CC AWS DevOps Graded Assignment – 3

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.

**Solution :**

Grading Criteria

1. Documentation explaining the solution architecture and template created - 10 marks

In this solution, we will use Terraform to create an Autoscaling Group (ASG) on Amazon Web Services (AWS) and ensure that instances launched within this ASG have Nginx installed. The ASG will automatically adjust the number of instances based on the defined scaling policies. We will also associate the ASG with an Application Load Balancer (ALB) to distribute incoming traffic.

**Prerequisites:**

1. An AWS account with appropriate permissions.
2. Terraform installed on local machine.

**Directory Structure**

We will organize our project files in the following structure:

* **main.tf**: The main Terraform configuration file for creating the ASG and related resources.
* **variables.tf**: File to define input variables.

1. Shell script used to install Nginx 5 marks

#!/bin/bash

            yum update -y

              yum install -y nginx

            service nginx start

            chkconfig nginx on

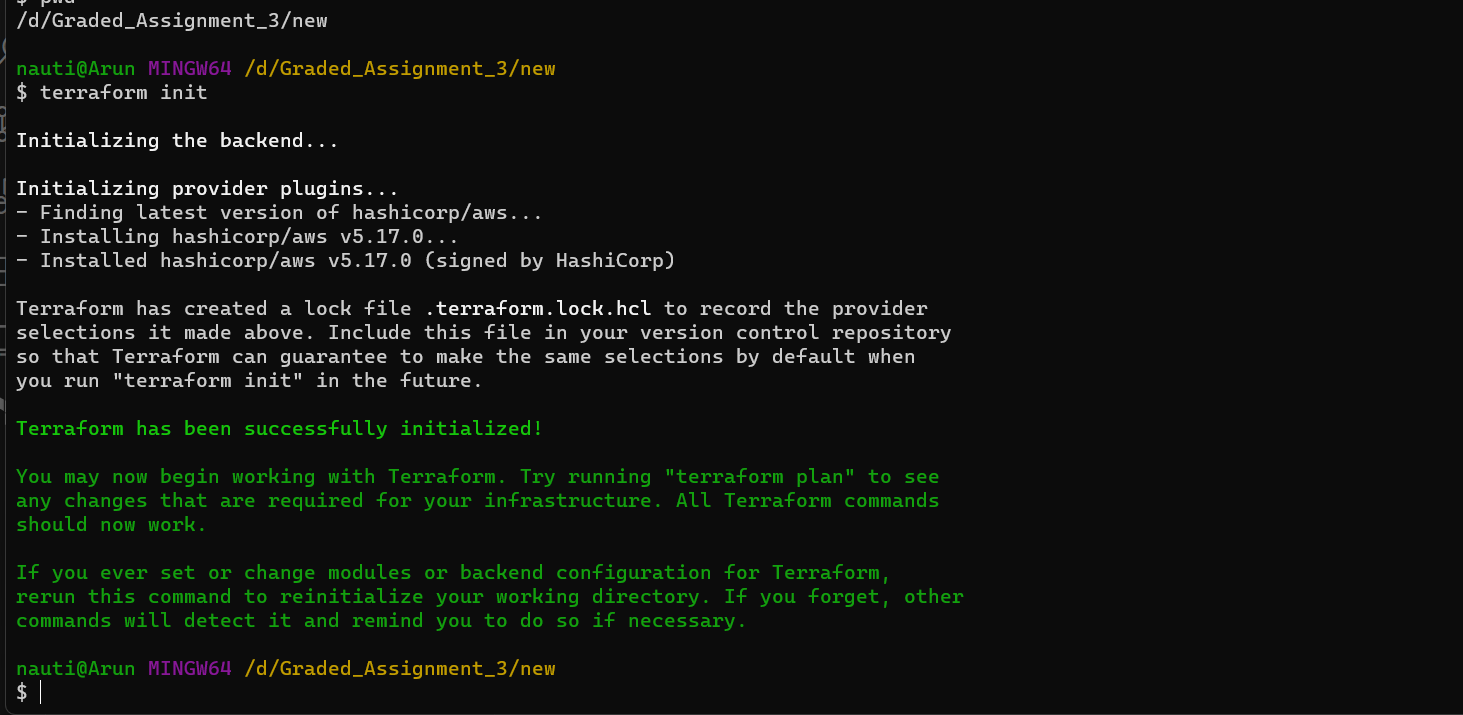
1. Separate variables file 5 marks
2. variable "vpc\_id" {
3. default = "vpc-0a29bcc23978eda9d"
4. }
5. variable "subnet1" {
6. default = "subnet-0659c2f344309404f"
7. }
8. variable "subnet2" {
9. default = "subnet-099b097e0e5ca2088"
10. }
11. variable "instance\_type" {
12. description = "The EC2 instance type for the ASG instances."
13. default     = "t2.micro"  # Change this to your desired instance type
14. }
15. variable "key\_name" {
16. description = "The name of the AWS Key Pair for SSH access to instances."
17. default     = "test\_access\_key" # Change this to your key pair name
18. }
    1. Output file showing DNS of load balancer

