Chapter 5

Lab 5.4

Objective: Run load testing on the Dromedary application using Artillery, learn different methods it can be done, and assess which methods are more viable for different situations

Prerequisites

You must first complete Labs 5.1, 5.2, and 5.3 in Chapter 5.

Installing Artillery

Install Artillery using **npm**:

```
$ npm install -g artillery
```

Check that Artillery has been properly installed:

```
$ artillery dino
```

Quick Testing with Artillery

Artillery has a “quick test” mode, often used for ad-hoc testing. It is quick and easy, and it requires only a single command to execute (a change to the port Dromedary is running on):

```
$ artillery quick --duration 60 --rate 20 http://localhost:<port_in_use>
```

This command executes HTTP requests at a rate of 20 requests per second, over a 60-second period. Feel free to practice different variations of this command on the running Dromedary instance.

Using Scripts to Load Tests with Artillery

There are multiple file types that can be used as scripts to execute load tests. One of these types is **`JSON`**; an example **`script.json`** is provided below (change "**`port_in_use`**" to the port Dromedary is running on):

```
{
  "config": {
    "target": "http://localhost:<port_in_use>",
```



```
    "phases": [
      {"duration": 10, "arrivalRate": 5, "name": "Low load phase"},
      {"pause": 5, "name": "Waiting..."},
      {"duration": 10, "arrivalRate": 50, "name": "High load phase"}
    ]
  },
  "scenarios": [
    {
      "flow": [
        { "get":
          {
            "url": "/increment?color=black"
          }
        }
      ]
    }
  ]
}
```

This specific script runs three separate tests or "phases". Note the low load, a waiting period, and a heavy load section. The "**scenarios**" and "**flow**" sections control what is actually being executed. In this case, a **get** request is hitting a **url** that simulates clicking on the black color.

Running the Script Load Test

1. Copy the above-mentioned code into a file titled **script.json**.

2. Start the Dromedary application again, and run **PORT=<port_in_use> gulp** inside of the **dromedary** directory. If the application does not start, make sure that **npm install** runs inside of the **dromedary** directory.

3. Make sure that the **<port_in_use>** has been set up with port forwarding on the virtual machine so that it can be accessed at **http://localhost:<port_in_use>**.

4. Leave the application running and open up a new terminal window in the same environment. If using Vagrant, go to the directory with the **Vagrantfile** and run **vagrant ssh**.

5. From the new terminal window, run **artillery run script.json**. This will start the load test, simulating clicks on the black color. There will be a noticeable amount of activity happening on the terminal where Dromedary is running. Navigate to **http://localhost:<port_in_use>** to see how the value for black increases as the test runs.

6. On the terminal where the test ran, there will be output describing what has taken place. For a more detailed description, there will also be a log file provided right after **artillery run script.json**

has been entered. This log will be populated with more information than the standard output that Artillery provides.

