Performance Testing with K6

Chris A James — @chrisalwinjames

Agenda

— What is k6?
— Core Concepts
— Anatomy of a Test
— Writing or recording tests
— Result Visualisation
— Scaling out your test
— Questions

What is k6?

Chris A James — @chrisalwinjames

Unit-testing for performance

Chris A James — @chrisalwinjames

What is k6?

— Modern
— Flexible
— User Friendly

Core Concepts

Core Concepts

— What's a test?
— What are Virtual Users (VUs)?
— Types of tests
— Metrics
— Checks and Thresholds
— Test runtime and functions

What's a test?

What's a test?

A test can be as simple as this:

```
// script.js

import { sleep } from "k6";
import http from "k6/http";

export default function() {
  http.get("https://test.loadimpact.com/");
  sleep(3);
}
```

Virtual Users (VUs)

— An entity that executes a test, and makes requests.
— Simulate an actual user session.
— They run concurrently, and keep repeating the test until the test is over.
— When testing anything that is “User journey” related, webapps, websites, or API endpoints in a specific order you should think in terms of Virtual Users.

How many VUs?

$$VUs = \frac{numHourlySessions * avgSessionDurationInSecs}{3600}$$

- Use data from different times to:
  - simulate regular traffic
  - simulate busiest/peak hour
  - stress test your system

Types of tests

Types of tests

Baseline

```
export let options = {
  stages: [
    { duration: "1m", target: 10 },
    { duration: "10m", target: 10 },
    { duration: "1m", target: 0 }
  ],
};
```

Types of tests

Load

```
export let options = {
  stages: [
    { duration: "5m", target: 2000 },
    { duration: "15m", target: 2000 },
    { duration: "5m", target: 0 }
  ],
};
```

Types of tests

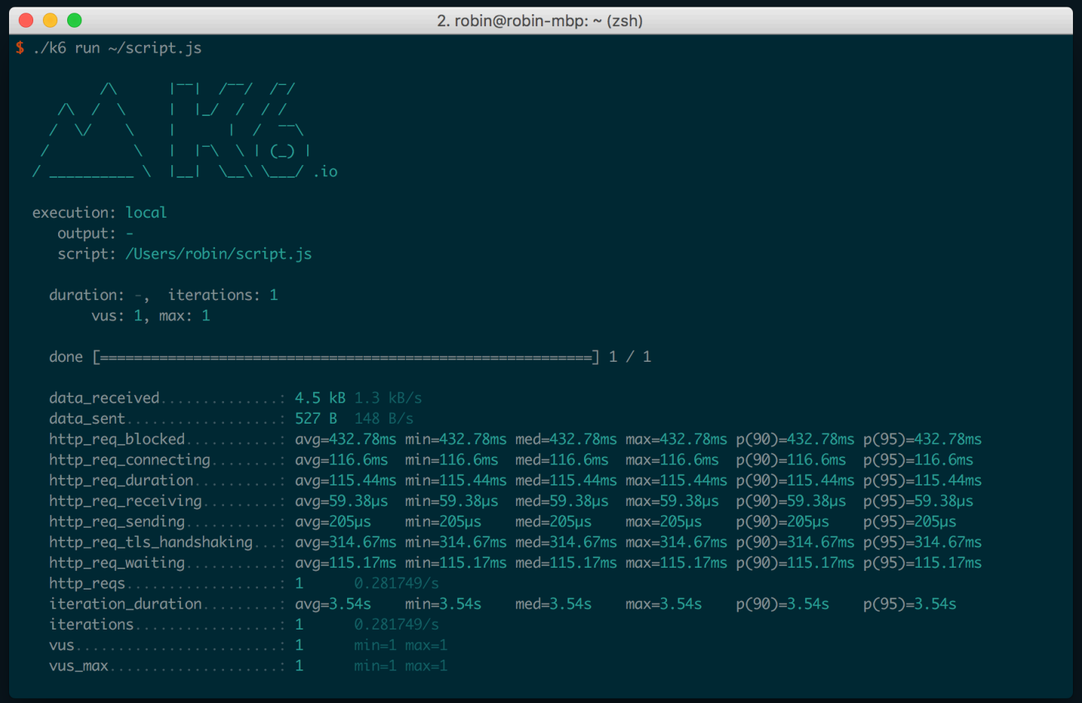
Spike

```
export let options = {
  stages: [
    { duration: "1m", target: 2000 },
    { duration: "9m", target: 2000 },
    { duration: "3m", target: 10000 },
    { duration: "7m", target: 10000 },
    { duration: "10m", target: 0 }
  ],
};
```

