Eco-Friendly Fleet Management

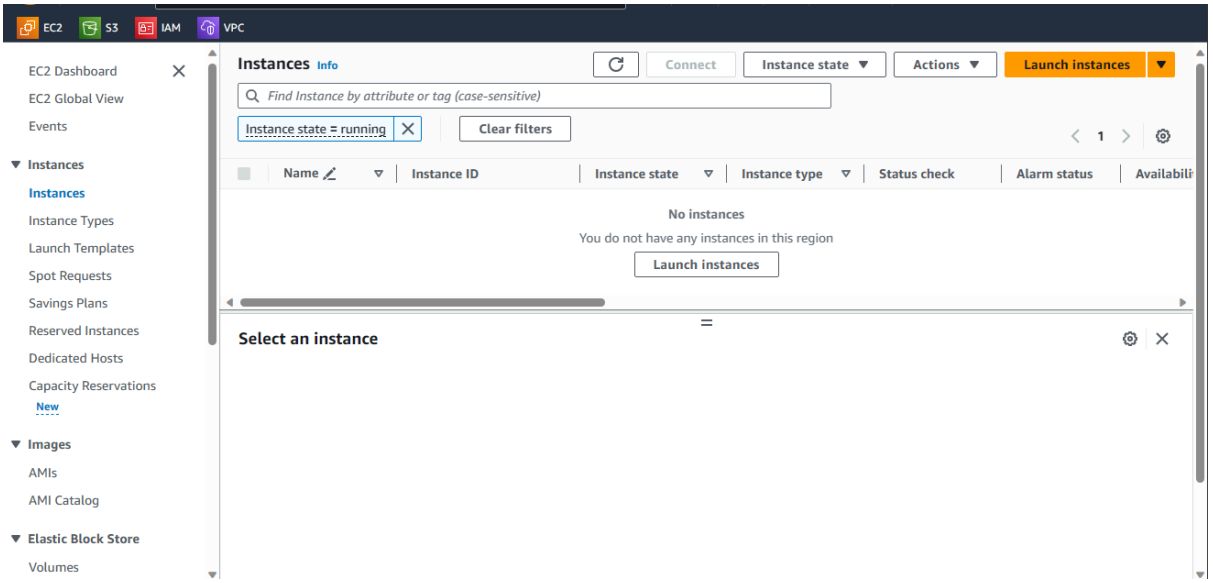
Author	Olatunji Olayinka Oluwaseun
LinkedIn	https://www.linkedin.com/in/olatunji-olayinka-coder/
GitHub	https://github.com/olatunji-weber/eco-friendly-fleet-management.git
Status	Done

Problem Statement:

You've been tasked with creating a flexible infrastructure for an eco-friendly car-sharing service. The service should automatically adapt to varying usage demands while keeping costs in check.

Guidelines/Goals:

1. **Create EC2 Instances and Load Balancer:**
    - Launch EC2 instances using a predefined AMI.
    - Create an Elastic Load Balancer (ELB) to distribute traffic.
2. **Set Up Auto Scaling:**
    - Create an Auto Scaling Group for the EC2 instances.
    - Configure scaling policies to add or remove instances based on CPU utilization.
3. **Optimize Costs:**
    - Implement scaling policies that consider cost optimization.
    - Use Auto Scaling to minimize instances during periods of low demand.



Here is a glimpse of the code in “main.tf” that will achieve the provisioning and de-provisioning via Terraform.

The scripts in this repository will automate the deployment of a scalable and highly available web application architecture on Amazon Web Services (AWS). The infrastructure includes:

- **Network Setup:** Creates a Virtual Private Cloud (VPC), subnets in different availability zones, and associates them with a custom route table and an internet gateway for internet access.
- **Security Configuration:** Establishes security groups controlling inbound and outbound traffic to EC2 instances.
- **Instance Configuration:** Sets up launch configurations for EC2 instances running an Apache web server and generates an HTML page displaying instance metadata.
- **Auto Scaling and Load Balancing:** Implements an Auto Scaling Group (ASG) and an Application Load Balancer (ALB) for distributing traffic among instances.

Infrastructure Provisioning Phase:

```terraform
# create vpc
resource "aws_vpc" "vpc" {
  cidr_block = "10.0.0.0/16"
  #10.0.0.0/16 means we have 65,536 IP addresses available to us in this VPC
  #Ip range is from 10.0.0.0 - 10.0.255.255
  tags = {
    Name = "eco_friendly_vpc"
  }

}
#end of vpc creation

#internet gateway is used to allow traffic to enter and leave the VPC
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.vpc.id
}

#subnet is used to divide the VPC into smaller networks
resource "aws_subnet" "subnet_az1" {
  cidr_block = "10.0.1.0/24"
  #10.0.1.0/24 has a range from 10.0.1.0 - 10.0.1.255
  availability_zone = "us-west-2a"
  vpc_id            = aws_vpc.vpc.id
}

resource "aws_subnet" "subnet_az2" {
  cidr_block = "10.0.2.0/24"
  #10.0.2.0/24 has a range from 10.0.2.0 - 10.0.2.255
  availability_zone = "us-west-2c"
  vpc_id            = aws_vpc.vpc.id
}
#end of subnet creation

# Creating a route table
resource "aws_route_table" "custom_route_table" {
  vpc_id = aws_vpc.vpc.id

  # Define your routes here
  route {
    cidr_block = "0.0.0.0/0"
```

The next thing to do in order to begin provisioning the resources that will create our infrastructure for our eco-friendly car-sharing service is to type in and execute the “terraform init” command at the prompt.

```
PS C:\Users\Olatunji Olayinka\Documents\FormerHDD\Programming\Node-React-NPM&More\IntroToNode&Express\AzubiParole\Azubi2023Work-Cloud
Class\AzubiProjects\week14_projects\ExerciseDm\eco-friendly-fleet-management> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.26.0...
- Installed hashicorp/aws v5.26.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\Olatunji Olayinka\Documents\FormerHDD\Programming\Node-React-NPM&More\IntroToNode&Express\AzubiParole\Azubi2023Work-Cloud
Class\AzubiProjects\week14_projects\ExerciseDm\eco-friendly-fleet-management>
```

This will create the “.terraform.lock.hcl” in order to record provider selections made. The file is maintained by “terraform init”, so you do not need to do any manual updates here.

