1.0

TESTE DE CARGA PARA Q.A.

*contruindo Testes de Cargas eficientes com **K6***

wederson machado

trilhadequalidade.com.br

ÍNDICE

INÍCIO

QUEM SOU EU?

Olá! Meu nome é Wederson, um carioca apaixonado pela vida, nascido nas ensolaradas terras do Rio de Janeiro em 02 de fevereiro de 1991. Desde cedo, fui cativado pelo mundo da tecnologia, onde encontrei minha paixão e propósito.

Ao longo dos anos, desenvolvi uma afinidade especial com a área técnica, liderando equipes com entusiasmo e comprometimento. São mais de 5 anos de experiência como líder, guiando e inspirando colegas na jornada do sucesso profissional.

Minha trajetória no universo da tecnologia começou aos 19 anos, e desde então, tenho vivido e respirado esse mundo dinâmico e desafiador. Além disso, minha vida é preenchida com amor pelos meus amigos, pelos animais, especialmente pelos cachorros, e pela energia revigorante da praia.
Neste Ebook, compartilho com você minha expertise, na esperança de inspirar e agregar valor ao seu caminho. Juntos, vamos explorar as possibilidades infinitas desse fascinante universo digital. Seja bem-vindo(a) à nossa jornada!

APRESENTAÇÃO DO LIVRO

Objetivo do Livro

Este livro tem como objetivo proporcionar um guia completo e prático sobre a utilização do K6 para realizar testes de carga eficientes e eficazes. Através deste livro, os leitores aprenderão como configurar, executar e interpretar testes de carga, além de entender como o K6 pode ser integrado em diferentes ambientes e pipelines de CI/CD.

Público-alvo

O público-alvo deste livro inclui profissionais de Quality Assurance (QA), engenheiros de desempenho, desenvolvedores de software e qualquer pessoa interessada em garantir que suas aplicações possam suportar a carga esperada. Tanto iniciantes quanto usuários experientes encontrarão neste livro informações valiosas para aprimorar suas práticas de teste de carga.

Estrutura do Livro

Este livro está organizado da seguinte maneira:

- **Apresentação do Livro**: Objetivo, público-alvo e estrutura.
- **Introdução aos Testes de Carga**: Importância, cenários críticos e desafios comuns.

- **Visão Geral do K6**: História, características e comparação com outras ferramentas.
- **Configuração Inicial do K6**: Instalação, configuração e primeiros passos.
- **Escrevendo Testes com K6**: Sintaxe básica, criação de scripts e exemplos práticos.
- **Executando Testes de Carga**: Métodos de execução, análise de resultados e ajustes.
- **Integração com CI/CD**: Automação de testes de carga em pipelines.
- **Boas Práticas e Dicas Avançadas**: Estratégias para otimização e manutenção de testes.
- **Estudos de Caso**: Exemplos reais de utilização do K6 em diferentes contextos.

INTRODUÇÃO AOS TESTES DE CARGA

Os testes de carga são fundamentais para garantir que uma aplicação possa suportar o tráfego e a utilização esperados sem degradação de desempenho. Eles ajudam a identificar gargalos, otimizar recursos e garantir a satisfação dos usuários finais.

Cenários Típicos em que os Testes de Carga são Críticos

- **Lançamento de Novos Produtos**: Garantir que uma nova aplicação ou funcionalidade possa suportar um grande número de usuários simultâneos.

- **Eventos Promocionais**: Preparação para picos de tráfego durante promoções, vendas sazonais ou eventos especiais.
- **Escalabilidade**: Avaliar a capacidade de uma aplicação de escalar com o aumento da carga.
- **Migração para a Nuvem**: Validar o desempenho após a migração de sistemas para ambientes de nuvem.

Desafios Comuns em Testes de Performance

- **Ambientes de Teste Representativos:** Criar ambientes de teste que reflitam com precisão os ambientes de produção.
- **Dados Realistas**: Gerar dados de teste que sejam representativos do uso real.
- **Análise de Resultados**: Interpretar corretamente os resultados dos testes para identificar problemas e oportunidades de melhoria.
- **Automatização**: Integrar testes de carga em pipelines de CI/CD para testes contínuos e feedback rápido.

VISÃO GERAL DO K6

K6 foi desenvolvido pela Load Impact, uma empresa com vasta experiência em testes de desempenho. Lançado como uma ferramenta de código aberto, o K6 foi projetado para ser uma ferramenta moderna, orientada para desenvolvedores, com foco em facilidade de uso e integração.

