What is load testing?

Slides from Graphana team with minor adaptations (2025)

“Load testing is the process of putting demand on a system and measuring its response”

Load Testing

✅

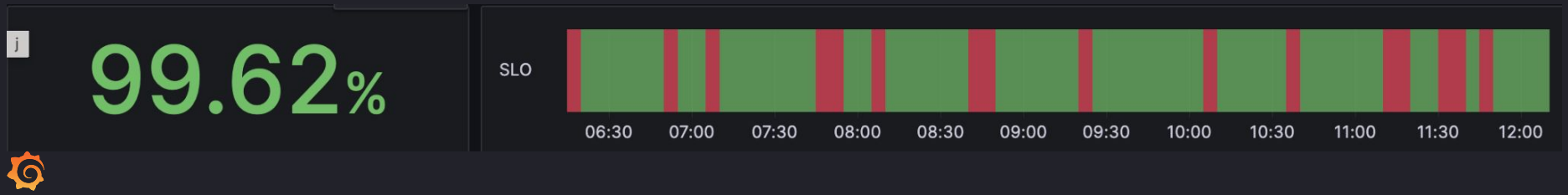
❌

Load Testing

Accept Errors: 9s of Reliability

Accept Outliers: percentiles 90/95/99/..

✅



Describe service expectations

SLI: Service Level Indicator

An **SLI** is a **quantitative measure** of some aspect of the level of service provided. It answers the question:

> *"How is the system actually performing?"*

Examples of SLIs in load testing:

- **Latency**: e.g., 95th percentile response time is 200 ms.
- **Throughput**: e.g., 500 requests per second.
- **Error rate**: e.g., 0.1% of requests result in a 5xx error.
- **Availability**: e.g., service is up 99.99% of the time.

SLO: Service Level Objective

An **SLO** is a **target value or range** for a given SLI. It defines the **expected or acceptable performance** level.

> *"What level of performance do we promise to maintain?"*

Examples of SLOs:

- "95% of requests should complete in under 300 ms."
- "Error rate should remain below 0.1% over a 30-day window."
- "Availability should be at least 99.9% each month."

What is load testing?

```
// LOAD
export const options = {
  vus: 100,
  duration: '2m',
};
```

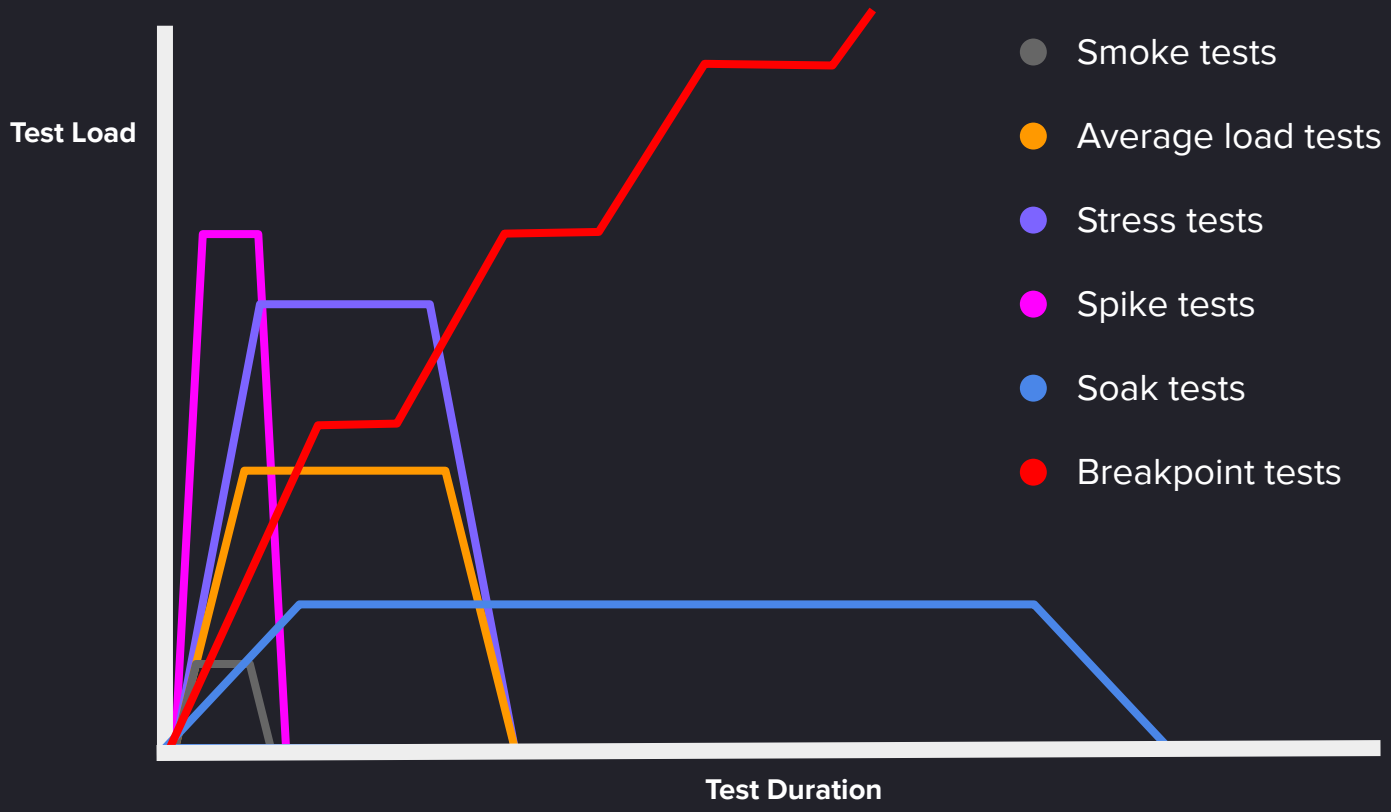
```
// SCENARIO
export default function() {

  http.get('https://myapi.com/products/');
  sleep(0.3);

  const data = {username: 'username',
                password: 'password'};
  http.post('https://myapi.com/login/', data);
  sleep(0.3);
}
```

“Applications and Systems perform differently under [distinct] traffic“

Common types of load tests



Why do load testing?

