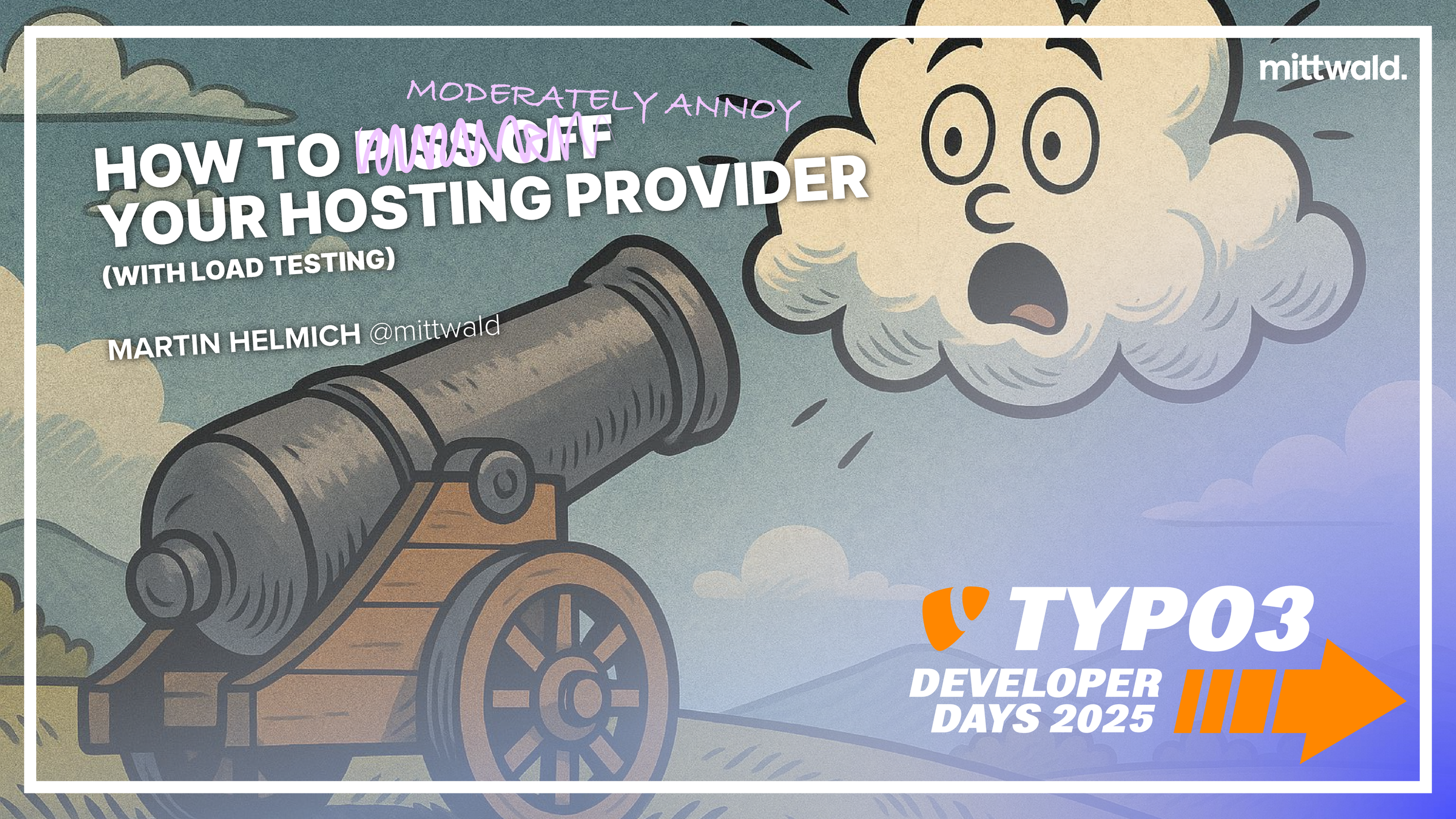

MODERATELY ANNOY
PISS OFF
HOW TO PISS OFF
YOUR HOSTING PROVIDER
(WITH LOAD TESTING)
MARTIN HELMICH @mittwald
mittwald.
TYPO3
DEVELOPER
DAYS 2025

mittwald.

MARTIN HELMICH

mittwald
Head of Architecture &
Developer Relations

TYPO3 Association
Board Member

PHWT
Lecturer, Software Engineering
& Cloud Computing


ONCE UPON A TIME...
mittwald.
https://pixabay.com/photos/mother-and-daughter-picnic-storytime-2629795/

mittwald.

ADD TO CART

MY LITTLE ONLINE SHOP

- Built on WooCommerce
- Built by a solo developer

mittwald.

This webpage is not available

ERR_SERVER_TOO_WEAK

Reload

MY LITTLE ONLINE SHOP

- Built on WooCommerce
- Built by a solo developer
- Sells only one (particularly ugly) article...
- ...for a major german rapper
- ...who had just released a new album

*) not the actual ugly sweater


mittwald.
DEVOPS
DEVOPS
DEVOPS
DEVOPS

mittwald.

REASONS FOR FAILURE

RESOURCE EXHAUSTION (CPU, MEMORY, BANDWIDTH) · ARCHITECTURAL BOTTLENECKS · SHITTY PROGRAMMING · SERVER MISCONFIGURATION · IT'S ALWAYS DNS

Tim Mossholder
https://unsplash.com/photos/white-smoke-on-black-background-GoHaYpu7-ks


mittwald.
STORY TIME...
https://pixabay.com/photos/mother-and-daughter-picnic-storytime-2629795/

Own work

mittwald.

Own work

The festival “under test”

zdrei.com

Z3
zdrei.com

The festival “under test”

Large(st) EDM Festival | NRW, Germany | ~225k Total visitors

TYPO3 CMS v13
Single Server Setup (192 Cores, 512GB RAM)

Note:
Image from original presentation not included for copyright reasons.

PARODKAVILLE

https://unsplash.com/photos/man-in-grey-t-shirt-playing-dj-mixer-nu7AOx73UOM


mittwald.
CHOOSE YOUR
TOOLS

```
martin @ local $ ab -k -c 100 -t 60 https://my-loadtest.example/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking my-loadtest.example (be patient)
Finished 50000 requests

Server Software:        Apache/2.4.59
Server Hostname:        my-loadtest.example
Server Port:            80

Document Path:          /
Document Length:        17620 bytes

Concurrency Level:      100
Time taken for tests:   6.045 seconds
Complete requests:      50000
Failed requests:        14
   (Connect: 0, Receive: 0, Length: 14, Exceptions: 0)
Keep-Alive requests:    49554
Total transferred:      894980321 bytes
HTML transferred:       880753320 bytes
Requests per second:    8270.99 [#/sec] (mean)
Time per request:       12.090 [ms] (mean)
Time per request:       0.121 [ms] (mean, across all concurrent requests)
Transfer rate:          144577.66 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       3
Processing:     0   12   5.2     11      59
Waiting:        0   12   5.2     11      59
Total:          0   12   5.3     11      62

Percentage of the requests served within a certain time (ms)
  50%     11
  66%     13
  75%     14
  80%     15
  90%     18
  95%     21
  98%     25
  99%     29
 100%     62 (longest request)
```

mittwald.

