Bachelor of Computing and Network Communications (Honours)

AWS & Terraform High Availability Infrastructure

Chance Page

Cloud Systems
SYST 35144
Professor. Felix Carapaica
April 6th, 2024

Table of Contents

1. VPC Module:

Code:

The VPC module defines a VPC with a specific CIDR block, DNS support, DNS host names, and a name tag. It also defines an Internet Gateway attached to the VPC, subnets within the VPC, a public route table, and associations between the route table and the subnets.

Variables:

The variables for the VPC module include the VPC CIDR block, the VPC name, the subnet CIDR blocks, and the availability zones.

Outputs:

The outputs for the VPC module are the VPC ID and the subnet IDs. These outputs are consumed by the High Availability Infrastructure module and the Security Group module. Specifically:

•The VPC ID is used by the Security Group module to create a security group within the VPC.

•The subnet IDs are used by the High Availability Infrastructure module to specify the subnets for the auto-scaling group and the load balancer.

```
#VPC main.tf

// Define the VPC
resource "aws_vpc" "vpc" {
  cidr_block           = var.vpc_cidr // The CIDR block for the VPC
  enable_dns_support   = true // Enable DNS support
  enable_dns_hostnames = true // Enable DNS hostnames
  tags                 = { Name = var.vpc_name } // Tags for the VPC
}

// Define the Internet Gateway
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.vpc.id // The ID of the VPC
  tags   = { Name = "igw-${var.vpc_name}" } // Tags for the Internet Gateway
}

// Define the Subnet
resource "aws_subnet" "subnet" {
  count                   = length(var.subnet_cidrs) // The number of subnets
  vpc_id                  = aws_vpc.vpc.id // The ID of the VPC
  cidr_block              = var.subnet_cidrs[count.index] // The CIDR block for the subnet
  availability_zone       = var.availability_zones[count.index] // The availability zone
  map_public_ip_on_launch = true // Enable public IP on launch

  tags = {
    Name = "${var.vpc_name}-SN-${count.index + 1}" // Tags for the subnet
  }
}

// Define the Route Table
resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.vpc.id // The ID of the VPC

  route {
    cidr_block = "0.0.0.0/0" // The CIDR block
    gateway_id = aws_internet_gateway.igw.id // The ID of the Internet Gateway
  }

  tags = { Name = "${var.vpc_name}-publicRT" } // Tags for the Route Table
}

// Define the Route Table Association
resource "aws_route_table_association" "public_ra" {
  count          = length(var.subnet_cidrs) // The number of associations
  subnet_id      = aws_subnet.subnet[count.index].id // The ID of the subnet
  route_table_id = aws_route_table.public_rt.id // The ID of the Route Table
}

// Output the VPC ID
output "vpc_id" {
  value       = aws_vpc.vpc.id
}

// Output the Subnet IDs
output "subnet_ids" {
  value       = aws_subnet.subnet[*].id
}
```

2. Security Group Module:

Code:

• The Security Group module creates a new AWS Security Group, allows SSH (port 22) and HTTP (port 80) inbound traffic from all IP addresses, allows all outbound traffic, and outputs the ID of the created security group.

Variables:

• The variables for the Security Group module would include the VPC IDs.

Outputs:

•The outputs for the Security Group module are the security group IDs. These outputs are consumed by the High Availability Infrastructure module. Specifically:

•The security group ID is used by the High Availability Infrastructure module to specify the security group for the launch template, the auto-scaling group, and the load balancer.

```
#Security Groups main.tf

// Create a new AWS Security Group
resource "aws_security_group" "sgtf" {
  count  = length(var.vpc_ids)
  name   = "sgtf_${count.index}"
  vpc_id = var.vpc_ids[count.index]
  tags   = { Name = "sgtf_${count.index}" }
}

/// Allow SSH (port 22) inbound traffic from all IP addresses
resource "aws_vpc_security_group_ingress_rule" "allow-ssh" {
  count = length(aws_security_group.sgtf)
  security_group_id = aws_security_group.sgtf[count.index].id
  cidr_ipv4 = "0.0.0.0/0"
  from_port = 22
  to_port = 22
  ip_protocol = "tcp"
}

// Allow HTTP (port 80) inbound traffic from all IP addresses
resource "aws_vpc_security_group_ingress_rule" "allow-http" {
  count = length(aws_security_group.sgtf)
  security_group_id = aws_security_group.sgtf[count.index].id
  cidr_ipv4 = "0.0.0.0/0"
  from_port = 80
  to_port = 80
  ip_protocol = "tcp"
}

// Allow all outbound traffic
resource "aws_security_group_rule" "allow_out" {
  count = length(aws_security_group.sgtf)
  security_group_id = aws_security_group.sgtf[count.index].id
  type        = "egress"
  from_port   = 0
  to_port     = 0
  protocol    = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}


// Output the ID of the created security group
output "sgtf_security_ids" {
  value = aws_security_group.sgtf[*].id
}
```