Metrics

Metrics

Standard Metrics



Metrics

Standard Metrics

— These are well documented on the k6 docs website.

Chris A James — @chrisalwinjames

HTTP-specific built-in metrics

There are also *built-in* metrics that will only be generated when/if HTTP requests are made:

Metric name	Type	Description
http_reqs	Counter	How many HTTP requests has k6 generated, in total.
http_req_blocked	Trend	Time spent blocked (waiting for a free TCP connection slot) before initiating request. `float`
http_req_looking_up	Trend	Time spent looking up remote host name in DNS. `float`
http_req_connecting	Trend	Time spent establishing TCP connection to remote host. `float`
http_req_tls_handshaking	Trend	Time spent handshaking TLS session with remote host.
http_req_sending	Trend	Time spent sending data to remote host. `float`
http_req_waiting	Trend	Time spent waiting for response from remote host (a.k.a. "time to first byte", or "TTFB"). `float`
http_req_receiving	Trend	Time spent receiving response data from remote host. `float`
http_req_duration	Trend	Total time for the request. It's equal to `http_req_sending + http_req_waiting + http_req_receiving` (i.e. how long did the remote server take to process the request and respond, without the initial DNS lookup/connection times). `float`

Metrics

Custom Metrics

```
import http from "k6/http";
import { Counter } from "k6/metrics";

let CounterErrors = new Counter("Errors");

export default function() {
    let res = http.get("https://test.loadimpact.com");
    let contentOK = res.html("h2").text().includes("Welcome to the LoadImpact.com demo site!");

    if (!contentOK) {
      CounterErrors.add(1);
    }
};
```

Metrics

Custom Metrics

Metrics

Custom Metrics

— Counter metrics
— Gauge metrics
— Rate metrics
— Trend metrics

Metrics

Counter Metrics

A metric that cumulatively sums added values.

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

var myCounter = new Counter("my_counter");

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

Metrics

Gauge metrics

A metric that stores the last value added to it.

```
import { Gauge } from "k6/metrics";

var myGauge = new Gauge("my_gauge");

export default function() {
    myGauge.add(3);
    myGauge.add(1);
    myGauge.add(2);
};
```

Metrics

Rate Metrics

A metric that tracks the percentage of added values that are non-zero.

```
import { Rate } from "k6/metrics";

var myRate = new Rate("my_rate");

export default function() {
    myRate.add(true);
    myRate.add(false);
    myRate.add(1);
    myRate.add(0);
};
```

