

A disponibilidade de alocação de memória RAM ou uso de CPU podem ser identificadas como um risco. Testes são realizados para investigar a utilização.

CAPACIDADE

Testes são realizados para avaliar a arquitetura dos sistemas quando a capacidade de números de usuários, volume de dados, entre outros foram identificados como risco.



TIPOS DE TESTE

Utilizamos duas classificações para os testes. Na primeira, temos os Testes Estáticos e Dinâmicos

Na segunda classificação, temos os
Testes de Carga
Testes de Estresse
Testes de Escalabilidade
Testes de Pico
Testes de Resistência
Testes de Concorrência
Testes de Capacidade

TIPOS DE TESTE

E ESTÁTICO

- Revisões de requisitos com foco em aspectos e riscos de performance
- Revisões de esquemas de banco de dados, diagramas de relacionamento de entidade, metadados, procedimentos armazenados e consultas
- Revisões do sistema e arquitetura de rede
- Revisões de segmentos críticos do código do sistema (p. ex., algoritmos complexos)

D DINÂMICO

Se o hardware customizado ou novo fizer parte do sistema, os testes iniciais de performance dinâmico podem ser realizados usando simuladores. No entanto, é uma boa prática começar a testar o hardware real o mais rápido possível, pois os simuladores geralmente não capturam adequadamente as restrições de recursos e os comportamentos relacionados à performance

TIPOS DE TESTE

Teste de carga

Lida com níveis crescentes de carga em um ambiente real em que são observadas o comportamento das transações suportadas quando aumentamos a carga de usuário em números significativos

Teste de estresse

Concentra-se na capacidade do sistema se sustentar com picos de carga dentro do limite e que ultrapasse o limite em utilização média. Pode ser utilizado também quando a disponibilidade de recursos é reduzida.

Teste de escabilidade

Determina a capacidade de um sistema em se expandir. É configurado considerando os limites de escabilidade já conhecidos e novamente definidos. O monitoramento desses limites são realizados em produção, observando e relatando possíveis falhas que estão prestes a acontecer.

TIPOS DE TESTE

Teste de pico

Testa a capacidade do sistema em suportar rajadas intempestivas de cargas consideráveis, conseguindo retornar ao seu estado estável.

Teste de resistência

Testa a disponibilidade de capacidade de recursos(vazamento de memória, cenexões de bancos de dados entre outros) que podem degradar a memória ou causar falhas. É um teste concentrado na estabilidade que o sistema proporciona.

Teste de concorrência

Quando muitos usuários fazem login ao mesmo tempo ou outras ações específicas ocorrem simultaneamente, o teste de concorrência foca nos impactos dessas ações. Problemas de concorrência são difíceis de encontrar e reproduzir, inclusive em produção

TIPOS DE TESTE

Teste de capacidade

Determina quantos usuários/transações um sistema suporta, considerando os objetivos da performance (como métricas definidas).

?

Combinação de Tipos de Testes

Os testes de performance podem ser aplicados à um determinado tipo de projeto isolados ou combinados, dependendo do objetivo

CONFIGURAÇÃO

A construção dos testes de performance exige a participação de vários stakeholders e uma verdadeira montagem de projeto.

É necessário que o cliente ou dono do sistema saiba claramente quais são as necessidades de performance para saber exatamente quais os testes serão aplicados.

Além disso, a definição de ambiente (homologação ou produção), níveis de carga e número de usuários, infraestrutura, ferramentas utilizadas, tempo de teste entre outros, devem ser definidos e alinhados com todos os responsáveis.

ENTENDA O QUE É NECESSÁRIO

Quando uma demanda de testes de performance chega, é preciso entender quais são as necessidades do cliente. Por isso, o planejamento começa entendendo quais pontos do sistema serão testados e o que é esperado com estes testes.

QUAIS TESTES SERÃO USADOS

Como vimos, existem diferentes tipos de testes podem ser utilizados, separados ou integrados em um projeto. Com o planejamento do que testar, definimos quais os testes serão aplicados.