Principais Características e Benefícios

- **Facilidade de Uso**: Scripts escritos em JavaScript, uma linguagem familiar para muitos desenvolvedores.
- **Alta Performance**: Capacidade de gerar grande quantidade de carga com eficiência.
- **Flexibilidade**: Suporte para múltiplos protocolos e integração com diversas ferramentas.
- **Escalabilidade**: Projetado para executar testes em escala, suportando desde pequenos testes locais até grandes execuções em nuvem.
- **Extensibilidade**: Possibilidade de estender funcionalidades através de módulos e bibliotecas.

Comparação com Outras Ferramentas de Teste de Carga

- **JMeter**: Embora amplamente utilizado, JMeter pode ser mais complexo e menos intuitivo para novos usuários em comparação com K6.
- **Gatling**: Oferece alta performance, mas possui uma curva de aprendizado mais íngreme devido à sua sintaxe em Scala.
- **Artillery**: Também baseado em JavaScript, mas K6 oferece maior performance e um conjunto de funcionalidades mais robusto.
- **Locust**: Escrito em Python, adequado para quem já está familiarizado com essa linguagem, mas K6 oferece melhor desempenho em ambientes de grande escala.

COMEÇANDO COM K6

INSTALAÇÃO E CONFIGURAÇÃO

Requisitos do Sistema

Antes de instalar o K6, verifique se seu sistema atende aos seguintes requisitos mínimos:

- **Sistema Operacional**: Windows, macOS ou Linux.
- **Memória RAM**: Mínimo de 4GB (8GB recomendado para grandes testes).
- **Espaço em Disco**: Pelo menos 100MB de espaço livre.
- **Dependências**: Node.js (para algumas funcionalidades avançadas).

Instalação do K6 em Diferentes Sistemas Operacionais

Windows

1. **Usando Chocolatey**:
    - Abra o PowerShell como Administrador.
    - Execute o comando:

Instalando K6 Windows

```
choco install k6
```

2. **Download Direto**:
   - Baixe o arquivo binário do K6 aqui.
   - Extraia o arquivo e adicione a pasta extraída ao PATH do sistema.

macOS

1. **Usando Homebrew:**
   - Abra o Terminal.
   - Execute o comando:

Instalando K6 macOS

```
brew install k6
```

2. **Download Direto**:
   - Baixe o arquivo binário do K6 aqui.
   - Extraia o arquivo e mova-o para o diretório /usr/local/bin.

Linux

1. **Usando APT (Debian/Ubuntu):**
   - Abra o Terminal.
   - Execute o comando:

Instalando K6 Linux

```
sudo apt update
sudo apt install k6
```

macOS

1. **Usando Homebrew:**
    - Abra o Terminal.
    - Execute o comando:

Configuração Inicial e Primeiros Passos

Após instalar o K6, você pode verificar se a instalação foi bem-sucedida executando:

Versão K6

```
k6 version
```

Se a instalação estiver correta, o comando retornará a versão do K6 instalada.

PRIMEIROS TESTES COM K6

Estrutura Básica de um Script K6

Um script K6 é escrito em JavaScript e segue uma estrutura básica. Aqui está um exemplo simples:

test.js

```
import http from 'k6/http';
import { sleep } from 'k6';

export let options = {
  vus: 10, // número de usuários virtuais
  duration: '30s', // duração do teste
};

export default function () {
  http.get('https://test.k6.io');
  sleep(1); // tempo de espera entre as requisições
}
```

Execução de um Teste Simples

Para executar o script acima, salve-o em um arquivo, por exemplo, test.js. Em seguida, execute o comando:

Rodando K6

```
k6 run test.js
```

Interpretação dos Resultados Iniciais

Após a execução do teste, o K6 fornecerá um resumo dos resultados no terminal. Aqui estão algumas métricas chave e suas interpretações:

- **http_reqs**: Número total de requisições HTTP realizadas.
- **vus_max**: Número máximo de usuários virtuais ativos.
- **iterations**: Número de iterações do teste executadas.
- **http_req_duration**: Tempo de resposta das requisições HTTP.
- **http_req_failed**: Percentual de requisições que falharam.

Essas métricas ajudam a avaliar o desempenho inicial da sua aplicação sob carga e identificar possíveis áreas de melhoria.

