Load Testing with k6

Roberson Liou



歡迎參加我們的每週四固定聚會

關於我

- Backend, CICD, Infra
- Communities
  - Microsoft MVP(2020-2022)
  - twMVC 核心成員
  - DevOps Taiwan 志工
  - GitKraken Ambassador
- BLOG - 工程良田的小球場



大綱

- k6 簡介
- 基本操作
- 進階 & 補充

https://github.com/robersonliou/k6-playground

tw MVC
https://mvc.tw

k6 簡介

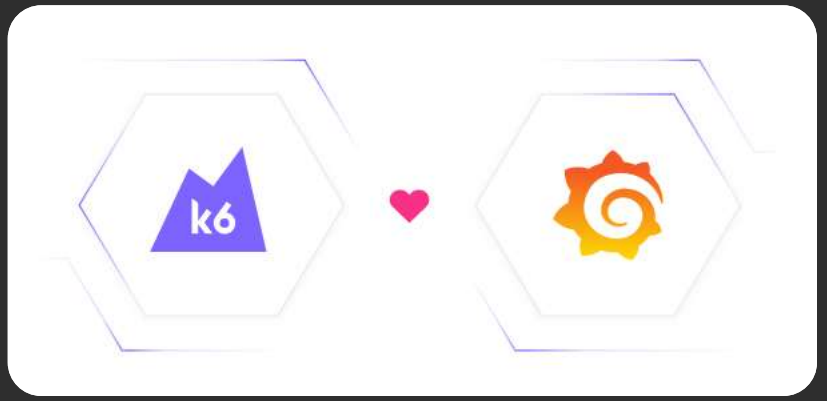


工程良田的小球場 – [Tools] - k6 壓測工具簡介

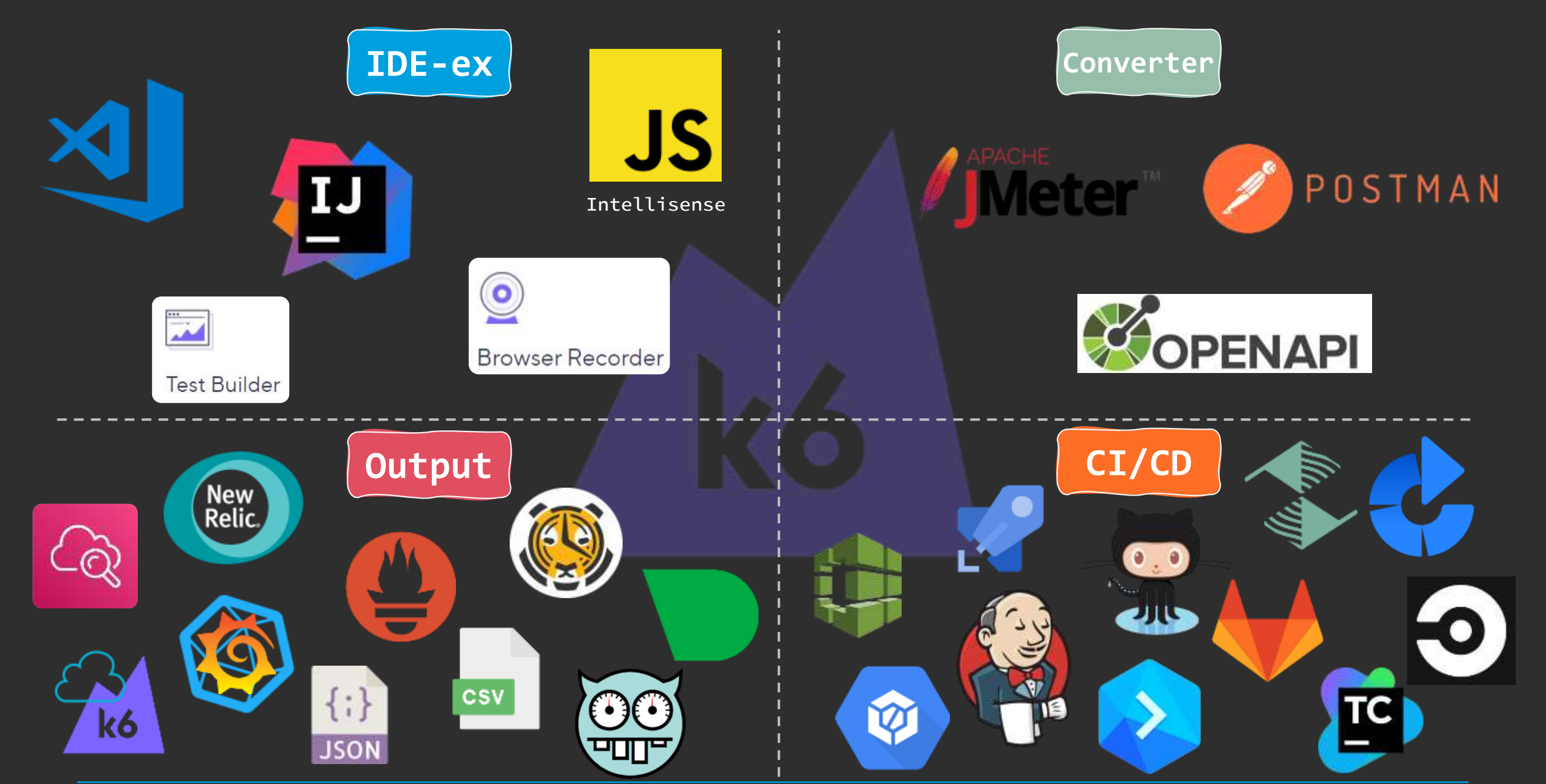


What is k6?

- Open source / 壓測工具（前身為 LoadImpact）
- 2021 年加入 Grafana Labs
- 號稱開發體驗最好的壓測工具
- 使用 ES6 JS 撰寫 / CLI 執行
- 豐富的生態圈整合



tw MVC
https://mvc.tw



https://k6.io/docs/integrations/

k6 is built for engineering teams



Developers & SDETs

- The k6 API and CLI are easy-to-use, flexible and powerful.
- Write tests in Javascript that mimic real-world scenarios.



DevOps & SREs

- Automate your tests to ensure that your infrastructure and application remain performant.
- Configure SLOs in your k6 script to test the health of your services.



Software Testers & QAs

- Create tests faster with our test builder, recorder or converters (JMeter, Postman, Swagger).
- Extensive documentation, awesome community and first class support.

