# k6 Load testing

**init** imports, options → once → **setup()** returns data → data → **default()** loops per VU → data → **teardown()** cleanup

## CONCEPTS

Scripts are JavaScript, but k6 runs them in its own Go runtime: no Node.js APIs, no npm install.

Options resolve in this order: defaults, script options, environment variables, then CLI flags (highest).

Failed checks do not change the exit code. Add a threshold on the checks metric to fail the run.

## RUNNING

**`k6 new [<file>]`**
Scaffold a script

**`k6 run <script>`**
Run a script locally

**`k6 run -`**
Read the script from stdin

**`--vus <n>` / `-u <n>`**
Concurrent virtual users

**`--duration <time>` / `-d <time>`**
Total run time, e.g. 3m

**`--iterations <n>` / `-i <n>`**
Total iterations across VUs

## RUNNING (cont.)

**`--stage <time>:<target>`**
Ramp VUs, repeatable (-s)

## VARIABLES AND TAGS

**`-e <NAME>=<value>` / `--env <NAME>=<value>`**
Set a variable in __ENV

**`--tag <name>=<value>`**
Tag every metric of the run

**`K6_<OPTION>`**
Set most options by env var

## OUTPUT AND SUMMARY

**`--out json=<file>` / `-o json=<file>`**
Write every metric point as JSON

**`--out csv=<file>`**
Write metric points as CSV

**`--out <name>=<target>`**
Stream to InfluxDB or another output

**`--summary-export <file>`**
Write the end summary as JSON

**`--summary-trend-stats <list>`**
Choose trend stats, e.g. avg,p(95),max

## OUTPUT AND SUMMARY (cont.)

**`--quiet / -q`**
Hide the progress bar

## DEBUGGING

**`--http-debug`**
Log HTTP requests and responses

**`--http-debug=full`**
Include request and response bodies

**`--no-thresholds`**
Run without evaluating thresholds

**`--insecure-skip-tls-verify`**
Accept any TLS certificate

## SCRIPT OPTIONS

### options block

```
export const options = {
  vus: 10,
  duration: '30s',
};
```

### Stages

```
stages: [
  { duration: '30s', target: 20 },
  { duration: '1m', target: 20 },
  { duration: '10s', target: 0 },
],
```

## SCRIPT OPTIONS (cont.)

### Thresholds

```
thresholds: {
  http_req_duration: ['p(95)<200'],
  http_req_failed: ['rate<0.01'],
  checks: ['rate>0.99'],
},
```

### Scenario

```
scenarios: {
  api: {
    executor: 'constant-arrival-rate',
    rate: 20, timeUnit: '1s',
    duration: '1m',
    preAllocatedVUs: 10,
maxVUs: 50,
  },
},
```

### Threshold syntax

| Metric type | Aggregations |
|---|---|
| `Counter` | count, rate |
| `Gauge` | value |
| `Rate` | rate |
| `Trend` | avg, min, max, med, p(N) |

A threshold key can filter by tag: 'http_req_duration{type:API}': ['p(95)<500'].

## SCRIPT API

### Minimal test

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

export default function () {
  const res = http.get('https://test.k6.io/');
  check(res, { 'status is 200': (r) => r.status === 200 });
  sleep(1);
}
```

**`check(<val>, { <name>: <fn> })`**
Assert without stopping the test

**`sleep(<seconds>)`**
Pause the VU between iterations

**`group('<name>', <fn>)`**
Group requests under a name

**`http.get(<url>)`**
GET request

**`http.post(<url>, <body>, <params>)`**
POST request

**`__ENV.<NAME>`**
Read a variable set with -e

**`export function setup()`**
Run once; return shared data