LINGUAGEM DE SCRIPT DO K6 (JAVASCRIPT)

Fundamentos do JavaScript Aplicados ao K6

O K6 utiliza JavaScript como sua linguagem de script, o que torna a criação e manutenção de testes acessível para muitos desenvolvedores. Aqui estão alguns conceitos básicos do JavaScript aplicados ao K6:

- **Variáveis e Tipos de Dados:**

Variáveis e Tipos de Dados

```
let stringVar = "Hello, K6!";
let numberVar = 42;
let booleanVar = true;
```

- **Funções:**

Funções

```
function greet(name) {
  return `Hello, ${name}!`;
}

console.log(greet("K6"));
```

- **Loops e Condicionais:**

Loops e Condicionais

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}

if (booleanVar) {
  console.log("This is true");
}
```

Principais Funções e APIs do K6

O K6 oferece várias funções e APIs para facilitar a escrita de scripts de teste de carga:

- **http.get(url, params):** Faz uma requisição HTTP GET.

http.get(url, params)

```
import http from 'k6/http';

export default function () {
  let res = http.get('https://test.k6.io');
}
```

- **Loops e Condicionais:sleep(seconds):** Pausa a execução por um número de segundos.

sleep(seconds)

```
import { sleep } from 'k6';

export default function () {
  sleep(1);
}
```

- **check(response, conditions):** Valida as respostas recebidas.

check(response, conditions)

```
import { check } from 'k6';

export default function () {
  let res = http.get('https://test.k6.io');
  check(res, {
    'status is 200': (r) => r.status === 200,
  });
}
```

Organização de Scripts

Organizar seus scripts de forma limpa e modular é essencial para a manutenção. Estruture seus scripts com funções reutilizáveis e separe as configurações das implementações.

- **Exemplo de Estrutura Modular:**

Estrutura Modular

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

export let options = {
  vus: 10,
  duration: '30s',
};

function makeRequest() {
  let res = http.get('https://test.k6.io');
  check(res, {
    'status is 200': (r) => r.status === 200,
  });
}

export default function () {
  makeRequest();
  sleep(1);
}
```

ESCREVENDO TESTES BÁSICOS

Criando Usuários Virtuais (VUs)

Usuários virtuais (VUs) são instâncias simuladas de usuários reais que executam os scripts de teste. Configuramos VUs nas opções do script:

Criando usuário virtuais (VUs)

```
export let options = {
  vus: 50, // número de usuários virtuais
  duration: '1m', // duração do teste
};
```

Simulação de Cenários de Navegação Simples

Simular a navegação de um usuário através de diferentes páginas é essencial para testes realistas:

Navegação Simplies

```
import http from 'k6/http';
import { sleep } from 'k6';

export default function () {
  http.get('https://test.k6.io');
  sleep(1);
  http.get('https://test.k6.io/contact');
  sleep(1);
}
```

Validação de Respostas

Para garantir que as respostas são as esperadas, use a função check:

Validação de Resposta

```
import { check } from 'k6';

export default function () {
  let res = http.get('https://test.k6.io');
  check(res, {
    'status is 200': (r) => r.status === 200,
    'body size is 1176 bytes': (r) => r.body.length === 1176,
  });
}
```

PARÂMETROS E CONFIGURAÇÕES DE TESTE

Configuração de Duração e Intensidade dos Testes

Você pode configurar a duração e a intensidade dos testes nas opções do script:

Configuração de Duração e Intensidade

```
export let options = {
  stages: [
    { duration: '30s', target: 20 }, // Rampa de 0 a 20 usuários em 30 segundos
    { duration: '1m', target: 20 },  // Sustenta 20 usuários por 1 minuto
    { duration: '10s', target: 0 },  // Rampa de 20 a 0 usuários em 10 segundos
  ],
};
```

Uso de Diferentes Opções de Execução

O K6 permite configurar várias opções para ajustar como os testes são executados:

- **stages**: Define diferentes etapas de carga para o teste.

- **thresholds**: Define condições para determinar o sucesso do teste.

95% das requisições devem ser menores que 500ms

```
export let options = {
  thresholds: {
    'http_req_duration': ['p(95)<500'],
  },
};
```

- **tags:** Adiciona meta-informações aos testes.

