**Project Introduction**

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. So, for that reason, we have chosen a realistic scenario where you deploy an application (Apache Web Server) and you also pick up code (JavaScript and HTML) from S3 Storage and deploy it in the appropriate folder on the web server.

There will be two parts to this project:

* 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.
* The second part is to interpret the instructions as well as your own diagram and create a matching CloudFormation script.

# Problem

Your company is creating an Instagram clone called Udagram. Developers pushed the latest version of their code in a zip file located in a public S3 Bucket.

You have been tasked with deploying the application, along with the necessary supporting software into its matching infrastructure.

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

# Project Requirements

## Server specs

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.

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. Be sure to allocate at least 10GB of disk space so that you don't run into issues.

## Security Groups and Roles

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.

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 its software.

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.

The application needs to be deployed into private subnets with a **Load Balancer** located in a public subnet.

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.

### Starter Code

You will find starter code for the project in the .yaml file in the Resources section of the left sidebar of this page.

# Other Considerations

## Other Considerations

You can deploy your servers with an **SSH Key** into Public subnets while you are creating the script. This helps with troubleshooting. Once done, move them to your private subnets and remove the **SSH Key** from your **Launch Configuration**.

It also helps to test directly, without the load balancer. Once you are confident that your server is behaving correctly, increase the instance count and add the load balancer to your script.

While your instances are in public subnets, you'll also need the SSH port open (port 22) for your access, in case you need to troubleshoot your instances.

Log information for UserData scripts is located in this file: cloud-init-output.log under the folder: /var/log.

You should be able to destroy the entire infrastructure and build it back up without any manual steps required, other than running the **CloudFormation** script.

The provided UserData script should help you install all the required dependencies. Bear in mind that this process takes several minutes to complete. Also, the application takes a few seconds to load. This information is crucial for the settings of your load balancer health check.

It's up to you to decide which values should be parameters and which you will hard-code in your script.

See the provided supporting code for help and more clues.

If you want to go the extra mile, set up a bastion host to allow you to SSH into your private subnet servers. This bastion host would be on a Public Subnet with port 22 open only to your home IP address, and it would need to have the private key that you use to access the other servers.

Last thing: Remember to delete your **CloudFormation** stack when you're done to avoid recurring charges!

# Starter Code

## final-project-starter.yml

Parameters:

# whatever you consider a changing value, put it as a parameter instead of hard-coding it into your script

Resources:

#instace profile contains the IAM Role name(s) that we want to associate to our auto scaling EC2 Servers

ProfileWithRolesForOurApp:

Type: AWS::IAM::InstanceProfile

Properties:

Roles:

- UdacityS3ReadOnlyEC2

#!/bin/bash

apt-get update -y

apt-get install unzip awscli -y

apt-get install apache2 -y

systemctl start apache2.service

cd /var/www/html

aws s3 cp s3://udacity-demo-1/udacity.zip .

unzip -o udacity.zip

Outputs:

#Bonus points for useful outputs!

# Deploy a high-availability web app using CloudFormation

## Project Submission

In this project, you’ll deploy web servers for a highly available web app using CloudFormation. You will write the code that creates and deploys the infrastructure and application for an Instagram-like app from the ground up. You will begin with deploying the networking components, followed by servers, security roles and software. The procedure you follow here will become part of your portfolio of cloud projects. You’ll do it exactly as it’s done on the job - following best practices and scripting as much as possible.

Gather your files and submit them via a zipped folder or a GitHub link.

Be sure you have completed all of the project requirements.

Also, check your work against the rubric [here](https://review.udacity.com/#!/rubrics/2556/view) before submitting!

## Project Submission Checklist

**Before submitting your project, please review and confirm the following items.**

I am confident all rubric items have been met and my project will pass as submitted. (If not, I will discuss with my mentor prior to submitting.)

Project builds correctly without errors and runs.

All required functionality exists and my project behaves as expected per the project's specifications.

**Once you have checked all these items, you are ready to submit!**

# Project Specification

## Deploy a High-Availability Web App using CloudFormation

### The Basics

|  |  |
| --- | --- |
| Criteria | Meets Specifications |
| Parameters | The more the better, but an exaggerated number of parameters can be messy (say, 10 or more). 1 or 0 is definitely lacking. |
| Resources | This is the mandatory section of the script, we are looking for a LoadBalancer, Launch Configuration, AutoScaling group a health check, security groups and a Listener and Target Group. |
| Outputs | This is optional, but it would be nice to have a URL here with the Load Balancer DNS Name and “http” in front of it. |
| Working Test | If the student provides a URL to verify his work is running properly, it will be a page that says “it works! Udagram, Udacity” |

### Load Balancer

|  |  |
| --- | --- |
| Criteria | Meets Specifications |
| Target Group | The auto-scaling group needs to have a property that associates it with a target group. The Load Balancer will have a Listener rule associated with the same target group |
| Health Check and Listener | Port 80 should be used in Security groups, health checks and listeners associated with the load balancer |

### Auto-Scaling

|  |  |
| --- | --- |
| Criteria | Meets Specifications |
| Subnets | Students should be using PRIV-NET (private subnets) for their auto-scaling instances |
| Machine Specs | The machine should have 10 GB or more of disk and should be a t3.small or better. |
| SSH Key | There shouldn’t be a ‘keyname’ property in the launch config |

### Bonus

|  |  |
| --- | --- |
| Criteria | Meets Specifications |
| Output | Any values in the output section are a bonus |
| Bastion Host | Any resource of type AWS:EC2:Instance, optional, but nice to have. |

### Suggestions to Make Your Project Stand Out!

* Students can deploy Windows Servers instead of Linux and use PowerShell scripts to showcase their Windows management skills.
* Students can use AWS Parameter Store to save sensitive data, such as credentials to showcase their attention to security.
* Students can use CloudWatch Alarms and CloudWatch custom metrics to showcase their performance and monitoring skills.