SMOKE TEST

Testes com um número pequeno de carga são realizados para observar o comportamento de pontos definidos do sistema. Esses testes são essenciais para observar os tempos de resposta, a quantidade de usuários virtuais a serem utilizados, tempo de teste e o próximo passo: a infraestrutura

INFRAESTRUTURA

Realizados o smoke test, temos as informações necessárias para definirmos quantos usuários virtuais, máquinas virtuais, tempo de teste e ferramentas de métricas e relatórios serão necessárias.

PROJETO DE TESTES

Este é o momento de concluir a construção do projeto de testes com os dados coletados nos passos anteriores. Com isso, cada passo é configurado com o número de usuários necessários e outras informações.

EXECUÇÃO DOS TESTES

Pronto! Agora é a hora de realizar os testes no ambiente definido e horário marcado. Neste momento, é importante a presença de toda a equipe necessária, para observar os testes, auxiliar em problemas com a infraestrutura e coleta de métricas e informações em tempo real.

APRESENTAÇÃO E RELATÓRIO

Com os testes realizados seguindo todos os parâmetros definidos, é hora de reunir todas as informações obtidas e gerar um relatório consolidado. Este relatório precisa conter as informações desde as necessidades dos clientes até os dados gerados na execução do teste. É realizada uma apresentação do realtório aos stakeholders. Este relatório poderá ser utilizado na execução de novos testes no futuro.

Na concepção do relatório, estes são os dados comumentes analisados:

- Status de usuários simulados
- Tempo de resposta de transação
- Transações por segundo
- Falhas de transação
- Ocorrências de transção
- Resposta Http

