```
    propagate_at_launch = true
  }
}
```

Deploying a Load Balancer

At this point, you can deploy your ASG, but you'll have a small problem: you now have multiple servers, each with its own IP address, but you typically want to give your end users only a single IP to use. One way to solve this problem is to deploy a *load balancer* to distribute traffic across your servers and to give all your users the IP (actually, the DNS name) of the load balancer. Creating a load balancer that is highly available and scalable is a lot of work. Once again, you can let AWS take care of it for you, this time by using Amazon's *Elastic Load Balancer* (ELB) service, as shown in Figure 2-10.

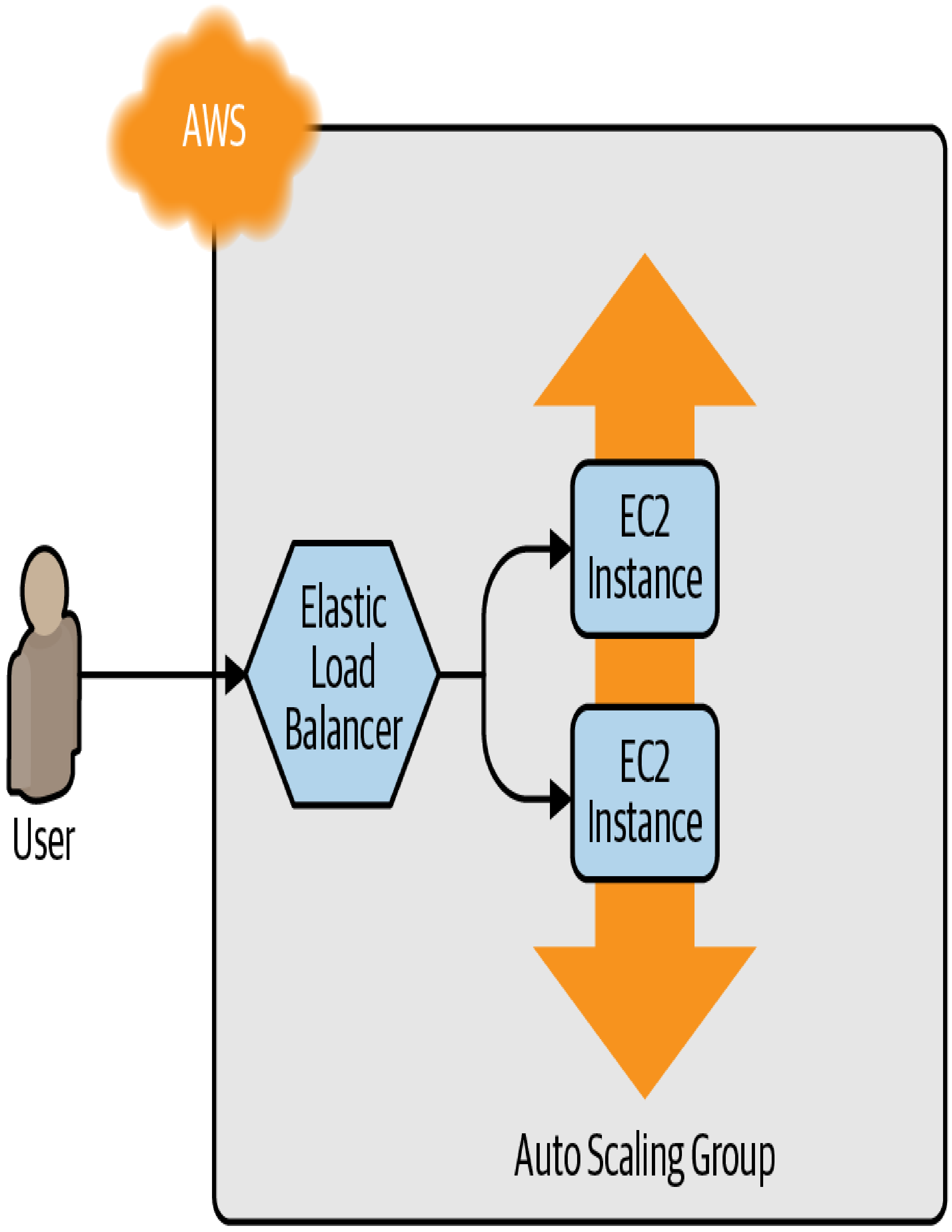


Figure 2-10. Use Amazon ELB to distribute traffic across the Auto Scaling Group.