Tags: Adiciona meta-informações aos testes.

```
export default function () {
  http.get('https://test.k6.io', { tags: { name: 'HomePage' } });
}
```

CENÁRIOS DE TESTE AVANÇADOS

Criação de Cenários Complexos

Para criar cenários de teste mais realistas e complexos, você pode combinar diferentes padrões de uso, simular comportamentos de usuários variados e incorporar múltiplas etapas de interação.

Cenários Complexos

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

export let options = {
  stages: [
    { duration: '1m', target: 50 },
    { duration: '3m', target: 50 },
    { duration: '1m', target: 0 },
  ],
};

export default function () {
  group('Visit Home Page', function () {
    http.get('https://test.k6.io');
    sleep(1);
  });

  group('Login and Browse', function () {
    let loginRes = http.post(
```

```
        'https://test.k6.io/login',
        { username: 'user', password: 'pass' }
    );
    sleep(1);

    let browseRes = http.get('https://test.k6.io/dashboard');
    sleep(1);
  });
}
```

Uso de Grupos e Tags para Organizar Testes

Os grupos e tags ajudam a organizar e categorizar diferentes partes dos testes, facilitando a análise e a compreensão dos resultados.

Uso de Grupos e Tags para Organizar Testes

```
import { group, check } from 'k6';

export default function () {
  group('Group 1', function () {
    let res = http.get('https://test.k6.io');
    check(
      res,
      { 'status was 200': (r) => r.status === 200 }
    );
  });

  group('Group 2', function () {
    let res = http.get('https://test.k6.io/about');
    check(
      res,
      { 'status was 200': (r) => r.status === 200 }
```

```
    );
  });
}
```

Testes de Carga com Diferentes Padrões de Uso

Simule diferentes padrões de uso para testar como a aplicação responde a diferentes cargas e comportamentos de usuários.

Testes de Carga com Diferentes Padrões de Uso

```
export let options = {
  scenarios: {
    constant_load: {
      executor: 'constant-vus',
      vus: 10,
      duration: '5m',
    },
    ramping_load: {
      executor: 'ramping-vus',
      startVUs: 0,
      stages: [
        { duration: '2m', target: 20 },
        { duration: '3m', target: 20 },
        { duration: '2m', target: 0 },
      ],
    },
  },
};
```

MEDIÇÕES E MÉTRICAS

Métricas de Performance Padrão do K6

O K6 fornece várias métricas padrão que ajudam a avaliar o desempenho dos testes de carga:

- **http_reqs**: Número total de requisições HTTP realizadas.
- **vus_max**: Número máximo de usuários virtuais ativos durante o teste.
- **iterations**: Número de iterações completas realizadas pelos usuários virtuais.
- **http_req_duration**: Tempo de resposta das requisições HTTP.

Personalização e Captura de Métricas Adicionais
Personalize e capture métricas adicionais para obter uma visão mais detalhada do desempenho.

Personalização e Captura de Métricas Adicionais

```
import { Trend } from 'k6/metrics';

let myTrend = new Trend('custom_trend');

export default function () {
  let res = http.get('https://test.k6.io');
  myTrend.add(res.timings.duration);
}
```

Análise Detalhada dos Resultados

Após a execução dos testes, utilize as métricas capturadas para uma análise detalhada dos resultados, identificando gargalos e áreas para melhoria.

Análise Detalhada dos Resultados

```
k6 run --out json=test-results.json script.js
```

Analise o arquivo JSON gerado para insights detalhados.

SIMULAÇÃO DE USUÁRIOS REAIS

Testes com Dados Dinâmicos

Utilize dados dinâmicos para simular interações mais realistas.

Testes com Dados Dinâmicos

```
export default function () {
  let userId = Math.floor(Math.random() * 1000);
  let res = http.get(`https://test.k6.io/user/${userId}`);
  check(res, { 'status was 200': (r) => r.status === 200 });
}
```

Uso de Arquivos e Variáveis de Ambiente

Carregue dados de arquivos externos e utilize variáveis de ambiente para configurar seus testes.

Uso de Arquivos e Variáveis de Ambiente

```
import { SharedArray } from 'k6/data';

let users = new SharedArray('users', function () {
  return JSON.parse(open('./users.json'));
});

export default function () {
  let user = users[Math.floor(Math.random() * users.length)];
  let res = http.get(`https://test.k6.io/user/${user.id}`);
  check(res, { 'status was 200': (r) => r.status === 200 });
}
```

Utilize variáveis de ambiente para configurar seus testes:

variáveis de ambiente