LOAD TEST REQUIREMENTS

- Mimic user behaviour realistically
- Verify if service level objectives (SLOs) are met
- CI/CD integration

mittwald.

k6.io

k6 Grafana Labs

OPEN SOURCE GRAFANA CLOUD K6 PRICING DOCUMENTATION ABOUT SIGN IN SIGN UP

The best developer experience for load testing

Open source and SaaS for engineering teams

OPEN SOURCE >_ GRAFANA CLOUD K6 >_

LOVED BY DEVELOPERS. TRUSTED BY BUSINESSES

amazon NOAA SEPHORA CITRIX Microsoft Yum! GitLab CARVANA

https://k6.io/

Untitled1

php • Untitled1 ×

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

export const options = {
  vus: 1,
  duration: '20s',
};

export default function() {
  const res = http.get('https://my-loadtest.example');

  check(res, {
    'is status 200': r => r.status === 200,
  });
}
```

Configure how many **VIRTUAL USERS** k6 should test with (and for how long)

The **DEFAULT EXPORT** determines how a virtual user should behave

30:1 PHP

mittwald.

mittwald.

..d/t3cm-k6/01-smoketest

~/Playground/t3cm-k6/01-smoketest (20.799s)

```
k6 run script.js
```

```
          /\      |‾‾| /‾‾/   /‾‾/
     /\  /  \     |  |/  /   /  /
    /  \/    \    |     (   /   ‾‾\
   /          \   |  |\  \ |  (‾)  |
  / __________ \  |__| \__\ \_____/ .io

     execution: local
        script: script.js
        output: -

     scenarios: (100.00%) 1 scenario, 1 max VUs, 50s max duration (incl. graceful stop):
              * default: 1 looping VUs for 20s (gracefulStop: 30s)


     ✓ is status 200

     checks.........................: 100.00% ✓ 2640       ✗ 0
     data_received..................: 127 MB  6.3 MB/s
     data_sent......................: 198 kB  9.9 kB/s
     http_req_blocked...............: avg=7.17µs   min=0s     med=3µs    max=1.85ms  p(90)=4µs     p(95)=5µs
     http_req_connecting............: avg=3.12µs   min=0s     med=0s     max=1.04ms  p(90)=0s      p(95)=0s
     http_req_duration..............: avg=7.44ms   min=6.39ms med=6.99ms max=66.27ms p(90)=8.54ms  p(95)=9.11ms
       { expected_response:true }...: avg=7.44ms   min=6.39ms med=6.99ms max=66.27ms p(90)=8.54ms  p(95)=9.11ms
     http_req_failed................: 0.00%   ✓ 0          ✗ 2640
     http_req_receiving.............: avg=121.01µs min=27µs   med=94µs   max=3.27ms  p(90)=178.1µs p(95)=262.09µs
     http_req_sending...............: avg=31.8µs   min=3µs    med=12µs   max=2.64ms  p(90)=20µs    p(95)=26µs
     http_req_tls_handshaking.......: avg=0s       min=0s     med=0s     max=0s      p(90)=0s      p(95)=0s
     http_req_waiting...............: avg=7.29ms   min=6.27ms med=6.84ms max=66.16ms p(90)=8.36ms  p(95)=8.88ms
     http_reqs......................: 2640    131.942282/s
     iteration_duration.............: avg=7.56ms   min=6.45ms med=7.1ms  max=66.38ms p(90)=8.72ms  p(95)=9.32ms
     iterations.....................: 2640    131.942282/s
     vus............................: 1       min=1        max=1
     vus_max........................: 1       min=1        max=1