3. High Availability Infrastructure Module:

Code:

• The High Availability Infrastructure module defines a launch template, an auto-scaling group, a load balancer, a listener, and a target group. It takes in the security group ID, the VPC ID, and the subnet IDs from the VPC module.

Variables:

• The variables for the High Availability Infrastructure module would include the security group ID, the VPC ID, and the subnet IDs.

Outputs:

• The outputs for the High Availability Infrastructure module are the DNS names of the load balancers. These outputs can be consumed by other services that need to route traffic to the load balancers. For example, a DNS service could use these DNS names to create DNS records that point to the load balancers.

```
# High Avail main.tf

# Define the Launch Template
resource "aws launch template" "webtest" {
  name                   = "${var.launch-template}-${var.vpc name}" // The name of the launch template
  image id               = data.aws ami.web-aurora.id // The ID of the AMI
  instance type          = var.chassis // The instance type
  vpc security group ids = [var.sgid] // The ID of the security group
  tag specifications {
    resource type = "instance" // The resource type
    tags          = { Name = "webtest-${var.vpc name}" } // The tags for the instance
  }
  lifecycle { create before destroy = true } // Lifecycle policy
}

# Define the Auto Scaling Group
resource "aws autoscaling group" "asg" {
  launch template {
    id      = aws launch template.webtest.id // The ID of the launch template
    version = "$Latest" // The version of the launch template
  }
  vpc zone identifier = var.subnetids // The ID of the VPC zone
  target group arns   = [aws lb target group.asg-tg.arn] // The ARN of the target group
  health check type   = "ELB" // The type of health check
  min size            = 2 // The minimum size of the group
  desired capacity    = 3 // The desired capacity of the group
  max size            = 3 // The maximum size of the group
  tag {
    key                 = "Name" // The key of the tag
    value               = "asg-${var.vpc name}" // The value of the tag
    propagate at launch = true // Whether to propagate the tag at launch
  }
}

# Define the Load Balancer
resource "aws lb" "elb-tf" {
  name               = "elb-tf-${var.vpc name}" // The name of the load balancer
  load balancer type = "application" // The type of load balancer
  subnets            = var.subnetids // The subnets for the load balancer
  security groups    = [var.sgid] // The security groups for the load balancer
}

# Define the Listener
resource "aws lb listener" "http" {
  load balancer arn = aws lb.elb-tf.arn // The ARN of the load balancer
  port              = 80 // The port for the listener
  protocol          = "HTTP" // The protocol for the listener
  default action {
    type = "fixed-response" // The type of action
    fixed response {
      content type = "text/plain" // The content type of the response
      message body = "200, OK" // The body of the response
      status code  = 200 // The status code of the response
    }
  }
}

# Define the Target Group
resource "aws lb target group" "asg-tg" {
  name     = "asg-tg-${var.vpc name}" // The name of the target group
  port     = 80 // The port for the target group
  protocol = "HTTP" // The protocol for the target group
  vpc id   = var.vpcid // The ID of the VPC
  health check {
    path                = "/" // The path for the health check
    protocol            = "HTTP" // The protocol for the health check
    matcher             = "200" // The matcher for the health check
    interval            = 15 // The interval for the health check
    timeout             = 3 // The timeout for the health check
    healthy threshold   = 2 // The healthy threshold for the health check
    unhealthy threshold = 2 // The unhealthy threshold for the health check
  }
}

# Define the Listener Rule
resource "aws lb listener rule" "asg-listen" {
  listener arn = aws lb listener.http.arn // The ARN of the listener
  priority     = 100 // The priority of the rule
  condition {
    path pattern {
      values = ["*"] // The values for the path pattern
    }
  }
  action {
    type             = "forward" // The type of action
    target group arn = aws lb target group.asg-tg.arn // The ARN of the target group
  }
}

# Output the DNS name of the Load Balancer
output "alb dns name" {
  value       = aws lb.elb-tf.dns name
}
```

4. Blue and Green VPCs:

Code:

• The code in main-root defines two separate VPC's, “vpcBlue” and “vpcGreen”, each with their own CIDR blocks, subnet CIDR blocks, and availability zones. It also defines two High Availability Infrastructure modules, one for each VPC.