```
K6_USER=user K6_PASS=pass k6 run script.js
```

Integração com APIs Externas

Integre seus testes de carga com APIs externas para simular cenários de uso real.

Integração com APIs Externas

```
export default function () {
  let apiKey = __ENV.API_KEY;
  let res = http.get(`https://api.external.com/data?api_key=${apiKey}`);
  check(res, { 'status was 200': (r) => r.status === 200 });
}
```

INTEGRAÇÃO COM CI/CD

Integrar K6 com pipelines de CI/CD (Continuous Integration/Continuous Deployment) permite executar testes de carga automaticamente em diferentes etapas do ciclo de desenvolvimento.

Exemplos com GitHub Actions

GitHub Actions é uma ferramenta poderosa para automação que pode ser facilmente configurada para executar testes de carga com K6.

Github Action

```yaml
name: k6 Load Test

on: [push]

jobs:
  load-test:
    runs-on: ubuntu-latest

    steps:
    - name: Check out repository
      uses: actions/checkout@v2
```

```
    - name: Set up K6
      uses: k6io/action@v0.2.0

    - name: Run K6 tests
      run: k6 run script.js
```

Boas Práticas para Automação de Testes de Carga

- Isolamento de Ambientes: Utilize ambientes de teste dedicados para evitar interferências.
- Limitação de Recursos: Controle o número de usuários virtuais para não sobrecarregar sistemas em produção.
- Automação Progressiva: Introduza testes de carga gradualmente para monitorar impactos.

INTEGRAÇÃO COM FERRAMENTAS DE MONITORAMENTO

Envio de Métricas para Grafana

Grafana é uma ferramenta popular para visualização de métricas. O K6 pode enviar dados diretamente para Grafana utilizando Prometheus ou InfluxDB como backends.

Integração com Prometheus

Configure K6 para enviar métricas para Prometheus:

Integração com Prometheus

```
export let options = {
  ext: {
    loadimpact: {
      projectID: 123456,
      token: 'YOUR_API_TOKEN'
    },
    system_tags: ['test_run_id', 'execution_type']
  },
  thresholds: {
    'http_req_duration{type:API}': ['p(95)<500'],
  },
  summaryTrendStats: ['avg', 'p(95)', 'p(99)', 'min', 'max'],
};
```

Integração com InfluxDB

Configure K6 para enviar métricas para InfluxDB:

Integração com InfluxDB

```
export let options = {
  ext: {
    influxdb: {
      // Configuration of the InfluxDB v1 client
      // Send test results to InfluxDB
      address: 'http://localhost:8086',
      database: 'k6',
      tags: { environment: 'test' },
      // Make sure the data tag names match your expected data types
      // tagsAsFields: ['region', 'app'],
    },
  },
};
```

Visualização e Análise Contínua dos Resultados

Configurar dashboards no Grafana para visualização contínua dos resultados de testes de carga. Monitore métricas chave como tempos de resposta, taxas de erro e throughput para identificar problemas de performance rapidamente.

INTEGRAÇÃO COM FERRAMENTAS DE MONITORAMENTO

Geração de Relatórios Detalhados

O K6 pode gerar relatórios detalhados que ajudam a entender o desempenho da aplicação sob carga. Utilize formatos de exportação como JSON ou CSV para análise aprofundada.

Geração de Relatórios Detalhados