running (20.0s), 0/1 VUs, 2640 complete and 0 interrupted iterations
default ✓ [======================================] 1 VUs  20s
```

~/Playground/t3cm-k6/01-smoketest

mittwald.

SCENARIO:
START OF TICKET SALE

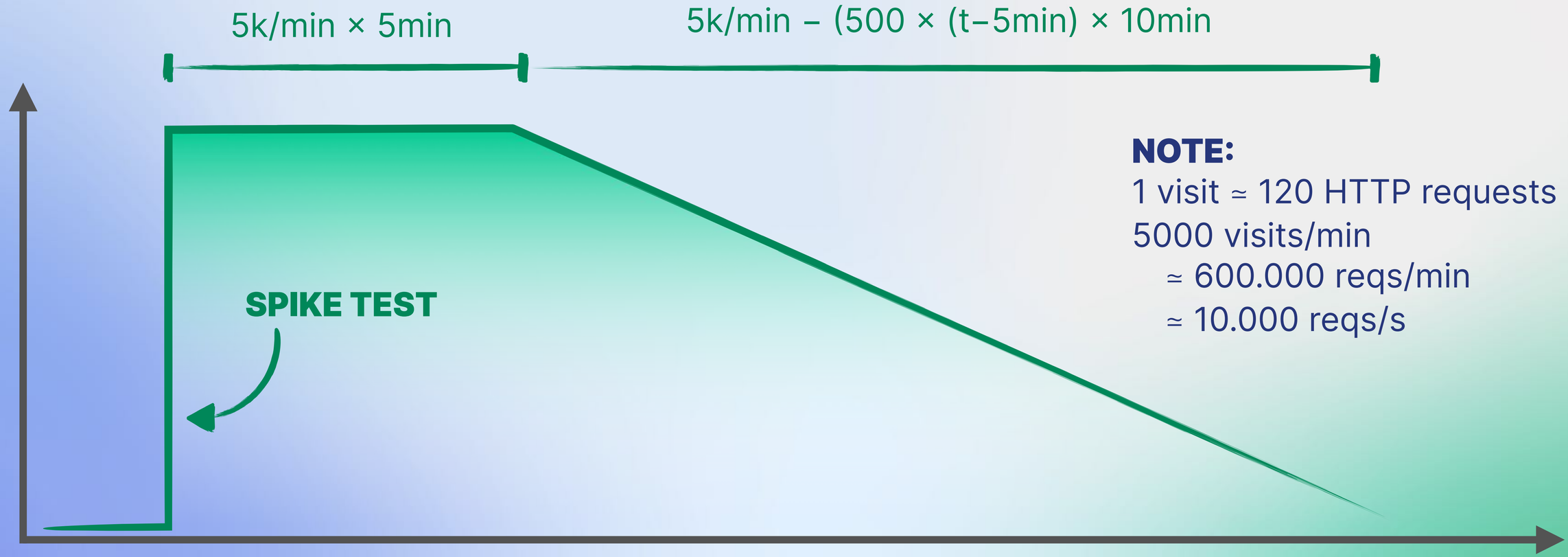
“50k visitors, 25k of those in the first 5min”

mittwald.

SCENARIO:

START OF TICKET SALE

50k visitors, 25k of those in the first 5min



NOTE:

1 visit ≃ 120 HTTP requests

5000 visits/min

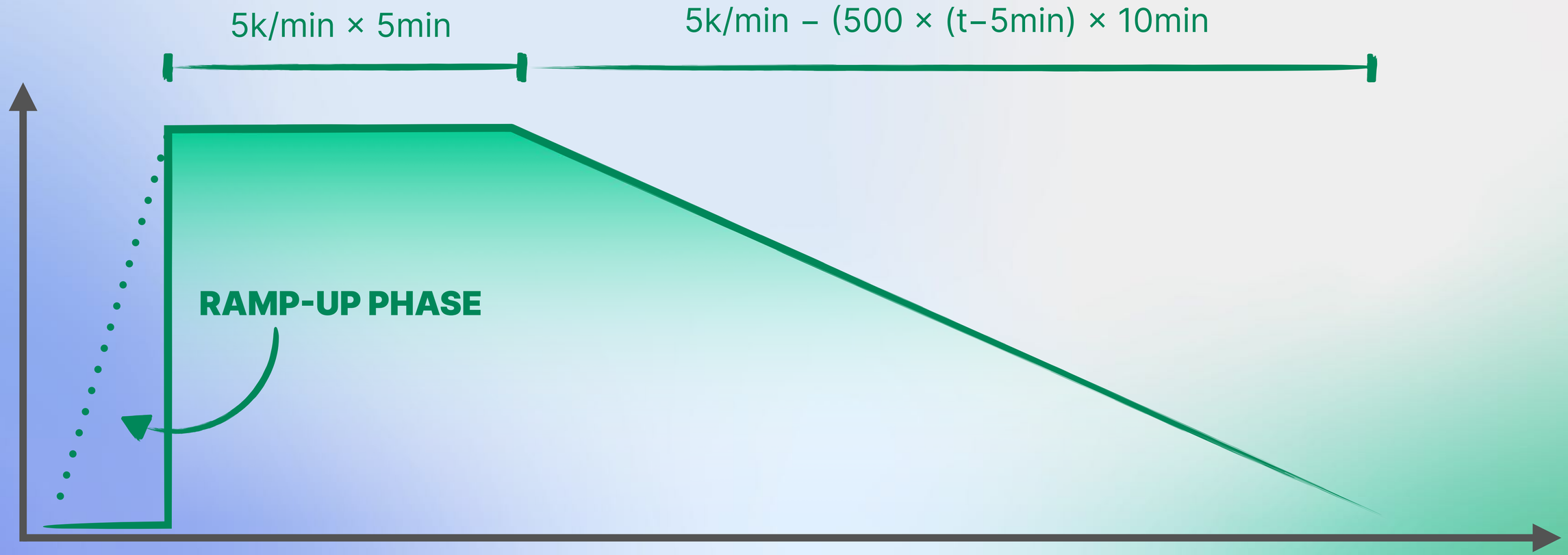
≃ 600.000 reqs/min

≃ 10.000 reqs/s

mittwald.

SCENARIO:
START OF TICKET SALE

50k visitors, 25k of those in the first 5min

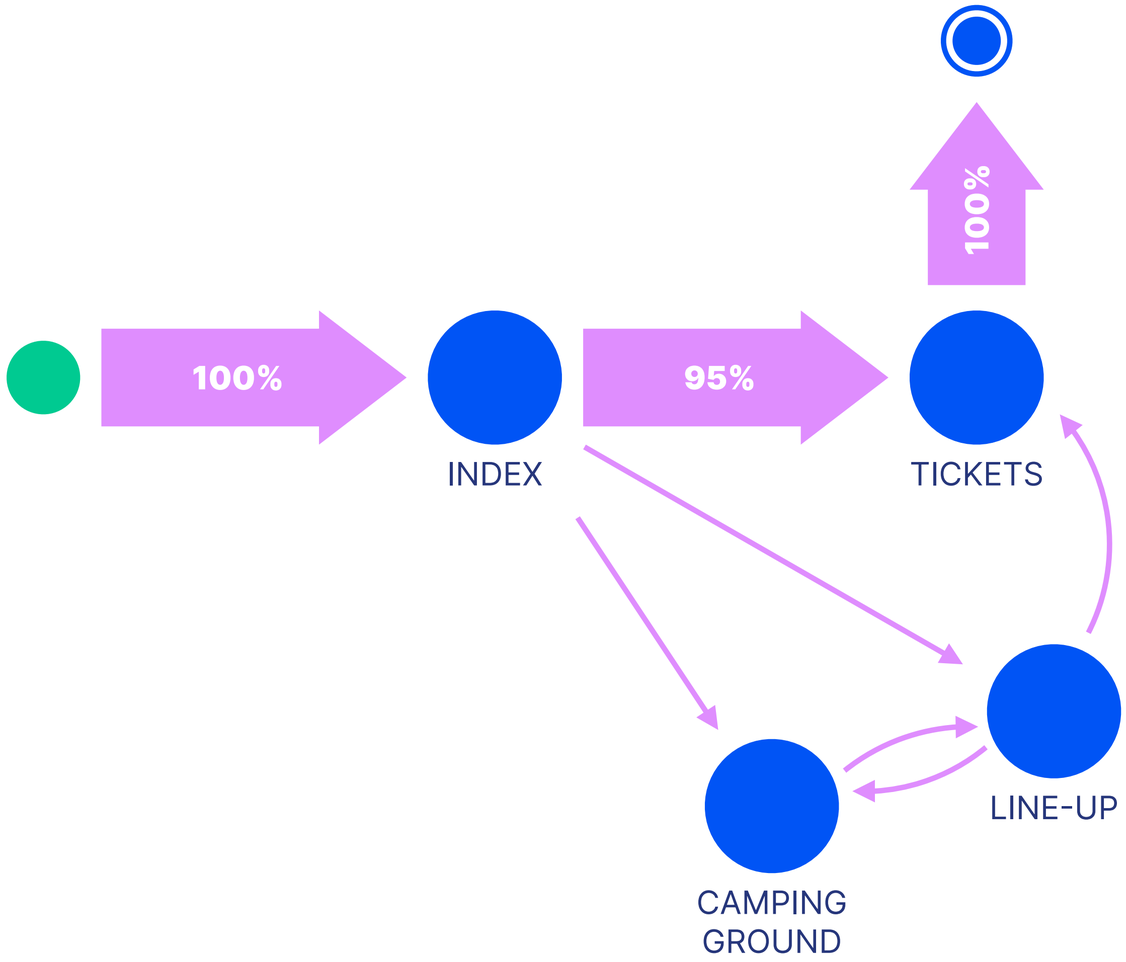


mittwald.

SPIKE TESTS

- If your caches aren’t warmed up already — it’s too late now.
- Auto-scaling won’t help you.
- Common DDoS mitigations might be actively harmful, because they might block legitimate traffic

mittwald.



SIMULATING A USER

(simplified)

SCENARIO:

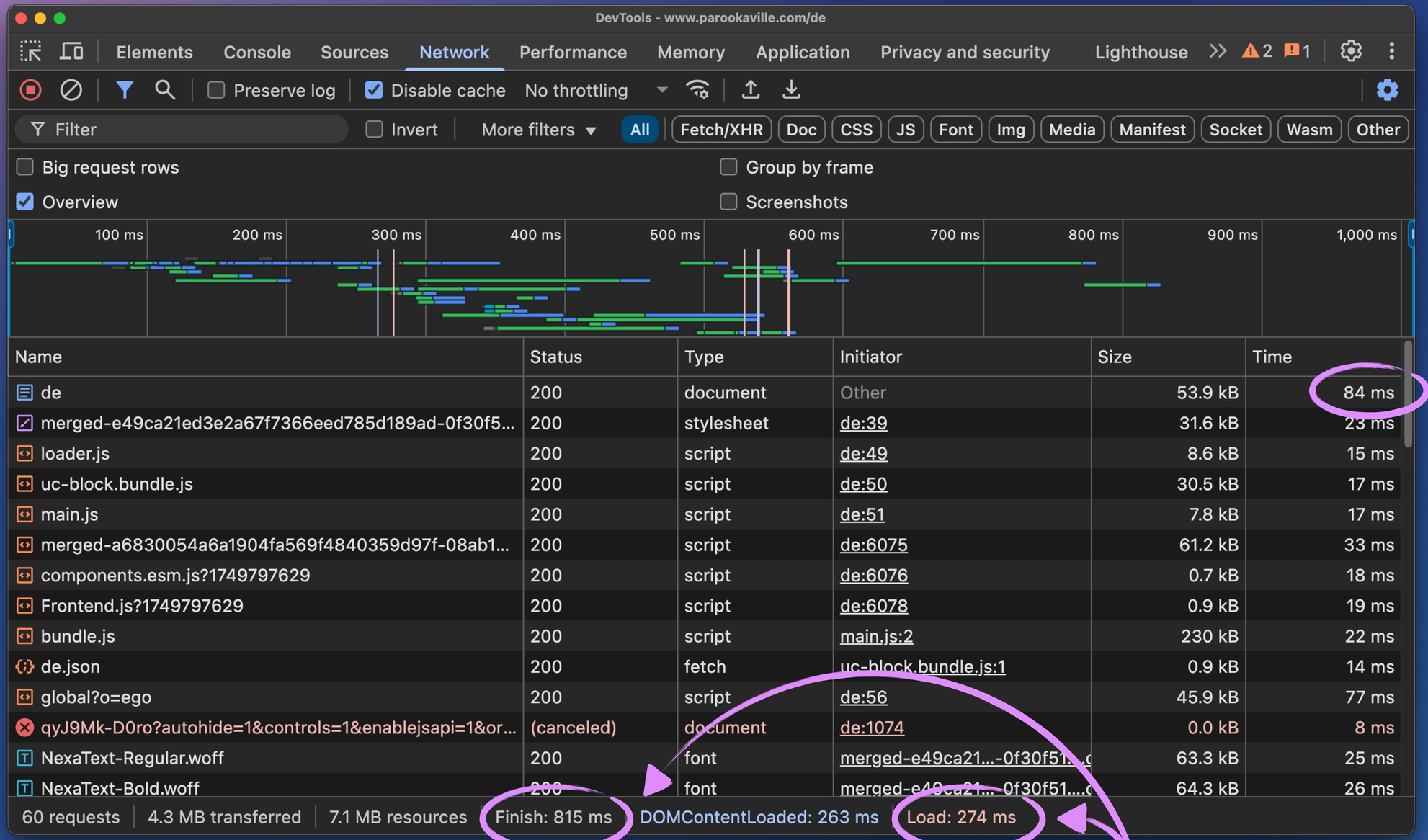