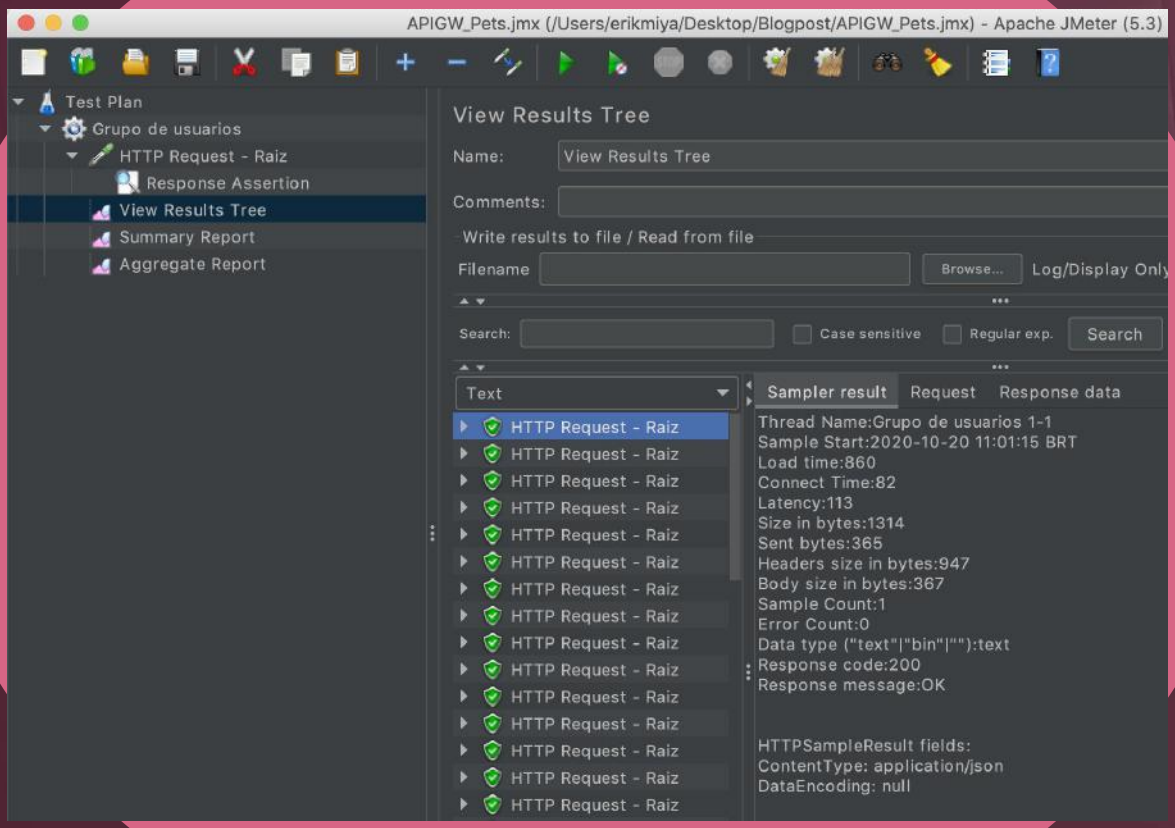
FERRAMENTAS
EXEMPLOS DE TESTE

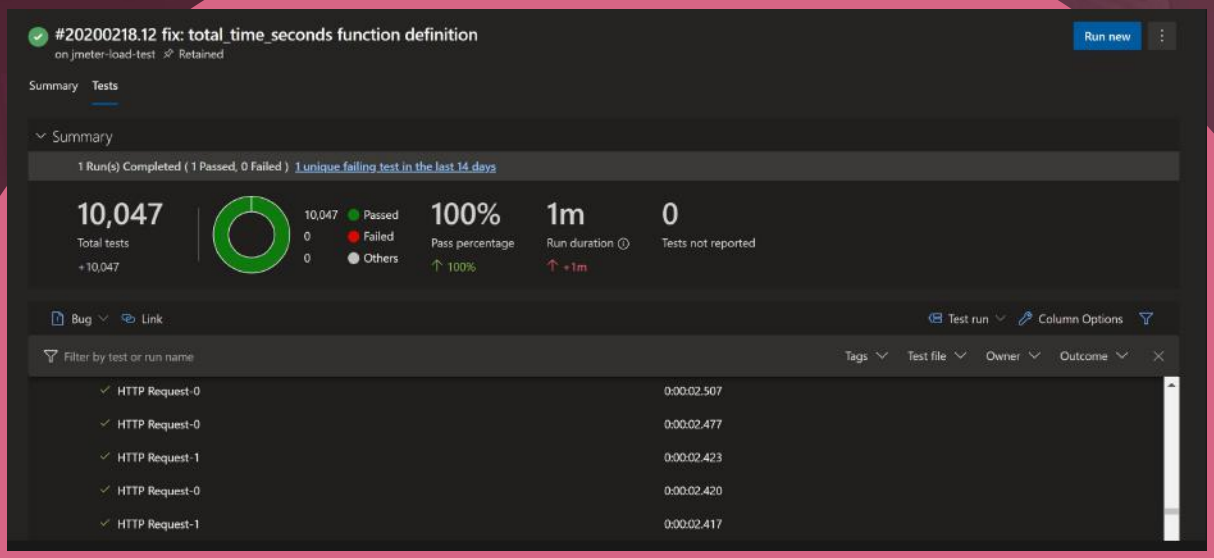
As ferramentas de teste precisam conter estruturas para fornecer um melhor suporte aos testes.