Variables:

• The variables for each VPC include the VPC CIDR block, the VPC name, the subnet CIDR blocks, and the availability zones.

Outputs:

• The outputs for each VPC would be the VPC ID and the subnet IDs. These outputs can be consumed by the Security Group module and the High Availability Infrastructure module in the same way as described above.

```
#main-root.tf

# Define the Blue VPC module
module "vpcBlue" {
  source = "./VPC" // The source of the module
  vpc_cidr = "100.64.0.0/16" // The CIDR block for the VPC
  vpc_name = "vpcBlue" // The name of the VPC
  subnet_cidrs = ["100.64.0.0/24", "100.64.1.0/24", "100.64.2.0/24"] // The CIDR blocks for the subnets
  availability_zones = ["us-east-1a", "us-east-1b", "us-east-1c"] // The availability zones
}

# Define the Blue High Availability module
module "HighAvailBlue" {
    source = "./HighAvail" // The source of the module
    sgid = module.SecurityGroups.sgtf_security_ids[0] // The ID of the security group
    vpcid = module.vpcBlue.vpc_id // The ID of the VPC
    subnetids = module.vpcBlue.subnet_ids // The IDs of the subnets
    vpc_name = "vpcBlue" // The name of the VPC
}

# Define the Green VPC module
module "vpcGreen" {
  source = "./VPC" // The source of the module
  vpc_cidr = "192.168.0.0/16" // The CIDR block for the VPC
  vpc_name = "vpcGreen" // The name of the VPC
  subnet_cidrs = ["192.168.0.0/24", "192.168.1.0/24", "192.168.2.0/24"] // The CIDR blocks for the subnets
  availability_zones = ["us-east-1a", "us-east-1b", "us-east-1c"] // The availability zones
}

# Define the Green High Availability module
module "HighAvailGreen" {
    source = "./HighAvail" // The source of the module
    sgid = module.SecurityGroups.sgtf_security_ids[1] // The ID of the security group
    vpcid = module.vpcGreen.vpc_id // The ID of the VPC
    subnetids = module.vpcGreen.subnet_ids // The IDs of the subnets
    vpc_name = "vpcGreen" // The name of the VPC
}

#Define the SecurityGroups module
module "SecurityGroups" {
    source = "./SecurityGroups" // The source of the module
    vpc_ids = [module.vpcBlue.vpc_id, module.vpcGreen.vpc_id] // The VPC ids being passed into the security group.
}

output "HighAvailLoadBalancerDNS" {
    value = {
        "Green" = module.HighAvailGreen.alb_dns_name
        "Blue" = module.HighAvailBlue.alb_dns_name
    }
}
```

5. Screenshots

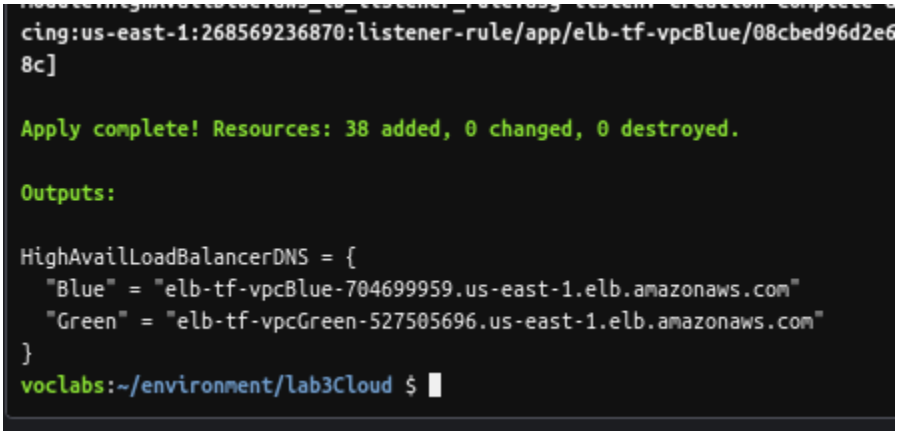


Figure 1: This is the terraform successful output. Also outputs the DNS names for each load balancer.