_ Start of ticket sale

OTHER SCENARIOS:

_ Artist announcements
_ On-location traffic during the festival
_ “Regular” visit

mittwald.

HTTP LOAD TESTING
vs
BROWSER TESTING



THIS ONE?

OR THIS?

WHAT METRIC TO OPTIMIZE FOR?

mittwald.

mittwald.

WHAT METRIC TO OPTIMIZE FOR?

CORE WEB VITALS

LOADING • Largest Contentful Paint (LCP), First Contentful Paint (FCP)

INTERACTIVITY • Interaction to Next Paint (INP)

VISUAL STABILITY • Cumulative Layout Shift (CLS)

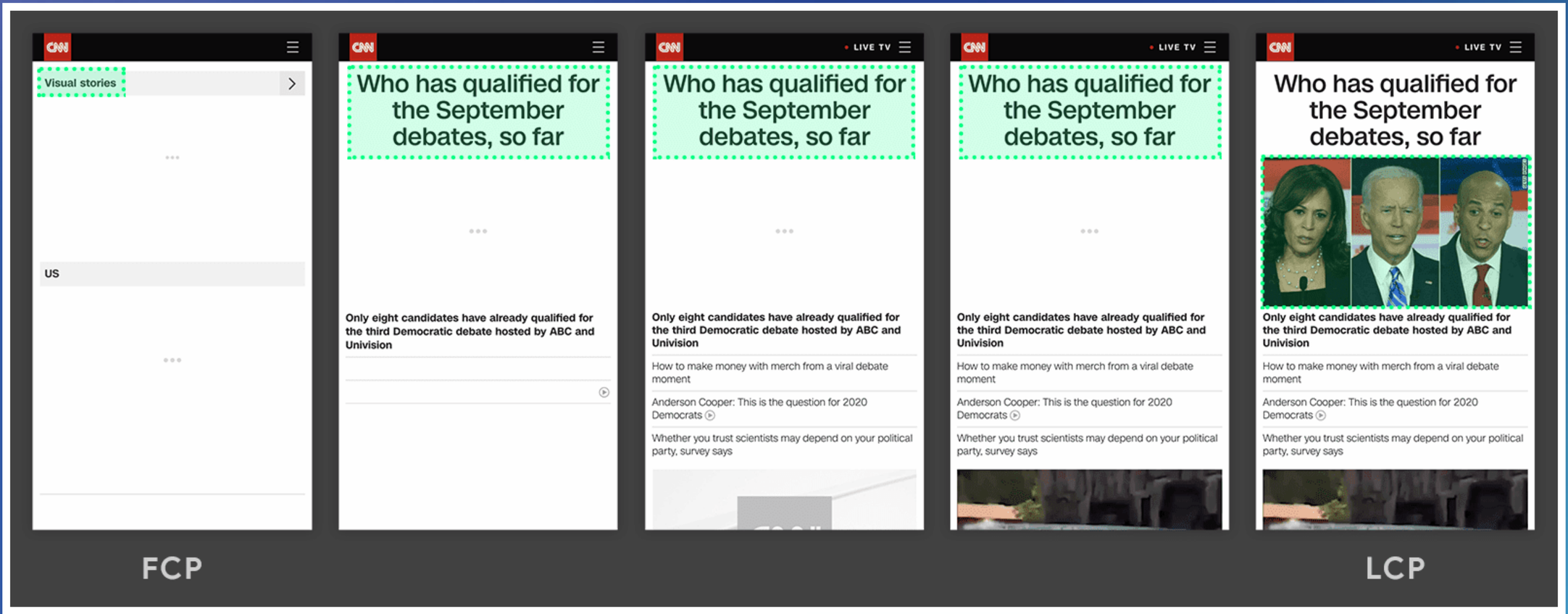
mittwald.

CORE WEB VITALS

LOADING • Largest Contentful Paint (LCP)

2.5sec 4sec

GOOD	NEEDS IMPROVEMENT	POOR



https://web.dev/articles/lcp

Untitled1

php • Untitled1 ×

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

export const options = {
  scenarios: {
    ui: {
      executor: 'constant-vus',
      vus: '20',
      duration: '15min',
      options: {
        browser: {
          type: 'chromium',
        },
      },
    },
  },




};
```