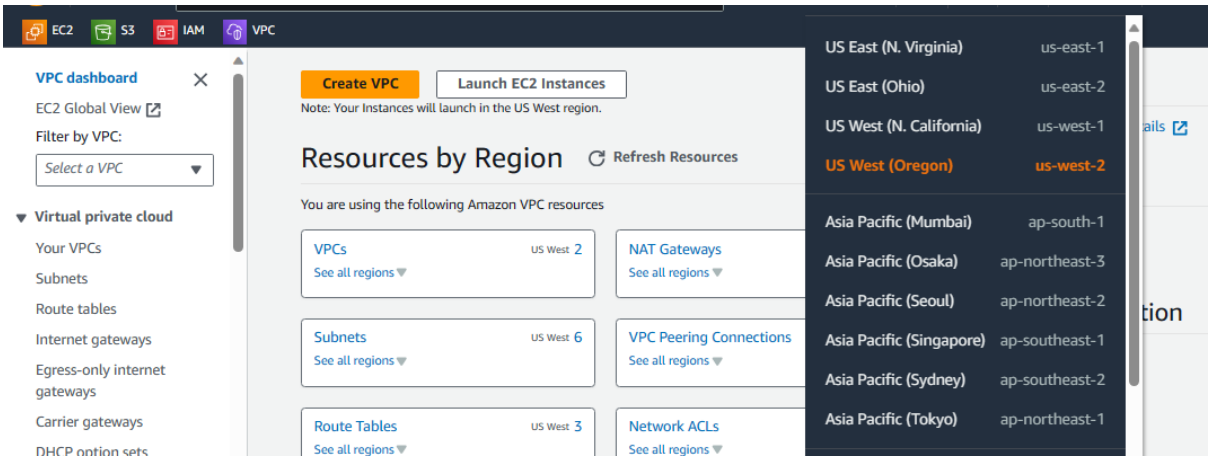
```
# This file is maintained automatically by "terraform init".
# Manual edits may be lost in future updates.

provider "registry.terraform.io/hashicorp/aws" {
  version = "5.26.0"
  hashes = [
    "h1:FAOnqlKh7dBZ/OTMFD8x0rO384Udr4yB7qySAKX3mnk=",
    "zh:11a4062491e574c8e96b6bc7ced67b5e9338ccfa068223fc9042f9e1e7eda47a",
    "zh:4331f85aeb22223ab656d04b48337a033f44f02f685c8def604c4f8f4687d10f",
    "zh:915d6c996390736709f7ac7582cd41418463cfc07696218af6fea4a282df744a",
    "zh:9306c306dbb2e1597037c54d20b1bd5f22a9cdcdb2b2b7bad657c8230bea2298",
    "zh:93371860b9df369243219606711bfd3cfbd263db67838c06d5d5848cf47b6ede",
    "zh:98338c17764a7b9322ddb6efd3af84e6890a4a0687f846eefdfb0fa03cec892d",
    "zh:9b12af85486a96aedd8d7984b0ff811a4b42e3d88dad1a3fb4c0b580d04fa425",
    "zh:a28c9d77a5be25bac42d99418365757e4eb65a2c7c6788828263772cf2774869",
    "zh:bd9c4648a090622d6b8c3c91dad513eec81e54db3dfe940ab6d155e5f37735e5",
    "zh:bde63db136cccdeb282489e2ec2b3f9a7566edc9df27911a296352ab00832261",
    "zh:ccd33f9490ce3f2d89efab995abf3b30e75579585f6a8a5b1f756246903d3518",
    "zh:d73d1c461eb9d22833251f6533fc214cf014bc1d3165c5bfaa8ca29cd295ffb2",
    "zh:db4ffb7eec5d0e1d0dbd0d65e1a3eaa6173a3337058105aec41fd0b2af5a2b46",
    "zh:eb36b933419e9f6563330f3b7d53d4f1b09e62d27f7786d5dc6c4a2d0f6de182",
    "zh:ec85ce1976e43f7d7fa10fa191c0a85e97326a3cb22387c0ed8b74d426ec94fd",
  ]
}
```

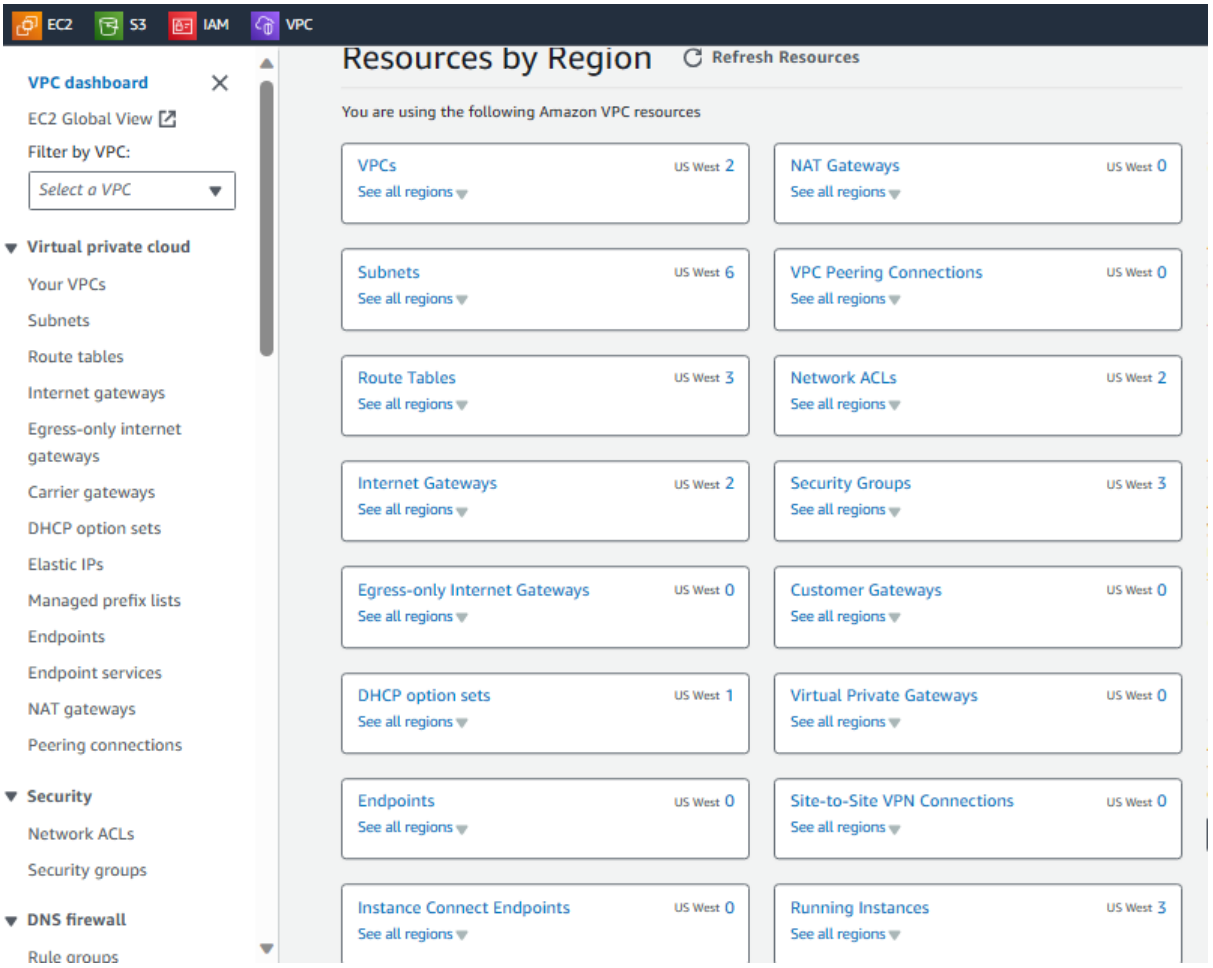
Since we have had a good idea of what will be provisioned, we can move on by issuing the command “terraform apply -auto-approve” in order to actually provision the resources that are needed for our infrastructure. The “-auto-approve” portion of the command helps us to disable the interactive approval prompt before the plan is applied.

Once all the resources are provisioned, we will see “Apply complete! Resources: 13 added, 0 changed, 0 destroyed.” and the prompt will be release back to us.