```
k6 run --out json=output.json script.js
```

Configuração de Alertas Automáticos para Falhas de Performance

Configure alertas automáticos para notificar a equipe quando os testes de carga detectarem problemas de desempenho. Utilize ferramentas como Grafana e Prometheus para configurar esses alertas.

Exemplo de Alerta com Grafana

1. Configure uma regra de alerta no Grafana.
2. Defina as condições de alerta, como "Se a latência média for maior que 500ms".
3. Configure os canais de notificação, como email ou Slack.

Compartilhamento e Interpretação de Resultados com a Equipe

Compartilhe os resultados dos testes de carga com a equipe para interpretação conjunta. Utilize relatórios visuais e dashboards interativos para facilitar a comunicação e a compreensão dos dados.

ESTUDOS DE CASO

Estudos de caso fornecem insights valiosos sobre como o K6 é utilizado em cenários reais. A seguir, apresentamos a análise de alguns projetos que empregaram K6 para testar suas aplicações.

Projeto 1: E-commerce de Grande Porte

Desafios Enfrentados:

- Picos de acesso durante promoções.
- Necessidade de garantir tempo de resposta rápido para milhares de usuários simultâneos.

Soluções Adotadas:

- Criação de cenários de teste que simulavam dias de promoção com carga aumentada.
- Utilização de scripts K6 para simular usuários navegando, adicionando itens ao carrinho e finalizando compras.

Resultados Obtidos e Lições Aprendidas:

- Identificação de gargalos na API de checkout.
- Implementação de otimizações que reduziram o tempo de resposta em 40%.

- A importância de testes contínuos para manter a performance em períodos de alta demanda.

Projeto 2: Plataforma de Streaming

Desafios Enfrentados:

- Alta demanda por streaming simultâneo em horários de pico.
- Garantir uma experiência de usuário fluida com baixa latência.

Soluções Adotadas:

- Testes de carga focados em simulação de múltiplas sessões de streaming.
- Uso de K6 para monitorar latência e taxa de transferência de dados.

Resultados Obtidos e Lições Aprendidas:

- Melhoria da infraestrutura de rede e balanceamento de carga.
- Redução de interrupções de streaming em 25%.
- A necessidade de monitoramento constante e ajustes proativos.

EXEMPLOS PRÁTICOS DE APLICAÇÕES

Teste de Carga em APIs RESTful

APIs RESTful são componentes críticos em muitas aplicações. Testar sua capacidade de lidar com grandes volumes de requisições é essencial.

- A importância de testes contínuos para manter a performance em períodos de alta demanda.

Projeto 2: Plataforma de Streaming

Desafios Enfrentados:

- Alta demanda por streaming simultâneo em horários de pico.
- Garantir uma experiência de usuário fluida com baixa latência.

Soluções Adotadas:

- Testes de carga focados em simulação de múltiplas sessões de streaming.
- Uso de K6 para monitorar latência e taxa de transferência de dados.

Resultados Obtidos e Lições Aprendidas:

- Melhoria da infraestrutura de rede e balanceamento de carga.
- Redução de interrupções de streaming em 25%.
- A necessidade de monitoramento constante e ajustes proativos.

EXEMPLOS PRÁTICOS DE APLICAÇÕES

Teste de Carga em APIs RESTful

APIs RESTful são componentes críticos em muitas aplicações. Testar sua capacidade de lidar com grandes volumes de requisições é essencial.

Exemplo:

Teste de Carga em APIs RESTful

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

export let options = {
  stages: [
    { duration: '30s', target: 20 },
    { duration: '1m', target: 50 },
    { duration: '30s', target: 0 },
  ],
};

export default function () {
  let res = http.get('https://api.example.com/data');
  check(res, { 'status was 200': (r) => r.status === 200 });
  sleep(1);
}
```

Esse script simula usuários enviando requisições GET para uma API RESTful, aumentando gradualmente a carga e depois diminuindo.

Teste de Carga em Aplicações Web Complexas

Aplicações web com múltiplas interações e dependências precisam de testes de carga abrangentes.

Exemplo:

Teste de Carga em Aplicações Web Complexas

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

export let options = {
  stages: [
    { duration: '1m', target: 10 },
    { duration: '3m', target: 30 },
    { duration: '1m', target: 10 },
  ],
};

export default function () {
  group('Homepage', function () {
    http.get('https://example.com');
    sleep(1);
  });

  group('Login', function () {
    http.post(
      'https://example.com/login',
      { username: 'user', password: 'pass' }
    );
    sleep(1);
  });

  group('Browse Items', function () {
    http.get('https://example.com/items');
    sleep(1);
  });
}
```

Esse script organiza a navegação por grupos, simulando usuários navegando, fazendo login e navegando por itens.

Simulação de Diferentes Cenários de Uso

Simule cenários variados para cobrir diferentes comportamentos de usuários e cargas de trabalho.

Exemplo:

Simulação de Diferentes Cenários de Uso

```
import http from 'k6/http';

export let options = {
  scenarios: {
    normal_usage: {
      executor: 'constant-vus',
      vus: 10,
      duration: '10m',
    },
    spike_test: {
      executor: 'ramping-vus',
      startVUs: 10,
      stages: [
        { duration: '2m', target: 50 },
        { duration: '2m', target: 0 },
      ],
    },
  },
};
```

