Setup Performance Testing of an Api, using Artillery and Faker

Craig Godden-Payne
Feb 7 · 3 min read ★

Here were are going to do some performance testing on a lambda, behind an api gateway to just prove that it scales well, and my initial thought was to turn to

something I've used before, such as Gatling, JMeter or Siege. I decided to try out artillery.io, and was very impressed!!

Artillery

According to their site:

Artillery is a modern, powerful & easy-to-use load testing toolkit. Use it to ship scalable applications that stay performant & resilient under high load

Its a library, in node, which is extremely easy to setup and use. The documentation isn't the best in terms of examples, but I managed to soldier on.

What I was testing was an api, which I needed to post a block of json. Since I wanted the payload to be dynamic (each payload needed to have a different email address), I used Faker, which is another node library which will generate random, but realistic style data for a variety of data fields such as emails, phone numbers, addresses, dates, colours images etc. It seems to be able to do pretty much anything!

Setup

First you need to install faker and artillery from npm

```
npm i artillery faker
```

Then you will need to create your artillery config file. Here is the one I used:

- fullapplication-test.yml

```
config:
  target: "https://url-changed.com"
  processor: "./random-fullapplication-payload.js"
  phases:
    - duration: 120
      arrivalRate: 5
scenarios:
  - name: "FullApplication Requests"
    flow:
      - function: "generateRandomPayload"
      - post:
          url: "/fullapplication"
          json: "{{payload}}"
```

This sends, for 2 minutes, 5 requests per second to https://url-changed.com/fullapplication using a post and the payload that I generate in the function “generateRandomPayload”.

- random-fullapplication-payload.js

```
'use strict';
var Faker = require('faker');
module.exports = {
  generateRandomPayload
};

function generateRandomPayload(userContext, events, done) {
  var payload = {
    "Data":"hasbeenremoved",
    "Email":"email"
  };

  payload.Email =  Faker.internet.exampleEmail();
  userContext.vars.payload = payload;
  return done();
}
```

When the function is called, the payload is generated based on my template, and I swap out the email, based on the value generated from Faker.

Runnning and reporting

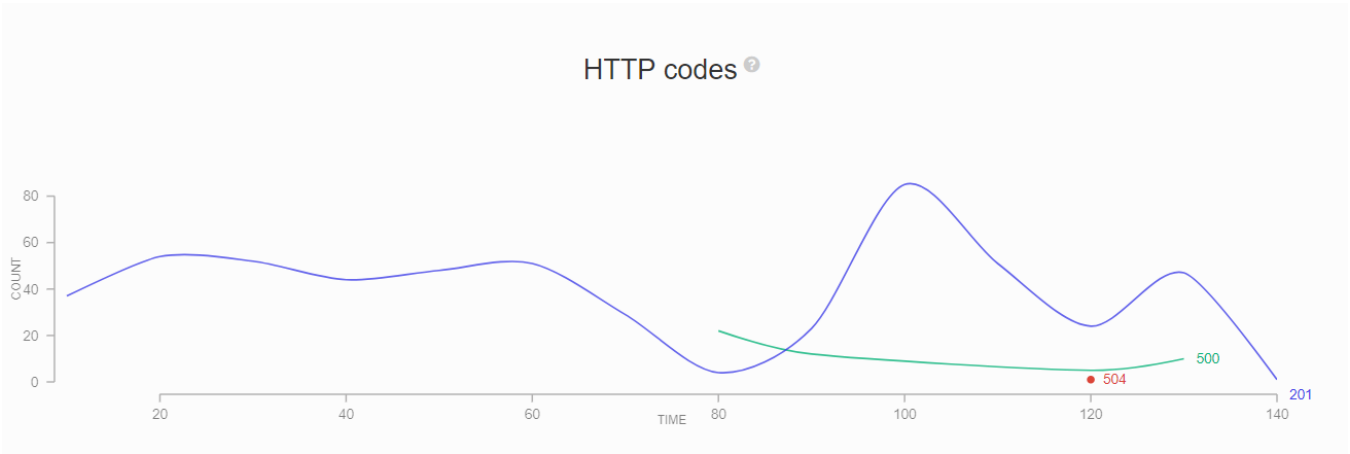
To run the test, and generate the report i ran the following command