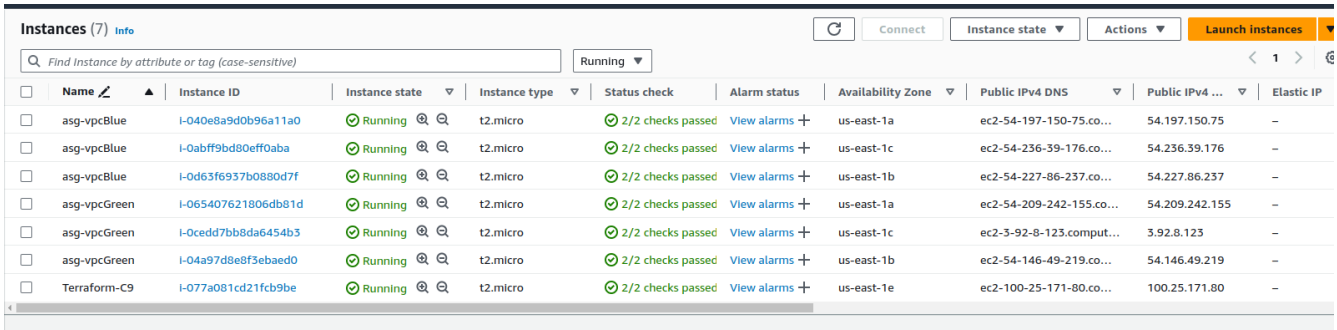


Figure 2: My instances showing a maximum of 3 per each VPC.

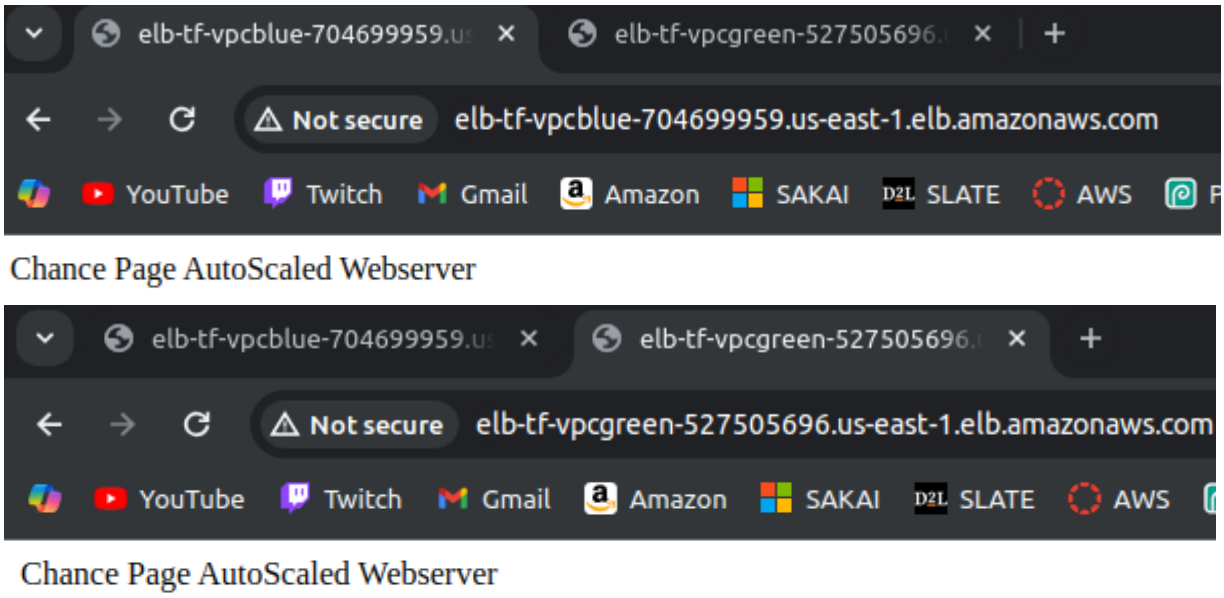


Figure 3: Load balancer DNS is redirecting me to the EC2's webpage.

