Options reference

Options define test-run behavior. Most options can be passed in multiple places. If an option is defined in multiple places, k6 chooses the value from the highest order of precedence.

Quick reference of options

Each option has its own detailed reference in a separate section.

OPTION	DESCRIPTION
Address	Address of the REST API server
Batch	Max number of simultaneous connections of a `http.batch()` call
Batch per host	Max number of simultaneous connections of a `http.batch()` call for a host
Blacklist IP	Blacklist IP ranges from being called
Block hostnames	Block any requests to specific hostnames
Compatibility mode	Support running scripts with different ECMAScript modes
Config	Specify the config file in JSON format to read the options values
Console output	Redirects logs logged by `console` methods to the provided output file
Discard response bodies	Specify whether response bodies should be discarded
DNS	Configure DNS resolution behavior
Duration	A string specifying the total duration of the test run; together with the vus option, it's a shortcut for a single scenario with a constant VUs executor
Execution segment	Limit execution to a segment of the total test
Exit on running	Exits when test reaches the running status
Extension options	An object used to set configuration options for cloud parameters and third-party collectors
Hosts	An object with overrides to DNS resolution
HTTP debug	Log all HTTP requests and responses
Include system Env vars	Pass the real system environment variables to the runtime
Insecure skip TLS verify	A boolean specifying whether k6 should ignore TLS verifications for connections established from code
Iterations	A number specifying a fixed number of iterations to execute of the script; together with the vus option, it's a shortcut for a single scenario with a shared iterations executor
Linger	A boolean specifying whether k6 should linger around after test run completion
Local IPs	A list of local IPs, IP ranges, and CIDRs from which VUs will make requests
Log output	Configuration about where logs from k6 should be send
LogFormat	Specify the format of the log output

Max redirects	The maximum number of HTTP redirects that k6 will follow
Minimum iteration duration	Specify the minimum duration for every single execution
No color	A boolean specifying whether colored output is disabled
No connection reuse	A boolean specifying whether k6 should disable keep-alive connections
No cookies reset	This disables resetting the cookie jar after each VU iteration
No summary	disables the end-of-test summary
No setup	A boolean specifying whether `setup()` function should be run
No teardown	A boolean specifying whether `teardown()` function should be run
No thresholds	Disables threshold execution
No usage report	A boolean specifying whether k6 should send a usage report
No VU connection reuse	A boolean specifying whether k6 should reuse TCP connections
Paused	A boolean specifying whether the test should start in a paused state
Quiet	A boolean specifying whether to show the progress update in the console or not
Results output	Specify the results output
RPS	The maximum number of requests to make per second globally (discouraged, use arrival-rate executors instead)
Scenarios	Define advanced execution scenarios
Setup timeout	Specify how long the `setup()` function is allow to run before it's terminated
Show logs	A boolean specifying whether the cloud logs are printed out to the terminal
Stages	A list of objects that specify the target number of VUs to ramp up or down; shortcut option for a single scenario with a ramping VUs executor
Supply environment variable	Add/override environment variable with `VAR=value`
System tags	Specify which System Tags will be in the collected metrics
Summary export	Output the end-of-test summary report to a JSON file (discouraged, use handleSummary() instead)
Summary trend stats	Define stats for trend metrics in the end-of-test summary
Summary time unit	Time unit to be used for *all* time values in the end-of-test summary
Tags	Specify tags that should be set test-wide across all metrics
Teardown timeout	Specify how long the teardown() function is allowed to run before it's terminated
Thresholds	Configure under what conditions a test is successful or not
Throw	A boolean specifying whether to throw errors on failed HTTP requests
TLS auth	A list of TLS client certificate configuration objects
TLS cipher suites	A list of cipher suites allowed to be used by in SSL/TLS interactions with a server
TLS version	String or object representing the only SSL/TLS version allowed
User agent	A string specifying the User-Agent header when sending HTTP requests

Verbose	A boolean specifying whether verbose logging is enabled
VUs	A number specifying the number of VUs to run concurrently

The following sections detail all available options that you can be specify within a script.