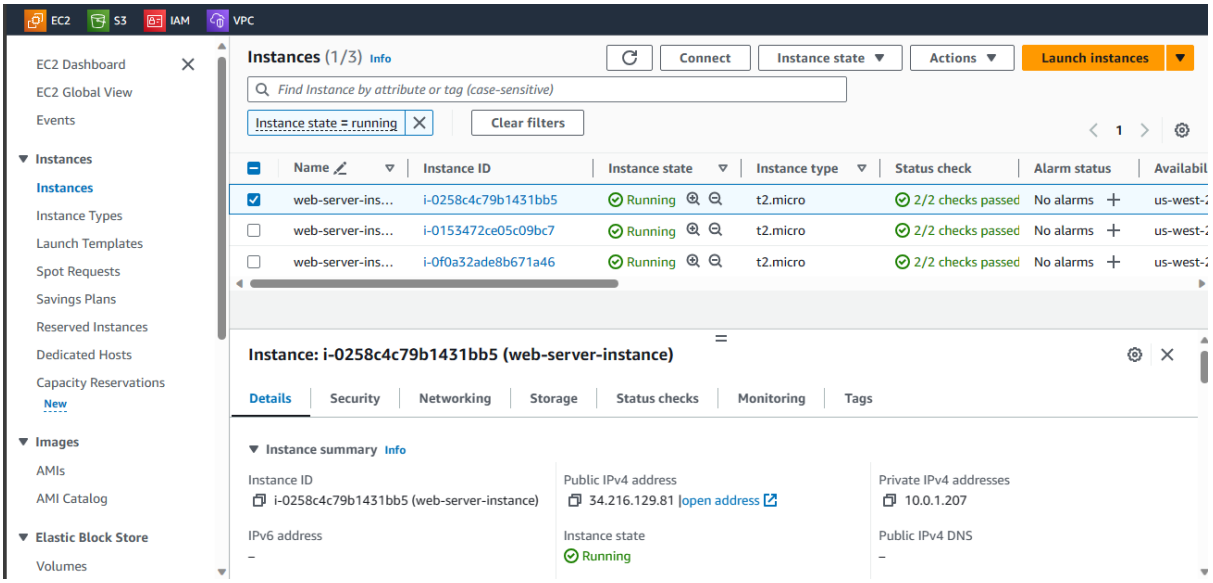
```
aws_launch_configuration.launch_config: Creation complete after 4s [id=eco-friendly-launch-config]
aws_route_table_association.subnet_az2_assoc: Creating...
aws_route_table_association.subnet_az1_assoc: Creating...
aws_autoscaling_group.asg: Creating...
aws_route_table_association.subnet_az1_assoc: Creation complete after 2s [id=rtbassoc-0aa6556691637eaa7]
aws_route_table_association.subnet_az2_assoc: Creation complete after 3s [id=rtbassoc-0ecebd8dce08fcf7a]
aws_lb.lb: Still creating... [10s elapsed]
aws_autoscaling_group.asg: Still creating... [10s elapsed]
aws_lb.lb: Still creating... [20s elapsed]
aws_autoscaling_group.asg: Still creating... [21s elapsed]
aws_lb.lb: Still creating... [30s elapsed]
aws_autoscaling_group.asg: Still creating... [31s elapsed]
aws_lb.lb: Still creating... [40s elapsed]
aws_autoscaling_group.asg: Still creating... [41s elapsed]
aws_autoscaling_group.asg: Creation complete after 45s [id=terraform-20231119134901499300000001]
aws_lb.lb: Still creating... [50s elapsed]
aws_lb.lb: Still creating... [1m0s elapsed]
aws_lb.lb: Still creating... [1m10s elapsed]
aws_lb.lb: Still creating... [1m20s elapsed]
aws_lb.lb: Still creating... [1m30s elapsed]
aws_lb.lb: Still creating... [1m40s elapsed]
aws_lb.lb: Still creating... [1m50s elapsed]
aws_lb.lb: Still creating... [2m0s elapsed]
aws_lb.lb: Still creating... [2m11s elapsed]
aws_lb.lb: Still creating... [2m21s elapsed]
aws_lb.lb: Still creating... [2m31s elapsed]
aws_lb.lb: Still creating... [2m41s elapsed]
aws_lb.lb: Creation complete after 2m50s [id=arn:aws:elasticloadbalancing:us-west-2:167834609858:loadbalancer/app/eco-friendly-lb/55ae12f546bc425b]
aws_lb_listener.listener: Creating...
aws_lb_listener.listener: Creation complete after 2s [id=arn:aws:elasticloadbalancing:us-west-2:167834609858:listener/app/eco-friendly-lb/55ae12f546
bc425b/f1a38a7d6b8885f3]

Apply complete! Resources: 13 added, 0 changed, 0 destroyed.
PS C:\Users\Olatunji Olayinka\Documents\FormerHDD\Programming\Node-React-NPM&More\IntroToNode&Express\AzubiParole\Azubi2023Work-CloudClass\AzubiProj
ects\week14_projects\ExerciseDm\eco-friendly-fleet-management>
```

All the resources were created in the US West (Oregon) - “us-west-2” as specified in the “main.tf” file. The VPC, Subnets, Routing Tables, Internet Gateway and Running Instances were all created as specified.

