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.

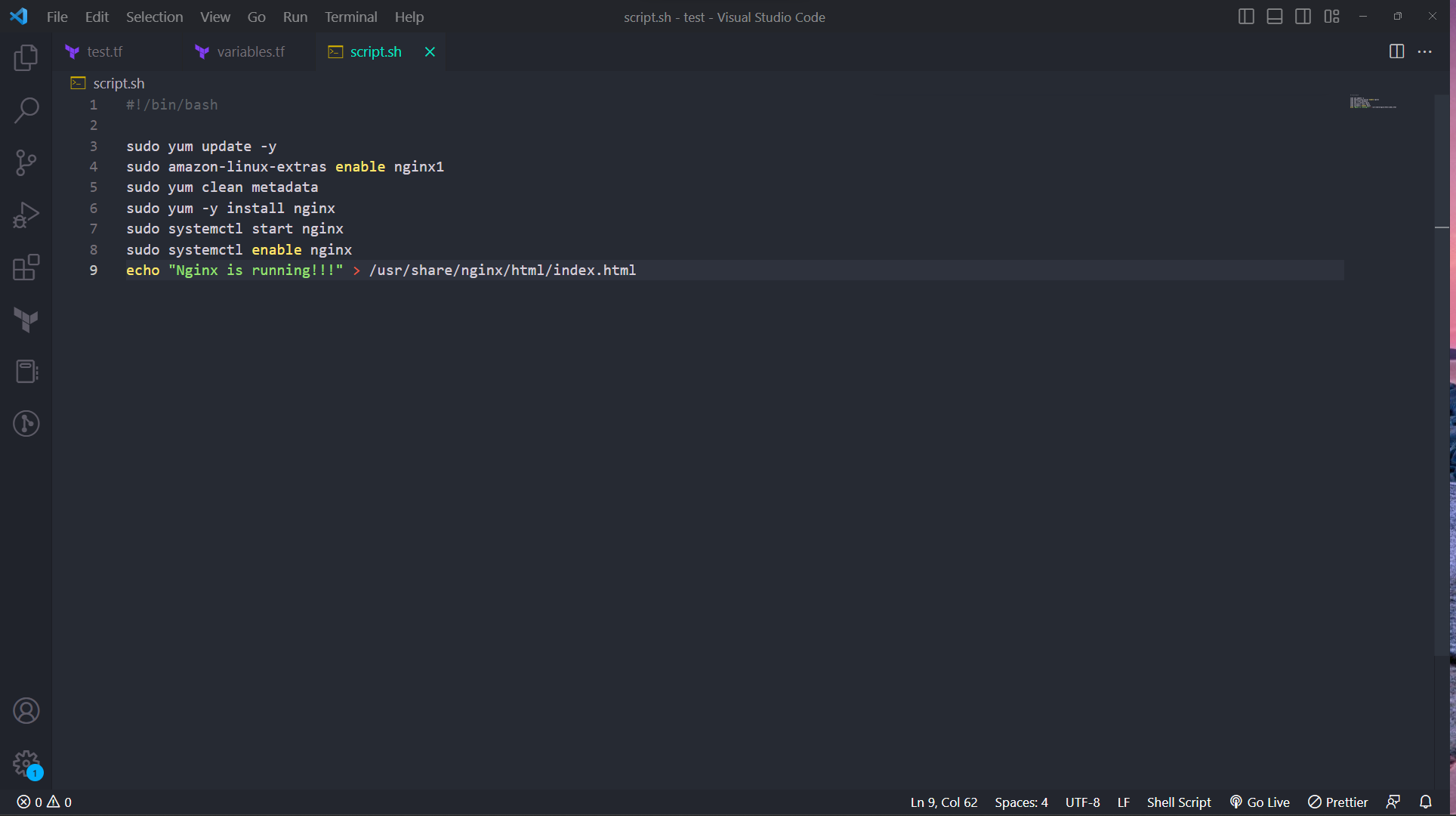
Grading Criteria

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

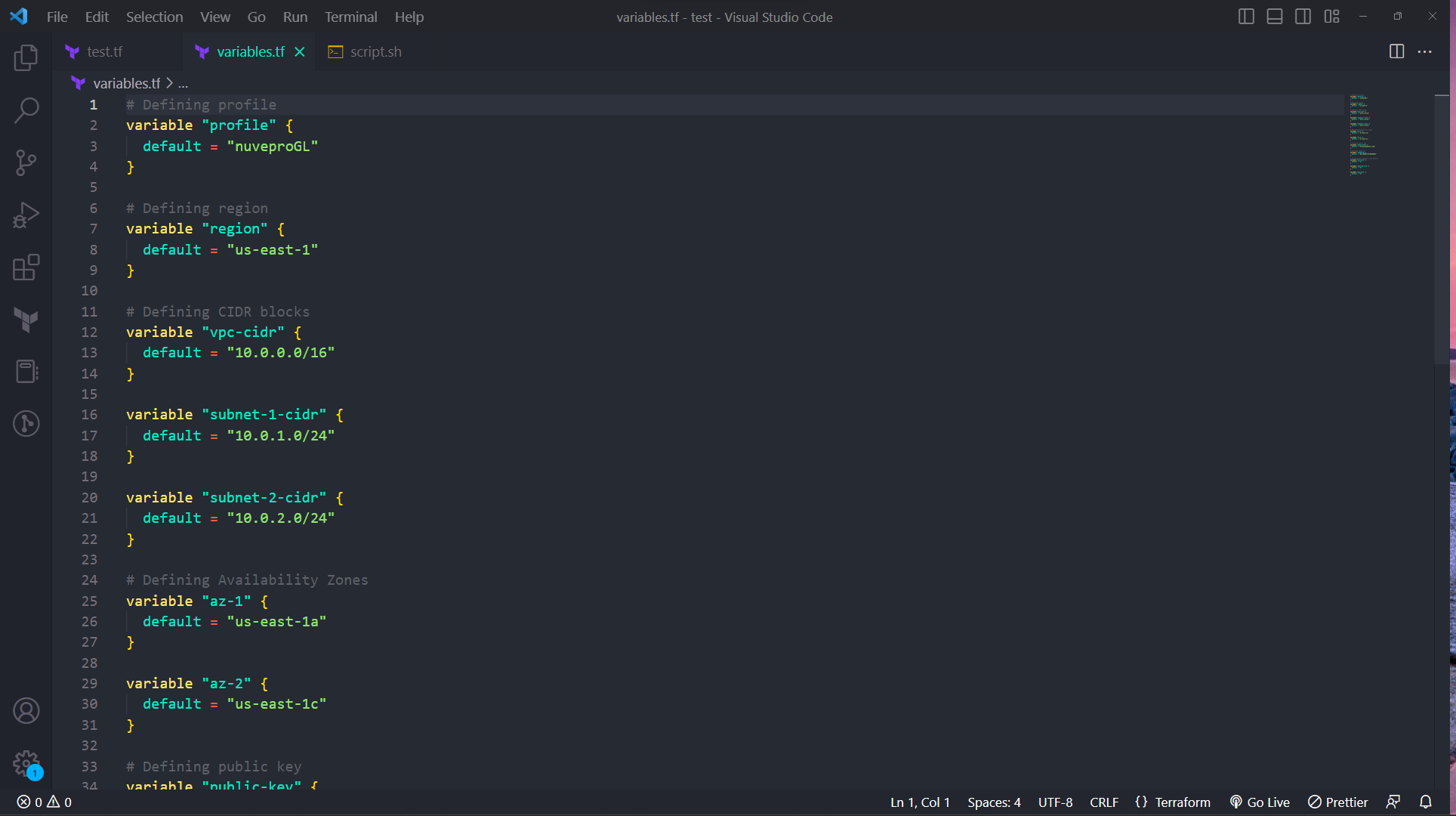
To create the Autoscaling group using terraform, the following resources need to be created:

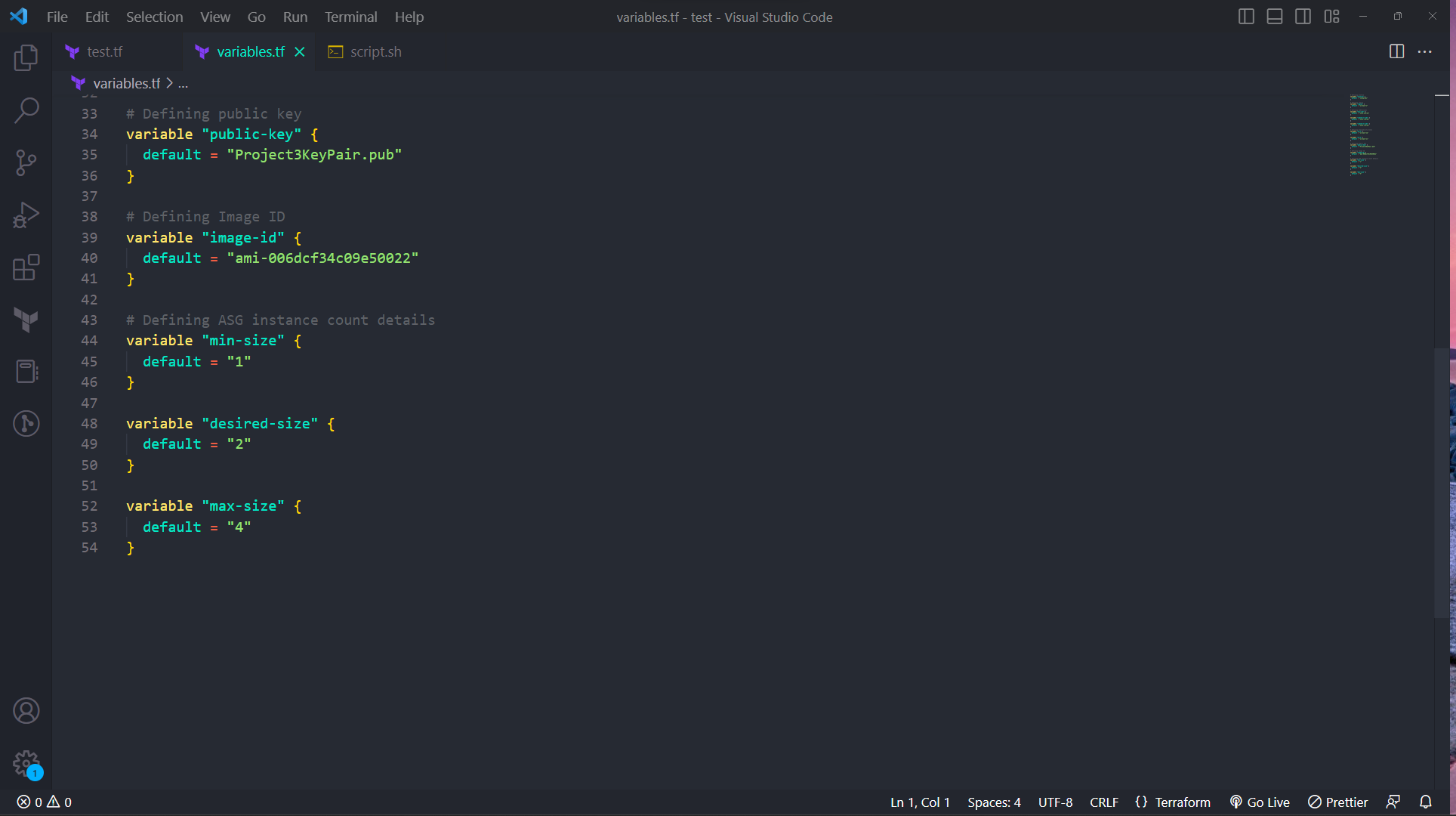
* 1. **VPC**
* CIDR block used: 10.0.0.0/16
  1. **2 Public subnets**
* Subnet 1:
  + CIDR block used: 10.0.1.0/24
  + Availability zone: us-east-1a
  + Map public IP on launch: TRUE
* Subnet 2:
  + CIDR block used: 10.0.2.0/24
  + Availability zone: us-east-1c
  + Map public IP on launch: TRUE
  1. **Internet gateway**
* Link to the created custom VPC
  1. **Route table**
* Route all traffic (0.0.0.0/0)
* Link to the created custom IGW
* Associate both subnets in the table
  1. **2 Security groups**
* Webserver security group:
  + Open port 80
  + Open port 22
  + Open port 443
