

K6 Overview

Installation

1. Download and install the latest .msi package.
2. Open command prompt and run: `k6`

If installation was successful, short k6 manual should be displayed:

[illegible]

Writing a simple k6 script

1. Create `google_search.js` file.
2. Paste the following code:

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

export default function () {
  const search_query = 'k6 installing';
  const search_url = `https://www.google.com/search?q=${search_query}`;

  const response = http.get('https://test.k6.io');
  console.log(response.body);
}
```

3. Open command prompt and run `k6 run google_search.js`

This script will execute search request in google. It terminal output you should see HTML response code, that will contain search results.

Using checks

1. Update `google_search.js` with the following code:

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

function check_google_response(res) {
  check(res, {
    'status was 200': (r) => r.status == 200,
    'response body wasn\'t empty': (r) => r.body.length > 0,
  });
}

export default function () {
  const search_query = 'k6 installing';
  const search_url = `https://www.google.com/search?q=${search_query}`;

  const response = http.get('https://test.k6.io');
  check_google_response(response);
}
```

It this code we added a function that will check response status and body length.

2. Open command prompt and run `k6 run google_search.js`

You should see checks result in terminal output.

```

C:\Users\tonib\Desktop>k6 run google_search.js

      M R K
     /  /  \
    /_____\  .io

execution: local
  script: google_search.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)

running (00m00.6s), 0/1 VUs, 1 complete and 0 interrupted iterations
default ✓ [=====] 1 VUs  00m00.6s/10m0s  1/1 iters, 1 per VU

✓ status was 200
✓ response body wasn't empty

checks.....: 100.00% ✓ 2 x 0
data_received.....: 17 kB  30 kB/s
data_sent.....: 693 B  1.2 kB/s
http_req_blocked.....: avg=441.43ms min=441.43ms med=441.43ms max=441.43ms p(90)=441.43ms p(95)=441.43ms
http_req_connecting.....: avg=126.8ms min=126.8ms med=126.8ms max=126.8ms p(90)=126.8ms p(95)=126.8ms
http_req_duration.....: avg=125.96ms min=125.96ms med=125.96ms max=125.96ms p(90)=125.96ms p(95)=125.96ms
  { expected_response:true }...: avg=125.96ms min=125.96ms med=125.96ms max=125.96ms p(90)=125.96ms p(95)=125.96ms
http_req_failed.....: 0.00% ✓ 0 x 1
http_req_receiving.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_sending.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_tls_handshaking.....: avg=313.62ms min=313.62ms med=313.62ms max=313.62ms p(90)=313.62ms p(95)=313.62ms
http_req_waiting.....: avg=125.96ms min=125.96ms med=125.96ms max=125.96ms p(90)=125.96ms p(95)=125.96ms
http_reqs.....: 1 1.753137/s
iteration_duration.....: avg=570.4ms min=570.4ms med=570.4ms max=570.4ms p(90)=570.4ms p(95)=570.4ms
iterations.....: 1 1.753137/s

```

Using options

If we want to implement a load testing, we should use VUs (Virtual Users), which are users that will execute requests in parallel.

1. Open command prompt and run `k6 run google_search.js --vus 5 --duration 10s`

This options will create 5 virtual users, which will execute requests for 10 seconds.


```
    check_google_response(response);
}
```

Ramping behavior using stages

You can also configure ramping behavior by adding stages option.

1. Update `google_search.js` with the following code:

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

export let options = {
  stages: [
    { duration: '5s', target: 5 },
    { duration: '10s', target: 10 },
    { duration: '5s', target: 5 },
  ],
};

function check_google_response(res) {
  check(res, {
    'status was 200': (r) => r.status === 200,
    'response body wasn\'t empty': (r) => r.body.length > 0,
  });
}

export default function () {
  const search_query = 'k6 installing';
  const search_url = `https://www.google.com/search?q=${search_query}`;

  const response = http.get('https://test.k6.io');
  check_google_response(response);
}
```

2. Open command prompt and run `k6 run google_search.js`

In this case, test running time will be 20 seconds divided into three stages:

1. 0-5s - will gradually increase the number of VUs from 0 to 5
2. 5-15 - will gradually increase the number of VUs from 5 to 10
3. 15-20 - will gradually reduce the number of VUs from 10 to 5