Myths about load testing

- Performance testing = load testing.
- Load testing is only for large companies.
- Load testing is expensive.
- Load testing should only be done in production.
- You don't need load testing if you have observability.

Test-type cheat sheet

The following table provides some broad comparisons.

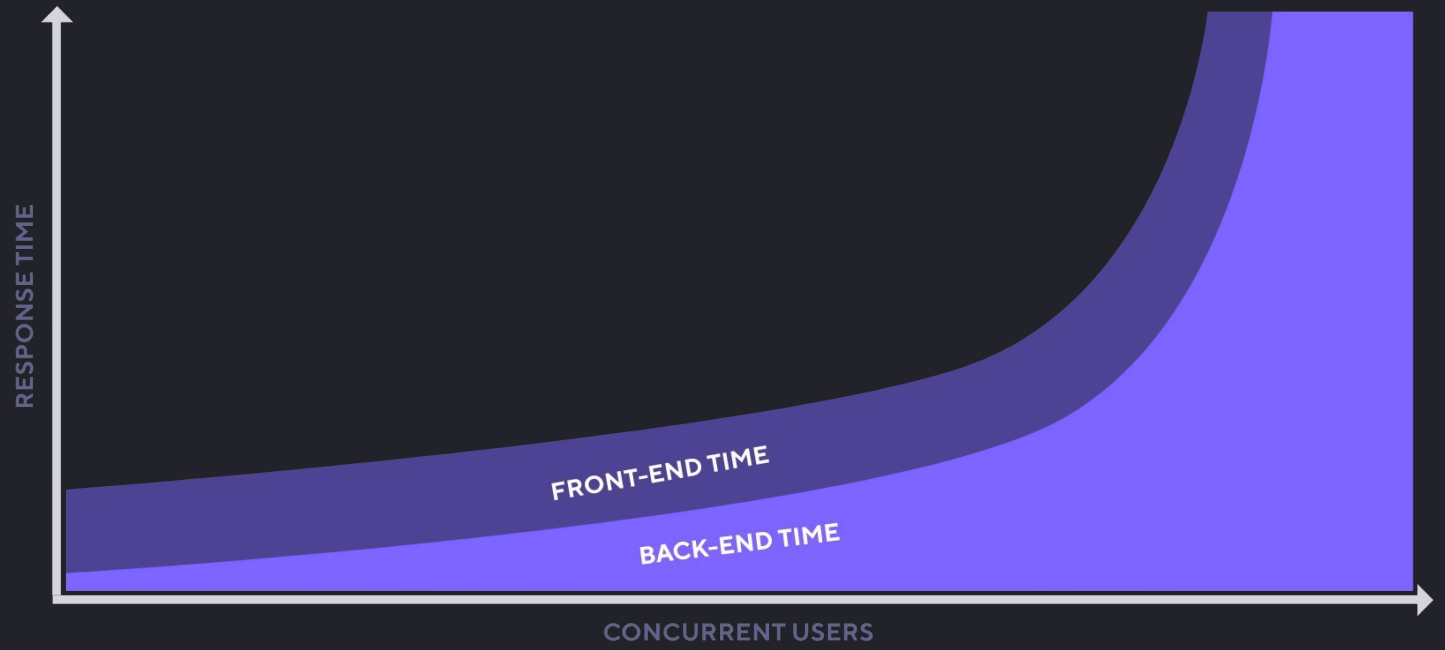
Expand table

Type	VUs/Throughput	Duration	When?
Smoke	Low	Short (seconds or minutes)	When the relevant system or application code changes. It checks functional logic, baseline metrics, and deviations
Load	Average production	Mid (5-60 minutes)	Often to check system maintains performance with average use
Stress	High (above average)	Mid (5-60 minutes)	When system may receive above-average loads to check how it manages
Soak	Average	Long (hours)	After changes to check system under prolonged continuous use
Spike	Very high	Short (a few minutes)	When the system prepares for seasonal events or receives frequent traffic peaks
Breakpoint	Increases until break	As long as necessary	A few times to find the upper limits of the system