  + Allow all traffic out
* Load balancer security group:
  + Open port 80
  + Allow all traffic out
  1. **Launch configuration**
* Create a key pair using the public key
* Create custom script to install nginx on each instance on spin up
* AMI used: ami-006dcf34c09e50022
* Instance type: t2.micro
* Use the key pair created for SSH
* Attach the created security group for webserver
* Attach the created script as user data
  1. **Elastic load balancer**
* Attach the created security group for load balancer
* Attach the created subnets
* Cross zone load balancing: TRUE
* Attach a listener on port 80 with HTTP protocol
  1. **Autoscaling group**
* Attach the created launch configuration
* Add the subnets as VPC zone identifiers
* Capacities used:
  + Minimum capacity: 1
  + Desired capacity: 2
  + Maximum capacity: 4
* Attach the created load balancer
* Assign ELB as health check type
* Enable metrics with mentioning granularity
  1. **2 Autoscaling policies**
* Scale UP policy:
  + Scaling adjustment: 1
  + Adjustment type: ChangeInCapacity
  + Cooldown(s): 300
  + Attach the created autoscaling group
* Scale DOWN policy:
  + Scaling adjustment: -1
  + Adjustment type: ChangeInCapacity
  + Cooldown(s): 300
  + Attach the created autoscaling group
  1. **2 Cloud watch alarms**
* Scale UP metric alarm:
  + Comparison operator: GreaterThanOrEqualToThreshold
  + Evaluation periods: 2
  + Metric name: CPUUtilization
  + Statistic: Average
  + Threshold: 75
  + Namespace: AWS/EC2
  + Period(s): 120
  + Attach the created autoscaling group
  + Attach the created scale UP policy
* Scale DOWN metric alarm:
  + Comparison operator: LessThanOrEqualToThreshold
  + Evaluation periods: 2
  + Metric name: CPUUtilization
  + Statistic: Average
  + Threshold: 25
  + Namespace: AWS/EC2
  + Period(s): 120
  + Attach the created autoscaling group
  + Attach the created scale DOWN policy

