K6

Install K6 (Linux)

```
sudo gpg -k
sudo gpg --no-default-keyring --keyring /usr/share/keyrings/k6-archive-keyring.gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys C5AD17C747E3415A3642D57D77C6C491D6AC1D69
echo "deb [signed-by=/usr/share/keyrings/k6-archive-keyring.gpg] 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
```

Run Test Script

Create New Script File

```
k6 run script.js
```

Add VUs

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

Set options

Instead of typing --vus 10 and --duration 30s each time you run the script, you can set the options in your JavaScript file:

```
import http from 'k6/http';
import { sleep } from 'k6';
export const options = {
  vus: 10,
  duration: '30s',
};
export default function () {
  http.get('http://test.k6.io');
  sleep(1);
}
```

Ramp VUs

Ramp Up -> Ramp Down

```
export const options = {
  stages: [
```

```
    { duration: '30s', target: 20 },
    { duration: '1m30s', target: 10 },
    { duration: '20s', target: 0 },
  ],
};
```

Execution Modes

k6 supports three execution modes to run a k6 test:

1. local

```
   k6 run script.js
```

3. distributed

```
apiVersion: k6.io/v1alpha1
kind: TestRun
metadata:
  name: k6-sample
spec:
  parallelism: 4
  script:
    configMap:
      name: 'k6-test'
      file: 'script.js'
```

```
kubectl apply -f /path/to/k6-testrun-resource.yaml
```

4. cloud

```
k6 cloud run script.js
```

Result Output Types

1. Metrics
    - http_req_duration
    - http_req_failed
    - iterations
2. End-of-Test Summary
    - Median & average values
    - Minimum & Maximum values
    - p90, p95 & p99 values
3. Custom Reports with handleSummary()

- At the end of the test, k6 automatically creates an object with all aggregated statistics. The handleSummary() function can process this object into a custom report in any text format: JSON, HTML, XML, and whatever else.

4. Time series and external outputs

```
k6 run \
--out json=test.json \
--out influxdb=http://localhost:8086/k6
```

The available built-in outputs include:

- Amazon CloudWatch
- Cloud
- CSV
- Datadog
- Dynatrace
- Elasticsearch
- Grafana Cloud Prometheus
- InfluxDB
- JSON
- Netdata
- New Relic
- Prometheus
- TimescaleDB
- StatsD

Using K6

K6 deps Command: This outputs dependency information in human-readable format to stdout.
`k6 deps [options] <script>`

Run a K6 test script

1. Run local machine:

   ```
   k6 run script.js
   ```

2. Run a test using Grafana Cloud K6:

```
k6 cloud login --token <API_TOKEN> --stack <STACK_SLUG>
k6 cloud run cloud_demo.js
```

Requests

1. HTTP Requests
    - Get request looks like

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

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

- Post request looks like

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

export default function () {
  const url = 'http://test.k6.io/login';
  const payload = JSON.stringify({
    email: 'aaa',
    password: 'bbb',
  });

  const params = {
    headers: {
      'Content-Type': 'application/json',
    },
  };

  http.post(url, payload, params);
}
```

The http module handles all kinds of HTTP requests and methods:

Name	Value
batch()	Issue multiple HTTP requests in parallel (like browsers do)
del()	Issue an HTTP DELETE request
get()	Issue an HTTP GET request
head()	Issue an HTTP HEAD request
options()	Issue an HTTP OPTIONS request
patch()	Issue an HTTP PATCH request
post()	Issue an HTTP POST request
put()	Issue an HTTP PUT request

Name	Value
request()	Issue any type of HTTP request

Follow Redirects

By default, k6 automatically follows a set number of redirects before stopping and returning the last response:

- maxRedirects
- Params.redirects

HTTP Request Tags

Name	Description
expected_response	By default, response statuses between 200 and 399 are true. Change using `setResponseCallback`.
group	When the request runs inside a group, the tag value is the group name. Default is empty.
name	Defaults to the requested URL
method	Request method (GET, POST, PUT, etc.)
scenario	When the request runs inside a scenario, the tag value is the scenario name. Default is `default`.
status	Response status
url	Defaults to the requested URL

Built-in Metrics

Standard built-in metrics:

Metric Name	Type	Description
checks	Rate	The rate of successful checks.
data_received	Counter	The amount of received data. This example covers how to track data for an individual URL.
data_sent	Counter	The amount of data sent. Track data for an individual URL.
dropped_iterations	Counter	The number of iterations that weren't started due to lack of VUs (arrival-rate executors) or lack of time (expired `maxDuration` in iteration-based executors).
iteration_duration	Trend	Time to complete one full iteration, including setup and teardown.

Metric Name	Type	Description
iterations	Counter	Total number of times VUs execute the JS script (default function).
vus	Gauge	Current number of active virtual users.
vus_max	Gauge	Maximum possible number of virtual users (pre-allocated to avoid scaling performance impact).

HTTP-specific built-in metrics: For all http_req_* metrics, the timestamp is emitted the end of the request.

Metric Name	Type	Description
http_req_blocked	Trend	Time spent blocked (waiting for a free TCP connection slot) before initiating the request.
http_req_connecting	Trend	Time spent establishing TCP connection to the remote host.
http_req_duration	Trend	Total request time = sending + waiting + receiving (excludes DNS lookup/connection time).
http_req_failed	Rate	Rate of failed requests according to `setResponseCallback`.
http_req_receiving	Trend	Time spent receiving response data from the remote host.
http_req_sending	Trend	Time spent sending data to the remote host.
http_req_tls_handshaking	Trend	Time spent performing TLS handshake with the remote host.
http_req_waiting	Trend	Time spent waiting for response (Time To First Byte - TTFB).
http_reqs	Counter	Total number of HTTP requests generated by k6.

Browser metrics

Metric Name	Description
browser_web_vital_cls	Measures visual stability by quantifying unexpected layout shifts of visible page content (Cumulative Layout Shift).
browser_web_vital_fid	Deprecated in favor of INP. Measures delay between a user's first interaction and browser response (First Input Delay).
browser_web_vital_inp	Measures page responsiveness (Interaction to Next Paint).
browser_web_vital_lcp	Measures time for the largest content element on the page to become visible (Largest Contentful Paint).

Built-in WebSocket metrics

Metric Name	Type	Description
ws_connecting	Trend	Total duration for the WebSocket connection request.
ws_msgs_received	Counter	Total number of messages received.
ws_msgs_sent	Counter	Total number of messages sent.
ws_ping	Trend	Duration between a ping request and its pong reception.
ws_session_duration	Trend	Duration of the WebSocket session (from connection start to VU execution end).
ws_sessions	Counter	Total number of started WebSocket sessions.

Checks

Defination

- virtual users
- Ramp Up
- Ramp Down
- http_req_duration
- http_req_failed
- iterations
- p90
- p95
- p99
- metric