Problem Statement - Create an Autoscaling group using Terraform on AWS. The instances in the group should have Nginx installed on them. Parameters in the ASG can be set at your discretion and should be mentioned in the solution documentation.

Any variables in the script should be placed in a separate variables.tf file. The output of the script should be the DNS of the associated load balancer.

Grading Criteria

1. Documentation explaining the solution architecture and template created - 10 marks
2. Shell script used to install Nginx 5 marks
3. Separate variables file 5 marks
4. Output file showing DNS of load balancer 10 marks
5. Terraform script with given components
   1. Security groups 2 marks
   2. EC2 instances 3 marks
   3. Elastic Load balancer 5 marks
   4. Autoscaling group 5 marks
6. Screenshot showing successful execution of Terraform script 5 marks

# SOLUTION:

-------------------------------------------------------------------------------------------------------------------------------

Document the solution architecture and template created - 10 marks

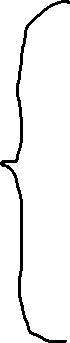
-------------------------------------------------------------------------------------------------------------------------------

Autoscaling environment:

NGINX server is installed and configured in the EC2 instances… using the userdata.sh. So any instance will point to this server. 2 availability zones are set in the infrastructure.

Created VPC, subnet1 , subnet2 , Route Table , and the create entries/associations in the route table for the subnets.

Then created a IAM role for the ASG and then created templates to support the ASG with Amazon Linux 2 and t2.micro instance types. The ASG has 3 desired EC2 and minimum of 2 and maximum of 4. Dynamic Scaling properties of the ASG is such that if the CPU utilization was below 10%, the ASG dynamic policy will remove an EC2 instance and if the CPU Utilization was more than 80% , it will scale up the units of EC2 instance.



ASG



Az2

Az2

Az1

DETAILS:

1. New VPC setup with Route table
2. 2 subnets created with 2 different availability zones (us-east-1a and us-east-1b) with CIDR blocks 10.0.1.0/24 and 10.0.2.0/24 respectively.
3. RT and subnets given associations to access the resources through SSH.
4. ASG created with 2, 3, and 4 as min, desired and max instance in the configuration.
5. A template is created with t2.micro and Amazon Linux 2
6. Load balancer is created with all the instances created through the template.
7. Dynamic scaling is performed by ASG if CPU utilization goes below 10% a instance is scaled down( a new policy)
8. Dynamic scaling is performed by ASG if CPU utilization goes above 80% a instance is added ( a new policy)
9. Outputs the DNS of the associated load balancer and VPC ID(optional, just FYI)

-------------------------------------------------------------------------------------------------------------------------------

Shell script used to install Nginx 5 marks

-------------------------------------------------------------------------------------------------------------------------------

#!/bin/bash

sudo su

sudo yum update -y

sudo amazon-linux-extras install nginx1 -y

sudo systemctl start nginx

sudo systemctl enable nginx

sudo systemctl status nginx

echo "newer NGINX Server v2 " > /usr/share/nginx/html/index.html

------------------------------------------------------------------------------------------------------------------------------

Separate variables file 5 marks

-------------------------------------------------------------------------------------------------------------------------------

variable "ami" {

  type    = string

  default = "ami-006dcf34c09e50022"

}

variable "instance\_type" {

  type        = string

  description = "EC2 instance type for launch template"

  default     = "t2.micro"

}

variable "project" {

  default = "AutoscalingNGINXProject"

}

variable "PATH\_TO\_PUBLIC\_KEY" {

  description = "Public key file to create the key pair"

  default     = "new1.pub"

}

variable "ec2-sg" {

  default = "ssh-nginx-sg"

}

variable "Nginx\_port" {

  type        = string

  description = "Nginx port"

  default     = "80"

}

-------------------------------------------------------------------------------------------------------------------------------