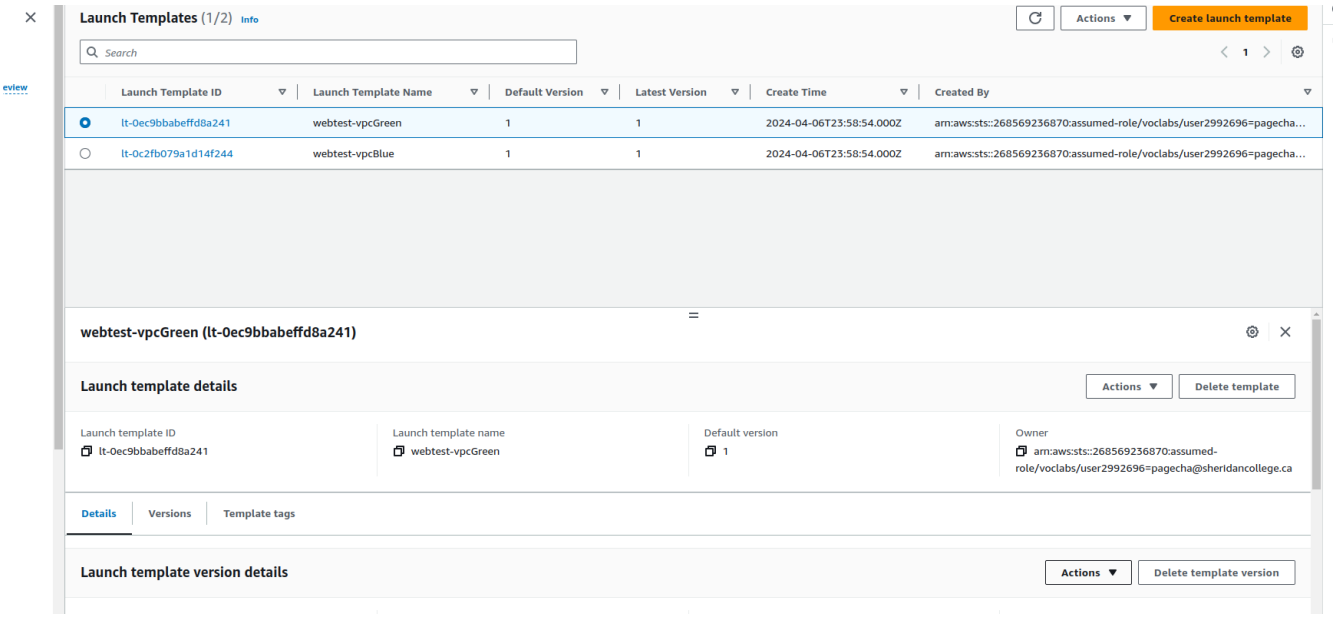


Figure 4: These are my 2 launch templates created by each High Availability Cluster