1. Shell script used to install Nginx 5 marks

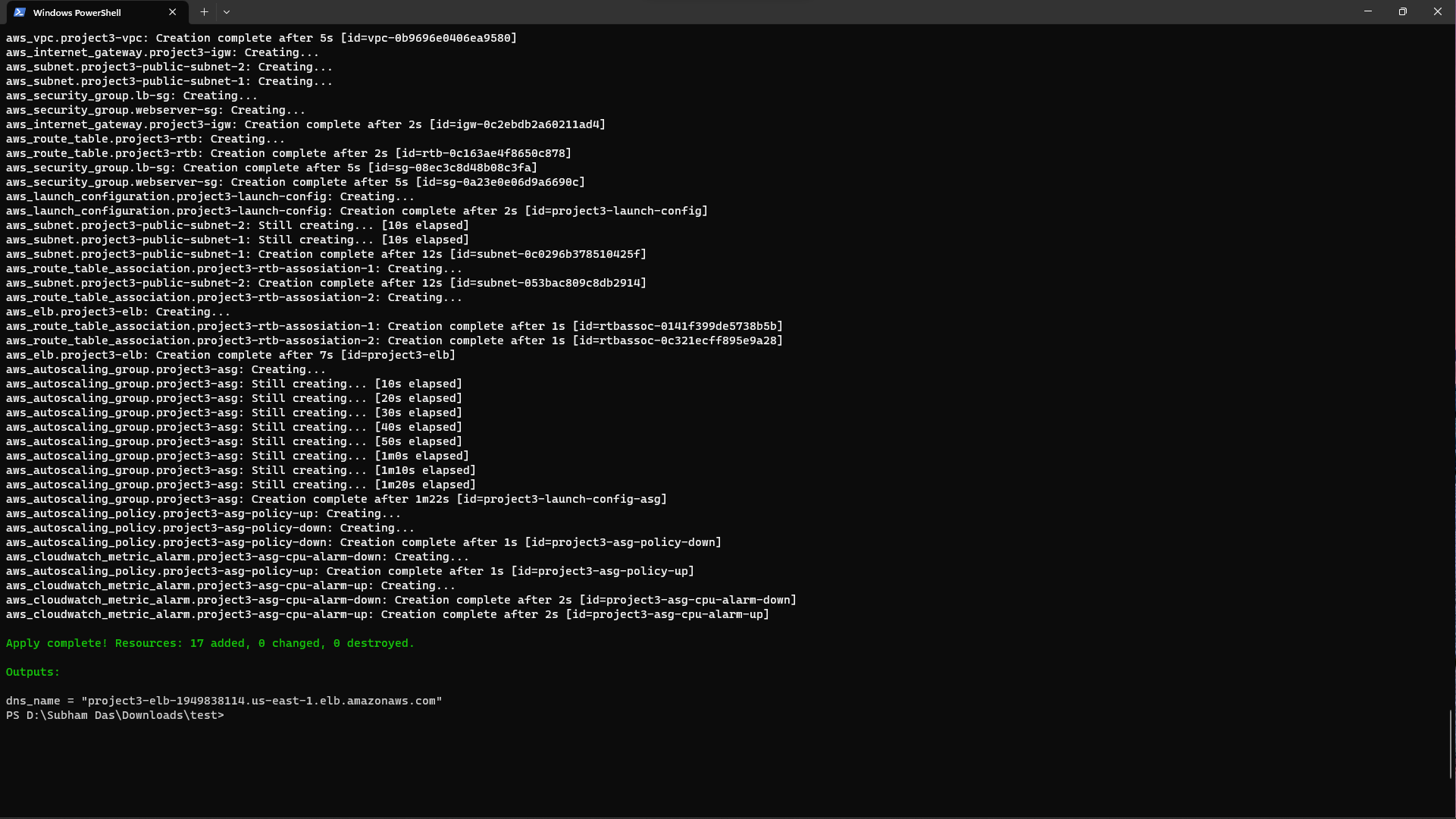


1. Separate variables file 5 marks