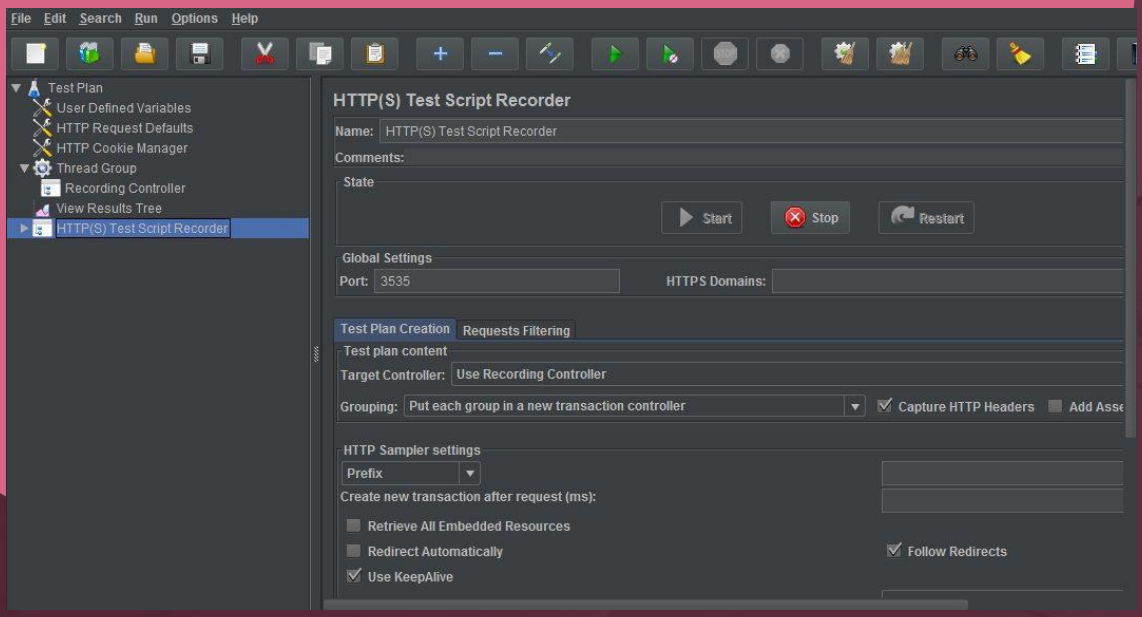
Geradores de carga, com o auxílio de uma IDE, criam e executam instâncias que simulam o comportamento de um usuário. Console de Gerenciamento de Carga fornece o controle de start e stop dos geradores de carga.

Os geradores permitem também a visualização de relatórios e gráficos. As **Ferramentas de Monitoramento** supervisionam, registram e analisam o comportamento do sistema.
Exemplos de ferramentas e scripts de testes são apresentados a seguir.

O aplicativo Apache JMeter™ é um software de código aberto, um aplicativo Java 100% puro projetado para carregar o comportamento funcional de teste e medir desempenho. Ele foi originalmente projetado para testar aplicativos da Web, mas desde então, se expandiu para outras funções de teste.




#20200218.12 fix: total_time_seconds function definition
on jmeter-load-test
Retained
Run new
Summary
Tests
Summary
1 Run(s) Completed (1 Passed, 0 Failed) 1 unique failing test in the last 14 days
10,047
Total tests
+10,047
10,047 Passed
0 Failed
0 Others
100%
Pass percentage
100%
1m
Run duration
+1m
0
Tests not reported
Bug
Link
Test run
Column Options
Filter by test or run name
Tags
Test file
Owner
Outcome
HTTP Request-0 0:00:02.507
HTTP Request-0 0:00:02.477
HTTP Request-1 0:00:02.423
HTTP Request-0 0:00:02.420
HTTP Request-1 0:00:02.417


File Edit Search Run Options Help
Test Plan
User Defined Variables
HTTP Request Defaults
HTTP Cookie Manager
Thread Group
Recording Controller
View Results Tree
HTTP(S) Test Script Recorder
HTTP(S) Test Script Recorder
Name: HTTP(S) Test Script Recorder
Comments:
State
Start
Stop
Restart
Global Settings
Port: 3535
HTTPS Domains:
Test Plan Creation
Requests Filtering
Test plan content
Target Controller: Use Recording Controller
Grouping: Put each group in a new transaction controller
Capture HTTP Headers
HTTP Sampler settings
Prefix
Create new transaction after request (ms):
Retrieve All Embedded Resources
Redirect Automatically
Follow Redirects
Use KeepAlive

Gatling Load Testing é uma ferramenta para testes de performance baseada em Scala_Akka-Netty que pode ser facilmente utilizada, gerando relatórios diferenciados.