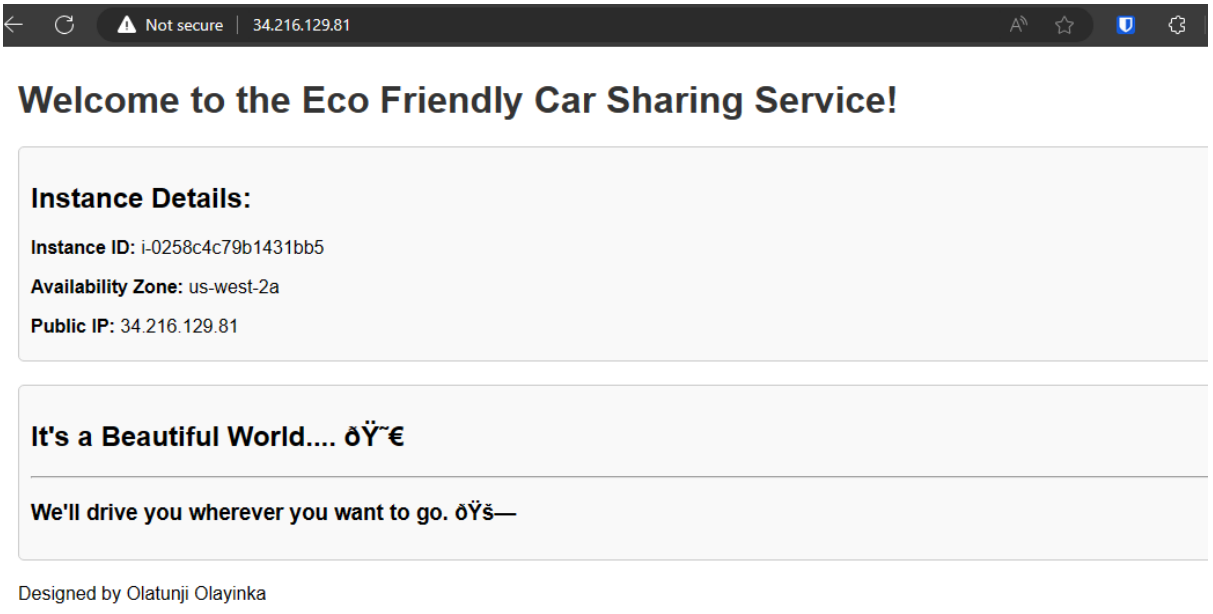




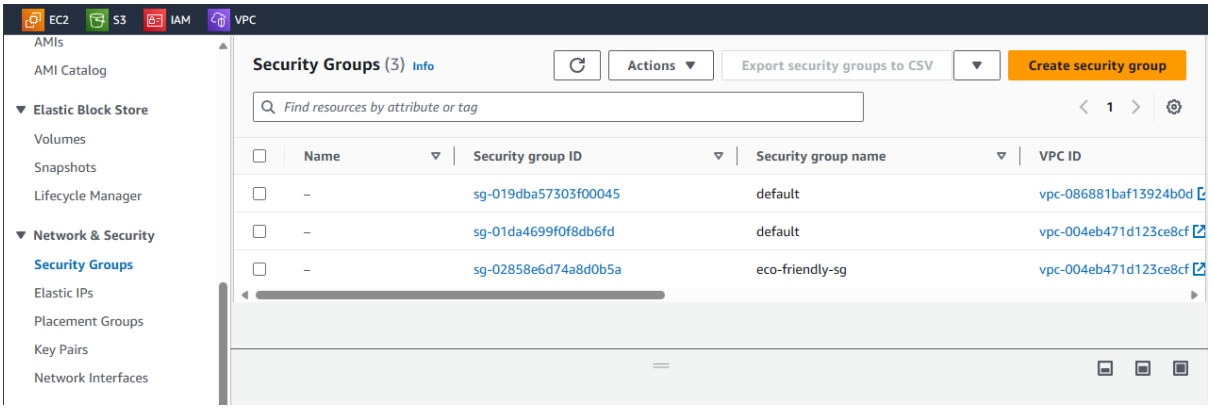
Now, if we go and take a look at the instances section of our EC2 service in our AWS Account, we will literally see that 3 EC2 instances have been provisioned.



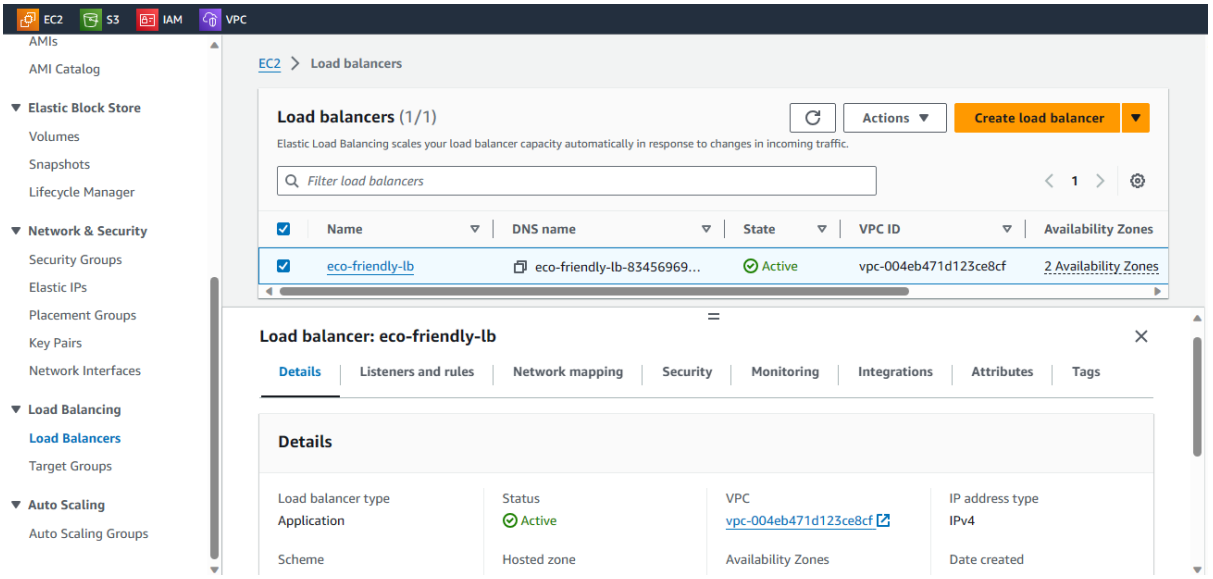
And we can also view the web page that is hosted on the instance if we access the public ip via any web browser (…meaning that Eco Friendly car service is now live). And that will literally be the case if we try and access the public ips or the other EC2 instances too.



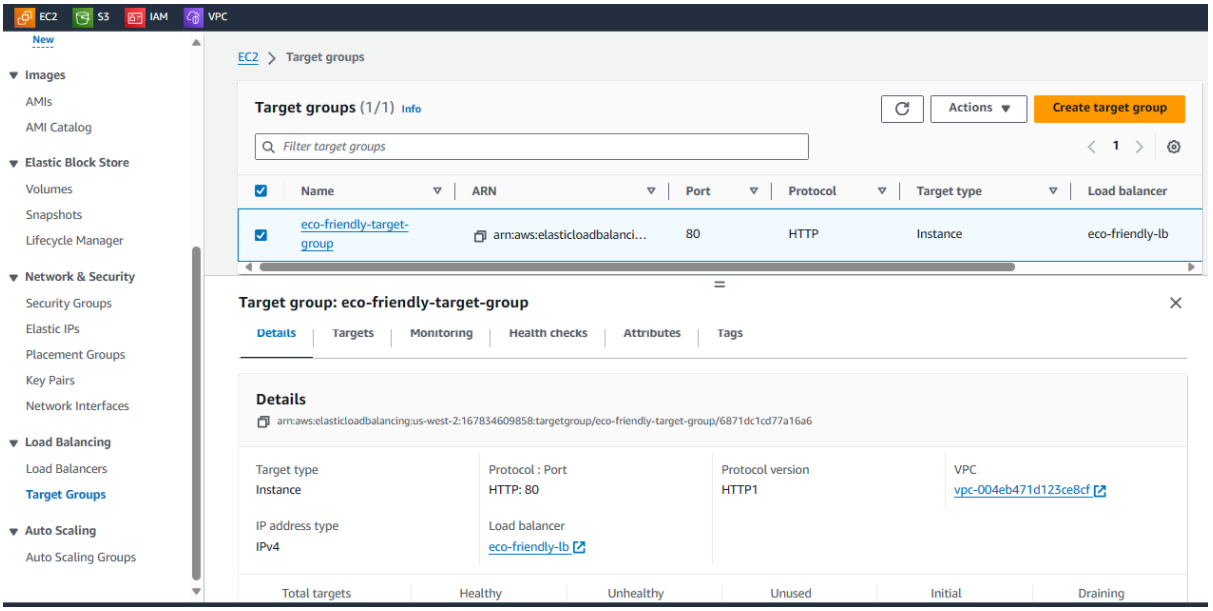
If we go ahead and check other sections like Security Groups, Load Balancers, Target Groups and Auto Scaling Group, we will find that they were all provisioned too as illustrated by the screenshots below.