https://grafana.com/docs/k6/latest/testing-guides/test-types/

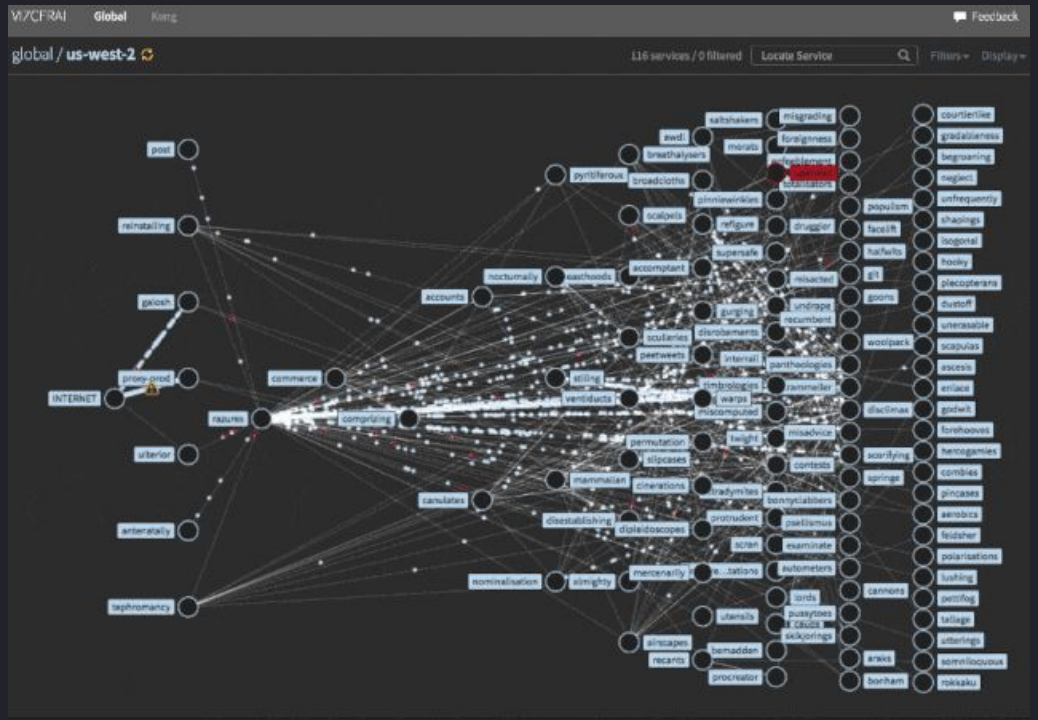
Why do load testing today?

UX



Why load testing today?

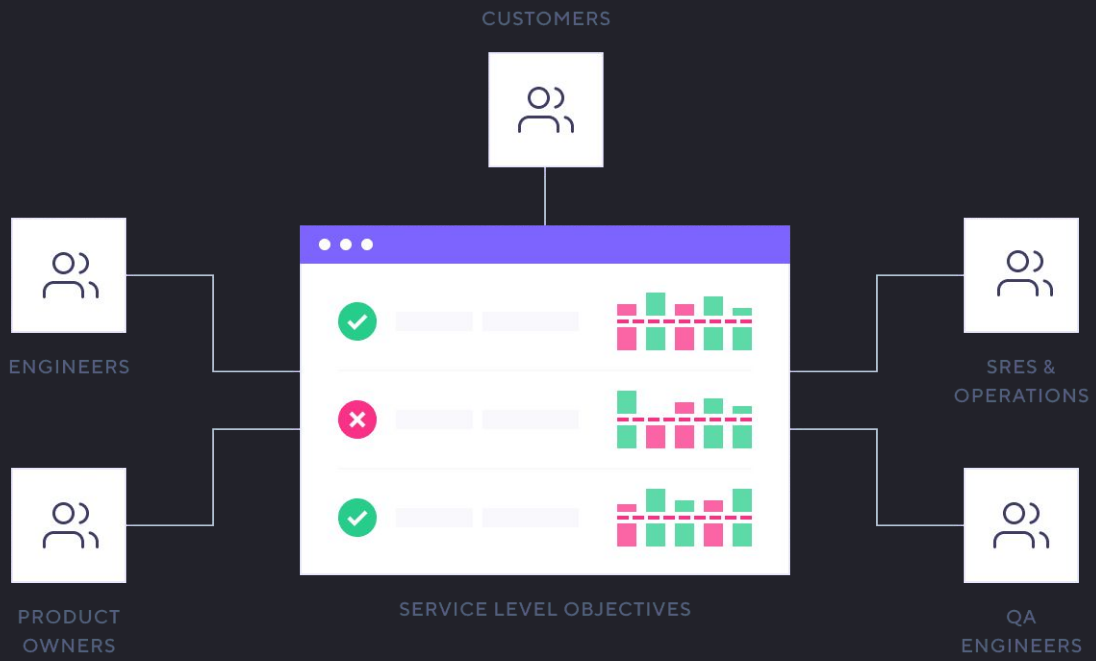
Distributed systems



Why load testing today?

SLOs - Reliability

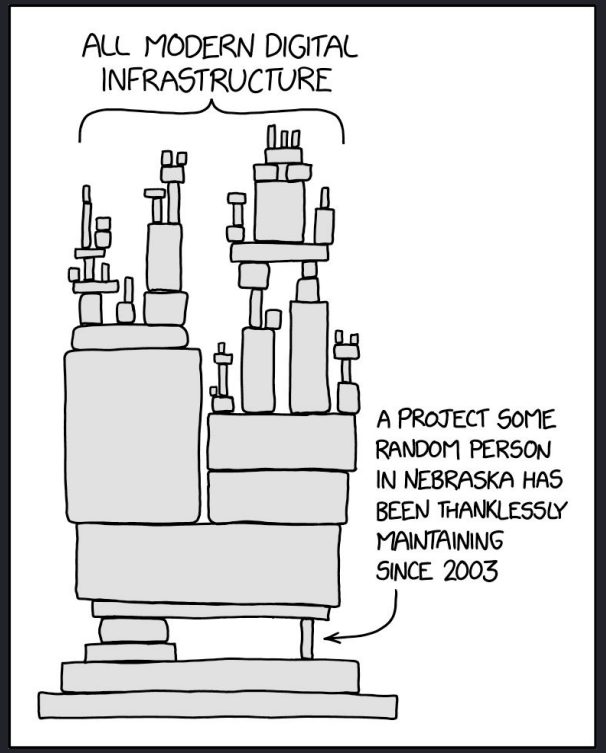
Be proactive



Why load testing today?