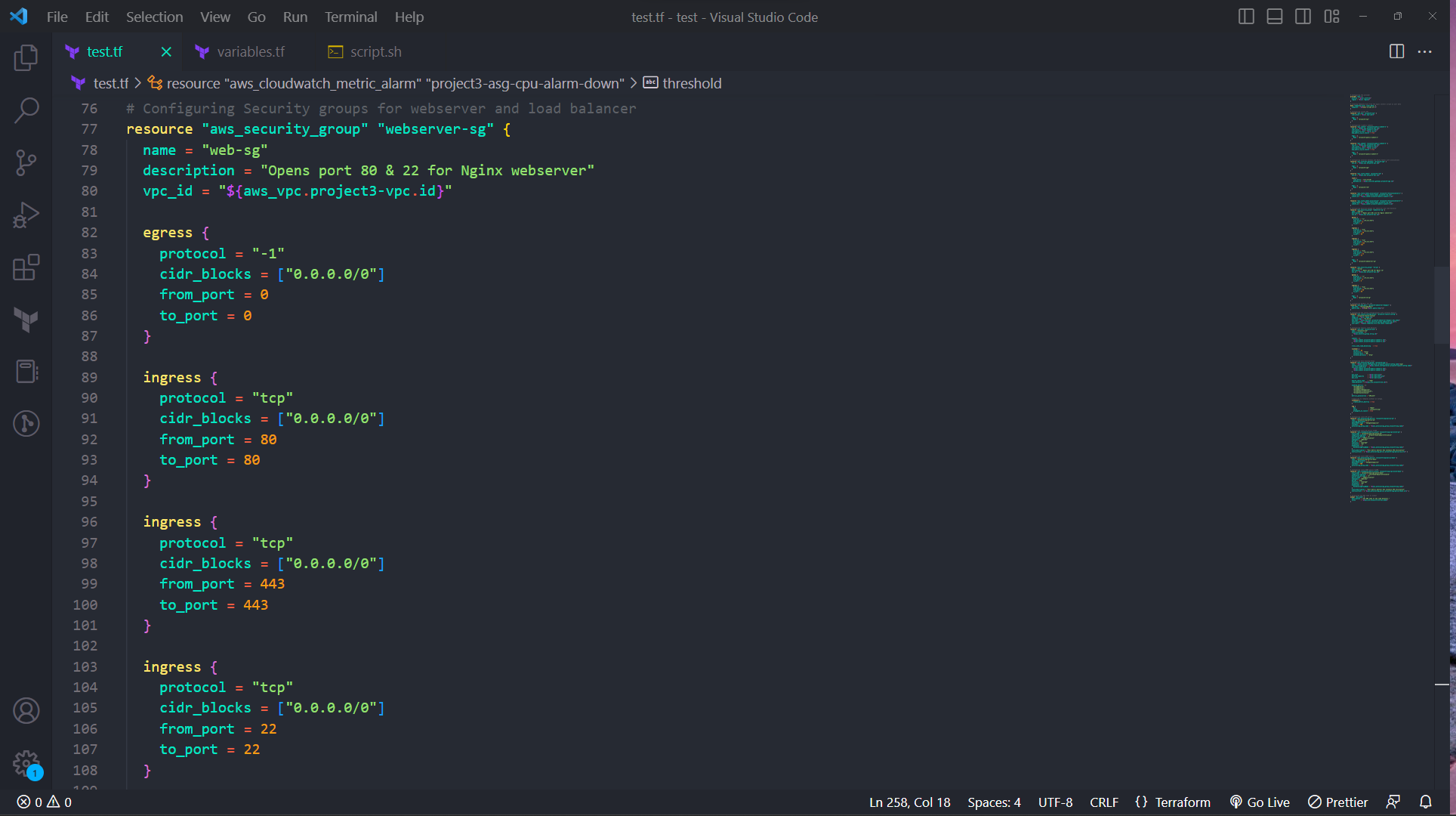


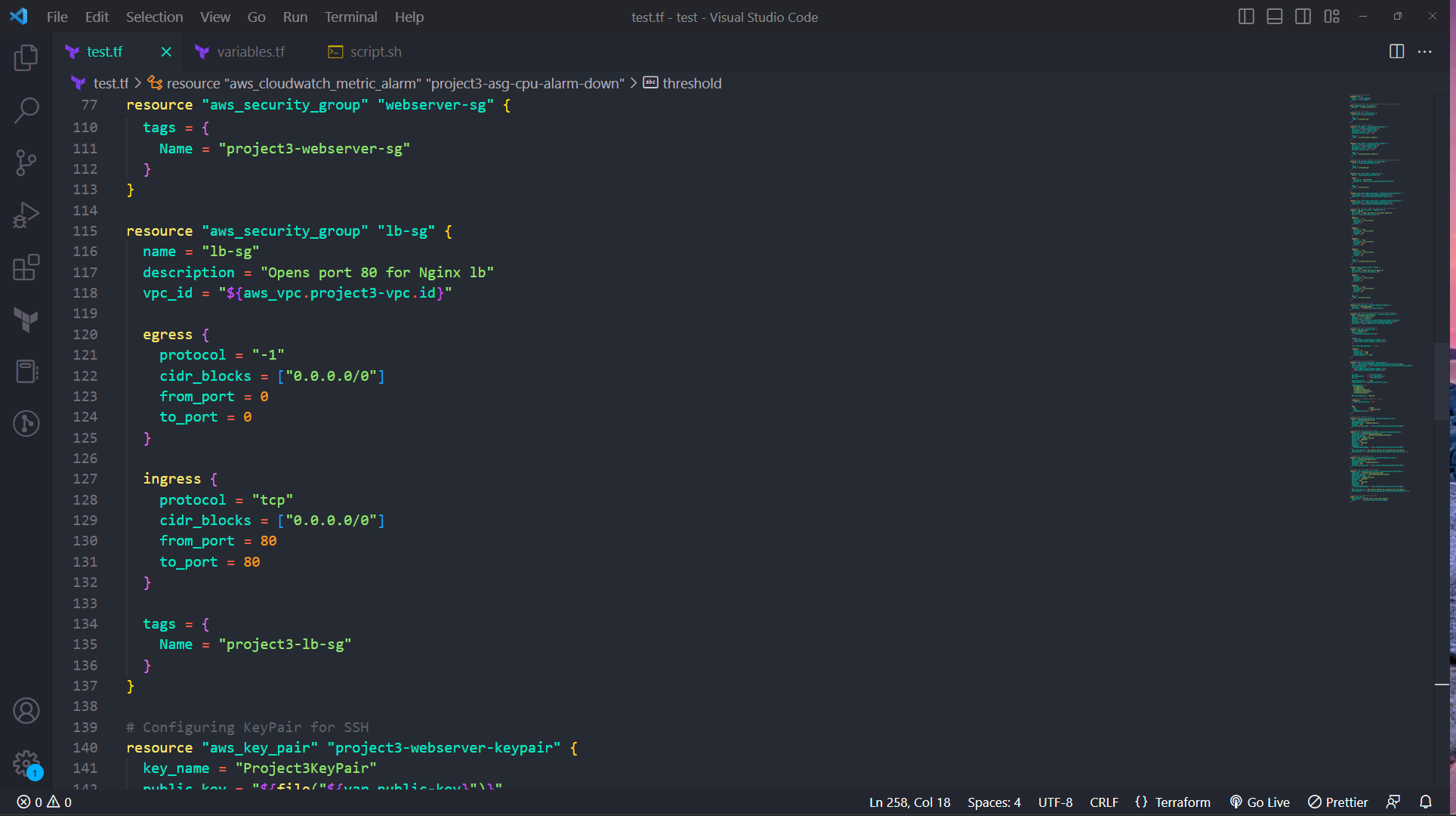


1. Output file showing DNS of load balancer 10 marks

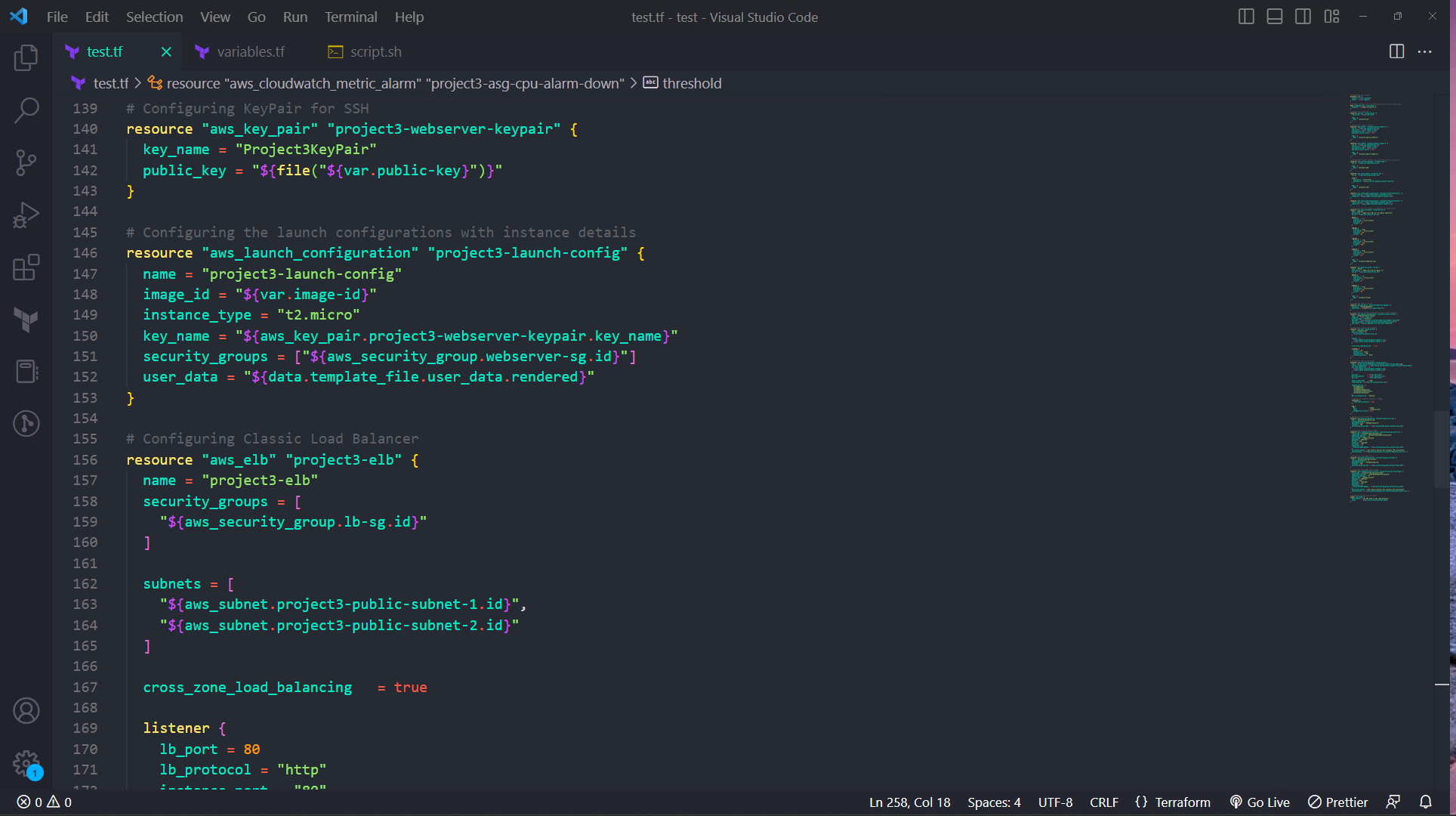


