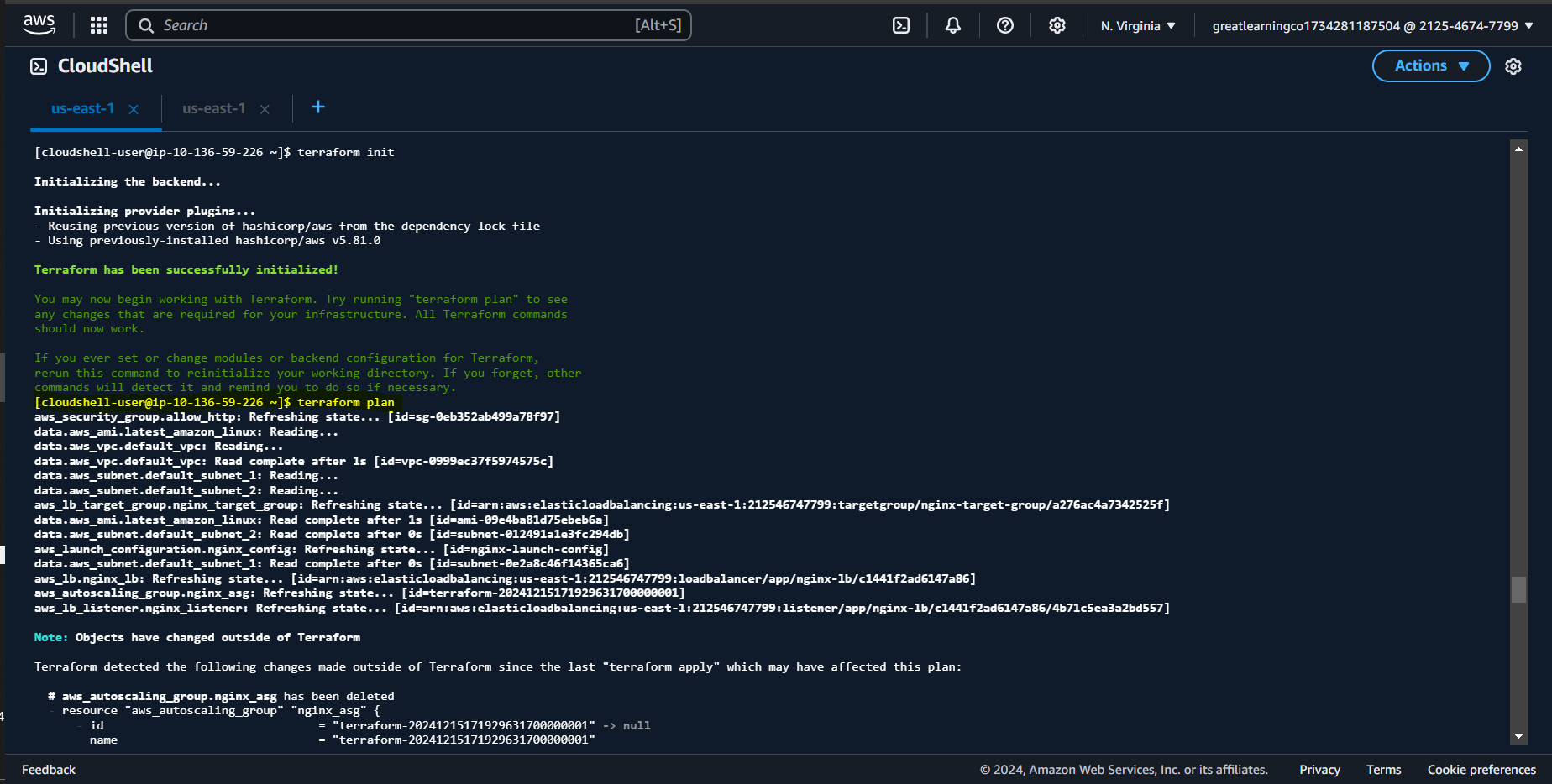
Problem Statement - Create an Autoscaling group using Terraform on AWS. The instances in the group should have Nginx installed on them. Parameters in the ASG can be set at your discretion and should be mentioned in the solution documentation.

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.