define a **BROWSER TEST**

30:1 PHP

mittwald.

Untitled1

php • Untitled1 ×

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

export const options = {
  scenarios: {
    ui: {
      executor: 'constant-vus',
      vus: '20',
      duration: '15min',
      options: {
        browser: {
          type: 'chromium',
        },
      },
    },
  },
  thresholds: {
    checks: [‘rate>0.98'],
    browser_web_vital_ttfb: ['p(95)<100'],
    browser_web_vital_lcp: ['p(95)<4000'],
  },
};
```

CORE WEB VITALS are exposed as metrics

30:1 PHP

mittwald.

Untitled1

php • Untitled1 ×

```
export default async function () {
  const context = await browser.newContext();
  const page = await context.newPage();

  try {
    await page.goto(baseURL);

    await Promise.all([
      page.waitForNavigation(),
      page.locator('a[title="Tickets"]').click(),
    ]);

    const header = await page.locator('h1').textContent();
    check(header, {
      "expected heading is found": (h) => h == 'Foo',
    });
  } finally {
    await page.close();
  }
}
```

The API is designed to be compatible with **PLAYWRIGHT**

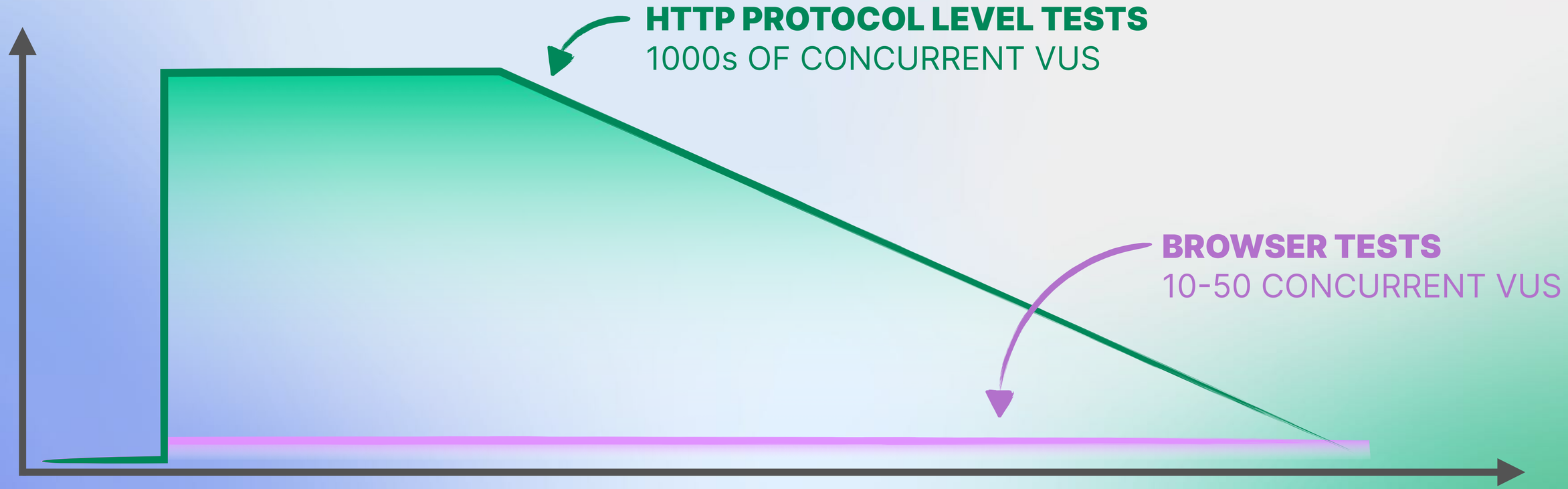
30:1 PHP

mittwald.

mittwald.

SCENARIO:
START OF TICKET SALE

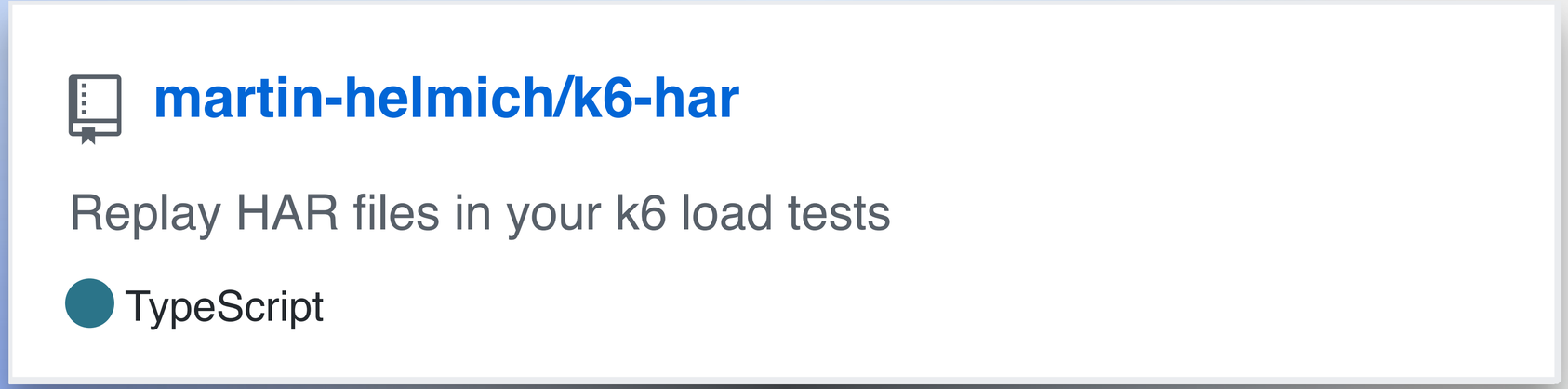
50k visitors, 25k of those in the first 5min



mittwald.

COMPROMISE

Simulating browser requests (at scale) without using a browser:
Replay an HAR file!



https://github.com/martin-helmich/k6-har

mittwald.

ONLINE
or
OFFLINE?

mittwald.

SIZING YOUR TESTING ENVIRONMENT

mittwald.

SIZING YOUR TESTING ENVIRONMENT

mittwald.

SIZING YOUR TESTING ENVIRONMENT

KERNEL PARAMETERS

```
$ sysctl -w net.ipv4.ip_local_port_range="1024 65535"
$ sysctl -w net.ipv4.tcp_tw_reuse=1
$ sysctl -w net.ipv4.tcp_timestamps=1
$ ulimit -n 250000
```

mittwald.

SIZING YOUR TESTING ENVIRONMENT

MEMORY CONSIDERATIONS

- 1-5MB per VU
- 300-500MB per VU for browser-based tests

CPU CONSIDERATIONS

- “it depends”
- In our case: 1500 VUs (w/ 200 reqs/iter) used 100% of 64 cores

mittwald.

SIZING YOUR TESTING ENVIRONMENT

MIND YOUR BANDWIDTH!

- also “it depends”
- From the original scenario: 83 visits/sec × 200 MB traffic/visit ≃ 16 GB/s ≃ 130 Gbit/s

mittwald.