1. Terraform script with given components
   1. Security groups 2 marks

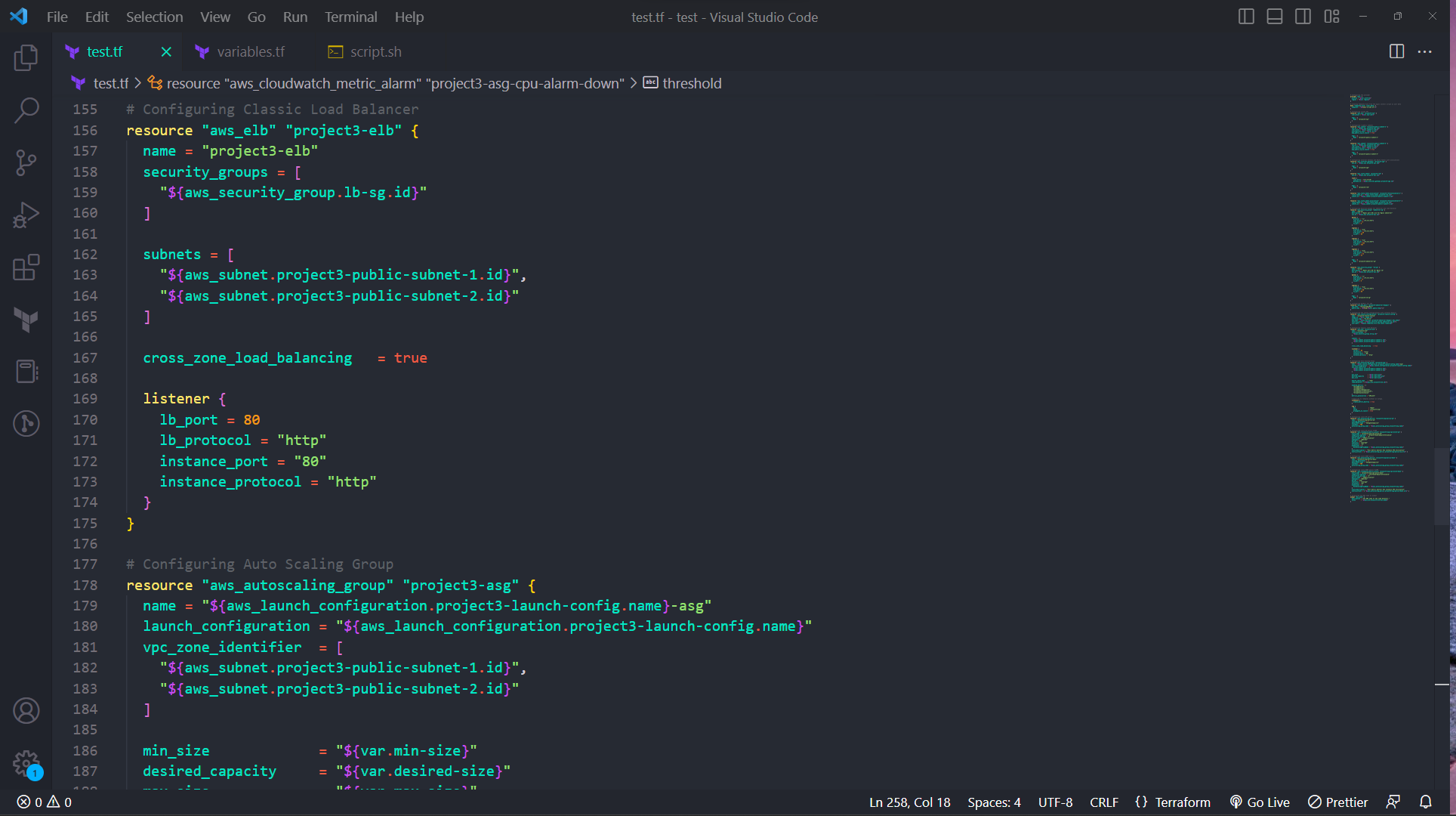




