Módulo 4 – Fundamentos Essenciais do k6

Objetivo do módulo

Compreender como o k6 executa um teste de performance, o que acontece em cada fase do ciclo de vida e como escrever o primeiro script completo e funcional.

---

1. Entendendo o Ciclo de Vida no k6

Todo teste no k6 segue **um fluxo de execução bem definido**, dividido em quatro fases:

Fase	Descrição	Executa uma vez?
init	Onde variáveis, imports e configurações são carregadas.	✅ Sim
setup	Fase opcional antes da execução principal, usada para preparar dados, autenticar ou buscar tokens.	✅ Sim
default	Onde ocorre o teste propriamente dito. Cada usuário virtual (VU) executa essa função repetidamente.	🔁 Repetidamente
teardown	Fase opcional após o teste. Usada para limpar dados, enviar logs ou gerar relatórios.	✅ Sim

---

Como o ciclo funciona na prática

1. **Init** é executado **uma única vez** por instância de execução. Tudo que for declarado fora das funções (`import`, `options`, variáveis globais) é carregado aqui.
2. **Setup** é executado **antes do teste principal**, geralmente para preparar dados, criar massa de teste ou autenticar.
3. **Default** é o **núcleo do teste**, onde cada usuário virtual executa as requisições configuradas.

4. **Teardown** é executado **após o teste**, e serve para limpar dados ou enviar logs.

---

2. Estrutura básica de um script

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

export const options = {
  vus: 5,            // Quantos usuários virtuais
  duration: '5s',    // Tempo total de execução
};

export default function () {
  const res = http.get('http://localhost:3000/produtos');

  check(res, {
    'status é 200': (r) => r.status === 200,
  });

  sleep(1); // pausa de 1s entre iterações
}
```

Explicando passo a passo

- `import http from 'k6/http'` : importa o módulo de requisições HTTP.
- `import { check, sleep } from 'k6'` : traz funções auxiliares para validação e pausas.
- `options` : define as **configurações do teste**, como quantidade de usuários e duração.
- `default` : função principal onde o teste acontece.
- `http.get()` : envia uma requisição GET.
- `check()` : valida a resposta (exemplo: status 200).
- `sleep()` : simula o tempo de espera entre ações humanas.

Entendendo o comportamento

Com `vus: 5` e `duration: '5s'`, o k6 cria **5 usuários virtuais** que fazem requisições simultaneamente por **5 segundos**.

Isso gera múltiplas execuções da função `default`, simulando um pequeno tráfego real.

---

3. Exemplo com `setup` e `teardown`

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

export const options = {
  vus: 5,
  duration: '10s',
};

export function setup() {
  console.log('Iniciando o teste...');
  return { baseUrl: 'http://localhost:3000' };
}

export default function (data) {
  http.get(`${data.baseUrl}/produtos`);
  sleep(1);
}

export function teardown() {
  console.log('Encerrando o teste.');
}
```

Explicação

- `setup()` é executado **uma única vez**, antes do teste, e pode **retornar dados** para uso dentro da função principal.
- O parâmetro `data` em `default` recebe o retorno de `setup()`.
- `teardown()` roda **uma única vez ao final**, ótimo para logs, limpeza ou relatórios.