Deploy a high-availability web app using CloudFormation

6 minutes remaining

As your final project, you'll be faced with a real scenario.

Creating this project will give you the hands-on experience you need to confidently talk about infrastructure as code. We have chosen a realistic scenario where you will deploy a dummy application (a sample JavaScript or HTML file) to the Apache Web Server running on an EC2 instance.

There will be two parts to this project:

* **Diagram**: You'll first develop a diagram that you can present as part of your portfolio and as a visual aid to understand the CloudFormation script.
* **Script (Template and Parameters)**: The second part is to interpret the instructions and create a matching CloudFormation script.

**Starter Code**

You will find starter code for the project in this [Github repository](https://github.com/udacity/nd9991-c2-Infrastructure-as-Code-v1). The same repo has the resources used in the lesson demos/exercises for a quick reference.

### Deploy a high-availability web app using CloudFormation

6 minutes remaining

## Scenario

Your company is creating an Instagram clone called Udagram.

Developers want to deploy a new application to the AWS infrastructure.

You have been tasked with provisioning the required infrastructure and deploying a **dummy application**, along with the necessary supporting software.

This needs to be automated so that the infrastructure can be discarded as soon as the testing team finishes their tests and gathers their results.

**Optional** - To add more challenge to the project, once the project is completed, you can try deploying sample website files located in a public S3 Bucket to the Apache Web Server running on an EC2 instance. Though, it is not the part of the project rubric.

## Server specs

1. You'll need to create a **Launch Configuration** for your application servers in order to deploy four servers, two located in each of your private subnets. The launch configuration will be used by an auto-scaling group.
2. You'll need two vCPUs and at least 4GB of RAM. The Operating System to be used is Ubuntu 18. So, choose an Instance size and Machine Image (AMI) that best fits this spec.
3. Be sure to allocate at least 10GB of disk space so that you don't run into issues.

## Security Groups and Roles

1. Since you will be downloading the application archive from an **S3 Bucket**, you'll need to create an **IAM Role** that allows your instances to use the S3 Service.
2. Udagram communicates on the default HTTP Port: 80, so your servers will need this inbound port open since you will use it with the **Load Balancer** and the **Load Balancer Health Check**. As for outbound, the servers will need unrestricted internet access to be able to download and update their software.
3. The load balancer should allow all public traffic (0.0.0.0/0) on port 80 inbound, which is the default HTTP port. Outbound, it will only be using port 80 to reach the internal servers.
4. The application needs to be deployed into private subnets with a **Load Balancer** located in a public subnet.
5. One of the output exports of the **CloudFormation** script should be the public URL of the **LoadBalancer**. **Bonus points** if you add http:// in front of the load balancer **DNS Name** in the output, for convenience.