Developer Operations
Assignment 2, March-April 2019

Objective

The objective of the second assignment is to demonstrate the deployment and automated management of a load-balanced auto-scaling web application.

Core assignment specification

Your demonstration should include the following:

1. Creation and configuration of a "master" instance of a web application. You may choose any web application, ideally one that relies on a third party or backend service. Note: any backend services/databases chosen should require minimal resources, e.g. use of micro instances and small amounts of storage.
2. Creation of a custom AMI based on your master instance, to be used by EC2 auto-scaling.
3. Creation of a VPC with public and private subnets into which your application will be deployed. Creation of suitable security groups.
4. Creation of a launch configuration based on that custom AMI.
5. Creation of an elastic load balancer.
6. Creation of an auto-scaling group based on your load balancer and launch configuration; making changes to this auto-scaling group.
7. Creation of an auto-scaling policy; making changes to this policy. You need only scale the front end server instances.
8. Using CloudWatch to trigger an increase in resources based on an alarm – based on a built-in metric such as CPU utilisation, network traffic, etc. You must support your choice of metric in your report.
9. Generation of test traffic to the load balancer – e.g. using curl/wget or a web testing tool.
10. Show that the load is distributed across more than one web server – e.g. by viewing web server or other logs. Screenshots and a brief explanation in your report will suffice for this.
11. Use of your own script to monitor some activity on your server. For example this could be web server or other server logs, or OS activity (CPU, memory, disk, number of processes, etc).

Additional functionality (at least one)

The above is the core assignment specification. In addition you are expected to explore one or more other tasks. The following are some examples of additional tasks:

- Use AWS Lambda functions in your architecture solution.
- Use one or more security services.
- Automate the basic specification, or part of it, using Python/boto.
- Deploy a database on a separate instance to your app.
- Deploy a DynamoDB database
- Capture your configuration using your own customised Cloud Formation script
- Scaling based on SQS

Deliverables

- **Report** with two major sections as follows
  A. Description of implementation
     - For steps 1-6, screenshots will suffice instead of a detailed explanation of each step. However you should introduce the document with your own customised architecture diagram and an explanation of this in your own words. You can create this diagram using *draw.io*. This link includes AWS icons: https://www.draw.io/?splash=0&libs=aws3
     - Provide more specific details on what you did for steps 7-11, including screenshots and discussion as appropriate.
  B. Analysis of the architecture, using the following categories of the AWS *Well-Architected Framework* (https://aws.amazon.com/architecture/well-architected/)
     - Security
     - Reliability
     - Performance Efficiency
     - Cost Optimization
     - Operational Excellence

- Completion of Excel "self-assessment" template (second worksheet tab in Excel file provided on Moodle).

- Any associated **scripts**.

- **Demonstration** in class in final week beginning April 29th. Schedule to be published later.

Marking scheme:

- 50% Core functionality – as specified
- 10% Additional functionality – to your own specification (at least one "extra")
- 30% Analysis according to five categories listed above
- 10% Document and presentation quality

Upload format

ZIP archive containing report in PDF format, completed Excel template and any associated scripts.

Deadline

Sunday April 28th (11:55 pm).