Fragility

Embrace and Prevent Failures



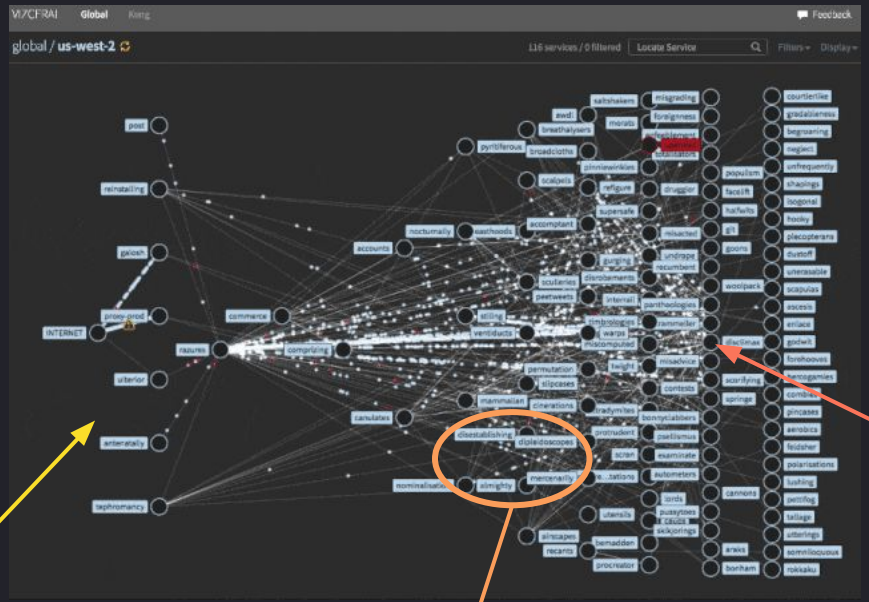
Modern testing
Why k6?

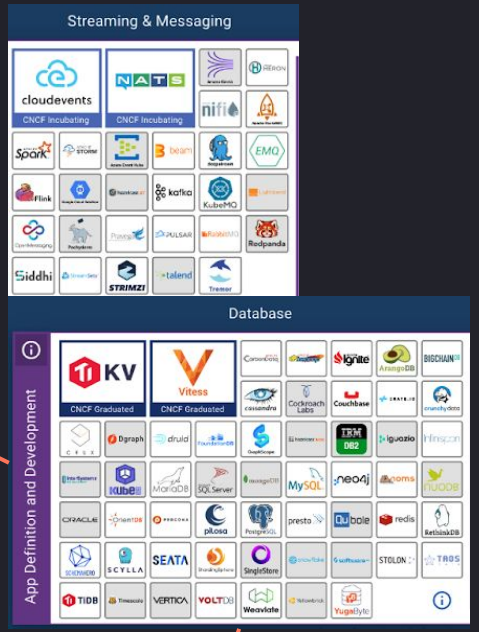
Modern testing. Why k6?

	PT Checklist / OLD WAY	DevOps / MODERN WAY
Release frequency	Quarterly or biannually	Weekly
Testing frequency	Before releases	AND nightly, feature branches, continuous, synthetic monitoring
How is initiated	Manually	Scheduled. Automatically as part of CI/CD
Testing environment	Test and Production	AND Staging, Long-lived and Short-lived ephemerals environment

Test Often - Continuous Testing

Modern testing. Distributed systems





Browser / HTTP / WS

HTTP / gRPC / SQS

End-to-end

Integration

Component

Start Simple and Test Frequently.
Fail Early.
Make Reliability a Team effort.

What is k6?

Free and open source tool

Paid performance testing platform

Open source

Same k6 test script for multiple execution modes

```
>_  k6 run script.js          LOCAL
```

```
>_  kubectl apply -f k6.yaml   K8S
```

```
>_  k6 cloud script.js         CLOUD
```

Test as Code - Programmable

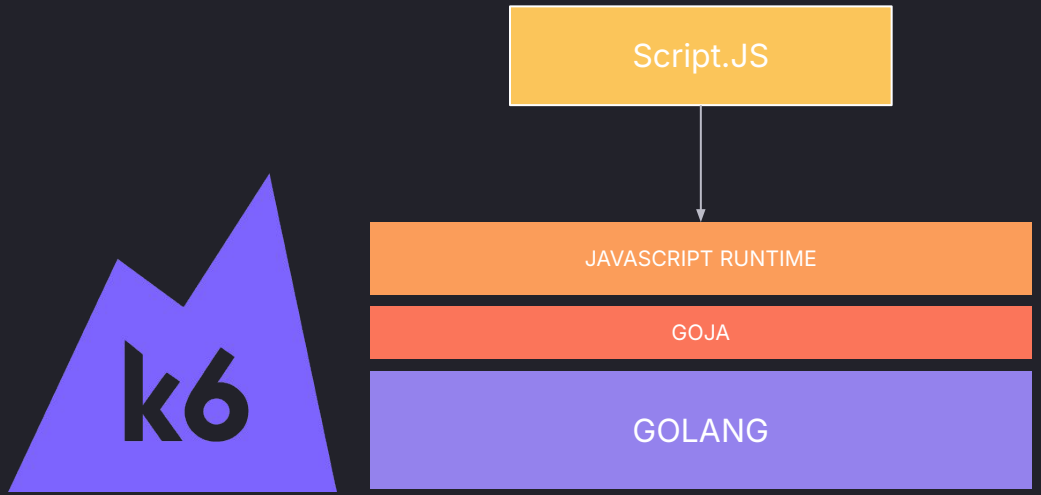
```
import { userFlowA, userFlow B } from './my-lib.js';

export default function () {
  userFlowA();
  userFlowB();
}
```