Chris A James — @chrisalwinjames

Metrics

Trend metrics

A metric that allows for calculating statistics on the added values (min, max, average and percentiles).

```
import { Trend } from "k6/metrics";

var myTrend = new Trend("my_trend");

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

Checks and Thresholds

Checks are like asserts but differ in that they don't halt execution.

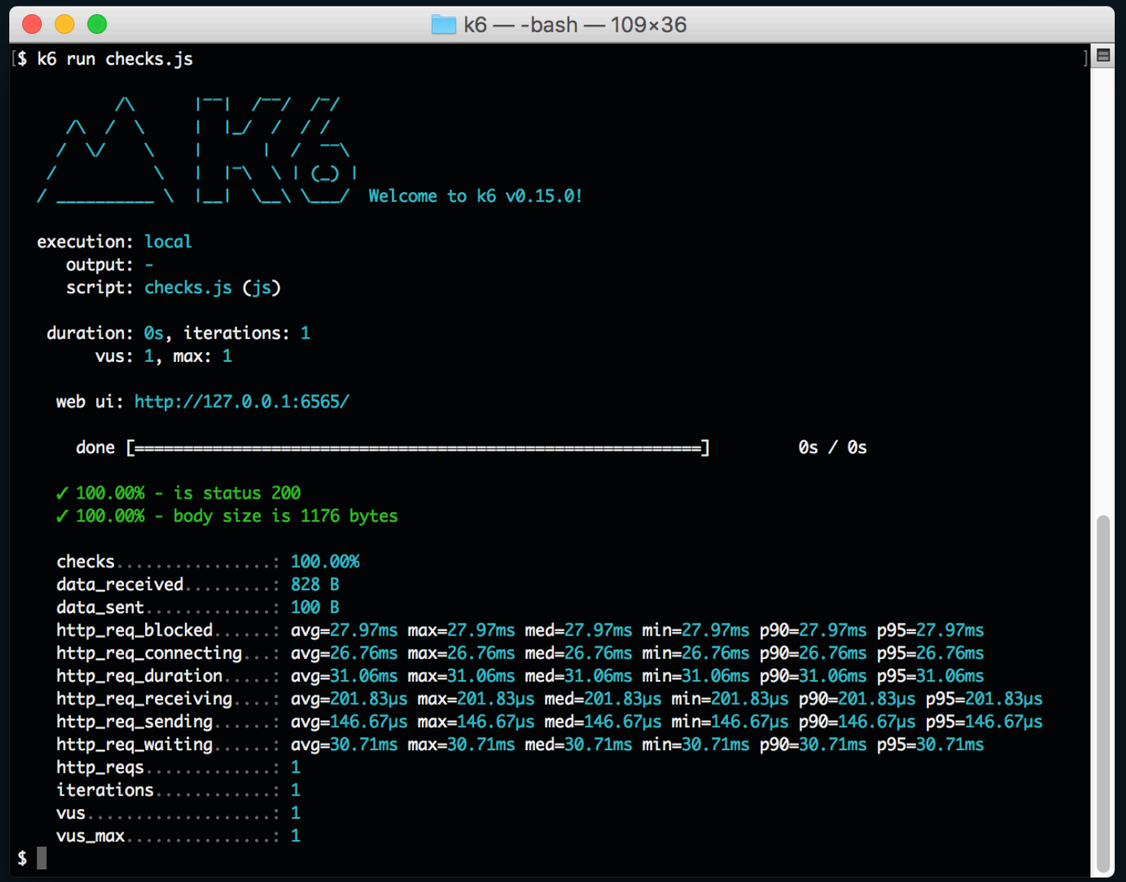
Thresholds are global pass/fail criteria that you can configure k6 to use.

Checks

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

export default function() {
  let res = http.get("http://test.loadimpact.com/");
  check(res, {
    "is status 200": (r) => r.status === 200,
    "body size is 1176 bytes": (r) => r.body.length == 1176
  });
}
```

— One important thing to understand regarding checks is that a failed check will not fail the whole load test.