https://k6.io/

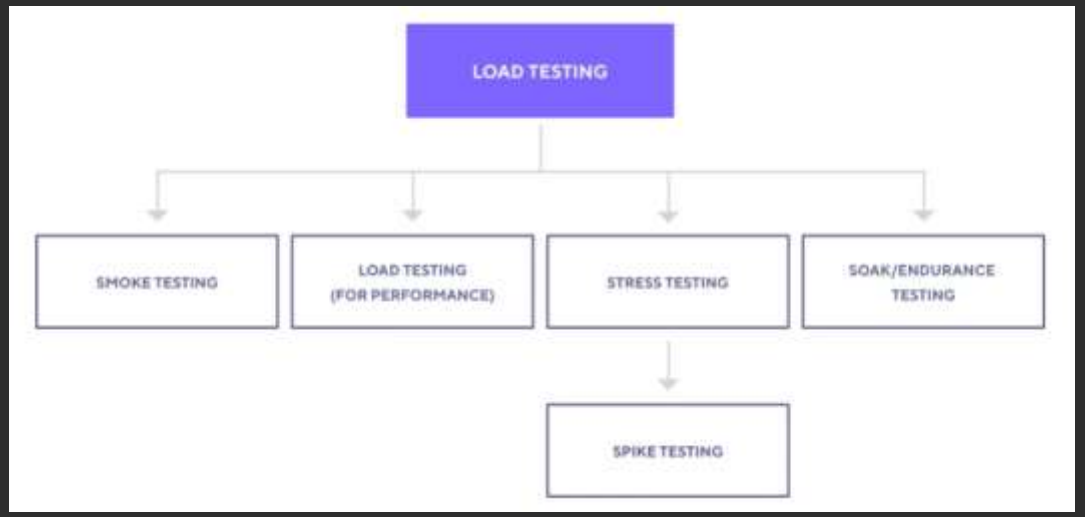
k6 - Load Testing Manifesto

- 有測總比沒測好
- 壓測應該是目標導向
- 壓測應該是開發者來執行
- 開發者體驗是非常重要的
- 在 pre-production 環境進行壓測

https://k6.io/our-beliefs/

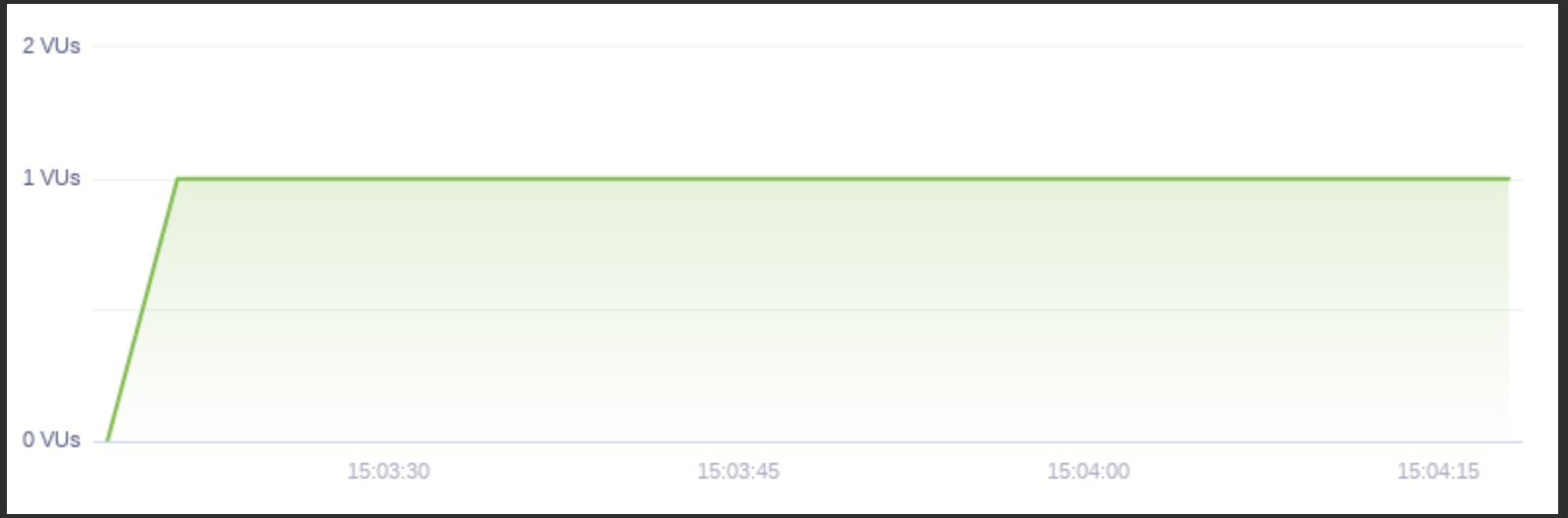
效能測試種類

- 冒煙測試（Smoke Testing）
- 負載測試（Load Testing）
- 壓力測試（Stress Testing）
- 浸泡測試（Soak Testing）



