Rackspace Technical Assessment

For Amazon Web Services

Table of contents

Introduction

Thank you for your interest in working at Rackspace – we're looking forward to meeting you! To help drive the conversation and let you demonstrate your technical skills we'd like you to complete a small project. You shouldn't need to spend more than about four hours (possibly less!) on this.

Overview

This small development project is helpful for us in evaluating your DevOps skills. When designing the solution, be sure to practice good architecture (following AWS best practices) and good coding practices. Feel free to create whatever templates, files, scripts, AMI, security groups, etc, that you feel necessary to accomplish the task. Please note that there are two possible approaches below. One involves VPC, Load Balancers, and EC2. The alternative approach leverages API Gateway, Lambda, and DynamoDB. Both present their own unique requirements and challenges. **You are only required to complete one of them.**

Please make sure you deploy your solution into AWS before replying to your recruiter. Also ensure you provide a copy of your Infrastructure as Code to your recruiter as well. Your solution will be evaluated in AWS as well as via your code presented.

You may use the Internet as a reference tool. The only restriction is that you must not contact any other individuals for help. This includes through instant or text messaging, e-mail, phone calls or posts to discussion forums. You may read through previous discussion forum posts, however.

Optional Approaches

Remember, you are only required to complete one of these approaches.

Approach 1

1.1 Objective

Launch a simple web site in a load balanced, highly available and highly resilient manner utilizing automation and AWS best practices.

1.2 Requirements

- VPC with private/public subnets and all required dependent infrastructure matching AWS highly available best practices (DO NOT USE THE DEFAULT VPC)
- ELB to be used to register web server instances
- Include a simple health check to make sure the webservers are responding
- The health check should automatically replace instances if they are unhealthy, and the instances should come back into service on their own

- Auto Scaling Group and Launch Configuration that launches EC2 instances and registers them to the ELB
- Establish a minimum, maximum, and desired server count based on AWS best practices that scales up / down based on a metric of your choice (and be able to demonstrate a scaling event)
- Security Group allowing HTTP traffic to load balancer from anywhere (not directly to the instance(s))
- Security Group allowing only HTTP traffic from the load balancer to the instance(s)
- Remote management ports such as SSH and RDP must not be open to the world
- Some kind of automation or scripting that achieves the following:
    - Install a webserver (your choice – Apache and Nginx are common examples)
    - Deploys a simple “hello world” page for the webserver to serve up
    - Can be written in the computer language of your choice (HTML, PHP, etc)
    - Can be sourced from the location of your choice (S3, cookbook file / template, etc)
    - Must include the server’s hostname in the “hello world” web page presented to the user
- All AWS resources must be created using Terraform or CloudFormation
- No resources may be created or managed by hand / manually other than EC2 SSH keys

Approach 2

2.1 Objective

Launch a simple API utilizing automation and AWS best practices.

2.2 Requirements

- DynamoDB table
    - Must contain multiple unique records (sample structure provided below, you may determine your own schema as desired)
- Lambda Function
    - Must be written in a language supported natively by AWS (Node.js, Java, C#, Go, Python – NO SHIMS)
- Must log execution output to CloudWatch Logs
- API Gateway with two stages (dev and prod) exposing endpoints for the Lambda function
- Must have a path to expose all Dynamo DB records as well as a single record
    - Example:
        - /id should return a list of all records
    - /id/5 should return the details of just record #5 in DynamoDB

{ “id”: “5”, “details” {

“firstName”: “Onica”,

“lastName”: “Test”} }

- CloudWatch Logs
  - Must have a retention policy of 30 days
- IAM Roles and Policies must be created in the least permissive model (Avoid using AWS Managed Policies)
- All AWS resources must be created using Serverless Framework, Terraform, or CloudFormation
- No resources may be created or managed by hand other than EC2 SSH keys