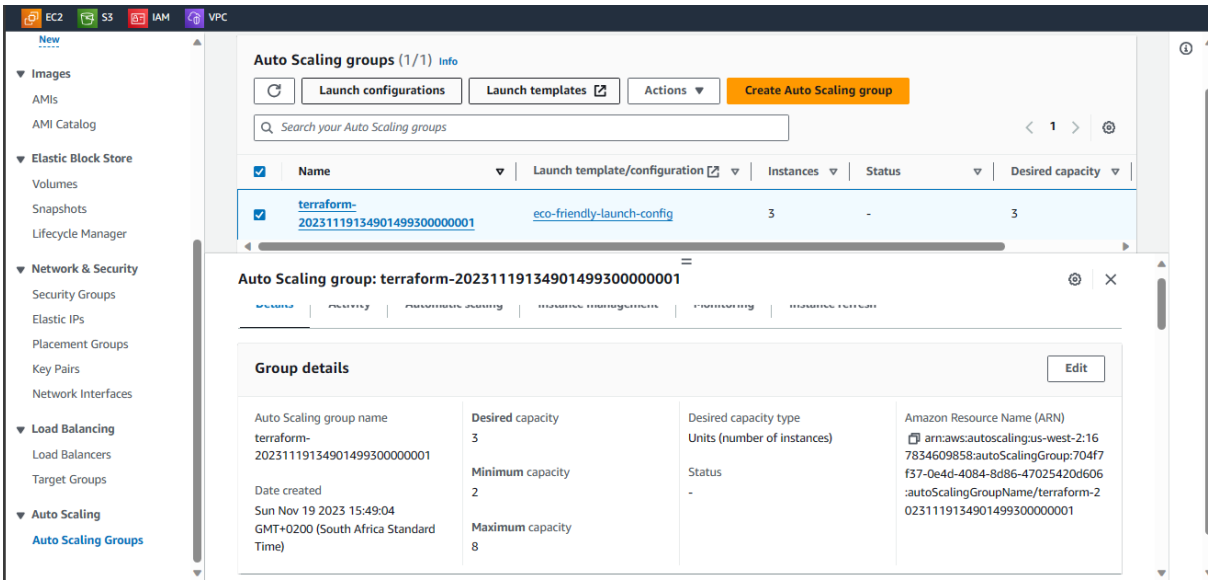
Load Balancer



Target group

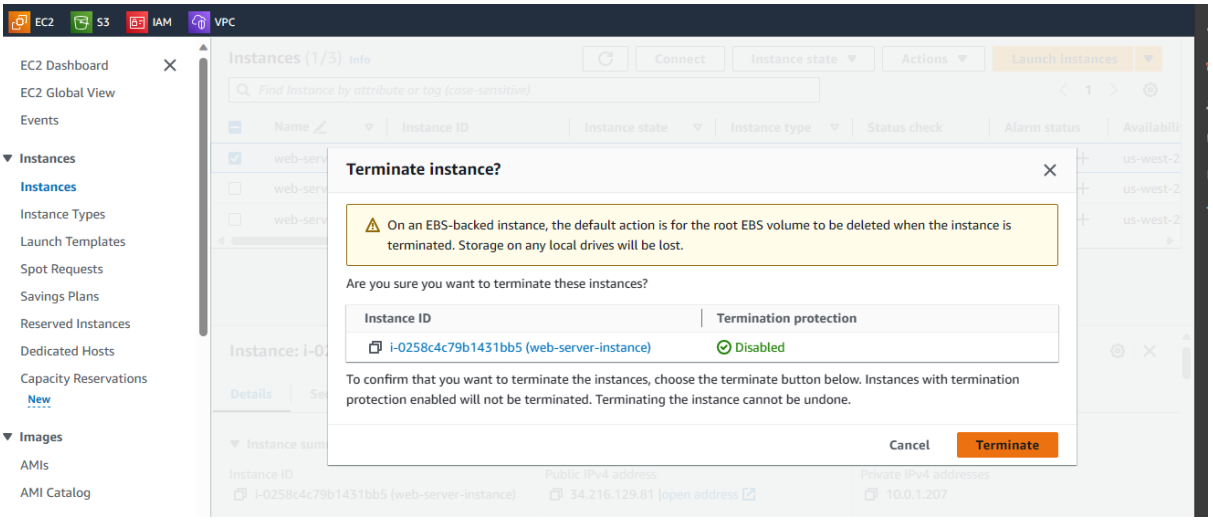


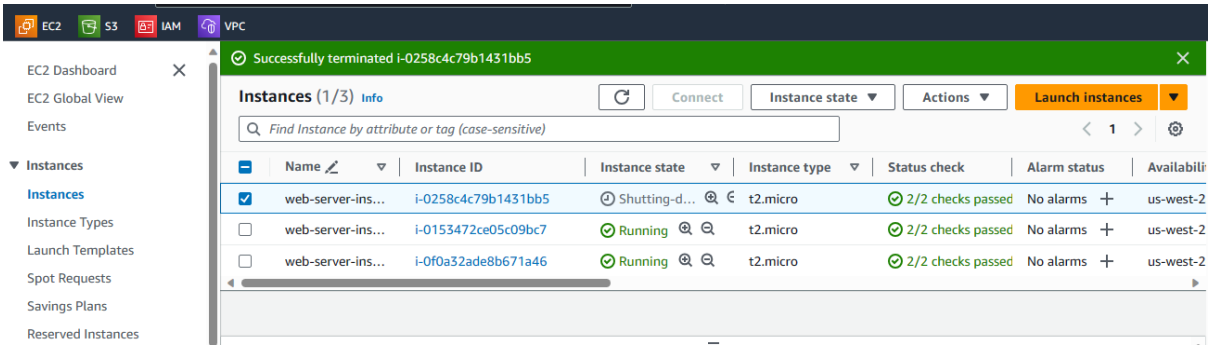
Auto Scaling group



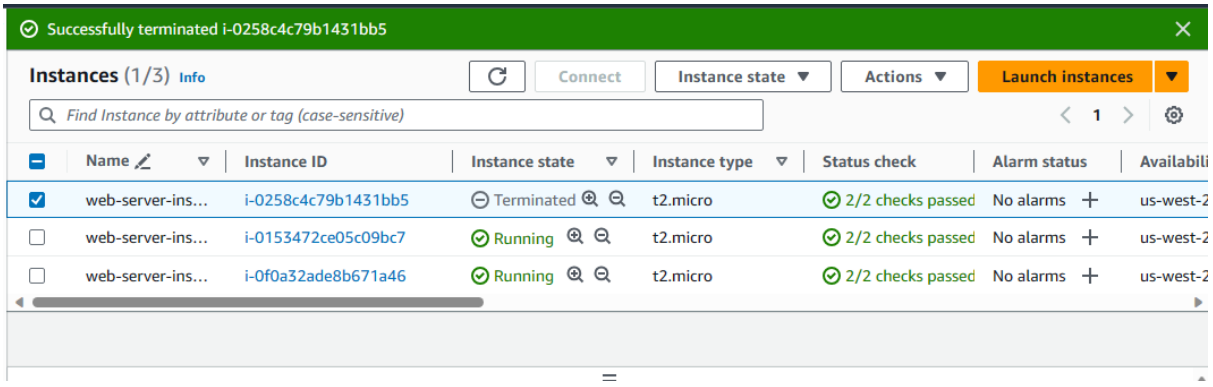
Testing Phase:

So, in order to test the function of the Auto Scaling group, we can terminate one of the EC2 instances and see that it will be re-provisioned by the Auto Scaling group.