```
     ✗ status is 200
       ↳  99% — ✓ 73931 / ✗ 154

     browser_data_received.....: 8.4 GB  8.5 MB/s
     browser_data_sent.........: 17 MB   17 kB/s
     browser_http_req_duration.....: avg=3.41s    min=489µs  med=461.45ms max=59.34s   p(90)=11.36s   p(95)=15.09s
     browser_http_req_failed.....: 0.66%   339 out of 50960
     browser_web_vital_cls.....: avg=0.016036 min=0      med=0.014317 max=0.086145 p(90)=0.068622 p(95)=0.086145
     browser_web_vital_fcp.....: avg=631.23ms min=44ms   med=264ms    max=20.04s   p(90)=1.47s    p(95)=1.78s
     browser_web_vital_fid.....: avg=549.69µs min=100µs  med=400µs    max=6.29ms   p(90)=799.99µs p(95)=1ms
   ✗ browser_web_vital_lcp.....: avg=2.24s    min=216ms  med=1.09s    max=25.83s   p(90)=4.78s    p(95)=7.3s
     browser_web_vital_ttfb.....: avg=314.17ms min=11.6ms med=53.69ms  max=7.57s    p(90)=1.02s    p(95)=1.23s
     checks.....................: 99.79%  73931 out of 74085
     data_received..............: 1.4 TB  1.4 GB/s
     data_sent..................: 8.4 GB  8.5 MB/s
     dropped_iterations.....: 20069   20.27137/s
     http_req_blocked...........: avg=72.88µs  min=72ns   med=337ns    max=3.12s    p(90)=491ns    p(95)=565ns
     http_req_connecting........: avg=31.39µs  min=0s     med=0s       max=3.1s     p(90)=0s       p(95)=0s
     http_req_duration..........: avg=778.54ms min=8.34ms med=553.55ms max=1m0s     p(90)=1.27s    p(95)=2.05s
       { expected_response:true }...: avg=736.48ms min=8.34ms med=550.16ms max=59.66s   p(90)=1.24s    p(95)=1.95s
     http_req_failed............: 1.16%   68064 out of 5853255
     http_req_receiving.....: avg=46.44ms  min=0s     med=2.83ms   max=38.17s   p(90)=74.71ms  p(95)=187.15ms
     http_req_sending...........: avg=41.69µs  min=3.28µs med=18.58µs  max=1.21s    p(90)=34.73µs  p(95)=45.87µs
     http_req_tls_handshaking.....: avg=24.67µs  min=0s     med=0s       max=692.48ms p(90)=0s       p(95)=0s
     http_req_waiting...........: avg=732.05ms min=3.35ms med=527.03ms max=1m0s     p(90)=1.2s     p(95)=1.83s
     http_reqs..................: 5853255 5912.277574/s
     iteration_duration.....: avg=35.72s   min=5.25s  med=19.43s   max=10m16s   p(90)=50.35s   p(95)=1m20s
     iterations.....: 32331   32.657017/s
     vus........................: 279     min=10              max=1510
     vus_max....................: 1510    min=20              max=1510


running (16m30.0s), 0000/1510 VUs, 32331 complete and 285 interrupted iterations
straightToTicket ✓ [======================================] 0279/1500 VUs  16m0s  00.26 iters/s
ui               ✓ [======================================] 10 VUs        15m0s
ERRO[0994] thresholds on metrics 'browser_web_vital_lcp' have been crossed
```

mittwald.

RUNNING LARGE TESTS

```
k6 run --execution-segment=0:1/3 \
  -o xk6-influxdb
```

```
k6 run --execution-segment=1/3:2/3 \
  -o xk6-influxdb
```

```
k6 run --execution-segment=2/3:1 \
  -o xk6-influxdb
```

Data sink

AVAILABLE TARGETS

- CloudWatch
- Kafka
- InfluxDB
- NewRelic
- OpenTelemetry
- Prometheus
- TimescaleDB
- (full list)

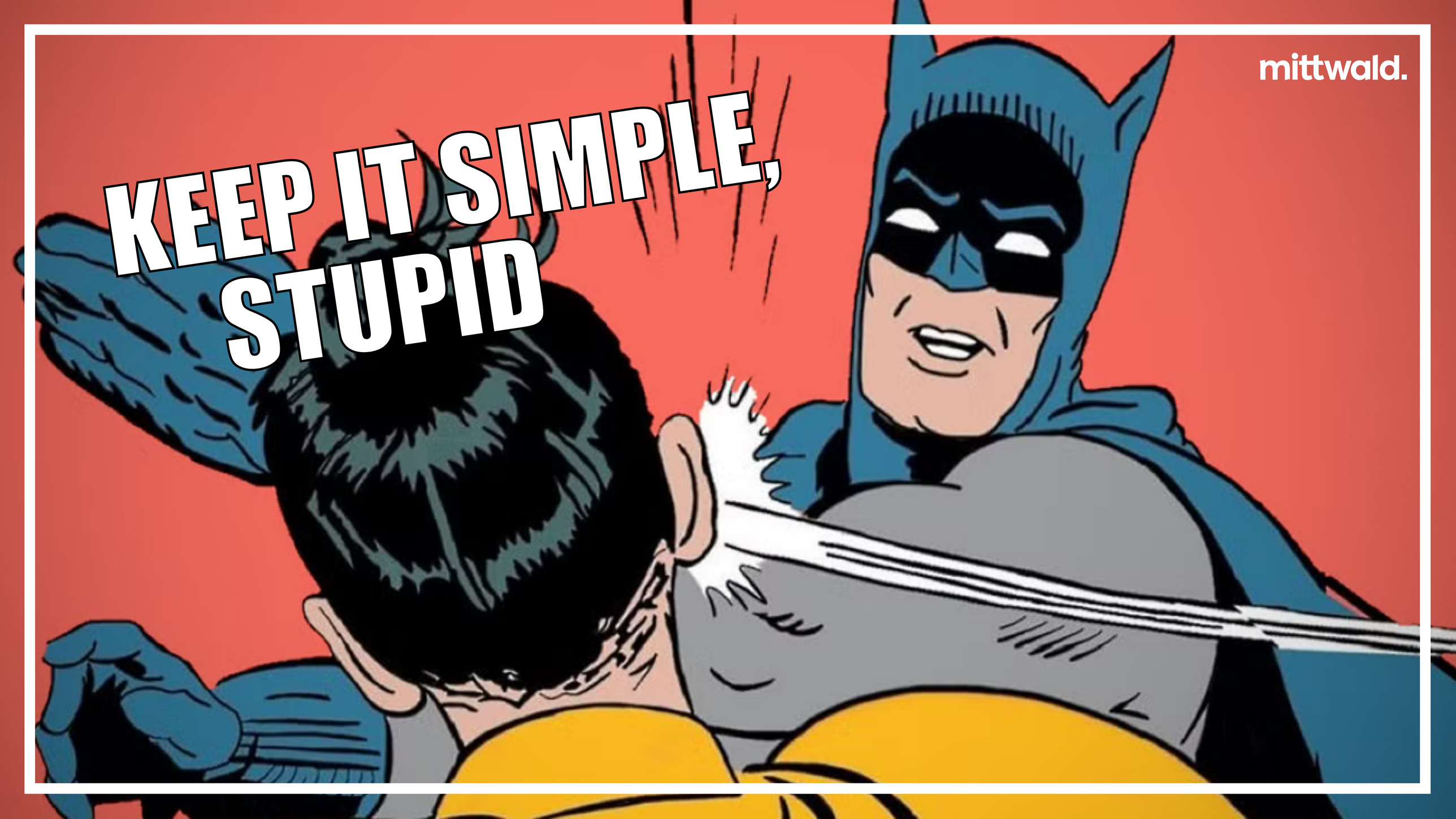
mittwald.

RUNNING LARGE TESTS