"lab-alb-892180244.us-east-1.elb.amazonaws.com"

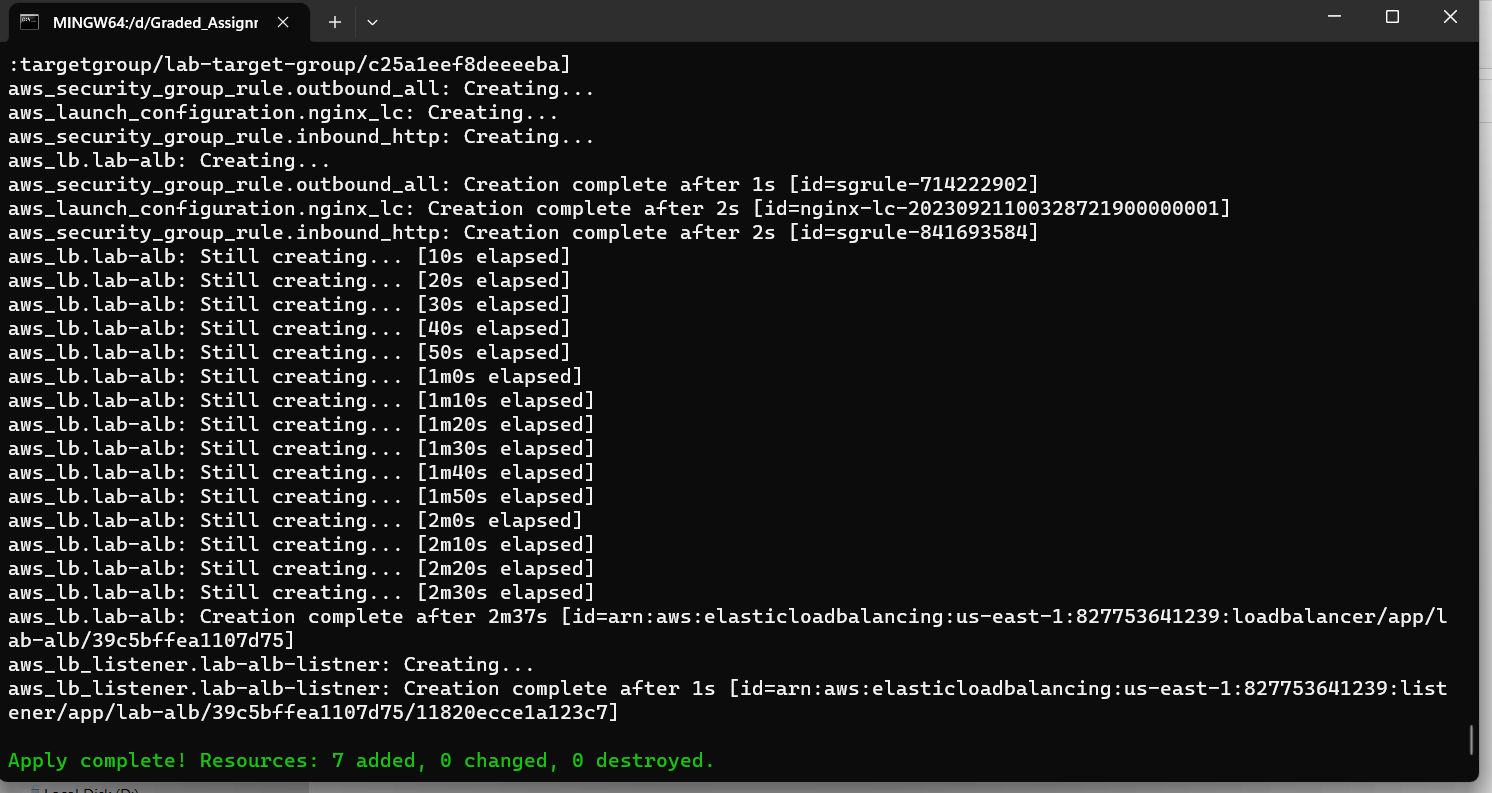
5) Terraform script with given components

