Grafana k6

a load testing tool for developers

Troy Miles, 19 May 2023

What is k6?

- An open-source load testing tool that makes performance testing easy and productive for engineering teams
- It is free, developer-centric, and extensible
- Developed by Grafana Lab and the community
- k6 is written in Go and has an embedding JavaScript runtime

Key Features

- CLI tool with developer-friendly APIs
- Scripting in JavaScript ES2016/ES6, with support for local and remote modules
- Checks and Thresholds for goal-oriented, automation-friendly load testing

Use Cases

- load testing
- performance testing
- *chaos and resilience testing*
- *performance and synthetic monitoring*

Load Testing Manifesto

- Simple testing is better than no testing
- Load testing should be goal oriented
- Load testing by developers
- Developer experience is super important
- Load test in a pre-production environment

What k6 does not

- Does not run natively in a browser
- Does not run in NodeJS

Supported Environments

- MacOS
- Windows
- Docker
- Linux

```
// MacOS

brew install k6

// Windows

choco install k6

winget install k6

// Docker

docker pull grafana/k6

// Linux*
```

Definitions

- VUs - virtual users
- options - configure test-run behavior, replaces CLI parameters
- stages - allows for ramping up/down the number of VUs
- metrics - measure how a system performs under test conditions
- checks - validate the boolean conditions in your test
- thresholds - pass/fail criteria that you define in your test metrics

Test Lifecycle

- initialization *(required)* runs once per VU
- setup *(optional)*
- VU code *(required)* runs parallel as many times as you have VUs
- teardown *(optional)*

```
// the four lifecycle stages

// 1. init code

export function setup() {
    // 2. setup code
}

export default function (data) {
    // 3. VU code
}

export function teardown(data) {
    // 4. teardown code
}
```

```
// (script.js) a simple script

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

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

```
// running the script from the command line

k6 run script.js
```

```
k6 run script.js
          /\      |‾‾| /‾‾/   /‾‾/
     /\  /  \     |  |/  /   /  /
    /  \/    \    |     (   /   ‾‾\
   /          \   |  |\  \ |  (‾)  |
  / __________ \  |__| \__\ \_____/ .io

  execution: local
     script: script.js
     output: -

  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)


     data_received..................: 17 kB 11 kB/s
     data_sent......................: 438 B 280 B/s
     http_req_blocked...............: avg=476.13ms min=476.13ms med=476.13ms max=476.13ms p(90)=476.13ms p(95)=476.13ms
     http_req_connecting............: avg=80.06ms  min=80.06ms  med=80.06ms  max=80.06ms  p(90)=80.06ms  p(95)=80.06ms
     http_req_duration..............: avg=83.19ms  min=83.19ms  med=83.19ms  max=83.19ms  p(90)=83.19ms  p(95)=83.19ms
       { expected_response:true }...: avg=83.19ms  min=83.19ms  med=83.19ms  max=83.19ms  p(90)=83.19ms  p(95)=83.19ms
     http_req_failed................: 0.00% ✓ 0       ✗ 1
     http_req_receiving.............: avg=236µs    min=236µs    med=236µs    max=236µs    p(90)=236µs    p(95)=236µs
     http_req_sending...............: avg=86µs     min=86µs     med=86µs     max=86µs     p(90)=86µs     p(95)=86µs
     http_req_tls_handshaking.......: avg=171.63ms min=171.63ms med=171.63ms max=171.63ms p(90)=171.63ms p(95)=171.63ms
     http_req_waiting...............: avg=82.87ms  min=82.87ms  med=82.87ms  max=82.87ms  p(90)=82.87ms  p(95)=82.87ms
     http_reqs......................: 1     0.640281/s
     iteration_duration.............: avg=1.56s    min=1.56s    med=1.56s    max=1.56s    p(90)=1.56s    p(95)=1.56s
     iterations.....................: 1     0.640281/s
     vus............................: 1     min=1      max=1
     vus_max........................: 1     min=1      max=1


running (00m01.6s), 0/1 VUs, 1 complete and 0 interrupted iterations
```

Command Line Options

- —vus - the number of Virtual Users (VUs)
- — duration - how long to run the test
- - e - Temporarily set an environment variable