```
k6 run --execution-segment=0:1/3 \
  -o xk6-influxdb
```

```
k6 run --execution-segment=1/3:2/3 \
  -o xk6-influxdb
```

```
k6 run --execution-segment=2/3:1 \
  -o xk6-influxdb
```


KEEP IT SIMPLE,
STUPID
mittwald.

mittwald.

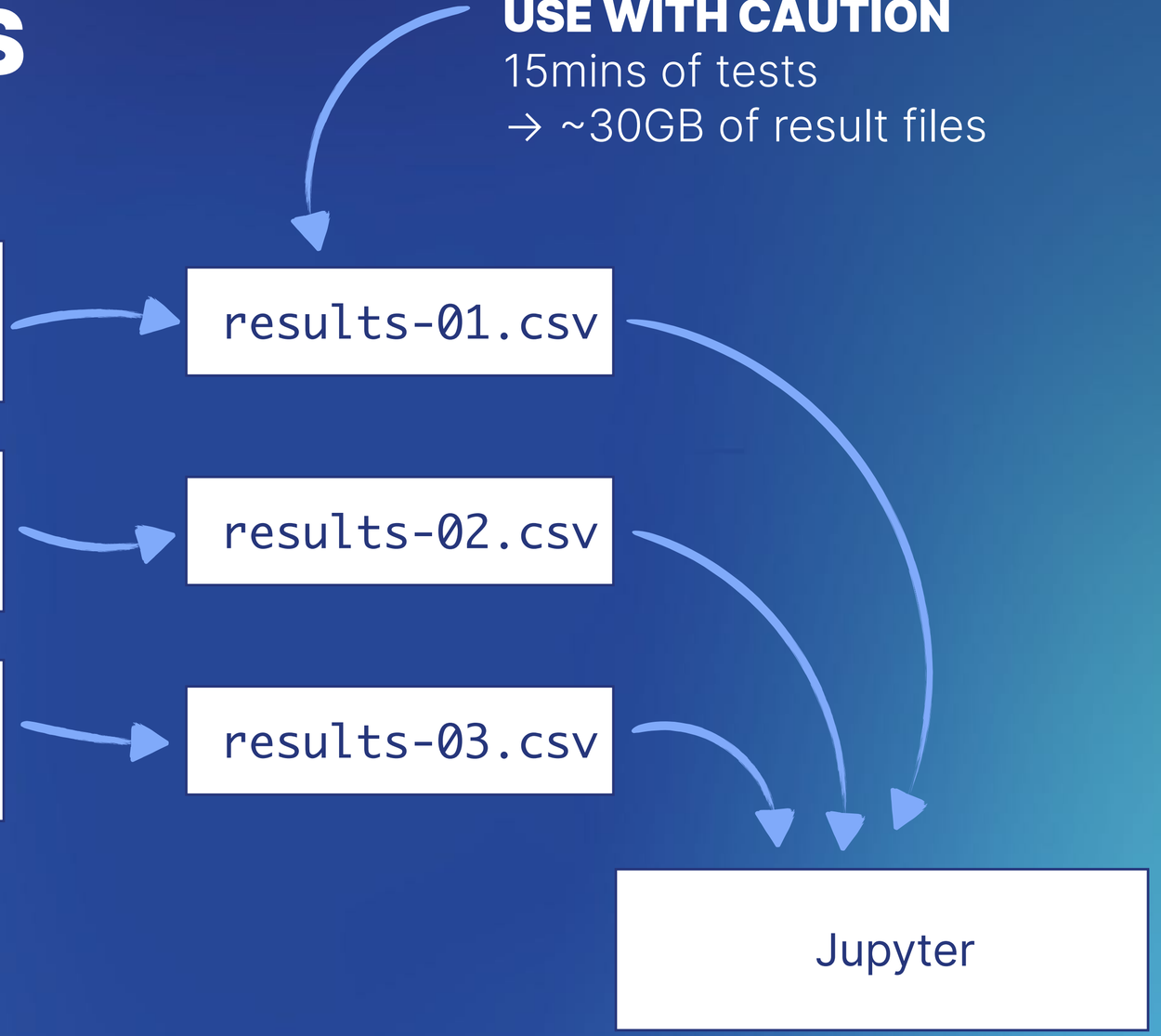
RUNNING LARGE TESTS

USE WITH CAUTION
15mins of tests
→ ~30GB of result files

```
k6 run --execution-segment=0:1/3 \
  -o csv=results-01.csv
```

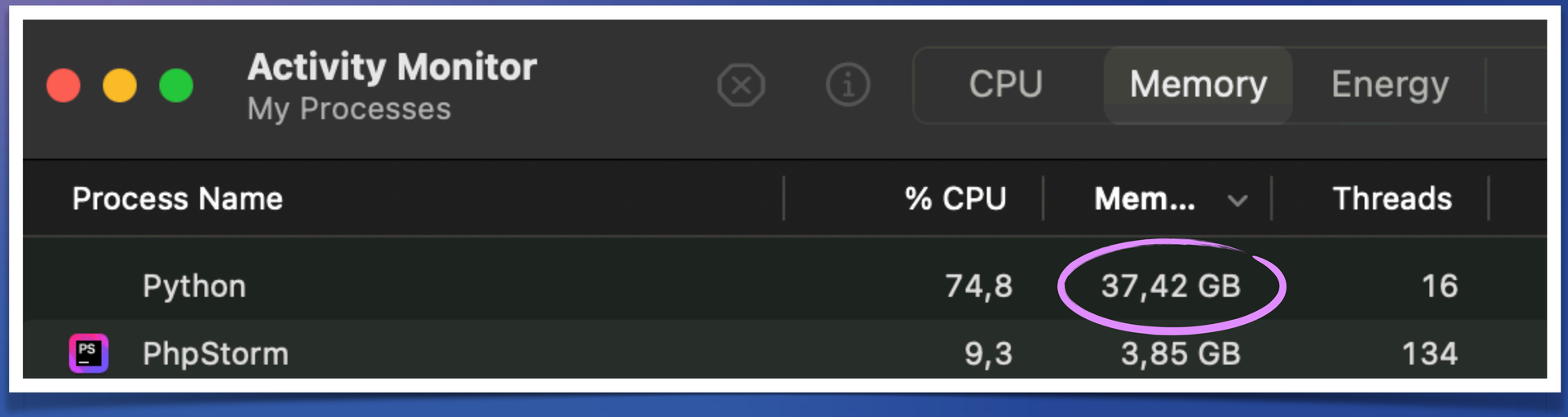
```
k6 run --execution-segment=1/3:2/3 \
  -o csv=results-02.csv
```