PetStoreSimulation1.scala

```scala
package com.octoperf.tutorials.two

import scala.concurrent.duration._

import io.gatling.core.Predef._
import io.gatling.http.Predef._
import io.gatling.jdbc.Predef._

class PetStoreSimulation1 extends Simulation {

    val httpProtocol = http
        .baseUrl("https://petstore.octoperf.com")
        .acceptHeader("text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8")
        .doNotTrackHeader("1")
        .acceptLanguageHeader("en-US,en;q=0.5")
        .acceptEncodingHeader("gzip, deflate")
        .userAgentHeader("Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:16.0) Gecko/20100101 Firefox/16.0")
        .inferHtmlResources(BlackList(), WhiteList("https://petstore.octoperf.com/.*"))

    val csvFeeder = csv("two/categories.csv").random

    val scn = scenario("PetStoreSimulation")
        .exec(http("Homepage").get("/actions/Catalog.action"))
        .feed(csvFeeder)
        .exec(http("Catalog ${categoryId}")
            .get("/actions/Catalog.action")
            .queryParam("viewCategory", "")
            .queryParam("categoryId", "${categoryId}"))

    setUp(scn.inject(atOnceUsers(1))).protocols(httpProtocol)
}
```

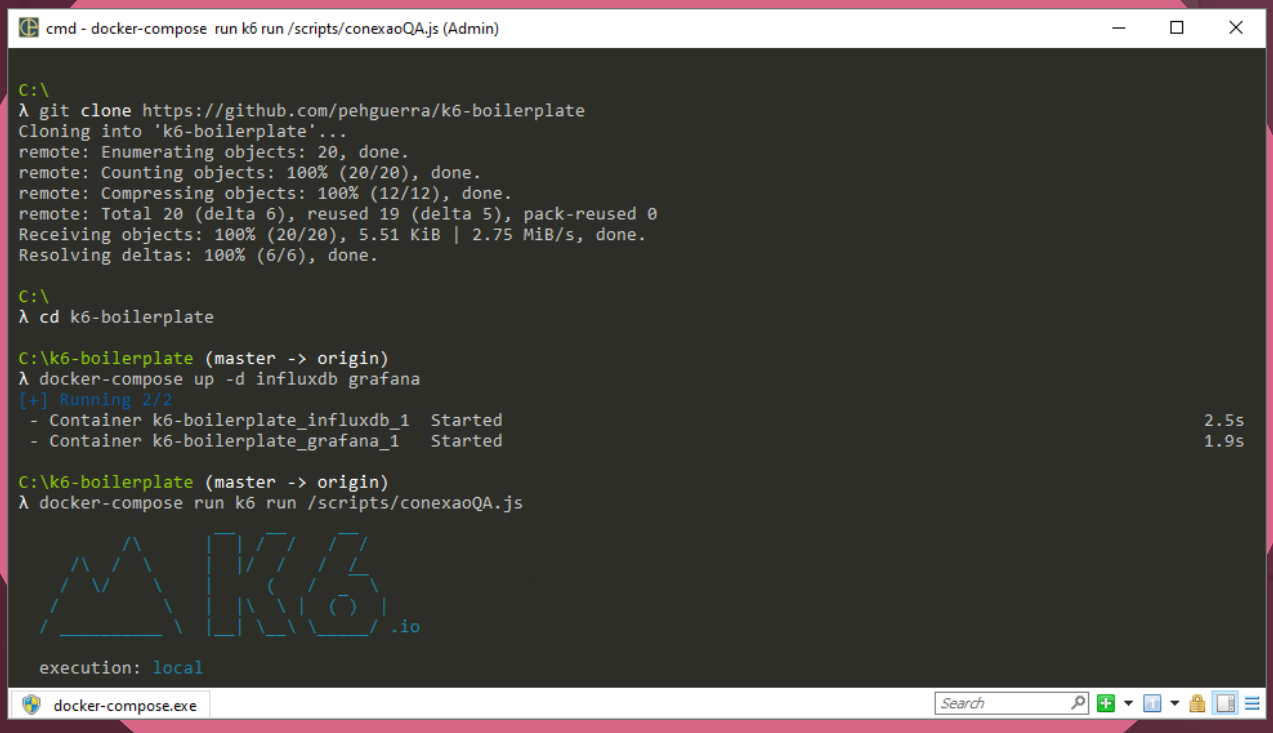
STATISTICS — Expand all groups | Collapse all groups