It also documents the equivalent command line flag, environment variables or option when executing `k6 run ...` and `k6 cloud ...`, which you can use to override options specified in the code.

Address

Address of the API server. When executing scripts with `k6 run` an HTTP server with a REST API is spun up, which can be used to control some of the parameters of the test execution. By default, the server listens on `localhost:6565`. Read more on k6 REST API.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
N/A	`--address`, `-a`	N/A	`localhost:6565`

```
$ k6 run --address "localhost:3000" script.js
```

Batch

The maximum number of simultaneous/parallel connections in total that an `http.batch()` call in a VU can make. If you have a `batch()` call that you've given 20 URLs to and `--batch` is set to 15, then the VU will make 15 requests right away in parallel and queue the rest, executing them as soon as a previous request is done and a slot opens. Available in both the `k6 run` and the `k6 cloud` commands

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_BATCH`	`--batch`	`batch`	`20`

```
1 export const options = {
2   batch: 15,
3 };
```

Batch per host

The maximum number of simultaneous/parallel connections for the same hostname that an `http.batch()` call in a VU can make. If you have a `batch()` call that you've given 20 URLs to the *same* hostname and `--batch-per-host` is set to 5, then the VU will make 5 requests right away in parallel and queue the rest, executing them as soon as a previous request is done and a slot opens. This will not run more request in parallel then the value of `batch`. Available in both the `k6 run` and the `k6 cloud` commands

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_BATCH_PER_HOST`	`--batch-per-host`	`batchPerHost`	`6`

```
1 export const options = {
2   batchPerHost: 5,
3 };
```

Blacklist IP

Blacklist IP ranges from being called. Available in `k6 run` and `k6 cloud` commands.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_BLACKLIST_IPS`	`--blacklist-ip`	`blacklistIPs`	`null`

```
1 export const options = {
2   blacklistIPs: ['10.0.0.0/8'],
3 };
```

Block hostnames

Blocks hostnames based on a list of glob match strings. The pattern matching string can have a single `*` at the beginning such as `*.example.com` that will match anything before that such as `test.example.com` and `test.test.example.com`. Available in `k6 run` and `k6 cloud` commands.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_BLOCK_HOSTNAMES`	`--block-hostnames`	`blockHostnames`	`null`

```
export const options = {
  blockHostnames: ['test.k6.io', '*.example.com'],
};
```

```
$ k6 run --block-hostnames="test.k6.io,*.example.com" script.js
```

Compatibility mode

Support running scripts with different ECMAScript compatibility modes.

Read about the different modes on the JavaScript Compatibility Mode documentation.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_COMPATIBILITY_MODE`	`--compatibility-mode`	N/A	`"extended"`

```
$ k6 run --compatibility-mode=base script.js
```

Config

Specify the config file in JSON format. If the config file is not specified, k6 looks for `config.json` in the `loadimpact/k6` directory inside the regular directory for configuration files on the operating system. Default config locations on different operating systems are as follows:

OS	DEFAULT CONFIG PATH
Unix-based	`${HOME}/.config/loadimpact/k6/config.json`
macOS	`${HOME}/Library/Application Support/loadimpact/k6/config.json`
Windows	`%AppData%/loadimpact/k6/config.json`

Available in `k6 run` and `k6 cloud` commands:

ENV	CLI	CODE / CONFIG FILE	DEFAULT
N/A	`--config <path>`, `-c <path>`	N/A	`null`

> **NOTE**
> When running tests in k6 Cloud and using a non-default config.json file, specify the cloud token inside your config file to authenticate.

Console output

Redirects logs logged by `console` methods to the provided output file. Available in `k6 cloud` and `k6 run` commands.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_CONSOLE_OUTPUT`	`--console-output`	N/A	`null`

```
$ k6 run --console-output "loadtest.log" script.js
```

Discard response bodies

Specify if response bodies should be discarded by changing the default value of responseType to `none` for all HTTP requests. Highly recommended to be set to `true` and then only for the requests where the response body

is needed for scripting to set responseType to `text` or `binary`. Lessens the amount of memory required and the amount of GC - reducing the load on the testing machine, and probably producing more reliable test results.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_DISCARD_RESPONSE_BODIES`	`--discard-response-bodies`	`discardResponseBodies`	`false`

```
export const options = {
  discardResponseBodies: true,
};
```

DNS

This is a composite option that provides control of DNS resolution behavior with configuration for cache expiration (TTL), IP selection strategy and IP version preference. The TTL field in the DNS record is currently not read by k6, so the `ttl` option allows manual control over this behavior, albeit as a fixed value for the duration of the test run.