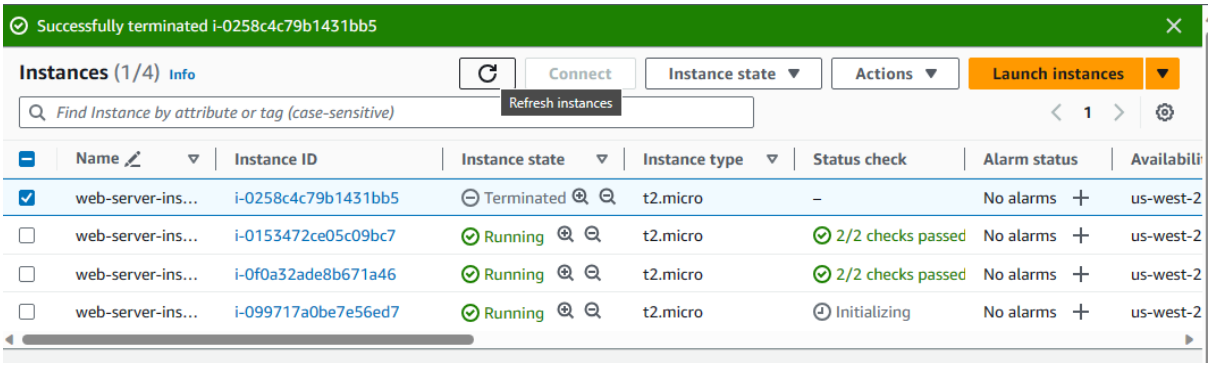




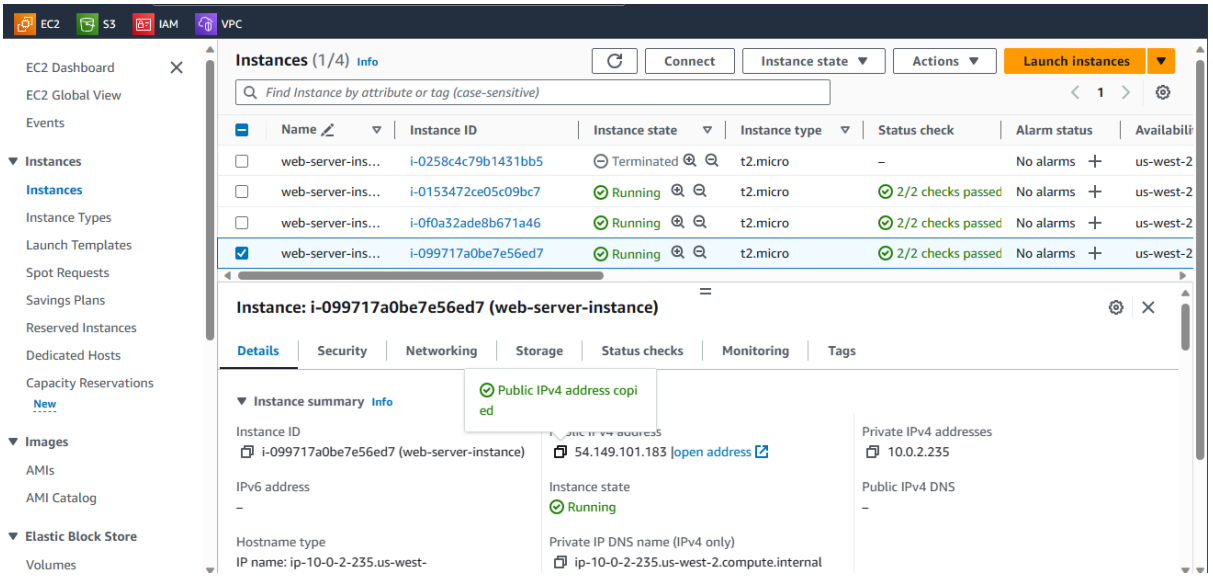
Terminated

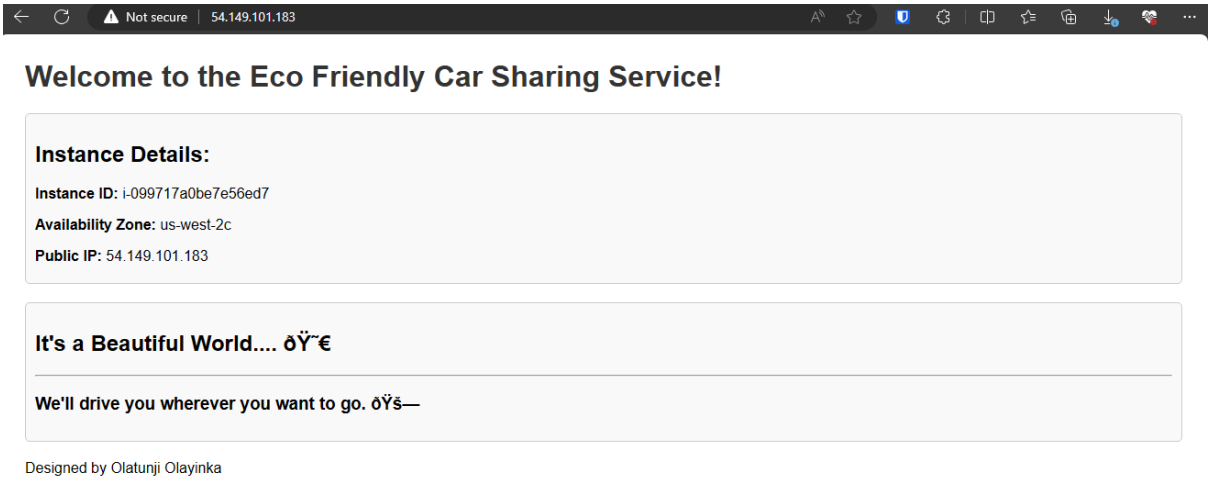


Now, we see that a new EC2 instance has been spun up and is initializing.



Then, after the EC2 instance that is being initialized has passed all status checks, you can access the web page hosted on it via it public ip also.





What is left for us to do now is to cleanup or de-provision our infrastructure by using the “terraform destroy” command at the prompt. This will literally destroy all the resources that were created by a particular Terraform configuration that we had in the “main.tf” file. Never forget this if you don’t want to wake up to crazy bills…😀

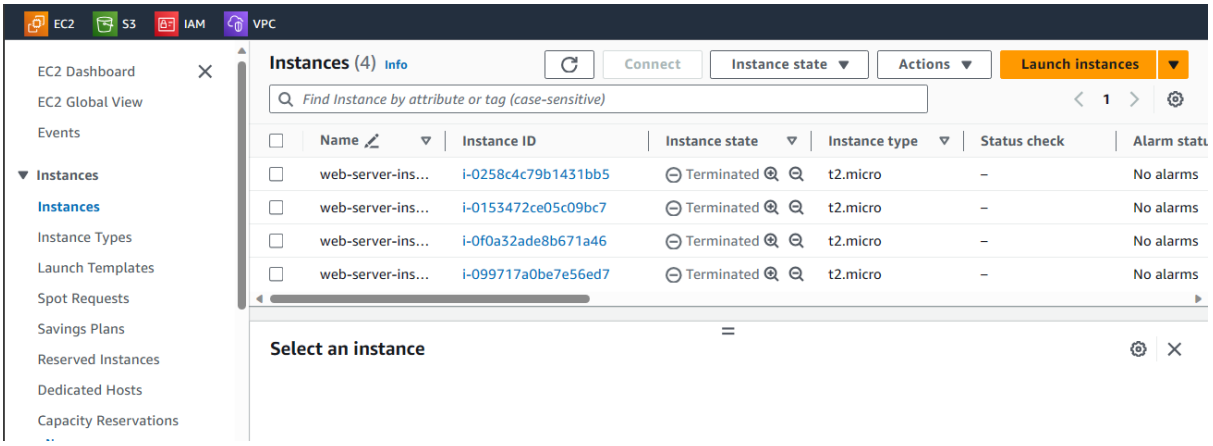
And if you don’t want terraform to be asking you silly questions like “Do you really want to delete these resources” or something like that (assuming that you know exactly what you are doing) you can include the “-auto-approve” flag to disable that feature.