```
export default function () {
  if (__ENV.SCENARIO === 'normal_usage') {
    http.get('https://example.com/normal');
  } else if (__ENV.SCENARIO === 'spike_test') {
    http.get('https://example.com/spike');
  }
}
```

Esse script define diferentes cenários de uso, simulando tanto o uso normal quanto picos de tráfego.

Esses exemplos práticos fornecem um ponto de partida para testar a performance de diferentes tipos de aplicações utilizando K6. Combinando essas práticas com uma análise contínua e ajustes baseados nos resultados dos testes, você pode garantir que suas aplicações sejam robustas e escaláveis, mesmo sob condições de carga intensa.

MELHORES PRÁTICAS PARA TESTES DE CARGA

Planejamento e Preparação de Testes
Planejar e preparar adequadamente os testes de carga é crucial para obter resultados precisos e significativos.

Definição de Objetivos Claros

- Estabeleça Metas: Determine o que você quer alcançar com os testes de carga, como tempos de resposta aceitáveis, taxa de transferência ou limites de usuários simultâneos.

- Cenários de Uso Realistas: Baseie seus testes em cenários que refletem o uso real do sistema.

Seleção de Métricas e KPIs

- Métricas Principais: Inclua métricas como latência, throughput, taxa de erros e tempo de resposta.

- KPIs Específicos: Defina KPIs relevantes para o contexto da aplicação, como desempenho de transações específicas.

Preparação do Ambiente de Teste

- Ambiente Isolado: Utilize um ambiente de teste que simule as condições de produção, mas isolado para evitar interferências.

- Recursos Adequados: Assegure-se de que os recursos de hardware e rede são suficientes para suportar a carga esperada.

Execução e Análise Contínua

A execução de testes de carga deve ser acompanhada por uma análise contínua dos resultados para ajustes rápidos e precisos.

Monitoramento em Tempo Real

- Ferramentas de Monitoramento: Utilize ferramentas como Grafana, Prometheus ou InfluxDB para monitorar métricas em tempo real.

- Alertas: Configure alertas para detectar problemas imediatamente durante a execução dos testes.

Análise Pós-Teste

- Relatórios Detalhados: Gere relatórios detalhados com todas as métricas relevantes.

- Revisão de Resultados: Analise os resultados para identificar padrões, gargalos e áreas de melhoria.

Otimização de Scripts e Recursos

Otimizar scripts e gerenciar recursos eficientemente ajuda a garantir testes de carga mais eficazes.

Refatoração de Scripts

- Código Limpo: Mantenha scripts de teste limpos e organizados.

- Funções Reutilizáveis: Utilize funções e módulos reutilizáveis para evitar duplicação de código.

Gestão de Recursos

- Alocação de Recursos: Gerencie a alocação de recursos de maneira eficiente para evitar sobrecargas desnecessárias.

- Uso de Dados Dinâmicos: Utilize dados dinâmicos para simular interações de usuários reais.

EVITANDO ARMADILHAS COMUNS

Identificação de Gargalos

- Análise de Métricas: Utilize métricas para identificar onde os gargalos ocorrem, como tempos de resposta elevados ou altas taxas de erro.

- Testes Incrementais: Aumente a carga gradualmente para identificar o ponto em que os problemas começam a surgir.

Resolução de Gargalos

- Ajustes de Configuração: Ajuste configurações de servidores, bancos de dados e redes.

- Otimização de Código: Revise e otimize o código da aplicação para melhorar a performance.

Manejo de Erros e Exceções em Testes

Gerenciar erros e exceções durante os testes de carga é crucial para obter resultados precisos.

Captura de Erros

- Logging Detalhado: Implemente logs detalhados para capturar erros e exceções.

- Monitoramento de Exceções: Utilize ferramentas de monitoramento para rastrear e analisar exceções.

Resolução de Erros

- Diagnóstico de Problemas: Analise logs e dados de monitoramento para diagnosticar a causa raiz dos erros.

- Correção e Reexecução: Corrija os problemas identificados e reexecute os testes para verificar a eficácia das correções.

Estratégias para Testes de Longa Duração

Testes de longa duração ajudam a identificar problemas que podem não ser visíveis em testes de curta duração.

Configuração de Testes de Longa Duração

- Planejamento: Planeje testes que duram várias horas ou dias para simular uso contínuo.

- Recursos: Assegure-se de que os recursos do sistema são suficientes para suportar testes prolongados.