Thresholds

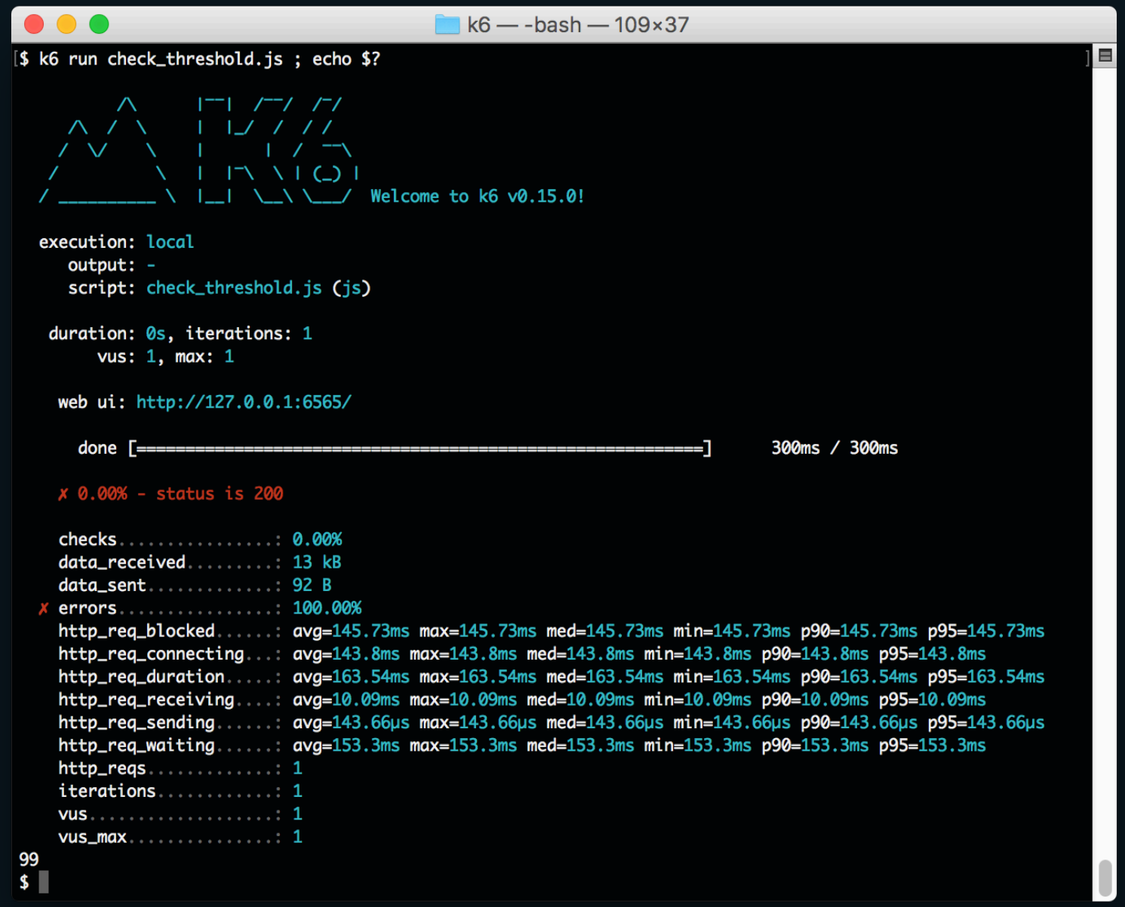
If you want to fail a load test,
use thresholds.

```
import http from "k6/http";
import { check } from "k6";
import { Rate } from "k6/metrics";

export let errorRate = new Rate("errors");

export let options = {
  thresholds: {
    "errors": ["rate<0.1"], // <10% errors
  }
};

export default function() {
  check(http.get("http://fakesite.com"), {
    "status is 200": (r) => r.status == 200
  }) || errorRate.add(1);
};
```