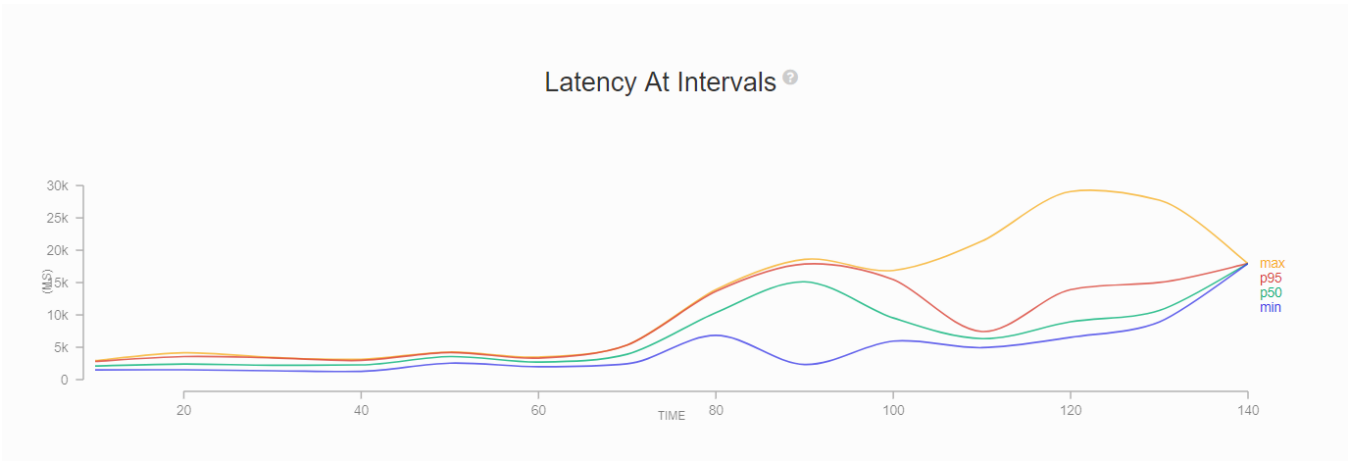
```
artillery run -o report.json fullapplication-test.yaml
artillery report report.json
```

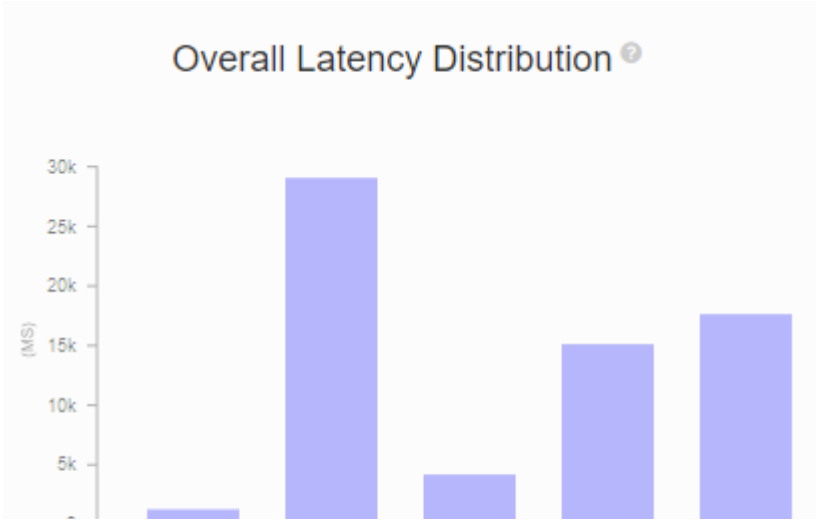
The html outputted report contains a really good breakdown of how the test performed. I was pretty impressed to get so much out of it, for only an hour or so’s work!!

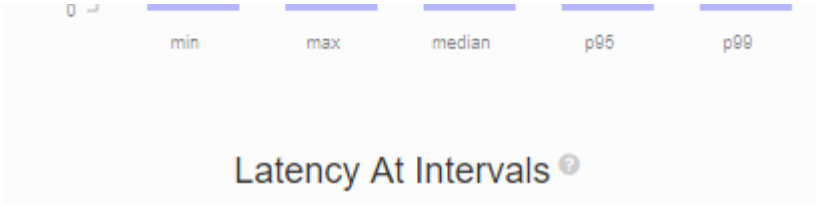
```
Summary report @ 12:13:28(+0100) 2018-06-21
 Scenarios launched:  600
 Scenarios completed: 600
 Requests completed:  600
 RPS sent: 4.42
 Request latency:
```

```
    min: 1269.7
    max: 29066
    median: 4183
    p95: 15118.6
    p99: 17637.6
  Scenario counts:
    FullApplication Requests: 600 (100%)
  Codes:
    201: 550
    500: 49
    504: 1
```









Written on June 21, 2018.

Originally published on https://craig.goddenpayne.co.uk/performance-testing-with-artillery/

Nodejs Performance Testing