Installation

Mac: `brew install k6`

Linux: `sudo apt-get install k6`

Windows: Download the installer from the K6 website

Usage

To create a K6 test script, create a new file with the .js extension. In the file, add the following code:

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

export default function() {
http.g et( " htt ps: //e xam ple
.co m");
}
```

This script will make a GET request to the example.com website.

To run the test script, use the following command:

```
k6 run test.js
```

This will start the K6 test runner and execute the test script.

Options

You can customize the behavior of the K6 test runner using the following options:

`vus`: The number of virtual users to simulate

`iterations`: The number of times to repeat the test script

`duration`: The duration of the test in seconds

Examples

To run a test with 10 virtual users for 60 seconds, use the following command:

```
k6 run --vus 10 --duration 60s t
est.js
```

To run a test with 100 virtual users for 120 seconds, use the following command:

```
k6 run --vus 100 --duration 120s
 test.js
```

Reporting

K6 comes with built-in metrics about the test load and the system response. Key metrics include:

`http_r eq_ dur ation`, the end-to-end time of all requests (that is, the total latency)

`http_r eq_ failed`, the total number of failed requests

`iterations`, the total number of iterations

Adding-checks-to-your-script

Note that you need to import the check from the k6 library:

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

And you need to put the actual check in the default function:

```
check( res ponse, {
 'Appli cation says hello': (r)
=> r.body.in clu des ('Hello wor
ld!')
   });
}
```

Setting-test-criteria-with-threshold

Types of thresholds:

1. Error rate

```
thresh olds: {
http_r eq_ failed: ['rate <=0.05
'],
},
```

2. Response time

```
thresh olds: {
 http_r eq_ dur ation: ['p(95 )<
= 5000'],
},
```

3. Checks

```
thresh olds: {
checks: ['rate >=0.9'],
},
```