Análise e Interpretação de Resultados

- Métricas de Longa Duração: Foque em métricas que indicam degradação ao longo do tempo, como aumento de latência ou uso de memória.

- Trends e Padrões: Identifique trends e padrões que possam indicar problemas de performance a longo prazo.

Seguindo essas práticas e estratégias, você estará bem equipado para conduzir testes de carga eficazes com K6, garantindo que suas aplicações possam suportar altos volumes de usuários e operações sem comprometer a performance ou a estabilidade.

CONCLUSÃO

A realização contínua de testes de carga é essencial para garantir que as aplicações mantenham um desempenho estável e robusto, mesmo sob condições de uso intensivo.

Testes regulares ajudam a identificar problemas antes que eles afetem os usuários finais, permitindo uma resposta proativa e a melhoria contínua da infraestrutura e do código.

Incentivamos você a continuar experimentando e inovando no campo dos testes de carga. Testes contínuos e a adoção de novas práticas e ferramentas ajudam a garantir que suas aplicações estejam sempre preparadas para atender às demandas de seus usuários.

Esperamos que este livro tenha fornecido uma base sólida e inspiradora para o uso do K6 em seus projetos de teste de carga. Continue explorando, aprendendo e melhorando suas habilidades para alcançar a excelência em performance e escalabilidade de aplicações.

RESUMO DOS APRENDIZADOS

Ao longo deste livro, exploramos de maneira abrangente o uso do K6 para a realização de testes de carga. A seguir, recapitulamos os pontos principais:

- **Introdução aos Testes de Carga**: Entendemos a importância dos testes de carga, cenários típicos onde são críticos e os desafios comuns enfrentados durante a execução desses testes.

- **Visão Geral do K6**: Discutimos a história, características e benefícios do K6, bem como sua comparação com outras ferramentas de teste de carga.

- **Instalação e Configuração**: Aprendemos a instalar e configurar o K6 em diferentes sistemas operacionais e a realizar os primeiros passos com a ferramenta.

- **Primeiros Testes com K6**: Exploramos a estrutura básica de um script K6, executamos testes simples e interpretamos os resultados iniciais.

- **Linguagem de Script do K6**: Abordamos os fundamentos do JavaScript aplicados ao K6, as principais funções e APIs, e a organização de scripts.

CONCLUSÃO

- **Escrevendo Testes Básicos**: Criamos usuários virtuais, simulamos cenários de navegação simples e validamos respostas.

- **Parâmetros e Configurações de Teste**: Configuramos a duração e intensidade dos testes, utilizando diferentes opções de execução.

- **Cenários de Teste Avançados**: Desenvolvemos cenários complexos, utilizamos grupos e tags, e realizamos testes de carga com diferentes padrões de uso.

- **Medições e Métricas**: Examinamos as métricas de performance padrão do K6, personalizamos e capturamos métricas adicionais, e analisamos os resultados detalhadamente.

- **Simulação de Usuários Reais:** Realizamos testes com dados dinâmicos, utilizamos arquivos e variáveis de ambiente, e integramos com APIs externas.

- **Integração com CI/CD**: Configuramos K6 em pipelines de CI/CD, exemplificamos com GitHub Actions e seguimos boas práticas para automação de testes de carga.

- **Integração com Ferramentas de Monitoramento**: Enviamos métricas para Grafana, integramos com Prometheus e InfluxDB, e visualizamos e analisamos continuamente os resultados.

- **Relatórios e Alertas**: Geramos relatórios detalhados, configuramos alertas automáticos para falhas de performance e compartilhamos e interpretamos os resultados com a equipe.

- **Estudos de Caso Reais:** Analisamos projetos reais que utilizaram K6, discutimos desafios enfrentados, soluções adotadas, resultados obtidos e lições aprendidas.

- **Exemplos Práticos de Aplicações**: Testamos carga em APIs RESTful e aplicações web complexas, simulando diferentes cenários de uso.

- **Melhores Práticas para Testes de Carga**: Planejamos e preparamos testes, executamos e analisamos continuamente, e otimizamos scripts e recursos.

- **Evitando Armadilhas Comuns**: Identificamos e resolvemos gargalos, manejamos erros e exceções em testes e desenvolvemos estratégias para testes de longa duração.

OBRIGADO!

ME SIGA NAS REDES SOCIAIS

WEDERSON-MACHADO

@TRILHADEQUALIDADE

TRILHADEQUALIDADE.COM.BR