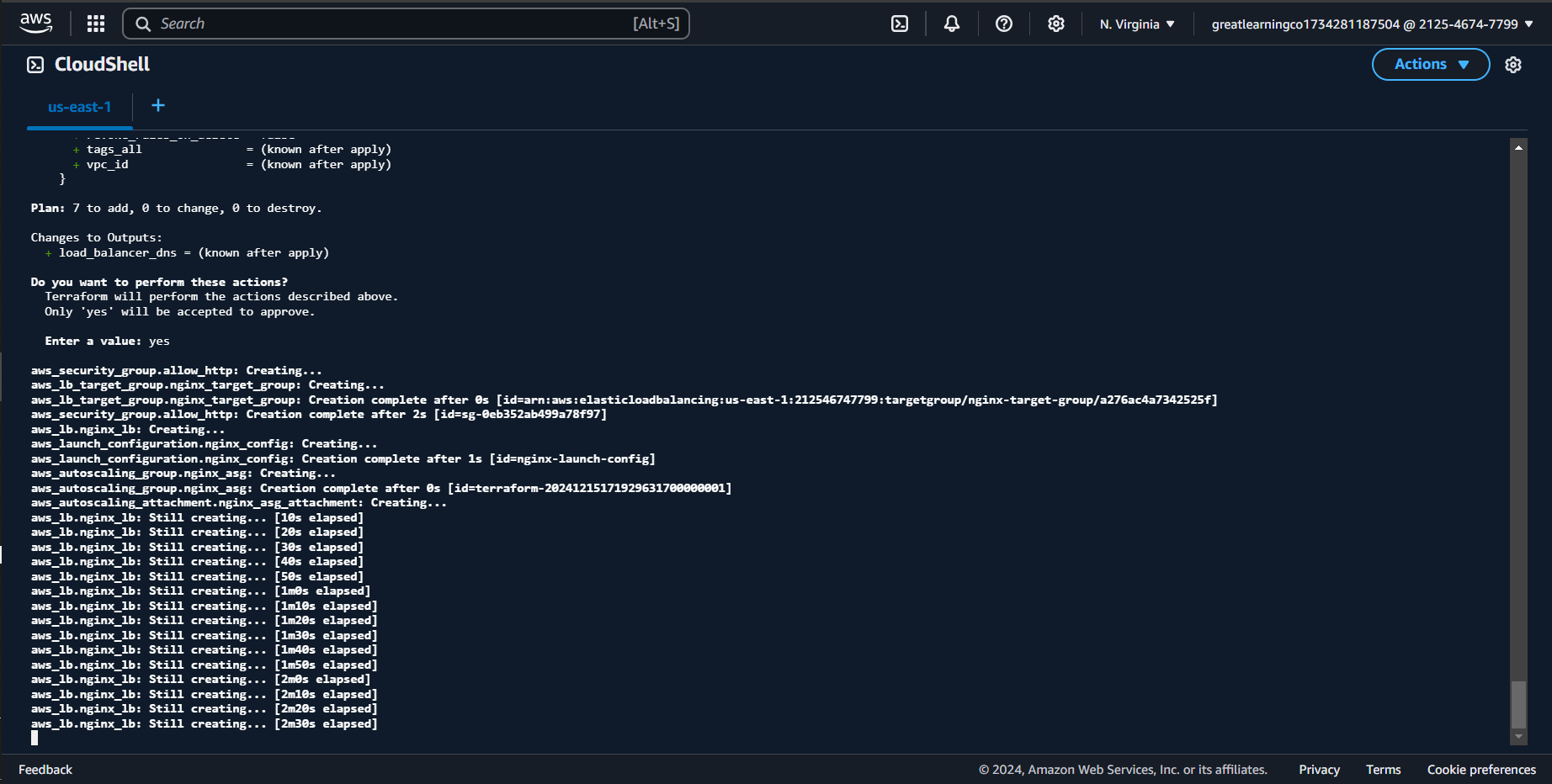
Grading Criteria

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



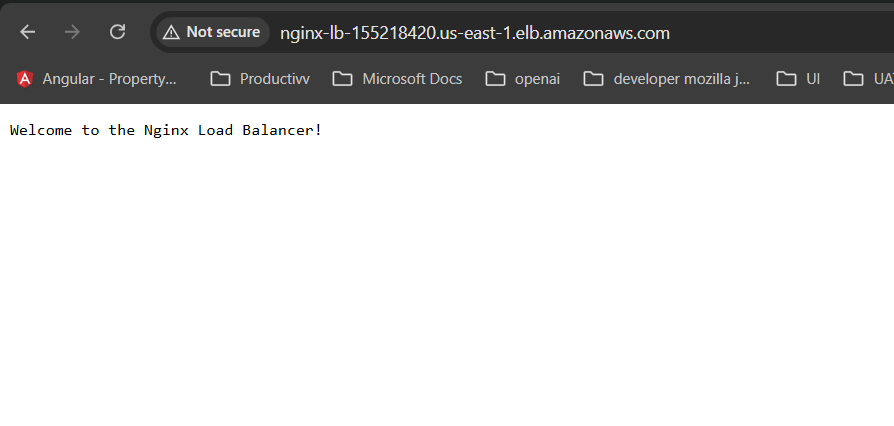
A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