Chris A James — @chrisalwinjames

Anatomy of a Test

Chris A James — @chrisalwinjames

Anatomy of a Test

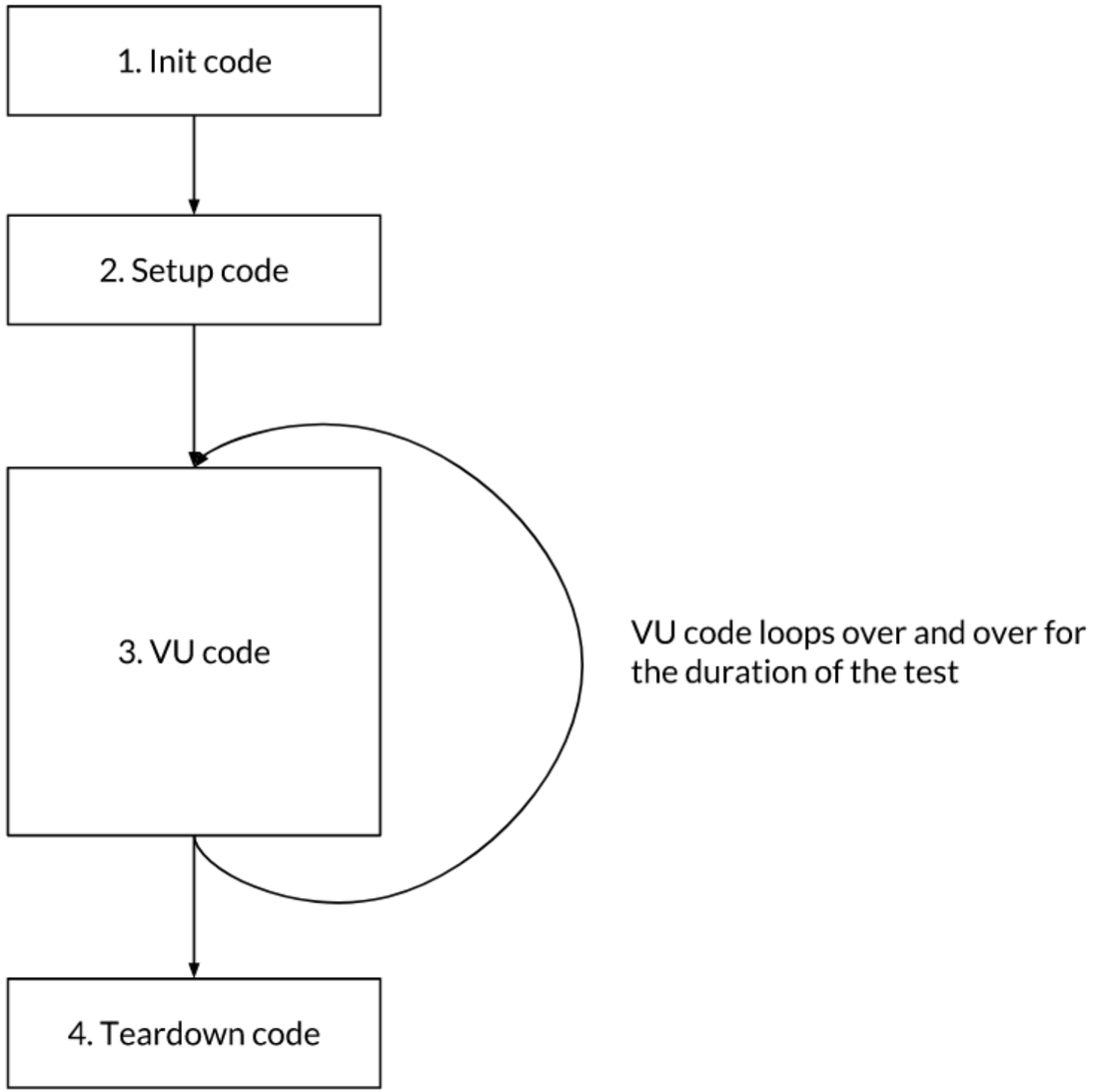
```
// 1. init code, imports and setup counters etc.

// OPTIONAL
export const options = {
  // ...
}

// OPTIONAL
export function setup() {
    // 2. setup code
}

export default function(data) {
    // 3. vu code
}

// OPTIONAL
export function teardown(data) {
    // 4. teardown code
}
```