AWS offers three types of load balancers:

Application Load Balancer (ALB)

Best suited for load balancing of HTTP and HTTPS traffic. Operates at the application layer (Layer 7) of the Open Systems Interconnection (OSI) model.

Network Load Balancer (NLB)

Best suited for load balancing of TCP, UDP, and TLS traffic. Can scale up and down in response to load faster than the ALB (the NLB is designed to scale to tens of millions of requests per second). Operates at the transport layer (Layer 4) of the OSI model.

Classic Load Balancer (CLB)

This is the "legacy" load balancer that predates both the ALB and NLB. It can handle HTTP, HTTPS, TCP, and TLS traffic but with far fewer features than either the ALB or NLB. Operates at both the application layer (L7) and transport layer (L4) of the OSI model.

Most applications these days should use either the ALB or the NLB. Because the simple web server example you're working on is an HTTP app without any extreme performance requirements, the ALB is going to be the best fit.

The ALB consists of several parts, as shown in Figure 2-11:

Listener

Listens on a specific port (e.g., 80) and protocol (e.g., HTTP).

Listener rule

Takes requests that come into a listener and sends those that match specific paths (e.g., `/foo` and `/bar`) or hostnames (e.g., `foo.example.com` and `bar.example.com`) to specific target groups.

Target groups

One or more servers that receive requests from the load balancer. The target group also performs health checks on these servers and sends requests only to healthy nodes.

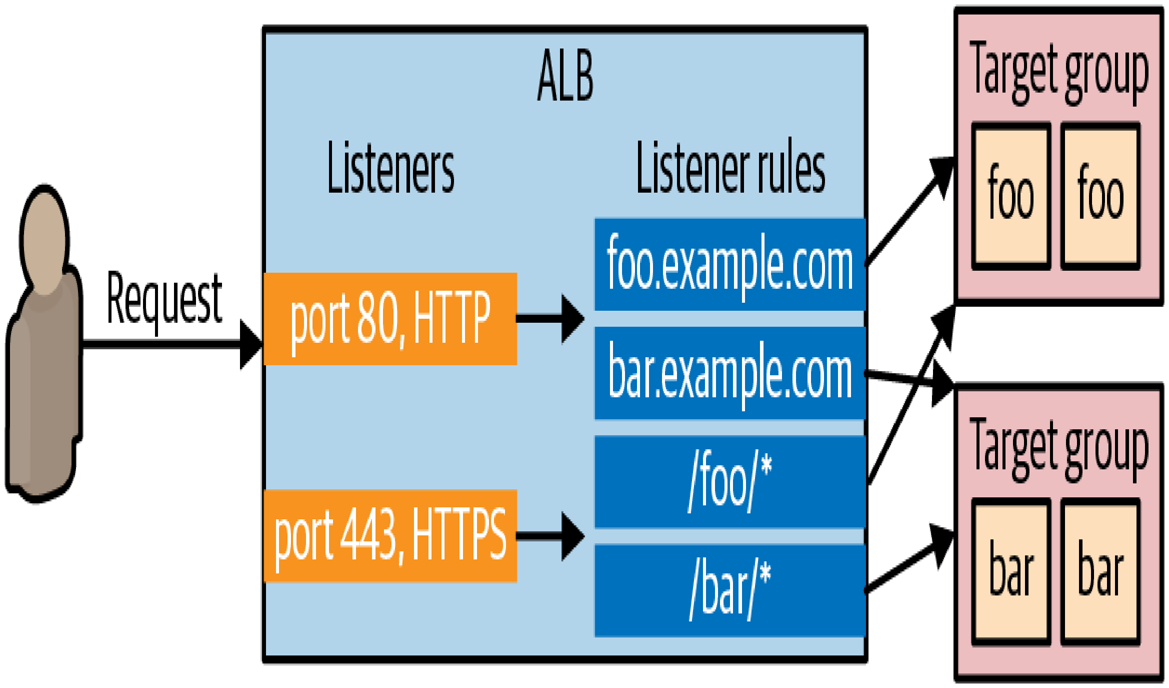


Figure 2-11. An ALB consists of listeners, listener rules, and target groups.

The first step is to create the ALB itself using the `aws_lb` resource:

```
resource "aws_lb" "example" {
  name               = "terraform-asg-example"
  load_balancer_type = "application"
  subnets            = data.aws_subnets.default.ids
}
```

Note that the `subnets` parameter configures the load balancer to use all the subnets in your Default VPC by using the `aws_subnets` data source.[16] AWS load balancers don't consist of a single server, but of multiple servers that can run in separate subnets (and, therefore, separate datacenters). AWS automatically scales the number of load balancer servers up and down based on traffic and handles failover if one of those servers goes down, so you get scalability and high availability out of the box.

