PROJECT-1:

----------

Approach 1

1.1 OBJECTIVE

Launch a simple web site in a load balanced and highly available manner utilizing automation

and AWS bestpractices

1.2 REQUIREMENTS

• VPC withprivate/public subnets andall requireddependentinfrastructure (DO NOT USE THE

DEFAULT VPC)

• ELB to be used to register web server instances.

• Include a simple health check to make sure the web servers 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 countthat scales up/downbased 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 web server (your choice – Apache and Nginx are common examples)

• Deploys a simple “hello world” page for the web server to serve up

• Can be written in the 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” presented to the user

• All AWS resources must be created using Terraform or CloudFormation

• No resources may be created or managed by hand other than EC2 SSH keys

PROJECT-2:

----------

1.3 OBJECTIVE

Launch a simple API utilizing automation and AWS best practices.

1.4 REQUIREMENTS

• DynamoDB table

o Must contain multiple unique records (sample structure provided below, you may

determine your own schema as desired)

• Lambda Function

o Must be written in a language supported natively by AWS (Node.js, Java, C#, Go, Python – NO SHIMS)

o Must log execution output to CloudWatch Logs

• API Gateway with two stages (dev and prod) exposing endpoints for the Lambda function

o Must have a path to expose all DynamoDB records as well as a single record

▪ Example:

• /idshould returnalist ofall records

1

2

3

4

5

• /id/5shouldreturnthedetails of record #5inDynamoDB

{

“id”: “5”,

“details” {

“firstName”: “Onica”,

“lastName”: “Test”

}

}

• CloudWatch Logs

o 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)

• AllAWSresourcesmustbecreatedusingServerlessFramework,Terraform,orCloudFormation

• No resources may be created or managed by hand other than EC2 SSH keys