Output file showing DNS of load balancer 10 marks

-------------------------------------------------------------------------------------------------------------------------------

output "vpc\_id" {

  description = "ID of project VPC"

  value       = aws\_vpc.proj-VPC.id

}

output "lb\_url" {

  description = "URL of load balancer"

  value       = aws\_lb.alb.dns\_name

}

-------------------------------------------------------------------------------------------------------------------------------

Terraform script with given components

* 1. Security groups 2 marks

-------------------------------------------------------------------------------------------------------------------------------

# ---------------------------------------------------------------------------------------------------------------------

# CREATE A SECURITY GROUP THAT CONTROLS WHAT TRAFFIC CAN GO IN AND OUT OF THE ALB AND EC2 INSTANCES

# ---------------------------------------------------------------------------------------------------------------------

resource "aws\_security\_group" "ec2-sg" {

  name        = var.ec2-sg

  description = "Autoscaling project instance security group"

  vpc\_id      = aws\_vpc.proj-VPC.id

  ingress {

    from\_port = var.Nginx\_port

    to\_port   = var.Nginx\_port

    protocol  = "tcp"

    # cidr\_blocks = [aws\_default\_vpc.default.cidr\_block]

    # cidr\_blocks = [aws\_vpc.proj-VPC.cidr\_block]

    cidr\_blocks = ["0.0.0.0/0"]

  }

  # ingress {

  #   from\_port   = 443

  #   to\_port     = 443

  #   protocol    = "tcp"

  #   cidr\_blocks = ["0.0.0.0/0"]

  # }

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  # Allow all outbound

  egress {

    from\_port = 0

    to\_port   = 0

    # -1 is semantically equivalent to "all." So all protocols are allowed

    protocol    = "-1"

    cidr\_blocks = ["0.0.0.0/0"]

  }

}

-------------------------------------------------------------------------------------------------------------------------------

Terraform script with given components

* 1. EC2 instances (Launch template) 3 marks

-------------------------------------------------------------------------------------------------------------------------------

# --------------------------------------------------------------------------------------------------------------------

# CREATE LAUNCH TEMPLATE FOR ASG/ EC2

# --------------------------------------------------------------------------------------------------------------------

#Subnet 2 instance resouce

resource "aws\_launch\_template" "template" {

  name          = "web-asg-template-1"

  instance\_type = var.instance\_type

  # image\_id      = "ami-006dcf34c09e50022"

  # image\_id = data.aws\_ami.ami.id

  image\_id = var.ami

  key\_name = aws\_key\_pair.mykeypair.key\_name

  network\_interfaces {

    associate\_public\_ip\_address = **true**

    security\_groups             = [aws\_security\_group.ec2-sg.id]

    device\_index                = "0"

    delete\_on\_termination       = "true"

  }

  iam\_instance\_profile {

    arn = aws\_iam\_instance\_profile.profile.arn

  }

  tag\_specifications {

    resource\_type = "instance"

    tags = {

      Name   = "Name"

      Source = "Autoscaling"

    }

  }

  user\_data = filebase64("userData1.sh")

}

-------------------------------------------------------------------------------------------------------------------------------

Terraform script with given components

* 1. Load balancer 5 marks

-------------------------------------------------------------------------------------------------------------------------------

# ---------------------------------------------------------------------------------------------------------------------

# CREATE AN APPLICATION LB TO ROUTE TRAFFIC ACROSS THE AUTO SCALING GROUP

# 2 AZs SET

# ---------------------------------------------------------------------------------------------------------------------

resource "aws\_lb" "alb" {

  name               = "proj-asg-alb"

  load\_balancer\_type = "application"

  security\_groups    = [aws\_security\_group.ec2-sg.id]

  # availability\_zones = ["${data.aws\_availability\_zones.all.names}"]

  subnets = [aws\_subnet.proj-Subnet1.id, aws\_subnet.proj-Subnet2.id]

  tags = {

    Environment = "${var.project}-lb-sg"

  }

}

-------------------------------------------------------------------------------------------------------------------------------

1. Terraform script with given components
   1. Autoscaling group 5 marks

-------------------------------------------------------------------------------------------------------------------------------