Performant

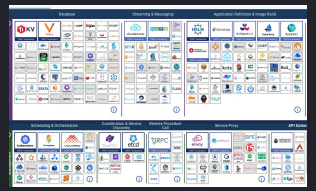
40K VUs in one load generator

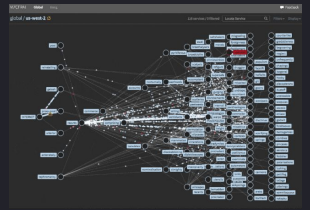
Limitations with NodeJS and browser libraries



Extensible

k6.io/docs/extensions/getting-started/explore/





Protocols

HTTP

Websocket

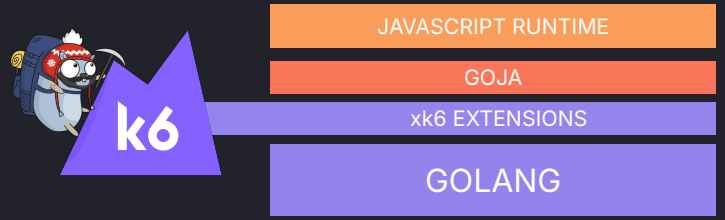
SQL / NoSQL

AMQP / Kafka

Crypto

Redis

....



Use cases

Browser testing

Infrastructure testing

Chaos testing

....

Outputs

InfluxDB

Prometheus

TimescaleDB

Grafana Cloud

NewRelic

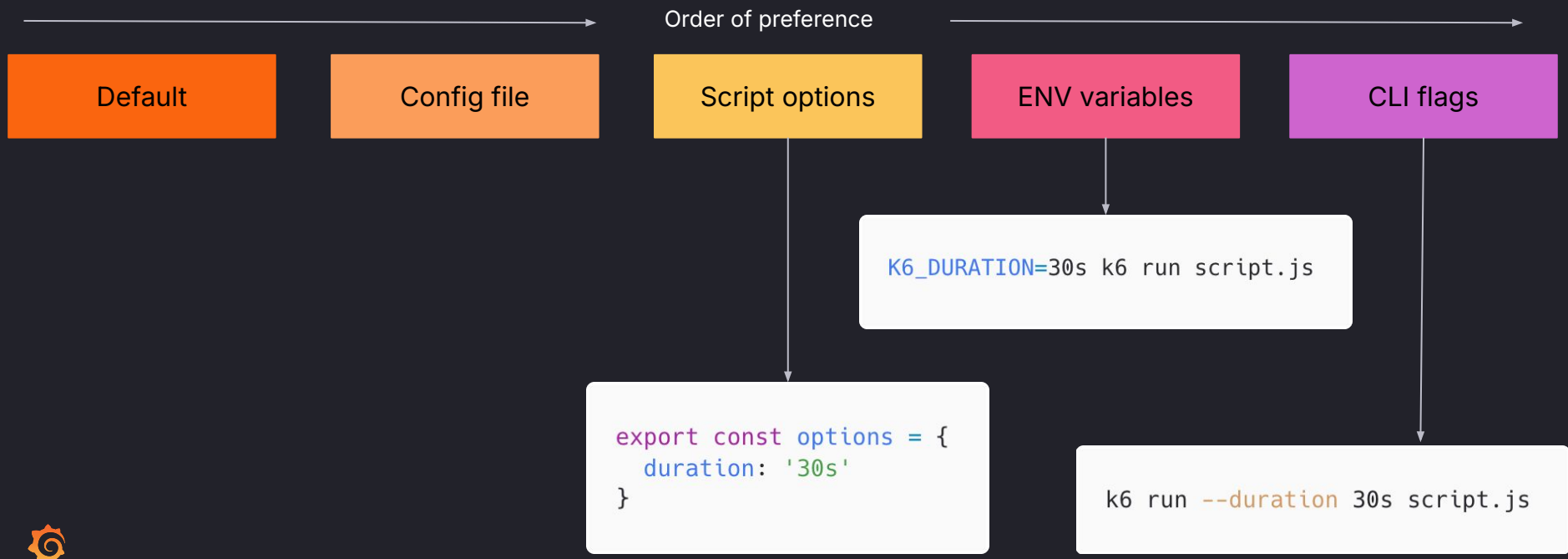
Dynatrace

....