**Steps**

**Step-by-Step Process**

**Step 1: Open AWS CloudShell**

**Step 2: Create a New Directory for Your Terraform Code**

mkdir terraform-nginx

cd terraform-nginx

**Step 3: Create the main.tf File**

**# main.tf**  
  
provider "aws" {

region = var.region # Use the region variable from variables.tf

}

# Fetch the default VPC

data "aws\_vpc" "default\_vpc" {

default = true

}

# Create a default security group

resource "aws\_security\_group" "allow\_http" {

name = "allow\_http"

description = "Allow HTTP and SSH traffic"

ingress {

from\_port = 80

to\_port = 80

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"]

}

}

# Automatically find the latest Amazon Linux 2 AMI (for EC2 instances)

data "aws\_ami" "latest\_amazon\_linux" {

most\_recent = true

owners = ["amazon"]

filter {

name = "name"

values = ["amzn2-ami-hvm-\*-x86\_64-gp2"]

}

}

# Launch Configuration to use the Amazon Linux 2 AMI and install Nginx

resource "aws\_launch\_configuration" "nginx\_config" {

name = "nginx-launch-config"

image\_id = data.aws\_ami.latest\_amazon\_linux.id

instance\_type = var.instance\_type # Use the instance type variable

security\_groups = [aws\_security\_group.allow\_http.id]

user\_data = file("nginx\_install.sh") # Executes the shell script to install Nginx

lifecycle {

create\_before\_destroy = true

}

}

# Automatically find default subnets in the default VPC

data "aws\_subnet" "default\_subnet\_1" {

vpc\_id = data.aws\_vpc.default\_vpc.id

availability\_zone = var.az\_1 # Use AZ variable

}

data "aws\_subnet" "default\_subnet\_2" {

vpc\_id = data.aws\_vpc.default\_vpc.id

availability\_zone = var.az\_2 # Use AZ variable

}

# Create the Auto Scaling Group (ASG)

resource "aws\_autoscaling\_group" "nginx\_asg" {

desired\_capacity = var.desired\_capacity

max\_size = var.max\_size

min\_size = var.min\_size

vpc\_zone\_identifier = [data.aws\_subnet.default\_subnet\_1.id, data.aws\_subnet.default\_subnet\_2.id]

launch\_configuration = aws\_launch\_configuration.nginx\_config.id

health\_check\_type = "EC2"

health\_check\_grace\_period = 300

force\_delete = true

wait\_for\_capacity\_timeout = "0"

}

# Create Elastic Load Balancer (ELB)

resource "aws\_lb" "nginx\_lb" {

name = "nginx-lb"

internal = false

load\_balancer\_type = "application"

security\_groups = [aws\_security\_group.allow\_http.id]

subnets = [data.aws\_subnet.default\_subnet\_1.id, data.aws\_subnet.default\_subnet\_2.id]

enable\_deletion\_protection = false

idle\_timeout = 60

}

# Create a Target Group for the Load Balancer

resource "aws\_lb\_target\_group" "nginx\_target\_group" {

name = "nginx-target-group"

port = 80

protocol = "HTTP"

vpc\_id = data.aws\_vpc.default\_vpc.id

health\_check {

interval = 30

path = "/"

port = 80

protocol = "HTTP"

timeout = 5

healthy\_threshold = 3

unhealthy\_threshold = 3

}

}

# Set up a listener for the Load Balancer to forward traffic

resource "aws\_lb\_listener" "nginx\_listener" {

load\_balancer\_arn = aws\_lb.nginx\_lb.arn

port = 80

protocol = "HTTP"

default\_action {

target\_group\_arn = aws\_lb\_target\_group.nginx\_target\_group.arn

type = "fixed-response"

fixed\_response {

status\_code = 200

content\_type = "text/plain"

message\_body = "Welcome to the Nginx Load Balancer!"

