

量化網站的極限

Allen

The k6 logo is located in the bottom right corner of the slide. It consists of a purple, jagged mountain-like shape. Inside this shape, the text "k6" is written in a bold, black, sans-serif font.

k6

目錄

- 什麼是性能測試？
- 什麼是 K6？
- 簡單說明 K6 的各種功能

什麼是性能測試 (Performance Testing) ?

利用對系統施加壓力，並觀察系統的運行狀況，以此來探測出系統的極限


性能測試涵蓋的範圍非常廣泛，其中又可以分為：

- 負載測試 (Load Testing)
- 尖峰測試 (Spike Testing)
- 壓力測試 (Stress Testing)
- 長時間穩定性測試 (Soak Testing / Endurance Testing)
- 可擴展性測試 (Scalability Testing)

什麼是 K6？

K6 是一個由 Grafana 推出並維護的負載測試工具

你可以使用 JavaScript 撰寫測試的腳本，並對系統進行負載測試



[OPEN SOURCE](#)
[GRAFANA CLOUD K6](#)
[PRICING](#)
[DOCUMENTATION](#)
[ABOUT ▾](#)

[SIGN IN ▾](#)
[SIGN UP](#)

The best developer experience for load testing

Open source and SaaS for engineering teams

[OPEN SOURCE >](#)
[GRAFANA CLOUD K6 >](#)



編寫一個壓力測試

首先我們建立一個 `test.js` 檔案, 並寫入以下內容

```
import http from "k6/http";

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

export default function () {
  http.get("https://docfunc.com");
}
```

編寫一個壓力測試

首先我們建立一個 `test.js` 檔案, 並寫入以下內容

vus: 虛擬用戶, 即模擬多少用戶訪問系統

duration: 測試總時間

執行測試的主要邏輯

```
import http from "k6/http";

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

export default function () {
  http.get("https://docfunc.com");
}
```

執行測試

使用 K6 執行剛剛寫好的 test.js 腳本

```
#!/bin/bash
```

```
k6 run test.js
```

測試結果

```
execution: local
  script: 01-test.js
  output: -
```

```
scenarios: (100.00%) 1 scenario, 10 max VUs, 40s max duration (incl. graceful stop):
    * default: 10 looping VUs for 10s (gracefulStop: 30s)
```

```

data_received.....: 17 MB 1.7 MB/s
data_sent.....: 56 kB 5.4 kB/s
http_req_blocked.....: avg=62.34ms min=0s med=1µs max=1.2s p(90)=3µs p(95)=586.15ms
http_req_connecting.....: avg=3.19ms min=0s med=0s max=80.27ms p(90)=0s p(95)=22.82ms
http_req_duration.....: avg=470.43ms min=255.97ms med=423.93ms max=1.09s p(90)=663.75ms p(95)=752.57ms
  { expected_response:true }...: avg=470.43ms min=255.97ms med=423.93ms max=1.09s p(90)=663.75ms p(95)=752.57ms
http_req_failed.....: 0.00% ✓ 0 × 191
http_req_receiving.....: avg=23.43ms min=2.54ms med=6.68ms max=283.74ms p(90)=9.45ms p(95)=191.09ms
http_req_sending.....: avg=69.84µs min=29µs med=65µs max=315µs p(90)=3µs p(95)=116.49µs
http_req_tls_handshaking.....: avg=4.39ms min=0s med=0s max=102.6ms p(90)=0s p(95)=31.32ms
http_req_waiting.....: avg=446.92ms min=248.48ms med=409.72ms max=1.08s p(90)=533.08ms p(95)=699.51ms
http_reqs.....: 191 18.484516/s
iteration_duration.....: avg=533.17ms min=256.27ms med=424.24ms max=2.29s p(90)=664.06ms p(95)=1.34s
iterations.....: 191 18.484516/s
vus.....: 10 min=10 max=10
vus_max.....: 10 min=10 max=10

```

```
running (10.3s), 00/10 VUs, 191 complete and 0 interrupted iterations
default ✓ [=====] 10 VUs 10s
```


測試結果說明

- `data_received`: 從「受測目標」接收到的傳輸量
- `data_sent`: 發送到「受測目標」的傳輸量
- `http_req_blocked`: 在發出要求前 TCP 連線的等待時間 (等候有空的 TCP 連線)
- `http_req_connecting`: 建立到「測試目標」的 TCP 連線時間
- `http_req_duration`: 整個 HTTP 的往返時間 (不含 DNS 查詢時間)
- `http_req_failed`: 失敗率
- `http_req_receiving`: 從「受測目標」接收到數據的時間
- `http_req_sending`: 發送數據到「受測目標」的傳輸時間
- `http_req_tls_handshaking`: 進行 TLS 交握的時間
- `http_req_waiting`: 等待伺服器回應的時間, 也就是俗稱的 time to first byte (TTFB) 時間
- `http_reqs`: 總共發出了多少 HTTP 要求
- `iteration_duration`: 一次完整 iteration 的時間
- `iterations`: 完成幾次 iteration

console.log()

你可以使用 `console.log()` 來印出程式碼中的值



```
import http from "k6/http";

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

export default function () {
  const res = http.get("https://docfunc.com");

  console.log(res.status);
}
```

Checks

可以使用 `check()` 檢查 HTTP 回應是否成功。在測試完後，測試結果會多一個 checks 結果

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

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

export default function () {
  const res = http.get("https://docfunc.com");

  check(res, {
    "status is 200": (r) => r.status === 200,
  });
}
```

✓ status is 200

checks.....	100.00%	✓ 215	× 0
data_received.....	20 MB	1.9 MB/s	
data_sent.....	58 kB	5.6 kB/s	
http_req_blocked.....	avg=10.42ms	min=0s	
http_req_connecting.....	avg=2.4ms	min=0s	
http_req_duration.....	avg=464.62ms	min=270.12ms	

Cookie

你可以在發送 HTTP 請求時附上 cookie，以此來模擬已經登入的狀態

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

export default function () {
  const res = http.get("https://docfunc.com/posts/create", {
    cookies: {
      docfunc_session: "...",
    },
  });

  check(res, {
    "status is 200": (r) => r.status === 200,
  });
}
```

Stage

我們可以對情境進行更詳細的設定，例如

rame-up 與 rame-down

- 花 10 秒的時間，從 1 VU 慢慢成長到 20 VUs
- 然後再花 10 秒的時間，漸漸的從 20 VUs 降低到 10 VUs
- 最後花 10 秒的時間，從 10 VUs 降低到 0 VUs

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

export const options = {
  stages: [
    // Ramp-up from 1 to 20 VUs in 10s
    { duration: "10s", target: 20 },
    // Ramp-down from 20 to 10 VUs for 10s
    { duration: "10s", target: 10 },
    // Ramp-down from 10 to 0 VUs for 10s
    { duration: "10s", target: 0 },
  ],
};

export default function () {
  const res = http.get("https://docfunc.com/");

  check(res, { "status was 200": (r) => r.status === 200 });
}
```

最後來個小 Demo

嘗試使用 K6 來進行登入

參考資料

- [什麼是 Performance test / Loading test](#)
- [Software performance testing](#)
- [軟體測試的種類 - 番外篇](#)
- [Performance Testing | Software Testing](#)
- [快速上手 Grafana k6 壓力測試工具](#)