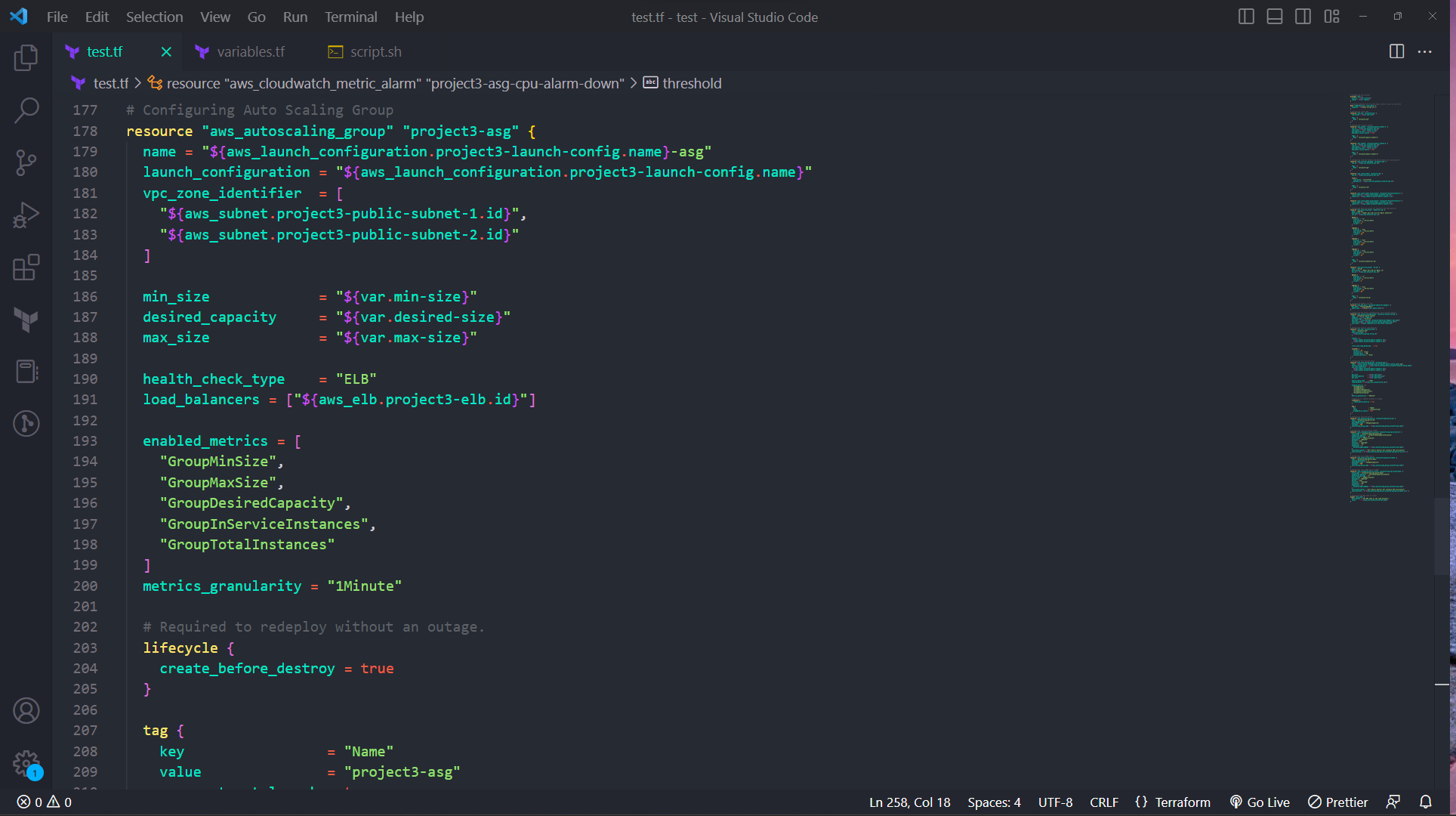
* 1. EC2 instances 3 marks

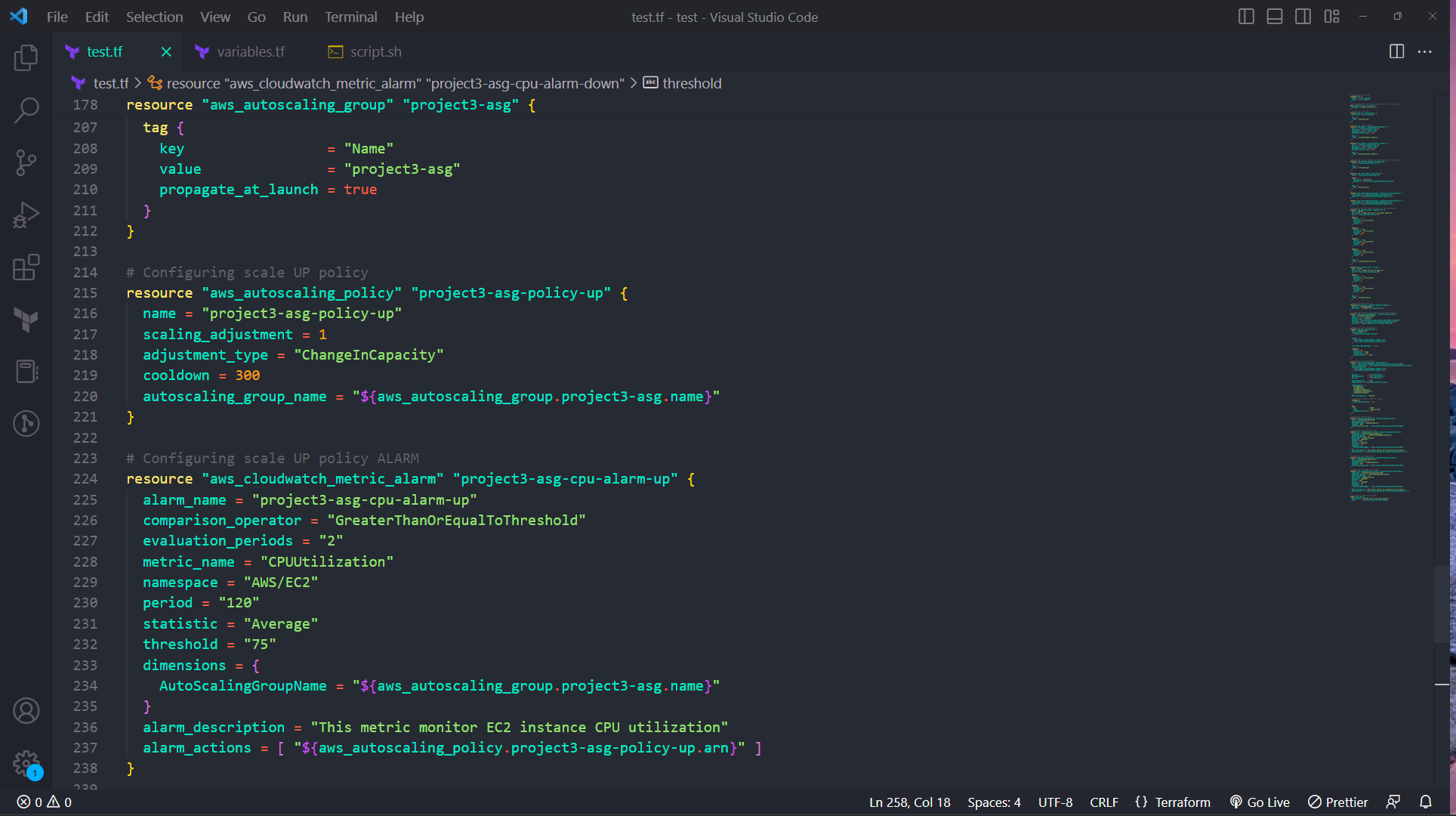


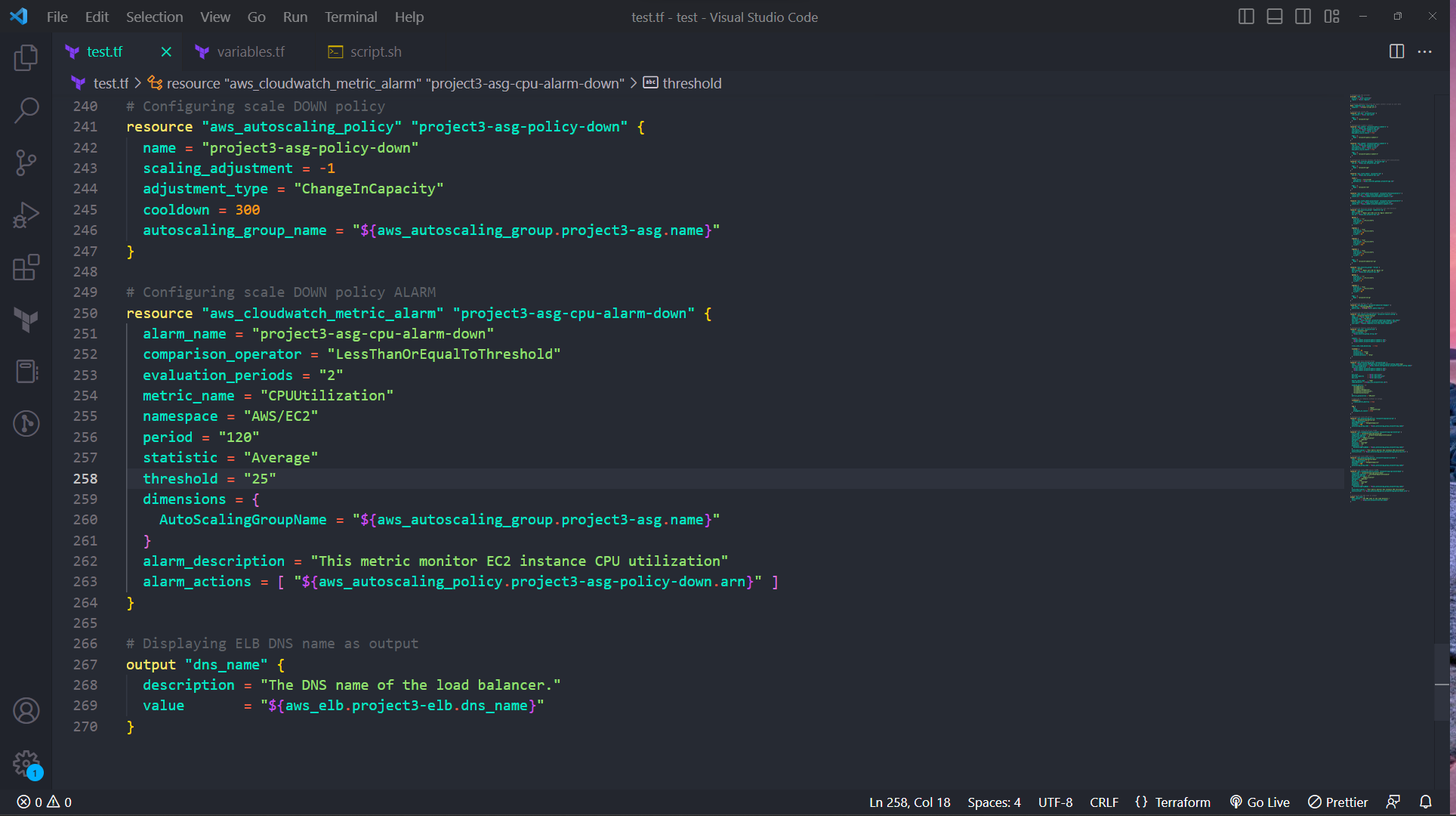
* 1. Elastic Load balancer 5 marks



* 1. Autoscaling group 5 marks







1. Screenshot showing successful execution of Terraform script 5 marks