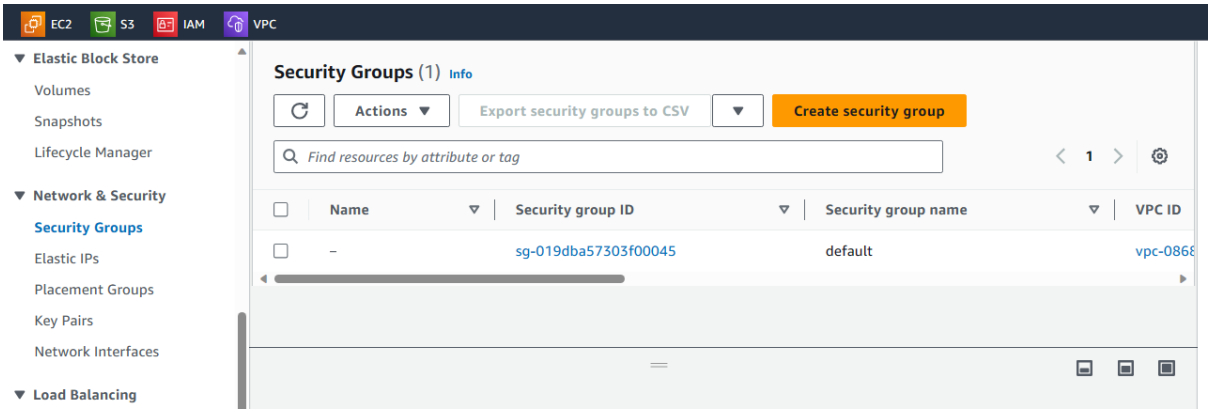
```
PS C:\Users\Olatunji Olayinka\Documents\FormerHDD\Programming\Node-React-NPM&More\IntroToNode&Express\AzubiParole\Azubi2023Work-CloudClass\AzubiProj
ects\week14_projects\ExerciseDm\eco-friendly-fleet-management> terraform destroy -auto-approve
aws_vpc.vpc: Refreshing state... [id=vpc-004eb471d123ce8cf]
aws_internet_gateway.igw: Refreshing state... [id=igw-097fa06b34db31ce6]
aws_subnet.subnet_az1: Refreshing state... [id=subnet-0881c1224df877e17]
aws_subnet.subnet_az2: Refreshing state... [id=subnet-05935dffb4f387116]
aws_lb_target_group.target_group: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:167834609858:targetgroup/eco-friendly-target-group/
6871dc1cd77a16a6]
aws_security_group.sg: Refreshing state... [id=sg-02858e6d74a8d0b5a]
aws_launch_configuration.launch_config: Refreshing state... [id=eco-friendly-launch-config]
aws_route_table.custom_route_table: Refreshing state... [id=rtb-084eb0bcaccb751d5]
aws_lb.lb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:167834609858:loadbalancer/app/eco-friendly-lb/55ae12f546bc425b]
aws_route_table_association.subnet_az2_assoc: Refreshing state... [id=rtbassoc-0ecebd8dce08fcf7a]
aws_route_table_association.subnet_az1_assoc: Refreshing state... [id=rtbassoc-0aa6556691637eaa7]
aws_lb_listener.listener: Refreshing state... [id=arn:aws:elasticloadbalancing:us-west-2:167834609858:listener/app/eco-friendly-lb/55ae12f546bc425b/
f1a38a7d6b8885f3]
aws_autoscaling_group.asg: Refreshing state... [id=terraform-20231119134901499300000001]
```

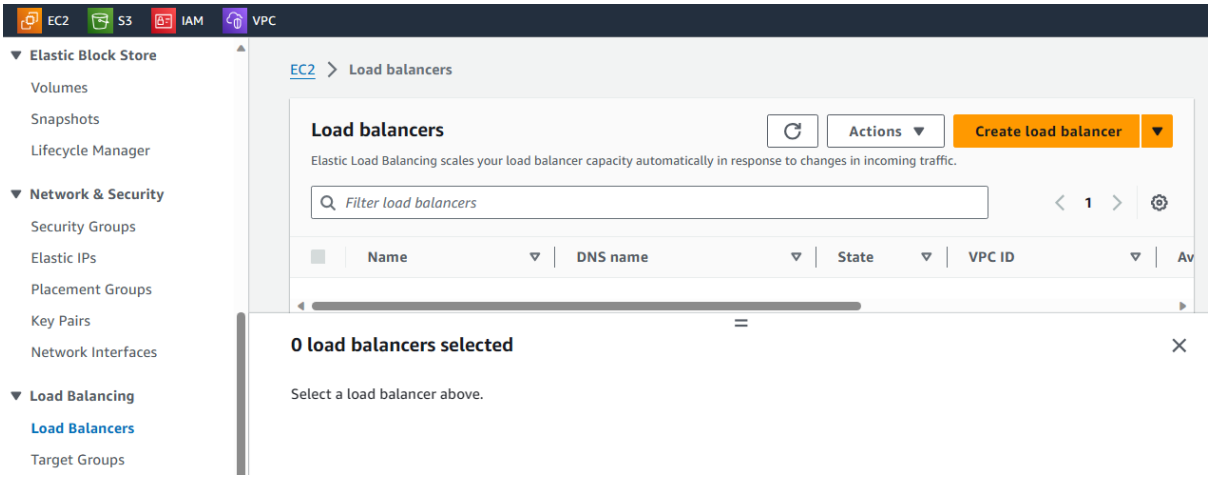
Once all resources have been completely deleted, you will see the message “Destroy complete! Resources: 13 destroyed.”