Note that DNS resolution is done only on new HTTP connections, and by default k6 will try to reuse connections if HTTP keep-alive is supported. To force a certain DNS behavior consider enabling the `noConnectionReuse` option in your tests.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_DNS`	`--dns`	`dns`	`ttl=5m,select=random,policy=preferIPv4`

Possible `ttl` values are:

- `0`: no caching at all - each request will trigger a new DNS lookup.
- `inf`: cache any resolved IPs for the duration of the test run.
- any time duration like `60s`, `5m30s`, `10m`, `2h`, etc.; if no unit is specified (e.g. `ttl=3000`), k6 assumes milliseconds.

Possible `select` values are:

- `first`: always pick the first resolved IP.
- `random`: pick a random IP for every new connection.
- `roundRobin`: iterate sequentially over the resolved IPs.

Possible `policy` values are:

- `preferIPv4`: use IPv4 addresses if available, otherwise fall back to IPv6.
- `preferIPv6`: use IPv6 addresses if available, otherwise fall back to IPv4.
- `onlyIPv4`: only use IPv4 addresses, ignore any IPv6 ones.
- `onlyIPv6`: only use IPv6 addresses, ignore any IPv4 ones.
- `any`: no preference, use all addresses.

Here are some configuration examples:

```
K6_DNS="ttl=5m,select=random,policy=preferIPv4" k6 cloud script.js
```

script.js

```
export const options = {
  dns: {
    ttl: '1m',
    select: 'roundRobin',
    policy: 'any',
  },
};
```

Duration

A string specifying the total duration a test run should be run for. During this time each VU will execute the script in a loop. Available in `k6 run` and `k6 cloud` commands.

Together with the `vus` option, `duration` is a shortcut for a single scenario with a constant VUs executor.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_DURATION`	`--duration`, `-d`	`duration`	`null`

```
export const options = {
  vus: 100,
  duration: '3m',
};
```

Extension options

An object used to set configuration options for cloud parameters and third-party collectors, like plugins. For more information about available parameters, refer to Cloud options.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
N/A	N/A	`ext`	`null`

This is an example of how to specify the test name (test runs/executions with the same name will be logically grouped for trending and comparison) when streaming results to k6 Cloud Performance Insights.

```
export const options = {
  ext: {
    loadimpact: {
      name: 'My test name',
    },
  },
};
```

Execution segment

These options specify how to partition the test run and which segment to run. If defined, k6 will scale the number of VUs and iterations to be run for that segment, which is useful in distributed execution. Available in `k6 run` and `k6 cloud` commands.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
N/A	`--execution-segment`	`executionSegment`	`"0:1"`
N/A	`--execution-segment-sequence`	`executionSegmentSequence`	`"0,1"`

For example, to run 25% of a test, you would specify `--execution-segment '25%'`, which would be equivalent to `--execution-segment '0:1/4'`, i.e. run the first 1/4 of the test. To ensure that each instance executes a specific segment, also specify the full segment sequence, e.g. `--execution-segment-sequence '0,1/4,1/2,1'`. This way one instance could run with `--execution-segment '0:1/4'`, another with `--execution-segment '1/4:1/2'`, etc. and there would be no overlap between them.

Exit on running

A boolean, specifying whether the script should exit once the test status reaches `running`. When running scripts with `k6 cloud` by default scripts will run until the test reaches a finalized status. This could be problematic in certain environments (think of Continuous Integration and Delivery pipelines), since you'd need to wait until the test ends up in a finalized state.