k6 Concepts

Scenarios: Options

57 configurable options → https://k6.io/docs/using-k6/k6-options/reference/

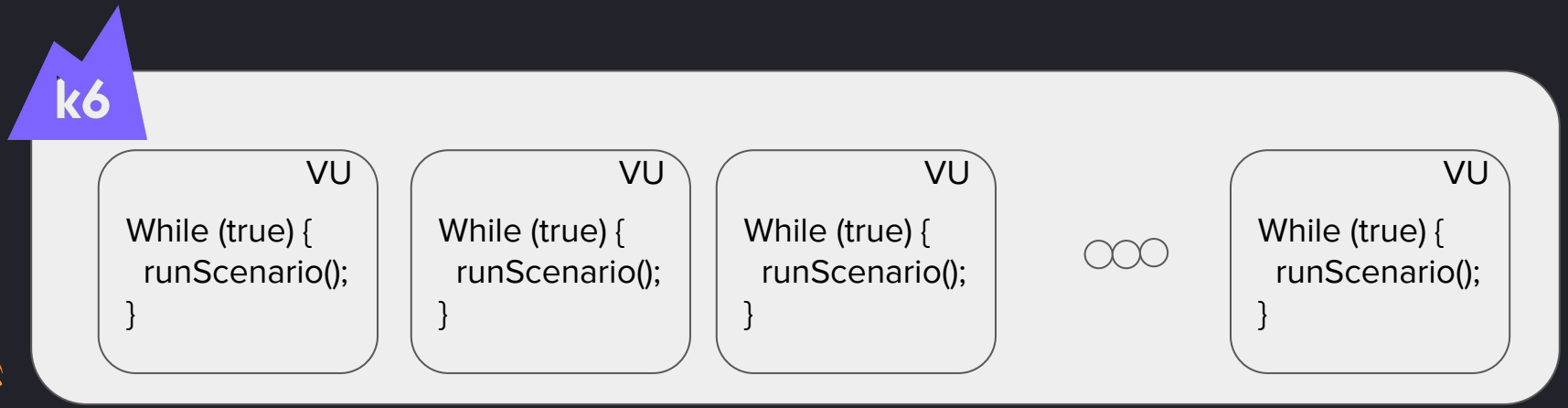


VU Duration Iterations Scenario

```
export const options = {
  vus: 100,
  duration: '1m'
}
```

```
export const options = {
  vus: 100,
  iterations: 200
}
```

```
export default function () {
  // test logic
}
```



Built-in Metrics

```
checks.........................: 100.00% ✓ 34       ✗ 0
data_received..................: 25 kB   2.6 kB/s
data_sent......................: 11 kB   1.1 kB/s
http_req_blocked...............: avg=275.57µs min=8µs    ...
http_req_connecting............: avg=137.6µs  min=0s     ...
http_req_duration..............: avg=154ms    min=7.47ms ...
  { expected_response:true }...: avg=154ms    min=7.47ms ...
http_req_failed................: 0.00%   ✓ 0        ✗ 35
http_req_receiving.............: avg=323.51µs min=81µs   ...
http_req_sending...............: avg=185.25µs min=46µs   ...
http_req_tls_handshaking.......: avg=0s       min=0s
http_req_waiting...............: avg=153.49ms min=4.64ms ...
http_reqs......................: 35      3.632738/s
iteration_duration.............: avg=1.09s    min=130µs  ...
iterations.....................: 34      3.528946/s
vus............................: 3       min=2      max=5
vus_max........................: 5       min=5      max=5
```

Custom Metrics

new Trend(‘metric_name’);

new Rate(‘metric_name’);

new Counter(‘metric_name’);

new Gauge(‘metric_name’);

```
import { Counter } from 'k6/metrics';

const myCounter = new Counter('my_counter');

export default function () {
  myCounter.add(1);
  myCounter.add(2);
}
```

Checks

Thresholds

Like Unit testing, for Performance

```
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,
  });
}
```

```
export const options = {
  thresholds: {
    http_req_failed: ['rate<0.01'],
    http_req_duration: ['p(95)<400'],
    'http_req_duration{type:API}': ['p(95)<200'],
   my_custom_metric: ['p(99)<200']
  },
};
```

Scenario Types

Constant VUs

Ramping VUs

Shared iterations

Per VU iterations

Constant Arrival Rate

Ramping Arrival Rate

```
export const options = {
  scenarios: {
    my_scenario: {
      executor: "constant-arrival-rate",
      duration: "10s",

      // 100 iters per second
      rate: 100,
      timeUnit: "1s",
      ...
```

Multiple Scenarios

```
export const options = {
  scenarios: {
    browser: {
      executor: 'constant-vus',
      exec: 'browserReadNews',
      vus: 1,
      duration: '3m',
    },
    news: {
      executor: 'constant-vus',
      exec: 'stressNewsAPI',
      vus: 5000,
      duration: '3m',
    },
  },
};
```