}

}

}

# Attach the Auto Scaling Group to the Load Balancer

resource "aws\_autoscaling\_attachment" "nginx\_asg\_attachment" {

autoscaling\_group\_name = aws\_autoscaling\_group.nginx\_asg.id

lb\_target\_group\_arn = aws\_lb\_target\_group.nginx\_target\_group.arn

depends\_on = [

aws\_lb\_target\_group.nginx\_target\_group # Ensure target group is created first

]

}

# Output the DNS of the Load Balancer

output "load\_balancer\_dns" {

description = "The DNS of the load balancer"

value = aws\_lb.nginx\_lb.dns\_name

}

**Step 4: Create a variables.tf File**

**# variables.tf**

variable "region" {

description = "The AWS region to create resources in"

default = "us-east-1" # Default AWS region

}

variable "instance\_type" {

description = "The EC2 instance type"

default = "t2.micro" # Default EC2 instance type

}

variable "az\_1" {

description = "The first availability zone to deploy resources"

default = "us-east-1a"

}

variable "az\_2" {

description = "The second availability zone to deploy resources"

default = "us-east-1b"

}

variable "desired\_capacity" {

description = "The desired number of instances in the Auto Scaling group"

default = 2

}

variable "max\_size" {

description = "The maximum size of the Auto Scaling group"

default = 3

}

variable "min\_size" {

description = "The minimum size of the Auto Scaling group"

default = 1

}

**Step 5: Create the Nginx Installation Script (nginx\_install.sh)**

**# nginx\_install.sh**

#!/bin/bash

# Install Nginx on Amazon Linux 2

sudo yum update -y

sudo amazon-linux-extras enable nginx1

sudo yum install nginx -y

sudo service nginx start

sudo chkconfig nginx on

**Step 6: Initialize Terraform**

terraform init

**Step 7: Plan the Terraform Deployment**

terraform plan

**Step 8: Apply the Terraform Configuration**

terraform apply

**Step 9: Output the Load Balancer DNS**

terraform output load\_balancer\_dns

**Another Approach with more explanation:**

===============================================================

1. **Autoscaling Group (ASG)**: The ASG automatically scales the number of EC2 instances based on load.
2. **EC2 Instances**: These are the virtual machines where the application will run. In this case, Nginx will be installed on these instances.
3. **Elastic Load Balancer (ELB)**: Distributes incoming traffic to the EC2 instances in the ASG.
4. **Security Group**: Defines the access control rules (e.g., allow HTTP traffic).
5. **Nginx Installation**: We will install Nginx on each EC2 instance via a shell script.

**Prerequisites:**

* You need an AWS account with the necessary permissions to create EC2 instances, security groups, ELB, and autoscaling groups.
* Terraform installed on your local machine (can download from [terraform.io](https://www.terraform.io/)).
* AWS credentials configured on your local machine (aws configure).

**Solution Architecture:**

* We will create a VPC and security group to allow HTTP traffic to the EC2 instances.
* An Elastic Load Balancer (ELB) will distribute traffic to the instances.
* EC2 instances will run a basic shell script to install Nginx and serve a basic web page.
* An Autoscaling Group will ensure that instances are launched and terminated as per load metrics.

**Step-by-Step Guide:**

**1. Define Provider and Setup AWS Credentials**

Create a main.tf file to define the provider and AWS region:

hcl

Copy code

# main.tf

provider "aws" {

region = var.aws\_region

}

**2. Create Variables File**

Create a variables.tf file to manage the variables.

hcl

Copy code

# variables.tf

variable "aws\_region" {

description = "The AWS region to deploy resources"

default = "us-east-1"

}

variable "instance\_type" {

description = "EC2 instance type"

default = "t2.micro"

}

variable "desired\_capacity" {

description = "Desired number of instances in ASG"

default = 2

}

variable "min\_size" {

description = "Minimum number of instances in ASG"

default = 1

}

variable "max\_size" {

description = "Maximum number of instances in ASG"

default = 3

}

variable "ami\_id" {

description = "AMI ID for EC2 instances"

}

**3. Create Security Groups**

Create a security group to allow HTTP traffic to the EC2 instances.

hcl

Copy code

# security\_groups.tf

resource "aws\_security\_group" "allow\_http" {

name = "allow\_http\_sg"

description = "Allow HTTP traffic to EC2 instances"