冒煙測試（Smoke Testing）

- 用最小的壓力執行測試
- 驗證測試程式及待測系統功能是否正常



https://k6.io/docs/test-types/smoke-testing/

負載測試（Load Testing）

- 驗證一般/尖峰時段下的系統行為
- 可作為持續驗證的效能指標
- 可搭配 CD pipeline 使用



https://k6.io/docs/test-types/load-testing/

壓力測試（Stress Testing）

- 測試系統在極端條件下的表現
- 測試系統的最大乘載量及自動擴展能力
- 測試系統發生錯誤時自我修復的能力



https://k6.io/docs/test-types/stress-testing/

尖峰衝擊測試（Spike Testing）

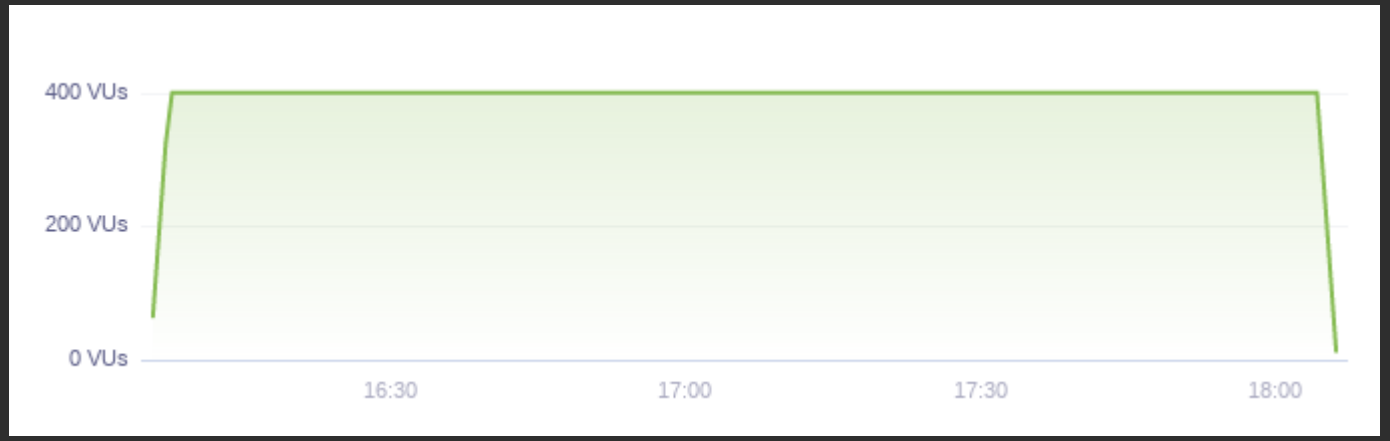
- 壓力測試的變形
- 瞬間大流量
- 典型情境：搶票



https://k6.io/docs/test-types/stress-testing/#spike-testing

浸泡測試（Soak Testing）

- 長期浸在穩定流量之上（80% capacity）
- 觀察系統是否能穩定運作
  - EX: memore leak /port leakage/race condition/disk space
- 測試第三方服務在一定流量下的穩定度
- 記得做好成本評估



https://k6.io/docs/test-types/soak-testing/

支援的協定標準

- HTTP/1.1, HTTP/2
- WebSockets
- gRPC
- Others: SQL, Kafka, Redis…(xk6 community)

基本操作



安裝方式

- 支援跨平台（Linux/MacOS/Windows）
- Linux：apt-get/dnf/yum
- MacOS：homebrew
- Windows：choco/winget/.msi
- Docker

https://k6.io/docs/getting-started/installation/

安裝 - WSL Ubuntu-18.04

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys C5AD17C747E3415A3642D57D77C6C491D6AC1D69
echo "deb https://dl.k6.io/deb stable main" | sudo tee /etc/apt/sources.list.d/k6.list
sudo apt-get update
sudo apt-get install k6
```

https://k6.io/docs/getting-started/installation/

Hello k6

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

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

```
$ k6 run hello-k6.js
```



Hello k6 - Output

```
execution: local
   script: hello-k6.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)
```

基本資訊

```
running (00m02.1s), 0/1 VUs, 1 complete and 0 interrupted iterations
default ✓ [======================================] 1 VUs  00m02.1s/10m0s  1/1 iters, 1 per VU
```

Metrics