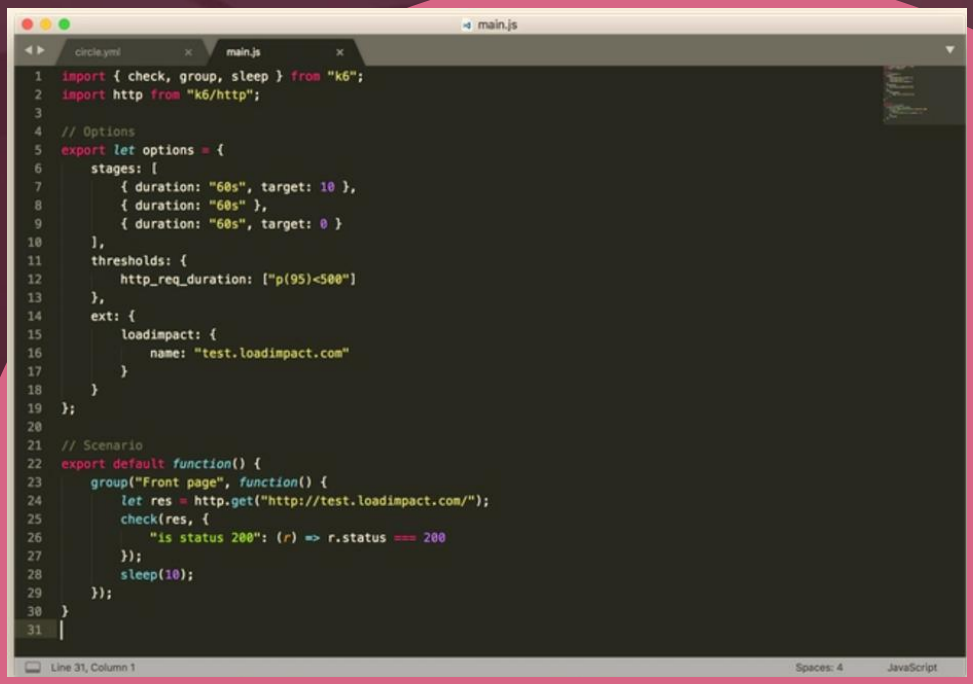
Requests	Executions				Response Time (ms)						
	Total	OK	KO	% KO	Min	Max	Mean	Std Dev	95th pct	99th pct	Req/s
Global Information	9661	9659	2	0 %	94	1145	159	108	383	663	140.6
Home Redirect 1	2410	2410	0	0 %	94	1145	150	101	335	625	35.06
Search	1205	1205	0	0 %	97	965	156	106	366	656	17.53
Select	1205	1205	0	0 %	95	980	155	108	352	684	17.53
Page 0	1205	1205	0	0 %	100	1134	169	122	389	733	17.53
Page 1	1205	1205	0	0 %	100	1122	163	113	374	687	17.53
Page 2	1205	1205	0	0 %	100	992	165	110	367	677	17.53
Page 3	1205	1205	0	0 %	100	978	163	101	365	561	17.53
Form	7	7	0	0 %	98	512	194	148	453	500	0.10
Post Redirect 1	14	12	2	14 %	99	483	150	103	354	457	0.20