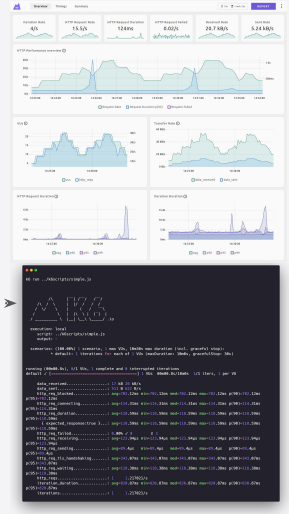
Much more in
k6 docs

k6 Results


/\ |‾‾| /‾‾/ /‾‾/
/\ / \ | |/ / / /
/ \/ \ | (/ ‾‾\
/ \ | |\ \ | (‾) |
/ ‾‾‾‾‾‾‾‾‾ \ |__| __\ _____/ .io
execution: local
script: supersimple.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)
running (00m00.5s), 0/1 VUs, 1 complete and 0 interrupted iterations
default ✓ [======================================] 1 VUs 00m00.5s/10m0s 1/1 iters, 1 per VU
data_received..................: 5.9 kB 12 kB/s
data_sent......................: 636 B 1.3 kB/s
http_req_blocked...............: avg=149.56ms min=136.61ms med=149.56ms max=162.51ms p(90)=159.92ms p(95)=161.22ms
http_req_connecting............: avg=78.84ms min=36.84ms med=78.84ms max=120.83ms p(90)=112.43ms p(95)=116.63ms
http_req_duration..............: avg=91.53ms min=31.86ms med=91.53ms max=151.19ms p(90)=139.26ms p(95)=145.22ms
{ expected_response:true }...: avg=91.53ms min=31.86ms med=91.53ms max=151.19ms p(90)=139.26ms p(95)=145.22ms
http_req_failed................: 0.00% ✓ 0 ✗ 2
http_req_receiving.............: avg=340.99µs min=250µs med=340.99µs max=432µs p(90)=413.8µs p(95)=422.9µs
http_req_sending...............: avg=152µs min=117µs med=152µs max=187µs p(90)=180µs p(95)=183.5µs
http_req_tls_handshaking.......: avg=49.78ms min=0s med=49.78ms max=99.57ms p(90)=89.61ms p(95)=94.59ms
http_req_waiting...............: avg=91.03ms min=31.49ms med=91.03ms max=150.57ms p(90)=138.66ms p(95)=144.62ms
http_reqs......................: 2 4.127439/s
iteration_duration.............: avg=482.66ms min=482.66ms med=482.66ms max=482.66ms p(90)=482.66ms p(95)=482.66ms
iterations.....................: 1 2.063719/s

