How to use Datadog alerts and Thresholds to fail your load test

Slava Karhaltsev

[Datadog](https://www.datadoghq.com/) is a monitoring and analytics platform that can help you to get full visibility of the performance of your applications. Here at [LoadImpact](https://loadimpact.com/) we use Datadog to monitor various different services of our platform. [Datadog alerts](https://docs.datadoghq.com/monitors/) give the ability to know when critical changes in your system are occurring. These triggered alerts appear in Datadog's [Event Stream](https://docs.datadoghq.com/graphing/event_stream/), allowing collaboration around active issues in your applications or infrastructure.

One potential performance issue is that a System Under Test(SUT) has high CPU consumption when under stress. This tutorial will show you how to fail your load test for this type of condition by using [Datadog's API](https://docs.datadoghq.com/api) and [thresholds](https://docs.k6.io/docs/thresholds) in LoadImpact.

The roadmap of this tutorial is going to follow the outline below:

* [Requirements](https://k6.io/blog/datadog-alerts-fail-load-test#requirements)
* [Create a Monitor in Datadog](https://k6.io/blog/datadog-alerts-fail-load-test#create-a-monitor-in-datadog)
* [Write your performance test](https://k6.io/blog/datadog-alerts-fail-load-test#write-your-performance-test)
* [Running k6 test](https://k6.io/blog/datadog-alerts-fail-load-test#running-k6-test)

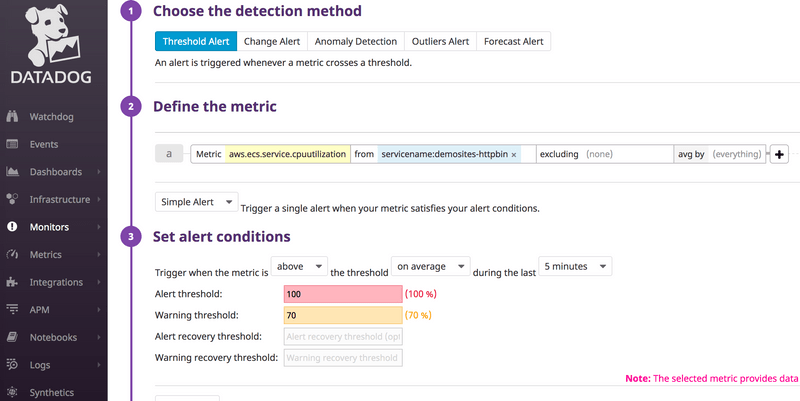
Requirements

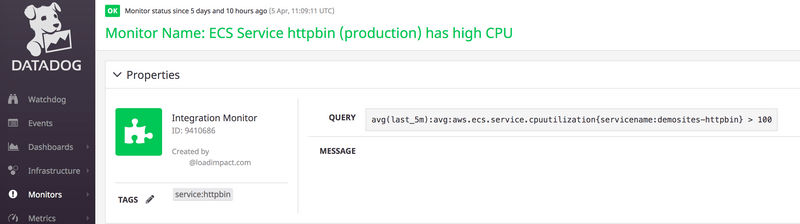
* A site/system to test. In this example, we will test a site already running as a ECS Service. This site is available at [https://httpbin.test.loadimpact.com](https://httpbin.test.loadimpact.com/).
* An already configured Datadog integration with a platform your site is running on. In our case it is Datadog integration with AWS, please refer to the official [Datadog AWS Integration Guide](https://docs.datadoghq.com/integrations/amazon_web_services) for details.
* k6 v0.25.0 (or above) is installed. If you do not have this installed, please refer to the official [k6 installation page](https://docs.k6.io/docs/installation%22). You can verify your current k6 version by command k6 version.
* An account in Datadog that allows us to create [monitors](https://docs.datadoghq.com/monitors).

Create a Monitor in Datadog

First, we want to create a monitor in Datadog which triggers an alert if CPU utilization reaches 100 units or more on the ECS Service. You may wish to monitor something else, so feel free to adjust this to meet your needs. While creating a monitor make next actions:

* Choose Threshold alert as a detection method
* Choose aws.ecs.service.cpuutilization metric from servicename:<your\_service\_name> in "Define the metric" step
* Configure "Alert threshold" to be 100
* Edit message and notification steps and save Monitor

[](https://k6.io/blog/static/a7214ba59d7bd03eadef2a2dc91932d8/0c09f/datadog_pull_2.png)

[](https://k6.io/blog/static/d4569db63886489264136e003ca9d914/1076c/datadog_pull_1.png)

Now the monitor will appear in the Datadog Event Stream if the metric threshold is reached. This is what we will look for when we evaluate the LoadImpact Thresholds later.

Write your performance test

Next, we will need a test script to run. Here is our example that we will use in this test:

// performance-test.js

import http from 'k6/http';

import { Counter } from 'k6/metrics';

import { check, group, sleep } from 'k6';

export const datadogHttpbinCpu = new Counter('Httpbin\_CPU\_Alert');

export let options = {

stages: [

{ duration: '30s', target: 150 },

{ duration: '600s', target: 150 },

],

thresholds: {

http\_req\_duration: ['p(95)<200'],

Httpbin\_CPU\_Alert: ['count < 1'],

},

};

const datadogApi = 'https://api.datadoghq.com/api/v1/'; // DataDogs API endpoint

const datadogApiKey = '<YOUR\_DATADOG\_API\_KEY>'; // DataDogs API key, read below how to get it

const datadogAppKey = '<YOUR\_DATADOG\_APPLICATION\_KEY>'; // DataDogs application key, read below how to get it

const getDataDogHeader = (tagName) => {

return {

headers: { ['Content-Type']: 'application/x-www-form-urlencoded' },

tags: { name: tagName },

};