ERRORS

Error	Count	Percentage
status.is(201), but actually found 200	2	100.0 %

K6 é uma ferramenta moderna de teste de carga, baseada em anos de experiência na indústria de testes de carga e desempenho. Ele fornece uma API de script limpa e acessível, execução local e na nuvem com configuração flexível.





```
import { check, group, sleep } from "k6";
import http from "k6/http";

// Options
export let options = {
    stages: [
        { duration: "60s", target: 10 },
        { duration: "60s" },
        { duration: "60s", target: 0 }
    ],
    thresholds: {
        http_req_duration: ["p(95)<500"]
    },
    ext: {
        loadimpact: {
            name: "test.loadimpact.com"
        }
    }
};

// Scenario
export default function() {
    group("Front page", function() {
        let res = http.get("http://test.loadimpact.com/");
        check(res, {
            "is status 200": (r) => r.status === 200
        });
        sleep(10);
    });
}
```

```
  scenarios: (100.00%) 1 scenario, 1 max VUs, 10m30s max duration (incl. graceful stop):
           * default: 1 iterations for each of 1 VUs (maxDuration: 10m0s, gracefulStop: 30s)


running (00m03.2s), 0/1 VUs, 1 complete and 0 interrupted iterations
default ✓ [======================================] 1 VUs  00m03.2s/10m0s  1/1 iters, 1 per VU

    data_received..................: 17 kB 5.2 kB/s
    data_sent......................: 511 B 157 B/s
    http_req_blocked...............: avg=2.09s    min=2.09s    med=2.09s    max=2.09s    p(90)=2.09s    p(95)=2.09s
    http_req_connecting............: avg=124.54ms min=124.54ms med=124.54ms max=124.54ms p(90)=124.54ms p(95)=124.54ms
    http_req_duration..............: avg=148.61ms min=148.61ms med=148.61ms max=148.61ms p(90)=148.61ms p(95)=148.61ms
      { expected_response:true }...: avg=148.61ms min=148.61ms med=148.61ms max=148.61ms p(90)=148.61ms p(95)=148.61ms
    http_req_failed................: 0.00% ✓ 0        ✗ 1
    http_req_receiving.............: avg=96µs     min=96µs     med=96µs     max=96µs     p(90)=96µs     p(95)=96µs
    http_req_sending...............: avg=97µs     min=97µs     med=97µs     max=97µs     p(90)=97µs     p(95)=97µs
    http_req_tls_handshaking.......: avg=432.09ms min=432.09ms med=432.09ms max=432.09ms p(90)=432.09ms p(95)=432.09ms
    http_req_waiting...............: avg=148.42ms min=148.42ms med=148.42ms max=148.42ms p(90)=148.42ms p(95)=148.42ms
    http_reqs......................: 1     0.307764/s
    iteration_duration.............: avg=3.24s    min=3.24s    med=3.24s    max=3.24s    p(90)=3.24s    p(95)=3.24s
    iterations.....................: 1     0.307764/s
    vus............................: 1     min=1      max=1
    vus_max........................: 1     min=1      max=1
```

CONCLUSÃO

O que, como e quando?

Entender o que são e quais os tipos diferentes de testes de performance, é essencial para a consolidação de um projeto de testes visando utilização de cargas, suporte do sistema e fornecimento de infraestrutura para todos os usuários.

Ao aplicar as técnicas compreendidas, o entendimento dos dados e relatórios obtidos, torna-se uma das etapas mais importantes do projeto de performance.

Conhecer a teoria, assim como entender a utilização de ferramentas, garante que ao realizar a prática, vamos concretizar da melhor forma um projeto de teste visando a coleta de métricas importantes e garantindo o melhor desenvolvimento e utilização do sistema.

Lourena Ohara

- Belo Horizonte/MG
- QA Senior
- lourena6ohara@gmail.com
- https://www.linkedin.com/in/lourena-ohara/