```
data_received..................: 17 kB 8.2 kB/s
data_sent......................: 511 B 249 B/s
http_req_blocked...............: avg=818.15ms min=818.15ms med=818.15ms max=818.15ms p(90)=818.15ms p(95)=818.15ms
http_req_connecting............: avg=234.19ms min=234.19ms med=234.19ms max=234.19ms p(90)=234.19ms p(95)=234.19ms
http_req_duration..............: avg=235.52ms min=235.52ms med=235.52ms max=235.52ms p(90)=235.52ms p(95)=235.52ms
  { expected_response:true }...: avg=235.52ms min=235.52ms med=235.52ms max=235.52ms p(90)=235.52ms p(95)=235.52ms
http_req_failed................: 0.00% ✓ 0        ✗ 1
http_req_receiving.............: avg=143µs    min=143µs    med=143µs    max=143µs    p(90)=143µs    p(95)=143µs
http_req_sending...............: avg=91.8µs   min=91.8µs   med=91.8µs   max=91.8µs   p(90)=91.8µs   p(95)=91.8µs
http_req_tls_handshaking.......: avg=533.16ms min=533.16ms med=533.16ms max=533.16ms p(90)=533.16ms p(95)=533.16ms
http_req_waiting...............: avg=235.29ms min=235.29ms med=235.29ms max=235.29ms p(90)=235.29ms p(95)=235.29ms
http_reqs......................: 1     0.486605/s
iteration_duration.............: avg=2.05s    min=2.05s    med=2.05s    max=2.05s    p(90)=2.05s    p(95)=2.05s
iterations.....................: 1     0.486605/s
vus............................: 1     min=1      max=1
vus_max........................: 1     min=1      max=1
```

Options

- 用來修改測試執行的設定參數
- 可彈性從多個注入點設定，優先順序由大到小為：
  1. 命令列參數（command-line flags）
  2. 環境變數（environment variables ）
  3. 測試程式（exported script options）
  4. 組態檔（config file）
  5. 預設值（defaults）
- 常見的有 VUs、Duration、Iterations 等等

https://k6.io/docs/using-k6/options/

Metrics

- 用來統計結果的測試執行數據，可依需求自訂
- Metric types
  - Counter：計算累加數值用，如 http_reqs
  - Gauge：計算最大、最小或特定數值用，如 vus_max
  - Rate：比例類型的數值，如 http_req_failed
  - Trend：基本統計用，如 iteration_duration

Checks

- 如同單元測試的 assertion
- 驗證失敗時不會中止測試程式

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

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


https://k6.io/docs/using-k6/checks/

/check/hello-check.js

Thresholds

- 可針對 Metric 訂定測試標準
  - 80% 的回應時間必須小於 200ms
  - http 失敗回應需小於 1%
- 測試失敗時可用 abortOnFail 強制中止測試程式
- 一個 metric 可使用陣列訂定多個 threshold

https://k6.io/docs/using-k6/thresholds

Simple threshold

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

export const options = {
    thresholds: {
        http_req_failed: ['rate<0.01'],
        http_req_duration: ['p(95)<200']
    },
};

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

Apply multi-threshold to single metric

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

export const options = {
    thresholds: {
        http_req_duration: ['p(90) < 400', 'p(95) < 800']
    }
};

export default function () {
    const resp = http.get('https://example.com');
    sleep(1);
}
```

Duplicated threshold mistake

- 勿對同一 metric 重複套用不同的 threshold（後蓋前）

```
export let options = {
    thresholds: {
        // 若針對同個 metric 重複套用不同的 threshold
        // 會發生「後蓋前」的狀況
        http_req_duration: ['p(90) < 400'],
        http_req_duration: ['p(95) < 800']
    }
};
```

Http requests module

- 可發送 HTTP 請求操作
- 可用 http.batch() 同時發送多個請求

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

- 動態網址可使用 URL grouping 做分組

https://k6.io/docs/using-k6/http-requests/

URL grouping

```
export default function () {
    for (let id = 1; id <= 10; id++) {
        http.get(`http://test.k6.io?id=${id}`);
    }
}
// tags.name="http://test.k6.io?id=1",...
// tags.name="http://test.k6.io?id=2",...
```