* 1. Security groups 2 marks
  2. EC2 instances 3 marks
  3. Elastic Load balancer 5 marks
  4. Autoscaling group 5 marks

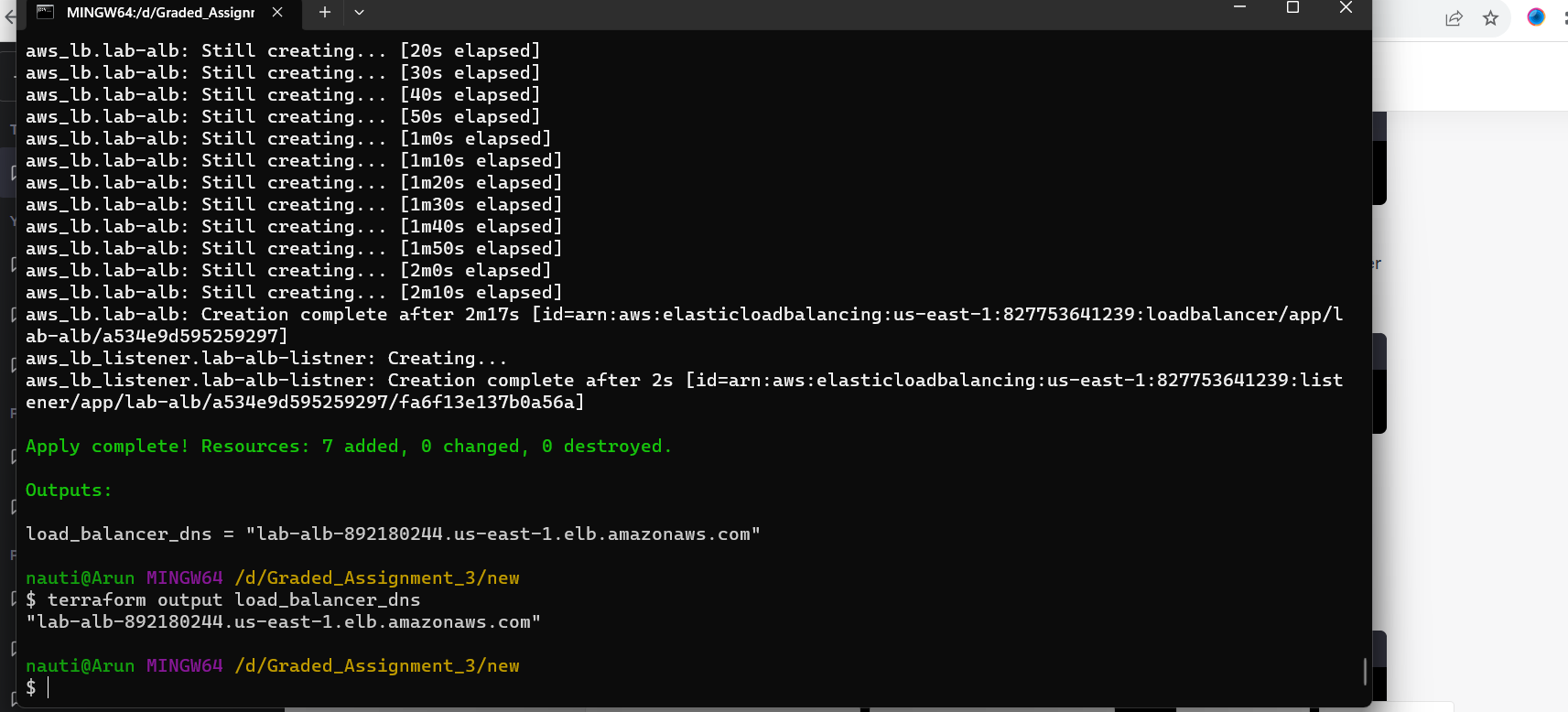
1. provider "aws" {
2. region = "us-east-1"
3. }
4. resource "aws\_launch\_configuration" "nginx\_lc" {
5. name\_prefix          = "nginx-lc-"
6. image\_id             = "ami-04cb4ca688797756f" # Specify your desired AMI ID
7. instance\_type        = var.instance\_type
8. key\_name             = var.key\_name
9. security\_groups = [
10. "${aws\_security\_group.lab-alb-sg.id}",
11. ]
12. associate\_public\_ip\_address = true
13. user\_data = <<-EOF
14. #!/bin/bash
15. yum update -y
16. yum install -y nginx
17. service nginx start
18. chkconfig nginx on
19. EOF
20. }
21. resource "aws\_lb\_target\_group" "lab-target-group" {
22. name        = "lab-target-group"
23. port        = 80
24. protocol    = "HTTP"
25. target\_type = "instance"
26. vpc\_id      = var.vpc\_id
27. health\_check {
28. interval            = 10
29. path                = "/"
30. protocol            = "HTTP"
31. timeout             = 5
32. healthy\_threshold   = 5
33. unhealthy\_threshold = 2
34. }
35. }
36. resource "aws\_lb" "lab-alb" {
37. name     = "lab-alb"
38. internal = false
39. ip\_address\_type    = "ipv4"
40. load\_balancer\_type = "application"
41. tags = {
42. Name = "lab-alb"
43. }
44. subnets = [
