Introduction

143

In this part of the project we will learn to programmatically manage AWS resources and explore a realistic deployment scenario with regards to infrastructure provisioning and load-balancing.

The Scenario

A company offers an online photo verification service and is planning to use Amazon EC2 as their user base has grown tremendously and they would like to scale accordingly. The website offers a service wherein users can upload a file and the service will verify it for them.

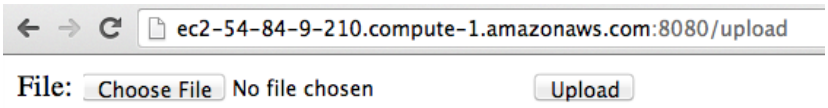
From the past experiences of the company's own internal load testing, they found the most time consuming portion of the website which overloads the server, is the actual file upload process. During peak loads, they find that their servers cannot handle all of the requests that come their way, resulting in loss of customers and potential revenue. They would like to be able to process 1000+ requests per second during peak load.

Your task will be to provision the web server for the company and perform load testing on the server. We will first explore the various loads that different instance types can provide for this application. Following this, we will try to distribute the load among multiple back-end server using an elastic load balancer. You will then arrive at the most scalable solution at an optimized cost for the company.

Click through to the next page when you are ready.

Web Servers and Load Testing Procedures

A Beta version of the web server that is being used by the company has already been packaged into an AMI with ID:`ami-69e3d500`. You can try spinning up a `t1.micro` instance with that AMI and configuring the security groups to allow ports 80 and 8080 and visit: `http://<ec2-dns-name>:8080/upload`, to verify that everything works OK. You should be able to upload a file on that page and get back an md5 hash of the file you just uploaded, as shown in the figures below:.



Before Upload

After Upload

SSH to the instance and visit the **$HOME/benchmark** folder, you should find **apache_bench.sh**, a custom load-testing script written to test the webserver. You can try running it locally using the following command:

```
./apache_bench.sh sample.jpg 100000 100 localhost logfile
```

This command will launch apache benchmark (**ab**) with 100,000 requests, running 100 at a time to **localhost** and store logging data in **logfile.csv** and **logfile.tsv**. The apache benchmark stores raw response time information in the log files and reports some aggregated metrics to **stdout**. The most important numbers to note here are the Requests/sec and the average response time (in milliseconds). **ab** will also bin the results to show what percentage of results responded in what amount of time..

For more information on apache benchmark, try **man ab** or visit the project homepage.