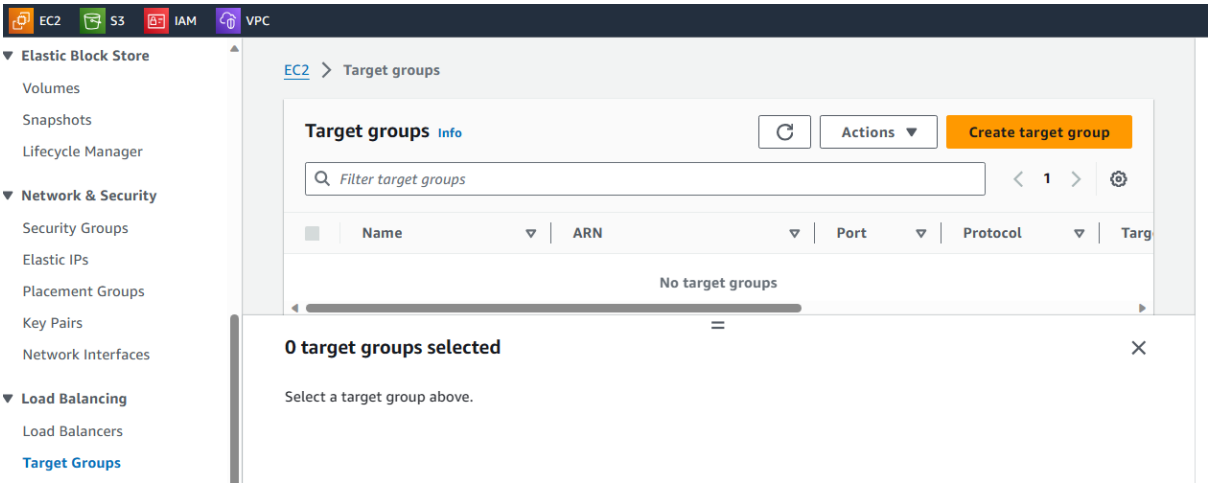
```
aws_route_table_association.subnet_az1_assoc: Destruction complete after 2s
aws_route_table.custom_route_table: Destroying... [id=rtb-084eb0bcaccb751d5]
aws_lb_listener.listener: Destruction complete after 3s
aws_lb_target_group.target_group: Destroying... [id=arn:aws:elasticloadbalancing:us-west-2:167834609858:targetgroup/eco-friendly-target-group/6871dc
1cd77a16a6]
aws_lb.lb: Destroying... [id=arn:aws:elasticloadbalancing:us-west-2:167834609858:loadbalancer/app/eco-friendly-lb/55ae12f546bc425b]
aws_lb_target_group.target_group: Destruction complete after 1s
aws_route_table.custom_route_table: Destruction complete after 2s
aws_internet_gateway.igw: Destroying... [id=igw-097fa06b34db31ce6]
aws_lb.lb: Destruction complete after 5s
aws_autoscaling_group.asg: Still destroying... [id=terraform-20231119134901499300000001, 10s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-097fa06b34db31ce6, 10s elapsed]
aws_autoscaling_group.asg: Still destroying... [id=terraform-20231119134901499300000001, 20s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-097fa06b34db31ce6, 20s elapsed]
aws_autoscaling_group.asg: Still destroying... [id=terraform-20231119134901499300000001, 30s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-097fa06b34db31ce6, 30s elapsed]
aws_autoscaling_group.asg: Still destroying... [id=terraform-20231119134901499300000001, 40s elapsed]
aws_internet_gateway.igw: Destruction complete after 36s
aws_autoscaling_group.asg: Still destroying... [id=terraform-20231119134901499300000001, 50s elapsed]
aws_autoscaling_group.asg: Still destroying... [id=terraform-20231119134901499300000001, 1m0s elapsed]
aws_autoscaling_group.asg: Destruction complete after 1m9s
aws_subnet.subnet_az1: Destroying... [id=subnet-0881c1224df877e17]
aws_launch_configuration.launch_config: Destroying... [id=eco-friendly-launch-config]
aws_subnet.subnet_az2: Destroying... [id=subnet-05935dffb4f387116]
aws_launch_configuration.launch_config: Destruction complete after 0s
aws_security_group.sg: Destroying... [id=sg-02858e6d74a8d0b5a]
aws_subnet.subnet_az2: Destruction complete after 3s
aws_subnet.subnet_az1: Destruction complete after 3s
aws_security_group.sg: Destruction complete after 3s
aws_vpc.vpc: Destroying... [id=vpc-004eb471d123ce8cf]
aws_vpc.vpc: Destruction complete after 2s

Destroy complete! Resources: 13 destroyed.
PS C:\Users\Olatunji Olayinka\Documents\FormerHDD\Programming\Node-React-NPM&More\IntroToNode&Express\AzubiParole\Azubi2023Work-CloudClass\AzubiProj
ects\week14_projects\ExerciseDm\eco-friendly-fleet-management>
```

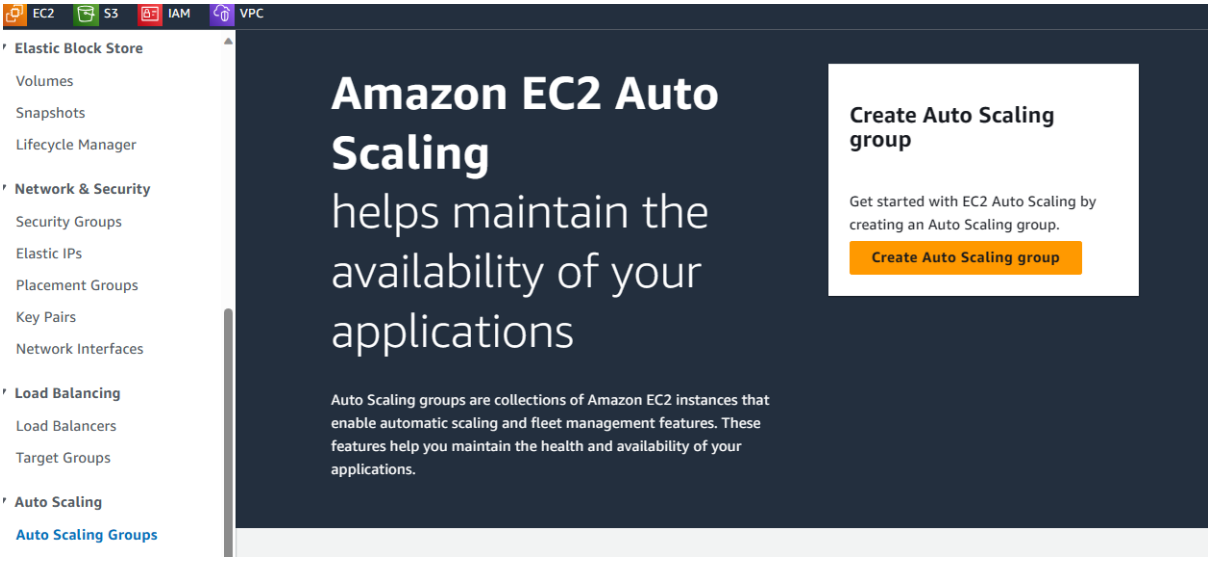
If you now go around your AWS console and check all the resources that were provisioned before, you will discover that they have all been de-provisioned now. EC2 instances have been terminated, Security Groups have been removed, Load Balancer has been removed, Target group has been deleted, Auto Scaling group has been removed too.


EC2
S3
IAM
VPC
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Instances (4) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
Name
Instance ID
Instance state
Instance type
Status check
Alarm statu
web-server-ins... i-0258c4c79b1431bb5 Terminated t2.micro – No alarms
web-server-ins... i-0153472ce05c09bc7 Terminated t2.micro – No alarms
web-server-ins... i-0f0a32ade8b671a46 Terminated t2.micro – No alarms
web-server-ins... i-099717a0be7e56ed7 Terminated t2.micro – No alarms
Select an instance


EC2
S3
IAM
VPC
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Security Groups (1) Info
Actions
Export security groups to CSV
Create security group
Find resources by attribute or tag
Name
Security group ID
Security group name
VPC ID
– sg-019dba57303f00045 default vpc-0868


EC2
S3
IAM
VPC
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
EC2 > Load balancers
Load balancers
Actions
Create load balancer
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Filter load balancers
Name
DNS name
State
VPC ID
0 load balancers selected
Select a load balancer above.





With this project, an illustration of the power of Autoscaling Groups and Load Balancers is achieved. And the fact that we were able to spin all these resources up effortlessly within a very short time is truly phenomenal, as it demonstrates the importance of Terraform in being able to provision and deprovision resources on the could quickly, thereby bringing immense possibilities to business applications.

And you can view the associated GitHub here - Eco Friendly Car Service