```
// running the script from the command line with some options

k6 run --vus 10 --duration 30s script.js
```

```
❯ k6 run --vus 10 --duration 30s script.js
          /\      |‾‾| /‾‾/   /‾‾/
     /\  /  \     |  |/  /   /  /
    /  \/    \    |     (   /   ‾‾\
   /          \   |  |\  \ |  (‾)  |
  / __________ \  |__| \__\ \_____/ .io

  execution: local
     script: script.js
     output: -

  scenarios: (100.00%) 1 scenario, 10 max VUs, 1m0s max duration (incl. graceful stop):
           * default: 10 looping VUs for 30s (gracefulStop: 30s)


     data_received..................: 3.3 MB 106 kB/s
     data_sent......................: 31 kB  993 B/s
     http_req_blocked...............: avg=8.04ms  min=1µs     med=8µs     max=232.28ms p(90)=19µs    p(95)=24.04µs
     http_req_connecting............: avg=3.17ms  min=0s      med=0s      max=91.28ms  p(90)=0s      p(95)=0s
     http_req_duration..............: avg=91.13ms min=75.44ms med=84.1ms  max=167.56ms p(90)=98.05ms p(95)=157.82ms
       { expected_response:true }...: avg=91.13ms min=75.44ms med=84.1ms  max=167.56ms p(90)=98.05ms p(95)=157.82ms
     http_req_failed................: 0.00%  ✓ 0        ✗ 280
     http_req_receiving.............: avg=7.44ms  min=19µs    med=100µs   max=84.81ms  p(90)=1.38ms  p(95)=76.08ms
     http_req_sending...............: avg=35.64µs min=6µs     med=28.5µs  max=528µs    p(90)=67.1µs  p(95)=75.09µs
     http_req_tls_handshaking.......: avg=3.65ms  min=0s      med=0s      max=110.57ms p(90)=0s      p(95)=0s
     http_req_waiting...............: avg=83.65ms min=75.22ms med=83.08ms max=129.9ms  p(90)=88.58ms p(95)=90.18ms
     http_reqs......................: 280    9.019212/s
     iteration_duration.............: avg=1.1s    min=1.07s   med=1.08s   max=1.31s    p(90)=1.15s   p(95)=1.16s
     iterations.....................: 280    9.019212/s
     vus............................: 1      min=1      max=10
     vus_max........................: 10     min=10     max=10


running (0m31.0s), 00/10 VUs, 280 complete and 0 interrupted iterations
default ✓ [======================================] 10 VUs  30s
```

```
// (script2.js) a script with the CLI options embedded

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

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


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

```
// the four lifecycle stages

// 1. init code

export function setup() {
    // 2. setup code
}

export default function (data) {
    // 3. VU code
}

export function teardown(data) {
    // 4. teardown code
}
```

Checks

- You can validate boolean conditions with *check*
- Import check from the k6 module
- You can test one or more conditions
- Each test consists of a string and a function
- The string names the test
- And the function tests it
- A failed check DOES NOT end the run

check demo

Thresholds

- Thresholds are pass/fail criteria you define
- If the test results don’t meet your criteria, the test fails
- Keep in mind that thresholds are based on the entire test run

threshold demo

Environment Variables

- k6 supports environment variables
- This allows us to run the same script in different environments (duh)
- basic format is: ${__ENV.<Variable name>}
- example: ${__ENV.HOSTNAME}

environment demo

Future Directions

- Creating our goals
- Integrating into our BFF pipeline
- Seeing if we can use the GitHub Action

Things for Web Teams

- k6 has a browser module
- It is still considered experimental
- It includes lots of essential browser metrics like:
  - browser_dom_content_loaded
  - browser_first_paint
  - browser_loaded
  - etc.

URLs

- https://k6.io/docs/
- https://k6.io/our-beliefs/
- https://github.com/marketplace/actions/k6-load-test
- https://github.com/grafana/k6-action
- https://k6.io/docs/using-k6-browser/overview/
- https://web.dev/vitals/#core-web-vitals
- https://k6.io/docs/javascript-api/k6-http/response/

Summary

- Grafana k6 is an easy-to-use load-testing tool
- It integrates readily into your CI/CD pipeline
- It uses JavaScript for scripting
- And is written in high performance Go for speed