With this option, you can exit early and let the script run in the background. Available in `k6 cloud` command.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_EXIT_ON_RUNNING`	`--exit-on-running`	N/A	`false`

```
$ k6 cloud --exit-on-running script.js
```

Hosts

An object with overrides to DNS resolution, similar to what you can do with `/etc/hosts` on Linux/Unix or `C:\Windows\System32\drivers\etc\hosts` on Windows. For instance, you could set up an override which routes all requests for `test.k6.io` to `1.2.3.4`.

k6 also supports ways to narrow or widen the scope of your redirects:

- You can redirect only from or to certain ports.
- Starting from v0.42.0, you can use an asterisk (`*`) as a wild card at the start of the host name to avoid repetition. For example, `*.k6.io` would apply the override for all subdomains of `k6.io`.

> **NOTE**
>
> This does not modify the actual HTTP `Host` header, but rather where it will be routed.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
N/A	N/A	`hosts`	`null`

```
export const options = {
  hosts: {
    'test.k6.io':     '1.2.3.4',
    'test.k6.io:443': '1.2.3.4:8443',
    '*.grafana.com':  '1.2.3.4',
  },
};
```

The preceding code will redirect requests made to `test.k6.io` to `1.2.3.4`, keeping the same port. If the request is done to port `443`, it will redirect it to port `8443` instead. It will also redirect requests to any subdomain of `grafana.com` to `1.2.3.4`.

HTTP debug

Log all HTTP requests and responses. Excludes body by default, to include body use `--http-debug=full`. Available in `k6 run` and `k6 cloud` commands.

Read more here.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_HTTP_DEBUG`	`--http-debug`, `--http-debug=full`	`httpDebug`	`false`

```
export const options = {
  httpDebug: 'full',
};
```

Include system env vars

Pass the real system environment variables to the runtime. Available in `k6 run` and `k6 cloud` commands.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
N/A	`--include-system-env-vars`	N/A	`true` for `k6 run`, but `false` for all other commands to prevent inadvertent sensitive data leaks.

Shell

```
$ k6 run --include-system-env-vars ~/script.js
```

Insecure skip TLS verify

A boolean, true or false. When this option is enabled (set to true), all of the verifications that would otherwise be done to establish trust in a server provided TLS certificate will be ignored. This only applies to connections created from code, such as HTTP requests. Available in `k6 run` and `k6 cloud` commands

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_INSECURE_SKIP_TLS_VERIFY`	`--insecure-skip-tls-verify`	`insecureSkipTLSVerify`	`false`

```
export const options = {
  insecureSkipTLSVerify: true,
};
```

Iterations

An integer value, specifying the total number of iterations of the `default` function to execute in the test run, as opposed to specifying a duration of time during which the script would run in a loop. Available both in the `k6 run` and `k6 cloud` commands.

Together with the `vus` option, `iterations` is a shortcut for a single scenario with a shared iterations executor.

By default, the maximum duration of a `shared-iterations` scenario is 10 minutes. You can adjust that time via the `maxDuration` option of the scenario, or by also specifying the `duration` global shortcut option.

Note that iterations aren't fairly distributed with this option, and a VU that executes faster will complete more iterations than others. Each VU will try to complete as many iterations as possible, "stealing" them from the total number of iterations for the test. So, depending on iteration times, some VUs may complete more iterations than others. If you want guarantees that every VU will complete a specific, fixed number of iterations

iterations than others. If you want guarantees that every VU will complete a specific, fixed number of iterations, use the per-VU iterations executor.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_ITERATIONS`	`--iterations`, `-i`	`iterations`	`1`

```
export const options = {
  vus: 5,
  iterations: 10,
};
```

Or, to run 10 VUs 10 times each:

```
export const options = {
  vus: 10,
  iterations: 100,
};
```

Linger

A boolean, true or false, specifying whether the k6 process should linger around after test run completion. Available in the `k6 run` command.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_LINGER`	`--linger`, `-l`	`linger`	`false`

```
export const options = {
  linger: true,
};
```

Local IPs

A list of IPs, IP ranges and CIDRs from which VUs will make requests. The IPs will be sequentially given out to VUs. This option doesn't change anything on the OS level so the IPs need to be already configured on the OS level for k6 to use them. Also IPv4 CIDRs with more than 2 IPs don't include the first and last IP as they are reserved for referring to the network itself and the broadcast address respectively.