# --------------------------------------------------------------------------------------------------------------------

# CREATE ASG WITH 3,2,4 AS DESIRED, MIN AND MAX RESPECTIVELY

# --------------------------------------------------------------------------------------------------------------------

resource "aws\_autoscaling\_group" "asg" {

  name                 = "web-asg"

  desired\_capacity     = 3

  min\_size             = 2

  max\_size             = 4

  termination\_policies = ["OldestInstance"]

  launch\_template {

    id      = aws\_launch\_template.template.id

    version = "$Latest"

  }

  vpc\_zone\_identifier = [aws\_subnet.proj-Subnet1.id, aws\_subnet.proj-Subnet2.id]

  health\_check\_type         = "EC2"

  health\_check\_grace\_period = 300

  force\_delete              = **true**

  # wait\_for\_capacity\_timeout = "3m"

  tag {

    key                 = "Name"

    value               = "asg-alb"

    propagate\_at\_launch = **true**

  }

  instance\_refresh {

    strategy = "Rolling"

    preferences {

      min\_healthy\_percentage = 85

    }

  }

}

# --------------------------------------------------------------------------------------------------------------------

# CREATE SCALE UP DYNAMIC SCALING POLICY

# --------------------------------------------------------------------------------------------------------------------

resource "aws\_autoscaling\_policy" "scale\_up" {

  name                   = "asg\_policy\_scale\_up"

  scaling\_adjustment     = 1

  adjustment\_type        = "ChangeInCapacity"

  cooldown               = 300

  autoscaling\_group\_name = aws\_autoscaling\_group.asg.name

}

# --------------------------------------------------------------------------------------------------------------------

# CREATE CLOUDWATCH ALARM FOR SCALE UP

# --------------------------------------------------------------------------------------------------------------------

resource "aws\_cloudwatch\_metric\_alarm" "scale\_up" {

  alarm\_description   = "Monitors CPU utilization for the ASG EC2s"

  alarm\_actions       = [aws\_autoscaling\_policy.scale\_up.arn]

  alarm\_name          = "proj\_asg\_scale\_up"

  comparison\_operator = "GreaterThanOrEqualToThreshold"

  namespace           = "AWS/EC2"

  metric\_name         = "CPUUtilization"

  threshold           = "80"

  evaluation\_periods  = "2"

  period              = "120"

  statistic           = "Average"

  dimensions = {

    AutoScalingGroupName = aws\_autoscaling\_group.asg.name

  }

}

# --------------------------------------------------------------------------------------------------------------------

# CREATE SCALE DOWN SCALING POLICY

# --------------------------------------------------------------------------------------------------------------------

resource "aws\_autoscaling\_policy" "scale\_down" {

  name                   = "asg\_policy\_scale\_down"

  autoscaling\_group\_name = aws\_autoscaling\_group.asg.name

  adjustment\_type        = "ChangeInCapacity"

  scaling\_adjustment     = -1

  cooldown               = 200

}

# --------------------------------------------------------------------------------------------------------------------

# CREATE CLOUDWATCH ALARM FOR SCALE DOWN

# --------------------------------------------------------------------------------------------------------------------

resource "aws\_cloudwatch\_metric\_alarm" "scale\_down" {

  alarm\_description   = "Monitors CPU utilization for the ASG"

  alarm\_actions       = [aws\_autoscaling\_policy.scale\_down.arn]

  alarm\_name          = "proj\_asg\_scale\_down"

  comparison\_operator = "LessThanOrEqualToThreshold"

  namespace           = "AWS/EC2"

  metric\_name         = "CPUUtilization"

  threshold           = "10"

  evaluation\_periods  = "6"

  period              = "120"

  statistic           = "Average"

  dimensions = {

    AutoScalingGroupName = aws\_autoscaling\_group.asg.name

  }

}

-------------------------------------------------------------------------------------------------------------------------------

Screenshot showing successful execution of Terraform script 5 marks

-------------------------------------------------------------------------------------------------------------------------------

Text

Description automatically generated