Writing and Running Tests

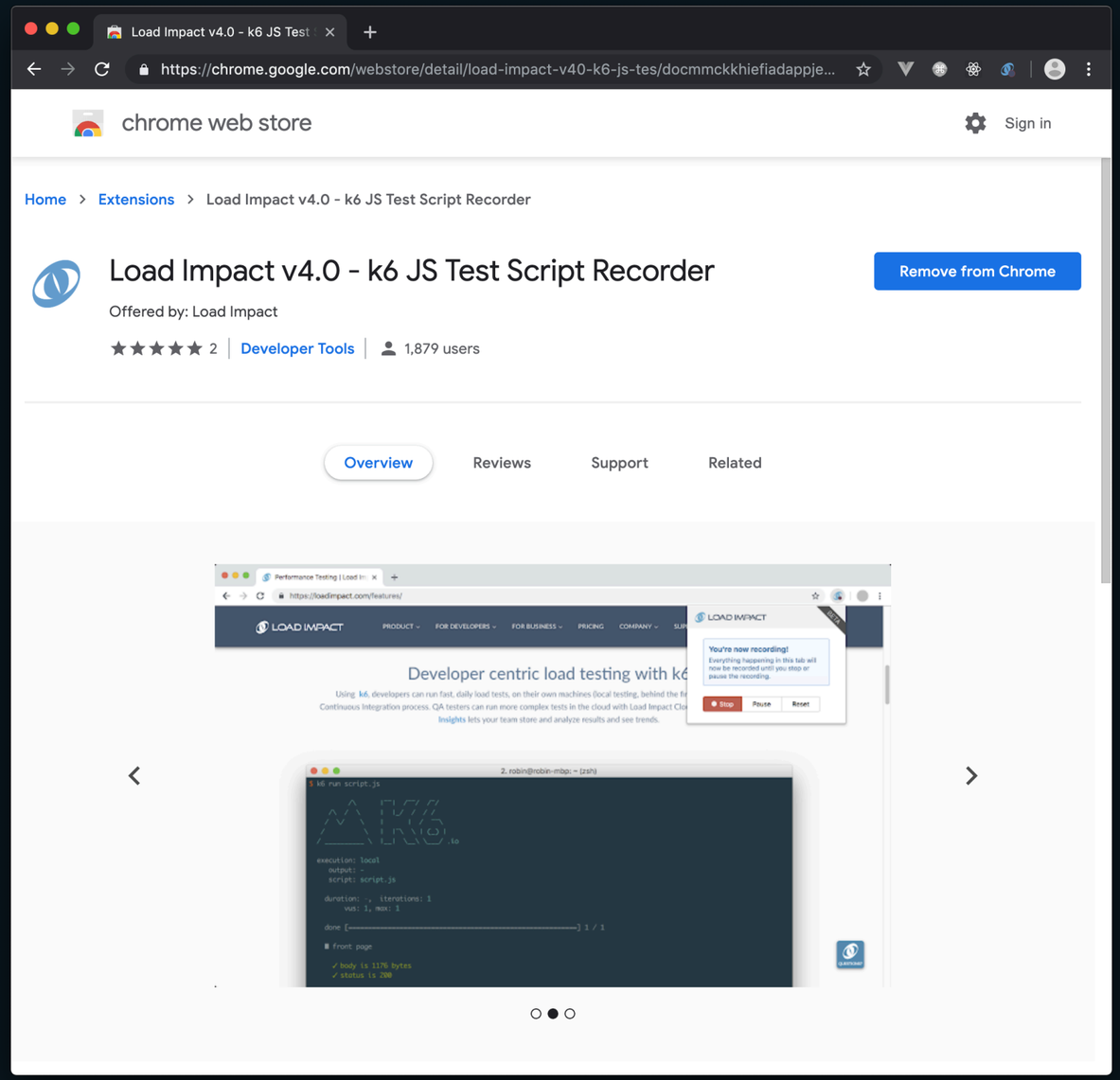
Chris A James — @chrisalwinjames

Demo

Recording Tests

— Writing these tests can get tedious, especially for websites.