This option can be used for splitting the network traffic from k6 between multiple network cards, thus potentially increasing the available network throughput. For example, if you have 2 NICs, you can run k6 with `--local-ips="<IP-from-first-NIC>,<IP-from-second-NIC>"` to balance the traffic equally between them - half of the VUs will use the first IP and the other half will use the second. This can scale to any number of NICs, and you can repeat some local IPs to give them more traffic. For example, `--local-ips="<IP1>,<IP2>,<IP3>,<IP3>"` will split VUs between 3 different source IPs in a 25%:25%:50% ratio.

Available in the `k6 run` command.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_LOCAL_IPS`	`--local-ips`	N/A	N/A

```
$ k6 run --local-ips=192.168.20.12-192.168.20.15,192.168.10.0/27 script.js
```

Log output

This option specifies where to send logs to and another configuration connected to it. Available in the `k6 run` command.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_LOG_OUTPUT`	`--log-output`	N/A	`stderr`

```
$ k6 run --log-output=stdout script.js
```

Possible values are:

- none - disable
- stdout - send to the standard output
- stderr - send to the standard error output (this is the default)

- stdout - send to the standard error output (this is the default)
- loki - send logs to a loki server
- file - write logs to a file

Loki

Use the `log-output` option to configure Loki as follows. For additional instructions and a step-by-step guide, check out the Loki tutorial.

```
$ k6 run --log-output=loki=http://127.0.0.1:3100/loki/api/v1/push,label.something=else,label.f
```

Where all but the url in the beginning are not required. The possible keys with their meanings and default values:

KEY	DESCRIPTION	DEFAULT VALUE
`nothing`	the endpoint to which to send logs	`http://127.0.0.1:3100/loki/api/v1/push`
allowedLabels	if set k6 will send only the provided labels as such and all others will be appended to the message in the form `key=value`. The value of the option is in the form `[label1,label2]`	N/A
label. `labelName`	adds an additional label with the provided key and value to each message	N/A
header. `headerName`	adds an additional HTTP header with the provided header name and value to each HTTP request made to Loki	N/A
limit	the limit of message per pushPeriod, an additional log is send when the limit is reached, logging how many logs were dropped	100
level	the minimal level of a message so it's send to loki	all
pushPeriod	at what period to send log lines	1s
profile	whether to print some info about performance of the sending to loki	false
msgMaxSize	how many symbols can there be at most in a message. Messages bigger will miss the middle of the message with an additional few characters explaining how many characters were dropped.	1048576

File

The file can be configured as below, where an explicit file path is required:

```
$ k6 run --log-output=file=./k6.log script.js
```

A valid file path is the unique mandatory field, the other optional fields listed below:

KEY	DESCRIPTION	DEFAULT VALUE
level	the minimal level of a message to write out of (in ascending order): trace, debug, info, warning, error, fatal, panic	trace

LogFormat

A value specifying the log format. By default, k6 includes extra debug information like date and log level. The other options available are:

- `json` : print all the debug information in JSON format.
- `raw` : print only the log message.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_LOG_FORMAT`	`--log-format` , `-f`	N/A	

```
$ k6 run --log-format raw test.js
```

Max redirects

The maximum number of HTTP redirects that k6 will follow before giving up on a request and erroring out. Available in both the `k6 run` and the `k6 cloud` commands.

ENV	CLI	CODE / CONFIG FILE	DEFAULT
`K6_MAX_REDIRECTS`	`--max-redirects`	`maxRedirects`	`10`

```
export const options = {
  maxRedirects: 10,
};
```

Minimum iteration duration

Specifies the minimum duration of every single execution (i.e. iteration) of the `default` function. Any iterations that are shorter than this value will cause that VU to sleep for the remainder of the time until the specified minimum duration is reached.

ENV	CLI	CODE / CONFIG FILE	DEFAULT