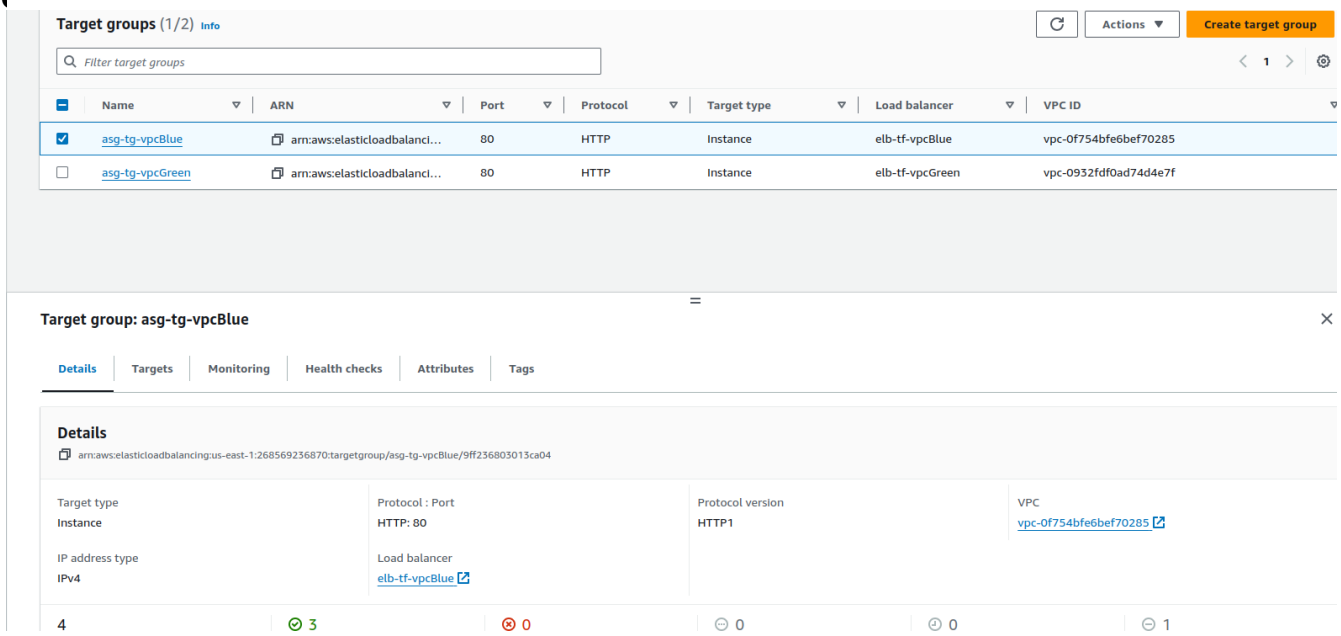


Figure 5: The target groups for each Load balancer pointing to specified EC2's.

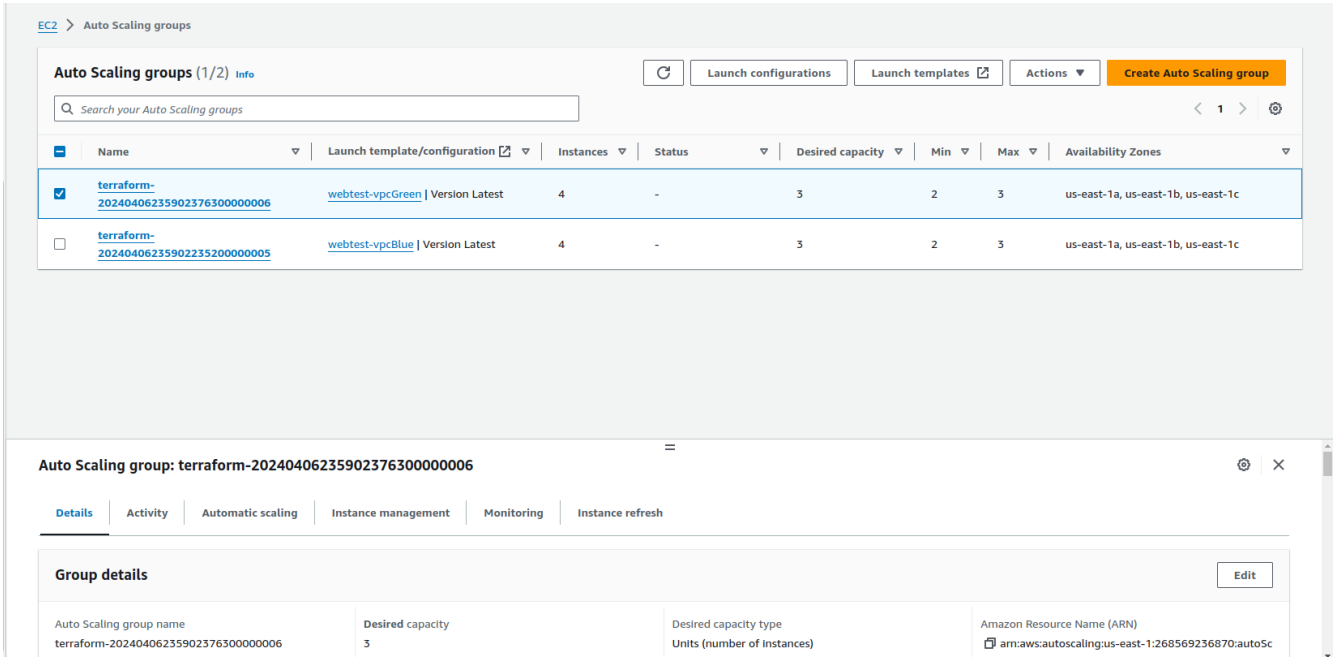


Figure 6: Here is each auto scaling group and you can see the specified min and maximums.