k6 Results

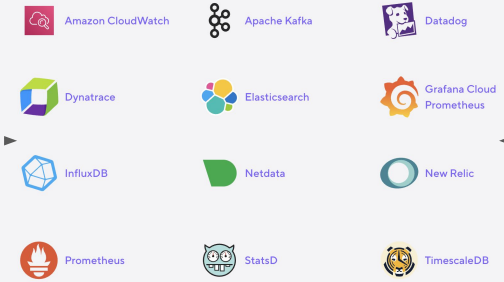
k6 run script.js



Web dashboard

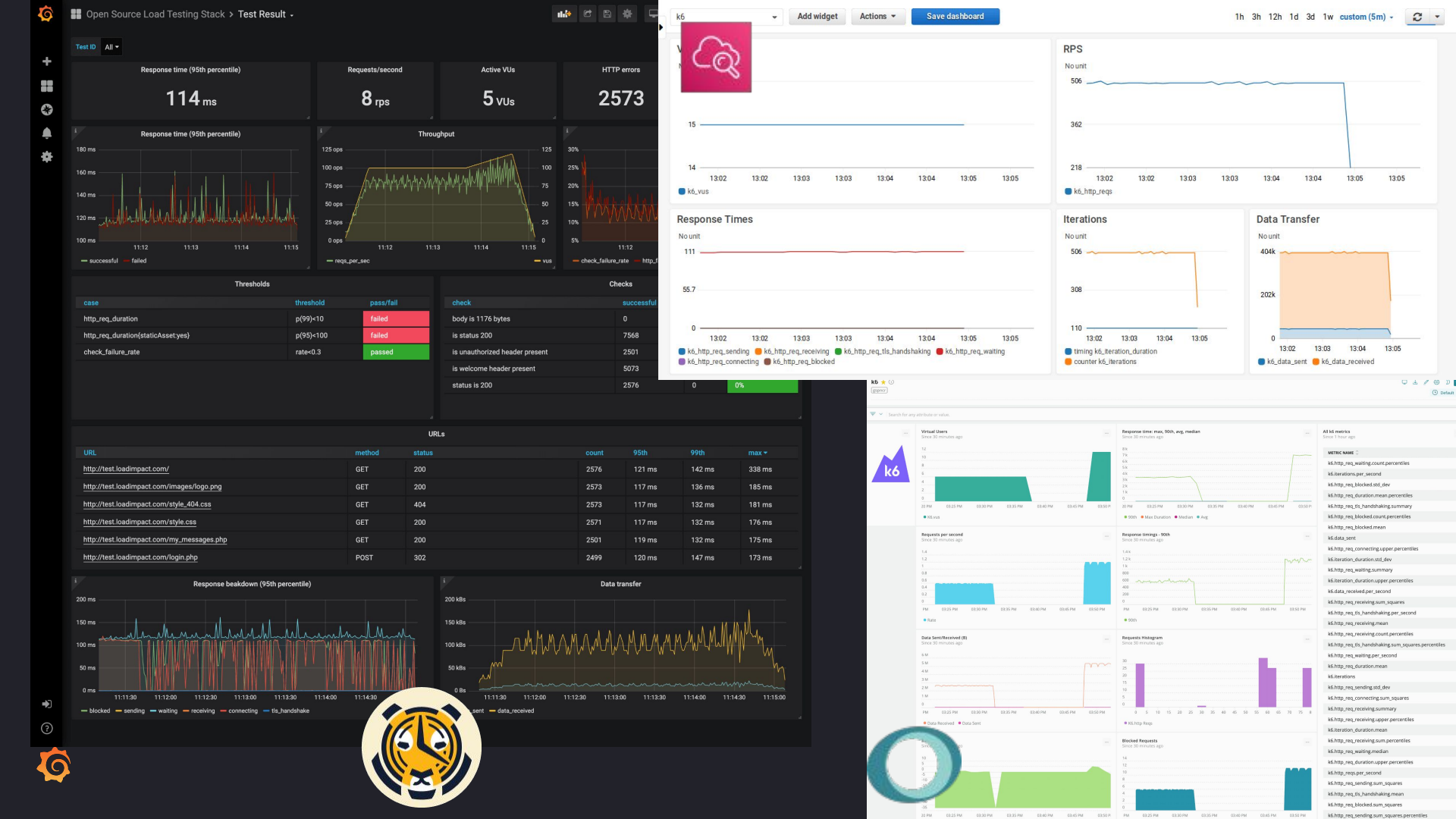
Terminal

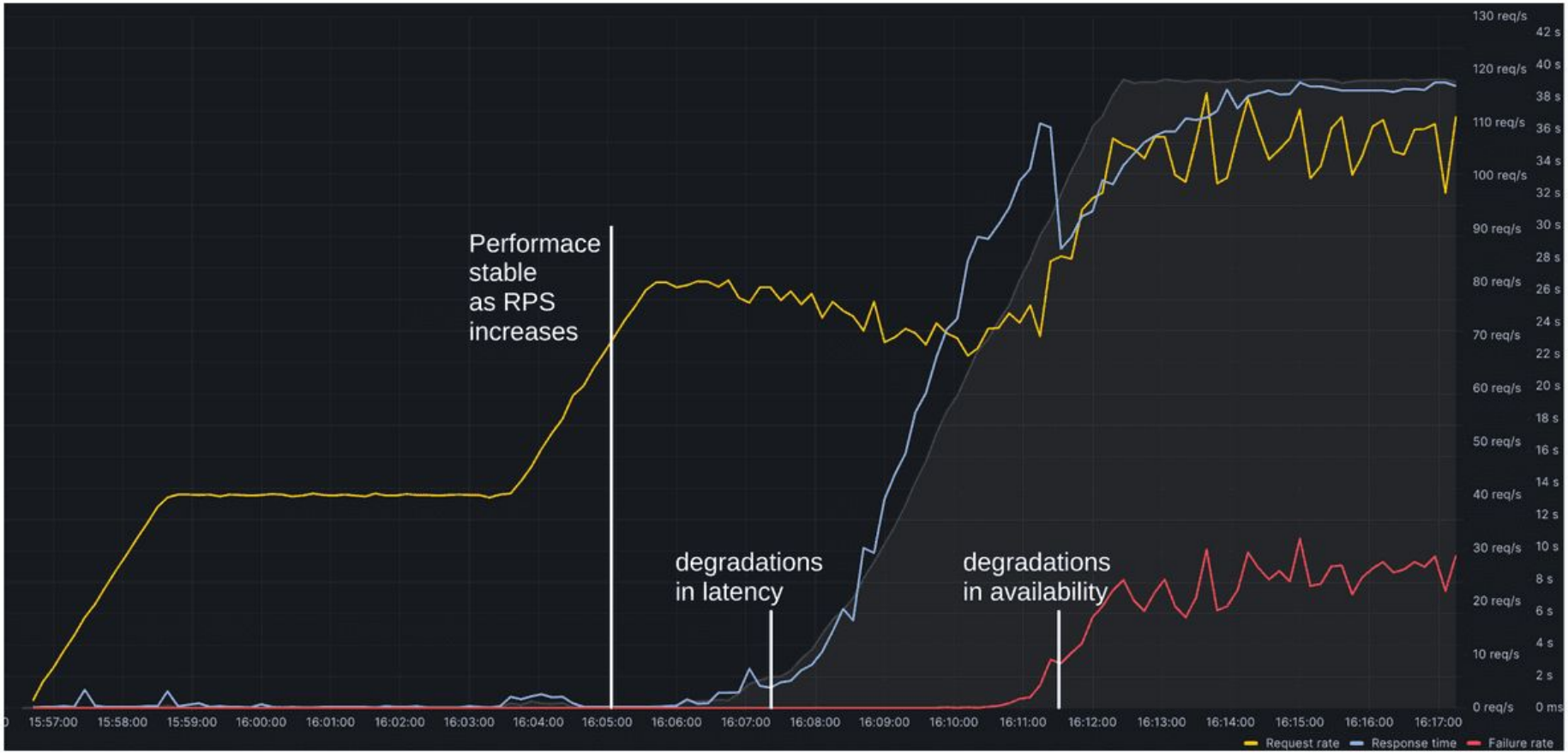
k6 run script.js -o output-xyz



Storage

Visualization


Open Source Load Testing Stack › Test Result
Test ID All
Response time (95th percentile)
114 ms
Requests/second
8 rps
Active VUs
5 VUs
HTTP errors
2573
Response time (95th percentile)
Throughput
Thresholds
case threshold pass/fail
http_req_duration p(99)<10 failed
http_req_duration{staticAsset:yes} p(95)<100 failed
check_failure_rate rate<0.3 passed
Checks
check successful
body is 1176 bytes 0
is status 200 7568
is unauthorized header present 2501
is welcome header present 5073
status is 200 2576 0 0%
URLs
URL method status count 95th 99th max
http://test.loadimpact.com/ GET 200 2576 121 ms 142 ms 338 ms
http://test.loadimpact.com/images/logo.png GET 200 2573 117 ms 136 ms 185 ms
http://test.loadimpact.com/style_404.css GET 404 2573 117 ms 132 ms 181 ms
http://test.loadimpact.com/style.css GET 200 2571 117 ms 132 ms 176 ms
http://test.loadimpact.com/my_messages.php GET 200 2501 119 ms 132 ms 175 ms
http://test.loadimpact.com/login.php POST 302 2499 120 ms 147 ms 173 ms
Response beakdown (95th percentile)
Data transfer
k6
Add widget
Actions
Save dashboard
1h 3h 12h 1d 3d 1w custom (5m)
RPS
No unit
Response Times
Iterations
Data Transfer



Healthy performance looks flat