};

export function setup() {

// function for getting start time of test, executed before actual load testing

let time = Date.now();

return time;

}

export default function () {

let res = http.get('https://httpbin.test.loadimpact.com/');

check(res, {

'is status 200': (r) => r.status === 200,

});

sleep(1);

}

export function teardown(time) {

// function which queries DataDogs event stream for alerts in time window of test run, executed after actual load testing

let endTime = Math.floor(Date.now() / 1000);

let startTime = Math.floor(time / 1000);

let monitorTags = ['servicename:demosites-httpbin'];

let reqString =

`events?api\_key=` +

datadogApiKey +

`&application\_key=` +

datadogAppKey +

'&start=' +

startTime +

'&end=' +

endTime +

'&tags=';

monitorTags.forEach((tag) => {

let response = http.get(datadogApi + reqString + tag, getDataDogHeader('Datadog Event Stream'));

let body = JSON.parse(response.body);

body.events.forEach((event) => {

if (event.tags.includes('servicename:demosites-httpbin')) {

datadogHttpbinCpu.add(true);

}

});

});

}

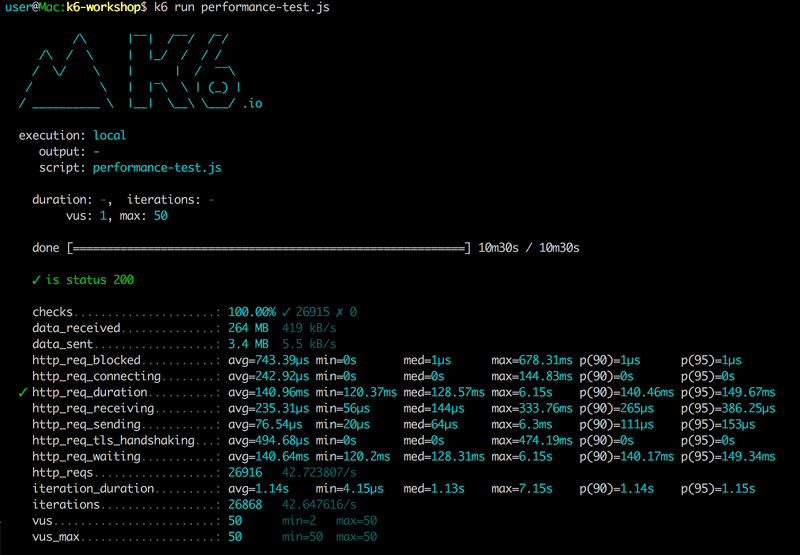
COPY

You can find how to manage your Datadog API and Application keys [here](https://docs.datadoghq.com/account_management/faq/api-app-key-management/).

Running k6 test

If you have installed k6 in your local machine, you could run your test locally in your terminal using the command: k6 run performance-test.js

Run your test and in our case since we configured our script to run only 50 VUs, that load will be not enough to trigger a CPU alert, therefore 1st run will be passed:

[](https://k6.io/blog/static/5e296b670a59963916c7627154d1f90b/9b427/datadog_pull_3.png)

Let's update our script to produce more load on our system under test. For this example, we achieve this by increasing number of VUs from 50 to 150:

export let options = {

stages: [

{ duration: '30s', target: 150 }, // increasing number of VUs from 50 to 150

{ duration: '600s', target: 150 }, // increasing number of VUs from 50 to 150

],

thresholds: {

http\_req\_duration: ['p(95)<200'],

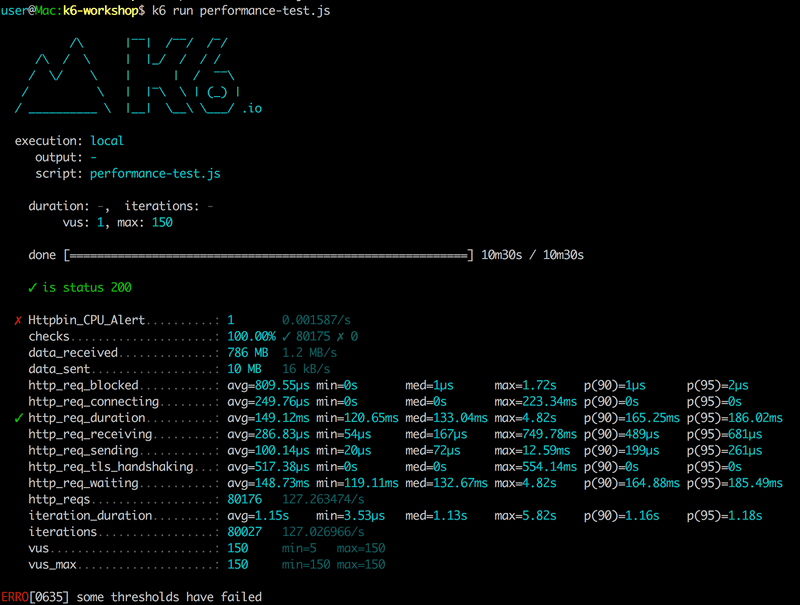
Httpbin\_CPU\_Alert: ['count < 1'],

},

};

COPY

As we can see, after increasing load our test was failed due to exceeding our defined threshold value:

[](https://k6.io/blog/static/078665e9d8f9a4fb2e5f8fac60d34ad0/089e1/datadog_pull_4.png)

See also

* [Datadog](https://www.datadoghq.com/)
* [Datadog Alerts](https://docs.datadoghq.com/monitors/)
* [Datadog Event Stream](https://docs.datadoghq.com/graphing/event_stream/)
* [Datadog API](https://docs.datadoghq.com/api)
* [Thresholds](https://docs.k6.io/docs/thresholds)

<https://docs.datadoghq.com/api/v1/monitors/>