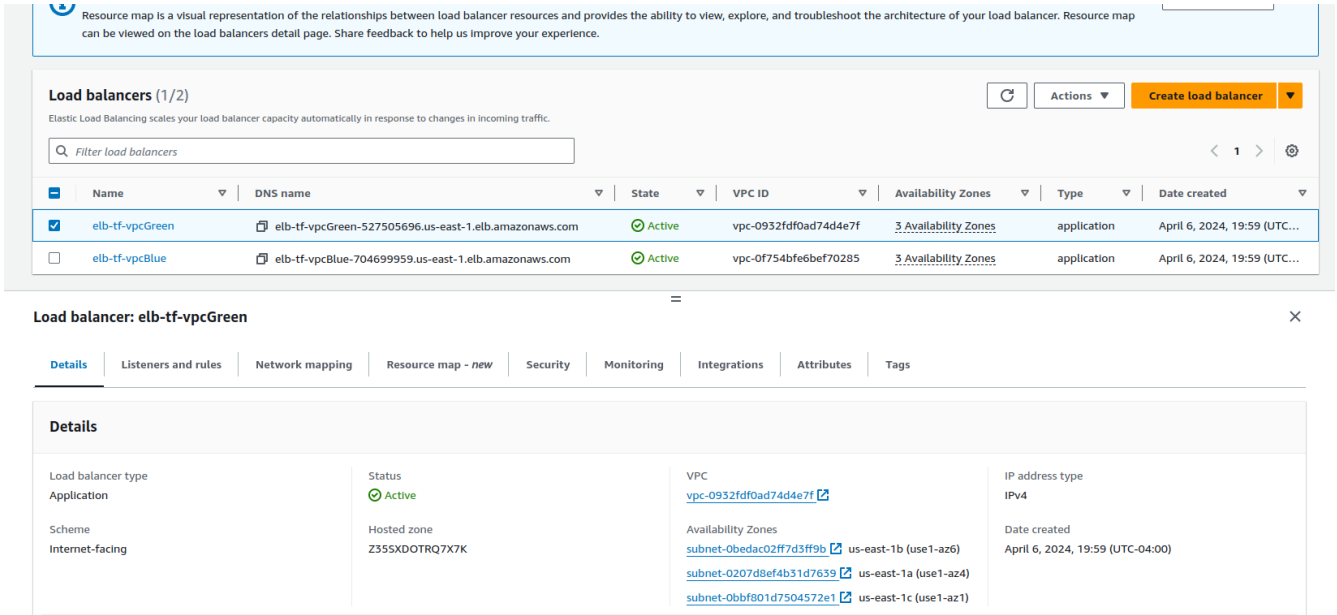


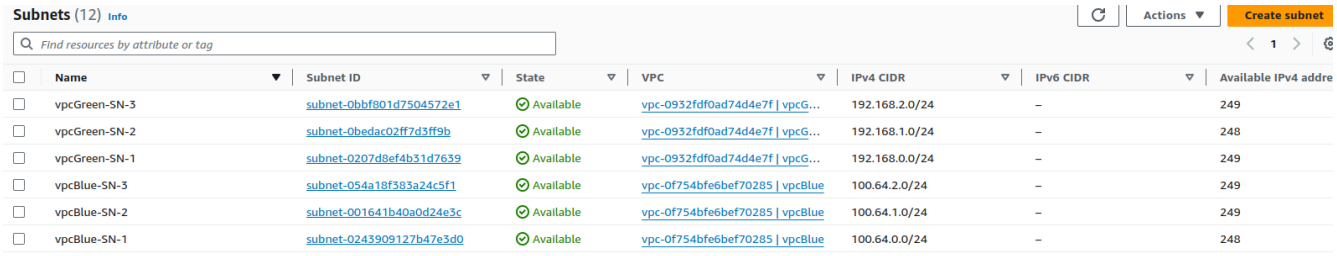
Figure 7: We can see the 2 load balancers and what their DNS names are as well as what vpc they occupy.

sgtf_1	sg-07580b40c9d06bac3	sgtf_1	vpc-0932fdf0ad74d4e7f	Managed by Terraform	268569236870
sgtf_0	sg-04ed09276d0dcec43	sgtf_0	vpc-0f754bfe6bef70285	Managed by Terraform	268569236870

Figure 8: These are the 2 security groups made for each VPC.

vpcGreen	vpc-0932fdf0ad74d4e7f	Available	192.168.0.0/16	–	dopt-03cb7abed8a48a1...	rtb-021340c9129779b76
vpcBlue	vpc-0f754bfe6bef70285	Available	100.64.0.0/16	–	dopt-03cb7abed8a48a1...	rtb-0d567f4c99e51ae76

Figure 9: These are the 2 created VPC's BLUE and GREEN.