45. "${var.subnet1}",
46. "${var.subnet2}",
47. ]
48. security\_groups = [
49. "${aws\_security\_group.lab-alb-sg.id}",
50. ]
51. }
52. resource "aws\_lb\_listener" "lab-alb-listner" {
53. protocol          = "HTTP"
54. port              = 80
55. load\_balancer\_arn = aws\_lb.lab-alb.arn
56. default\_action {
57. target\_group\_arn = aws\_lb\_target\_group.lab-target-group.arn
58. type             = "forward"
59. }
60. }
61. resource "aws\_security\_group" "lab-alb-sg" {
62. name   = "lab-alb-sg"
63. vpc\_id = var.vpc\_id
64. }
65. resource "aws\_security\_group\_rule" "inbound\_http" {
66. type              = "ingress"
67. from\_port         = 80
68. to\_port           = 80
69. protocol          = "tcp"
70. cidr\_blocks       = ["0.0.0.0/0"]
71. security\_group\_id = aws\_security\_group.lab-alb-sg.id
72. }
73. resource "aws\_security\_group\_rule" "outbound\_all" {
74. type              = "egress"
75. from\_port         = 0
76. to\_port           = 0
77. protocol          = "-1"
78. security\_group\_id = aws\_security\_group.lab-alb-sg.id
79. cidr\_blocks       = ["0.0.0.0/0"]
80. }
81. output "load\_balancer\_dns" {
82. value = aws\_lb.lab-alb.dns\_name
83. }
    1. Screenshot showing successful execution of Terraform script 5 marks
84. Terraform init



1. Terraform apply



1. terraform output load\_balancer\_dns



1. Terraform destroy