- 字串格式化的結果不如預期
- 可透過 explicit / wrapper 兩種方式調整

URL grouping – specify explicit name

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

export default function () {
    for (let id = 1; id <= 10; id++) {
        http.get(`http://test.k6.io?id=${id}`, {
            tags: { name: 'k6-id-query' }
        });
    }
}
// tags.name="k6-id-query",...
```

URL grouping – http url wrapper

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

export default function () {
    for (let id = 1; id <= 10; id++) {
        http.get(http.url`http://test.k6.io?id=${id}`);
    }
}
// tags.name="http://test.k6.io?id=${}"
```

Tags

- 作為測試結果分類用途
- System tags：k6 會自動標記
  - proto, status, method, url, name...
- User-defined tags
  - Checks, Thresholds, Custom metrics, Requests

https://k6.io/docs/using-k6/tags-and-groups/

Groups

- 可將同類的 request 分組
  - 會自動計算 group_duration metric
- 同組的 request 會被打上一樣的 group 標籤
  - 格式為 ::{group_name}
- 依功能將同組的程式碼抽出，即可重複利用

https://k6.io/docs/using-k6/tags-and-groups/

Simple group

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

export default function () {

    group('twMVC44', function () {
        http.get('https://test.k6.io');
    });
    sleep(1);
}
```

Nested group?

```
group('twMVC44', () => {
    group('k6', () => {
        http.get('https://test.k6.io');
    });

    group('google', () => {
        http.get('https://google.com');
    });
});
```

- 可成功產生 group tag
- 有 group_duration metric aggregate issue

https://github.com/grafana/k6/issues/2309

Environment variables - sample

- 可將環境變數傳入給測試程式使用
- 可用來設定 k6 Options（並非全部皆可設定）

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

export default function () {
    const res = http.get(`http://${__ENV.MY_HOST}/`);
    sleep(1);
}
```

Environment variables – CLI

- CLI Flag (-e / --env) [cross platform]

```
$ k6 run -e MY_HOST=test.k6.io pass-env-var-to-k6.js
```

- Bash

```
$ MY_HOST=test.k6.io k6 run pass-env-var-to-k6.js
```

- PowerShell

```
$ $env:MY_HOST="test.k6.io"; k6 run pass-env-var-to-k6.js
```

- Windows CMD

```
$ set "MY_HOST=test.k6.io" && k6 run pass-env-var-to-k6.js
```

Result output

- 預設會呈現彙整後的資訊
  - Metric, Threshold, Check, Group
  - 使用 `--no-summary` 可關閉此功能
- 可在 `handleSummary(data)` 客製化輸出結果
  - Send to remote server
  - stdout / stderr
  - Output file

https://k6.io/docs/results-visualization/end-of-test-summary/

Sample - handleSummary(data)

```
export function handleSummary(data) {
    //可回傳 dictionary 格式
    //key 必須為 stdout, stderr, {filePath} 其中之一
    return {
        'stdout': textSummary(data,
            { indent: ' ', enableColors: true }),
        'k6-result.xml': toXmlFormat(data),
        'k6-result.json': toJsonFormat(data),
    }
}
```

Test Life cycle - k6

```
// 1. init code -> 每個 VU 會獨立執行一次

export function setup() {
    // 2. setup code -> 所有測試只跑一次
}

export default function (data) {
    // 3. VU code -> 在測試階段會重複執行
}

export function teardown(data) {
    // 4. teardown code -> -> 所有測試只跑一次
}
```

https://k6.io/docs/using-k6/test-life-cycle/

實測 Life cycle

```
// 1. init code
const msg = function () {
    console.info(`hello init...`)
}();

export function setup() {
    // 2. setup code
    console.info(`hello setup with vu-${exec.vu.idInTest}...`)
}

export default function () {
    // 3. VU code
    console.info(`hello VU code with vu-${exec.vu.idInTest}...`)
}

export function teardown() {
    // 4. teardown code
    console.info(`hello teardown with vu-${exec.vu.idInTest}...`)
}
```