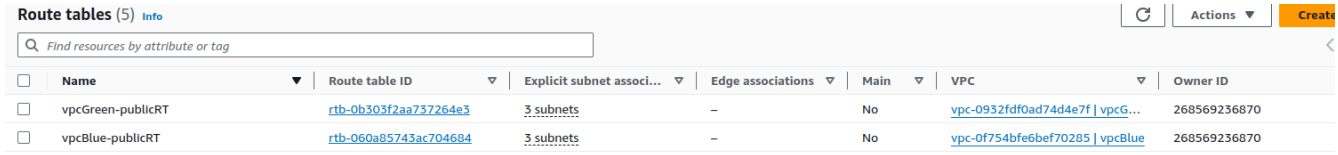
Subnets (12) Info

Actions | Create subnet

Find resources by attribute or tag

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv4 addre
vpcGreen-SN-3	subnet-0bbf801d7504572e1	Available	vpc-0932fdf0ad74d4e7f | vpcG...	192.168.2.0/24	–	249
vpcGreen-SN-2	subnet-0bedac02ff7d3ff9b	Available	vpc-0932fdf0ad74d4e7f | vpcG...	192.168.1.0/24	–	248
vpcGreen-SN-1	subnet-0207d8ef4b31d7639	Available	vpc-0932fdf0ad74d4e7f | vpcG...	192.168.0.0/24	–	249
vpcBlue-SN-3	subnet-054a18f383a24c5f1	Available	vpc-0f754bfe6bef70285 | vpcBlue	100.64.2.0/24	–	249
vpcBlue-SN-2	subnet-001641b40a0d24e3c	Available	vpc-0f754bfe6bef70285 | vpcBlue	100.64.1.0/24	–	249
vpcBlue-SN-1	subnet-0243909127b47e3d0	Available	vpc-0f754bfe6bef70285 | vpcBlue	100.64.0.0/24	–	248

Figure 10: All 6 subnets are created and in the correct CIDR blocks as well as correct VPC's.



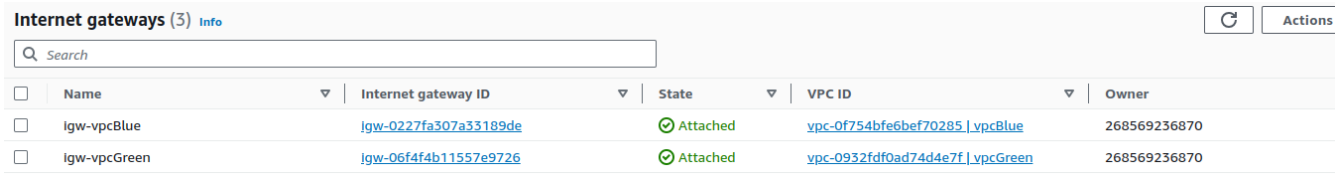
Route tables (5) Info

Actions | Create

Find resources by attribute or tag

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC	Owner ID
vpcGreen-publicRT	rtb-0b303f2aa737264e3	3 subnets	–	No	vpc-0932fdf0ad74d4e7f | vpcG...	268569236870
vpcBlue-publicRT	rtb-060a85743ac704684	3 subnets	–	No	vpc-0f754bfe6bef70285 | vpcBlue	268569236870

Figure 11: Here are the 2 public routing tables for each VPC, specifying that they can default route to anywhere through the internet gateway.



Internet gateways (3) Info

Actions

Search

Name	Internet gateway ID	State	VPC ID	Owner
igw-vpcBlue	igw-0227fa307a33189de	Attached	vpc-0f754bfe6bef70285 | vpcBlue	268569236870
igw-vpcGreen	igw-06f4f4b11557e9726	Attached	vpc-0932fdf0ad74d4e7f | vpcGreen	268569236870

Figure 12: These are the 2 internet gateways being used for each VPC.