```
k6 run --execution-segment=2/3:1 \
  -o csv=results-03.csv
```

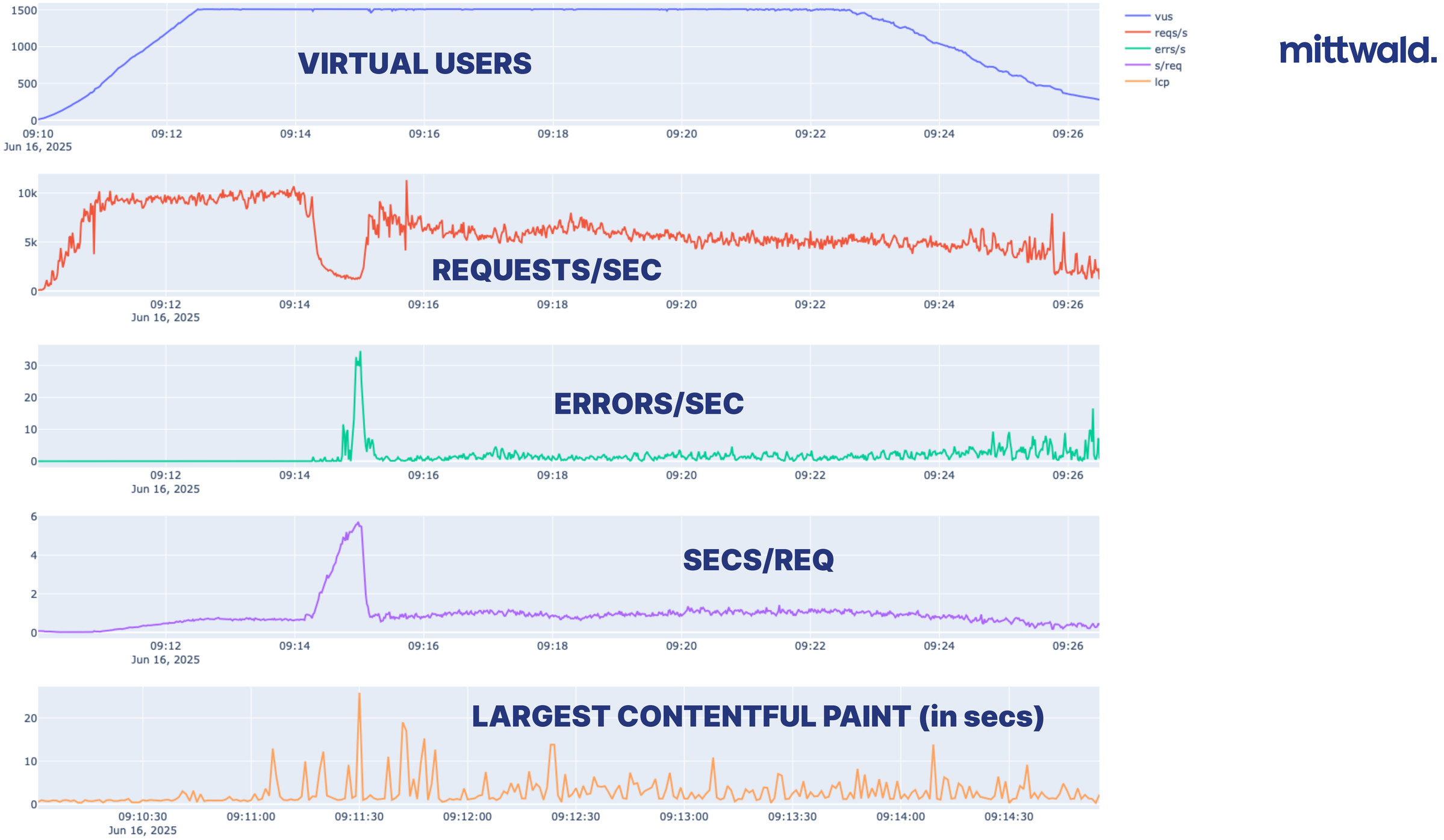


USE WITH CAUTION
15mins of tests
→ ~30GB of result files

mittwald.



"I PAID FOR 64 GIGS OF RAM,
THEREFORE I WILL USE 64 GIGS OF RAM"


vus
reqs/s
errs/s
s/req
lcp
VIRTUAL USERS
1500
1000
500
0
09:10
Jun 16, 2025
09:12
09:14
09:16
09:18
09:20
09:22
09:24
09:26
REQUESTS/SEC
10k
5k
0
09:12
Jun 16, 2025
09:14
09:16
09:18
09:20
09:22
09:24
09:26
ERRORS/SEC
30
20
10
0
09:12
Jun 16, 2025
09:14
09:16
09:18
09:20
09:22
09:24
09:26
SECS/REQ
6
4
2
0
09:12
Jun 16, 2025
09:14
09:16
09:18
09:20
09:22
09:24
09:26
LARGEST CONTENTFUL PAINT (in secs)
20
10
0
09:10:30
Jun 16, 2025
09:11:00
09:11:30
09:12:00
09:12:30
09:13:00
09:13:30
09:14:00
09:14:30

mittwald.

THINGS YOU CAN DO WHEN YOUR LOAD TEST FAILS

- Throw hardware at the problem
  (caution: might get expensive)
- Make smart choices
  - Configure web server and PHP-FPM for full resource utilization
  - Offload heavy assets into a CDN
  - Use caching proxies like Varnish

mittwald.

BUT WHAT ABOUT...

- **LOCUST**?
  - Works very similar, uses Python as scripting language instead of JavaScript
  - Same as k6: Open-Source, cloud option available
  - Written in Python (instead of Go); might (?) be slightly less performant
  - UI by default, but can also be used headlessly
  - Deciding factor: Personal preference

- **GATLING**?
  - Works very similar, uses JavaScript or Java as a scripting language
  - Basic open-source version available, but many features only available in Enterprise version
  - Deciding factor: Preference and your wallet

- **WRK**?
  - Uses Lua as a scripting language
  - More for raw throughput testing, not really optimized for complex user flows

mittwald.

mittwald.

IMPORTANT SERVICE ANNOUNCEMENT

- Some kinds of load tests (stress tests, or breakpoint tests) are **indistinguishable from DoS attacks**.
- If in doubt, get a **PtA (Permission to Attack)**. (your hosting provider will thank you 😉)
- When in the public cloud, **mind your (or your client's) traffic bill**.

Special thanks

https://zdrei.com/de/career-boost

zdrei.com

mittwald.

MARTIN HELMICH

m.helmich@mittwald.de
martin.helmich@typo3.org

WE'RE HIRING

Developer Relations Engineer
Infrastructure Engineer
CMS Support Engineer
Product Owner CMS Hosting
CMS Product Marketing Manager

(all genders, on-site)

https://www.mittwald.de/karriere