Result - 實測 Life cycle

```
$ k6 run lifecycle/print-life-cycle.js
```

```
INFO[0000] hello init...  1                              source=console
  execution: local
     script: lifecycle/print-life-cycle.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)

INFO[0000] hello init...  2                              source=console
INFO[0000] hello init...  3                              source=console
INFO[0000] hello setup with vu-0...                      source=console
INFO[0000] hello VU code with vu-1...                    source=console
INFO[0000] hello init...  4                              source=console
INFO[0000] hello teardown with vu-0...                   source=console

running (00m00.0s), 0/1 VUs, 1 complete and 0 interrupted iterations
default ✓ [======================================] 1 VUs  00m00.0s/10m0s  1/1 iters, 1 per VU
INFO[0000] hello init...  5                              source=console
```

Why init code run more than 1 times?

- 理論上只有當每個 VU 被建立時會被執行一次
  - 1st: read exported script options
  - 2nd: setup() VU init
  - 3rd: normal VUs init
  - 4th: teardown() VU init
  - 5th: handleSummary() VU init
- init 執行次數 = VU 數量 + 4

https://community.k6.io/t/why-init-called-4-times/973

進階 & 補充



Scenarios

- 官方針對常見的測試情境提供了內建的 executor
- Common option
  - executor
  - startTime：可指定情境於幾秒後再開始執行
  - exec：可指定情境要執行的方法（不同情境可分別執行不同方法）
  - env / tags

https://k6.io/docs/using-k6/scenarios/

Executors

- Shared iterations：由所有 VU 共同完成特定數量的測試次數
- Per VU iterations：一個 VU 必須獨自完成特定數量的測試次數
- Constant VUs：讓固定的 VU 數量在指定時間內持續執行
- Ramping VUs：在測試中階段性地調整 VU 數量
- Constant Arrival Rate：希望達到某個數量的 RPS 時
- Ramping Arrival Rate：在測試中階段性地調整 RPS

https://k6.io/docs/using-k6/scenarios/executors/

Browser recording

- 可直接針對網頁進行側錄（需安裝 extension）
  - Chrome（Edge）
  - Firefox
- 側錄的結果需輸出至 k6 cloud（免費）
  - Test builder（可轉成 script，推薦使用）
  - Script editor

https://k6.io/docs/test-authoring/recording-a-session/browser-recorder/

k6 x Azure DevOps

- 在 Marketplace 安裝 k6 extension
- 適合進行小規模壓測 (<= 500 RPS)
- 於 Pipeline 中載入 script 執行
- 可整合 Test Plan 使用

https://k6.io/blog/integrating-load-testing-with-azure-pipelines/

如何進行大型的壓測

- 推薦閱讀
  - Fine tune OS
  - Running large tests
- 以 ubuntu 為例（hosted on Azure）

```
sudo sysctl -w net.ipv4.ip_local_port_range="1024 65535"
sudo sysctl -w net.ipv4.tcp_tw_reuse=1
sudo sysctl -w net.ipv4.tcp_timestamps=1
ulimit -n 250000
```

關於壓測...

- 靠數據壓測，不要靠直覺
- 壓測前盡可能完善監控機制（AP / Infra）
  - Logging / Tracing / Metric collecting
- 訂定測試情境、測試人數及成本估計
- 分析結果以獲得反饋
- 自動化

結語 & QA

- k6 是一套非常適合開發者學習的壓測工具
- 官方文件及範例相當完整
- 如果有$$$，可以考慮一下 k6 cloud
- 壓測的執行流程應隨組織架構而有所調整

Thanks







Blog 是記錄知識的最佳平台



https://dotblogs.com.tw

SkillTree 為了確保內容與實務不會脫節，我們都是聘請企業顧問等級並且目前依然在職場的業界講師，我們不把時間浪費在述說歷史與沿革，我們並不是教您考取證照，而是教您如何上場殺敵，拳拳到肉的內容才是您花錢想要聽到的，而這也剛好是我們擅長的。



https://skilltree.my

天瓏資訊圖書