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

}

**4. Create Launch Configuration for EC2 Instances**

We need a Launch Configuration for the EC2 instances, which includes an instance type and an AMI.

hcl

Copy code

# launch\_configuration.tf

resource "aws\_launch\_configuration" "nginx\_lc" {

name = "nginx-launch-config"

image\_id = var.ami\_id

instance\_type = var.instance\_type

security\_groups = [aws\_security\_group.allow\_http.id]

user\_data = <<-EOT

#!/bin/bash

sudo apt update -y

sudo apt install -y nginx

sudo systemctl start nginx

sudo systemctl enable nginx

echo "<h1>Welcome to Nginx on Terraform!</h1>" | sudo tee /var/www/html/index.html

EOT

}

**5. Create Autoscaling Group**

Now, we’ll create the Autoscaling group that manages the EC2 instances.

hcl

Copy code

# autoscaling\_group.tf

resource "aws\_autoscaling\_group" "nginx\_asg" {

desired\_capacity = var.desired\_capacity

max\_size = var.max\_size

min\_size = var.min\_size

vpc\_zone\_identifier = [aws\_subnet.subnet.id]

launch\_configuration = aws\_launch\_configuration.nginx\_lc.id

target\_group\_arns = [aws\_lb\_target\_group.nginx\_target\_group.arn]

tag {

key = "Name"

value = "NginxInstance"

propagate\_at\_launch = true

}

}

**6. Create Elastic Load Balancer**

To distribute traffic among the EC2 instances, we'll create an Elastic Load Balancer (ELB).

hcl

Copy code

# load\_balancer.tf

resource "aws\_lb" "nginx\_lb" {

name = "nginx-lb"

internal = false

load\_balancer\_type = "application"

security\_groups = [aws\_security\_group.allow\_http.id]

subnets = [aws\_subnet.subnet.id]

enable\_deletion\_protection = false

enable\_cross\_zone\_load\_balancing = true

tags = {

Name = "nginx-lb"

}

}

resource "aws\_lb\_target\_group" "nginx\_target\_group" {

name = "nginx-target-group"

port = 80

protocol = "HTTP"

vpc\_id = aws\_vpc.vpc.id

health\_check {

interval = 30

path = "/"

port = "80"

protocol = "HTTP"

timeout = 5

healthy\_threshold = 3

unhealthy\_threshold = 3

}

}

resource "aws\_lb\_listener" "nginx\_listener" {

load\_balancer\_arn = aws\_lb.nginx\_lb.arn

port = "80"

protocol = "HTTP"

default\_action {

type = "fixed-response"

fixed\_response {

status\_code = 200

content\_type = "text/plain"

message\_body = "Welcome to the Nginx Load Balancer!"

}

}

}

**7. Create Output File**

Define the output that will show the DNS of the load balancer.

hcl

Copy code

# outputs.tf

output "load\_balancer\_dns" {

description = "DNS of the Load Balancer"

value = aws\_lb.nginx\_lb.dns\_name

}

**8. Create VPC and Subnet**

We need a VPC and a subnet for the instances to reside in. Here's a simple configuration:

hcl

Copy code

# vpc.tf

resource "aws\_vpc" "vpc" {

cidr\_block = "10.0.0.0/16"

}

resource "aws\_subnet" "subnet" {

vpc\_id = aws\_vpc.vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-1a"

map\_public\_ip\_on\_launch = true

}

**9. Initialize and Apply Terraform**

1. **Initialize Terraform:**

Run the following command to initialize your Terraform project:

bash

Copy code

terraform init

1. **Plan the Deployment:**

Run the plan to review the resources that Terraform will create:

bash

Copy code

terraform plan -out=tfplan

1. **Apply the Configuration:**

Apply the configuration to create the resources:

bash

Copy code

terraform apply "tfplan"

1. **Output Result:**

After the apply command is successful, Terraform will output the DNS name of the load balancer:

bash

Copy code

terraform output load\_balancer\_dns

**10. Test the Nginx Installation**

Once the Terraform script has successfully executed, navigate to the DNS of your load balancer in the browser. You should see the "Welcome to Nginx on Terraform!" page.

**Additional Points for the Solution:**

* **Shell Script for Nginx**: The script is included in the user\_data section of the launch configuration, which will install and start Nginx.
* **Autoscaling Group**: The ASG will adjust the number of instances between min\_size and max\_size based on load.