— Use the test recorder extension to simulate load and generate a baseline test.



Chris A James — @chrisalwinjames

Demo

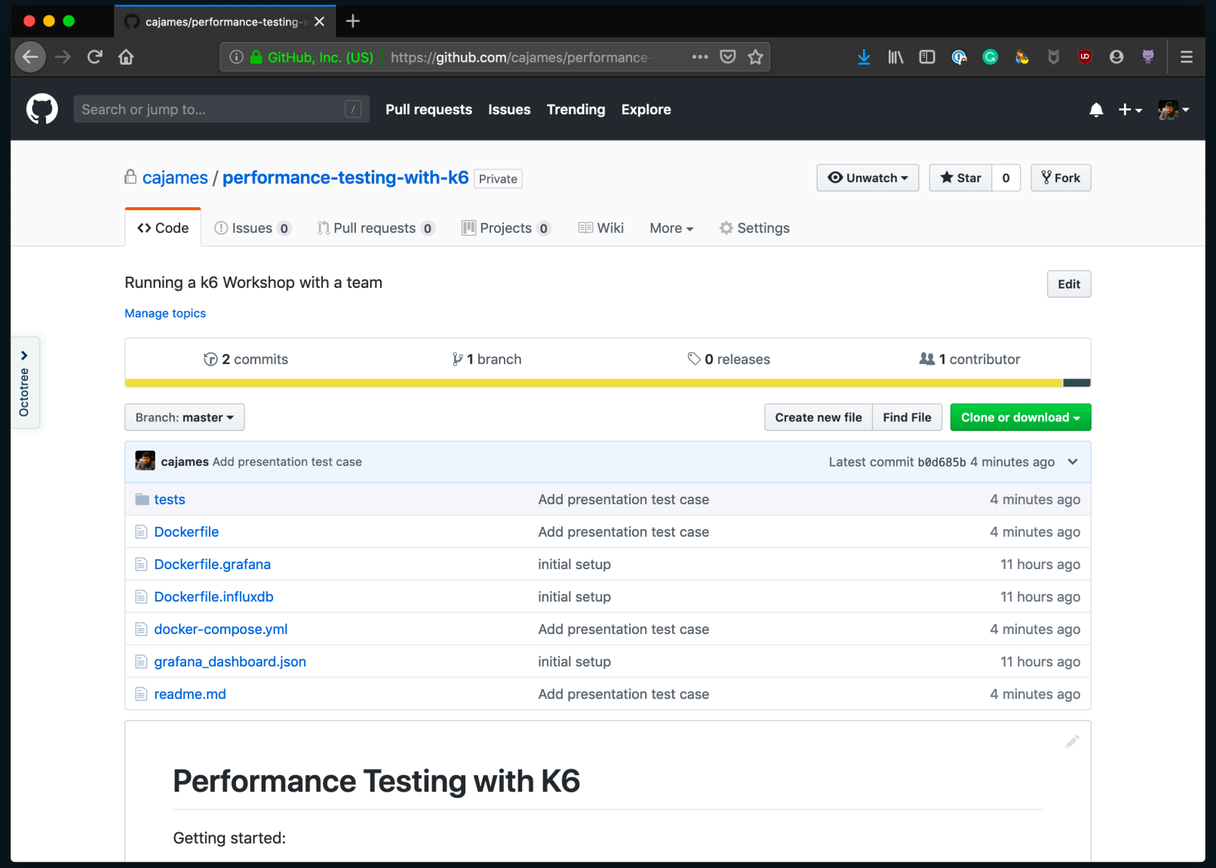
Result Visualisation

Demo

Scaling out your test

Chris A James — @chrisalwinjames

Demo



github.com/cajames/performance-testing-with-k6

Chris A James — @chrisalwinjames

Questions?

Thanks 🎉

Chris A James — @chrisalwinjames