The next step is to define a listener for this ALB using the `aws_lb_listener` resource:

```
resource "aws_lb_listener" "http" {
  load_balancer_arn = aws_lb.example.arn
  port              = 80
  protocol          = "HTTP"

  # By default, return a simple 404 page
  default_action {
    type = "fixed-response"

    fixed_response {
      content_type = "text/plain"
      message_body = "404: page not found"
      status_code  = 404
    }
  }
}
```

This listener configures the ALB to listen on the default HTTP port, port 80, use HTTP as the protocol, and send a simple 404 page as the default response for requests that don't match any listener rules.

Note that, by default, all AWS resources, including ALBs, don't allow any incoming or outgoing traffic, so you need to create a new security group specifically for the ALB. This security group should allow incoming requests on port 80 so that you can access the load balancer over HTTP, and allow outgoing requests on all ports so that the load balancer can perform health checks:

```
resource "aws_security_group" "alb" {
  name = "terraform-example-alb"

  # Allow inbound HTTP requests
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
```

```
    # Allow all outbound requests
    egress {
      from_port   = 0
      to_port     = 0
      protocol    = "-1"
      cidr_blocks = ["0.0.0.0/0"]
    }
  }
```

You’ll need to tell the `aws_lb` resource to use this security group via the `security_groups` argument:

```
  resource "aws_lb" "example" {
    name               = "terraform-asg-example"
    load_balancer_type = "application"
    subnets            = data.aws_subnets.default.ids
    security_groups    = [aws_security_group.alb.id]
  }
```

Next, you need to create a target group for your ASG using the `aws_lb_target_group` resource:

```
  resource "aws_lb_target_group" "asg" {
    name     = "terraform-asg-example"
    port     = var.server_port
    protocol = "HTTP"
    vpc_id   = data.aws_vpc.default.id

    health_check {
      path                = "/"
      protocol            = "HTTP"
      matcher             = "200"
      interval            = 15
      timeout             = 3
      healthy_threshold   = 2
      unhealthy_threshold = 2
    }
  }
```

This target group will health check your Instances by periodically sending an HTTP request to each Instance and will consider the Instance “healthy” only if the Instance returns a response that matches the configured `matcher` (e.g., you can configure a matcher to look for a 200 OK response). If an Instance fails to respond, perhaps because that Instance has gone down or is overloaded, it will be marked as "unhealthy," and the target group will automatically stop sending traffic to it to minimize disruption for your users.

How does the target group know which EC2 Instances to send requests to? You could attach a static list of EC2 Instances to the target group using the `aws_lb_target_group_attachment` resource, but with an ASG, Instances can launch or terminate at any time, so a static list won't work. Instead, you can take advantage of the first-class integration between the ASG and the ALB. Go back to the `aws_autoscaling_group` resource, and set its `target_group_arns` argument to point at your new target group:

```
resource "aws_autoscaling_group" "example" {
  launch_configuration = aws_launch_configuration.example.name
  vpc_zone_identifier  = data.aws_subnets.default.ids

  target_group_arns = [aws_lb_target_group.asg.arn]
  health_check_type = "ELB"

  min_size = 2
  max_size = 10

  tag {
    key                 = "Name"
    value               = "terraform-asg-example"
    propagate_at_launch = true
  }
}
```

You should also update the `health_check_type` to `"ELB"`. The default `health_check_type` is `"EC2"`, which is a minimal health check that considers an Instance unhealthy only if the AWS hypervisor says the VM is completely down or unreachable. The `"ELB"` health check is more robust, because it instructs the ASG to use the target group's health check to determine whether an Instance is healthy and to automatically replace Instances if the target group reports them as unhealthy. That way, Instances will be replaced not only if they are completely down but also if, for example, they've stopped serving requests because they ran out of memory or a critical process crashed.

Finally, it's time to tie all these pieces together by creating listener rules using the `aws_lb_listener_rule` resource:

```
resource "aws_lb_listener_rule" "asg" {
  listener_arn = aws_lb_listener.http.arn
  priority     = 100

  condition {
    path_pattern {
      values = ["*"]
    }
  }

  action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.asg.arn
  }
}
```

The preceding code adds a listener rule that sends requests that match any path to the target group that contains your ASG.

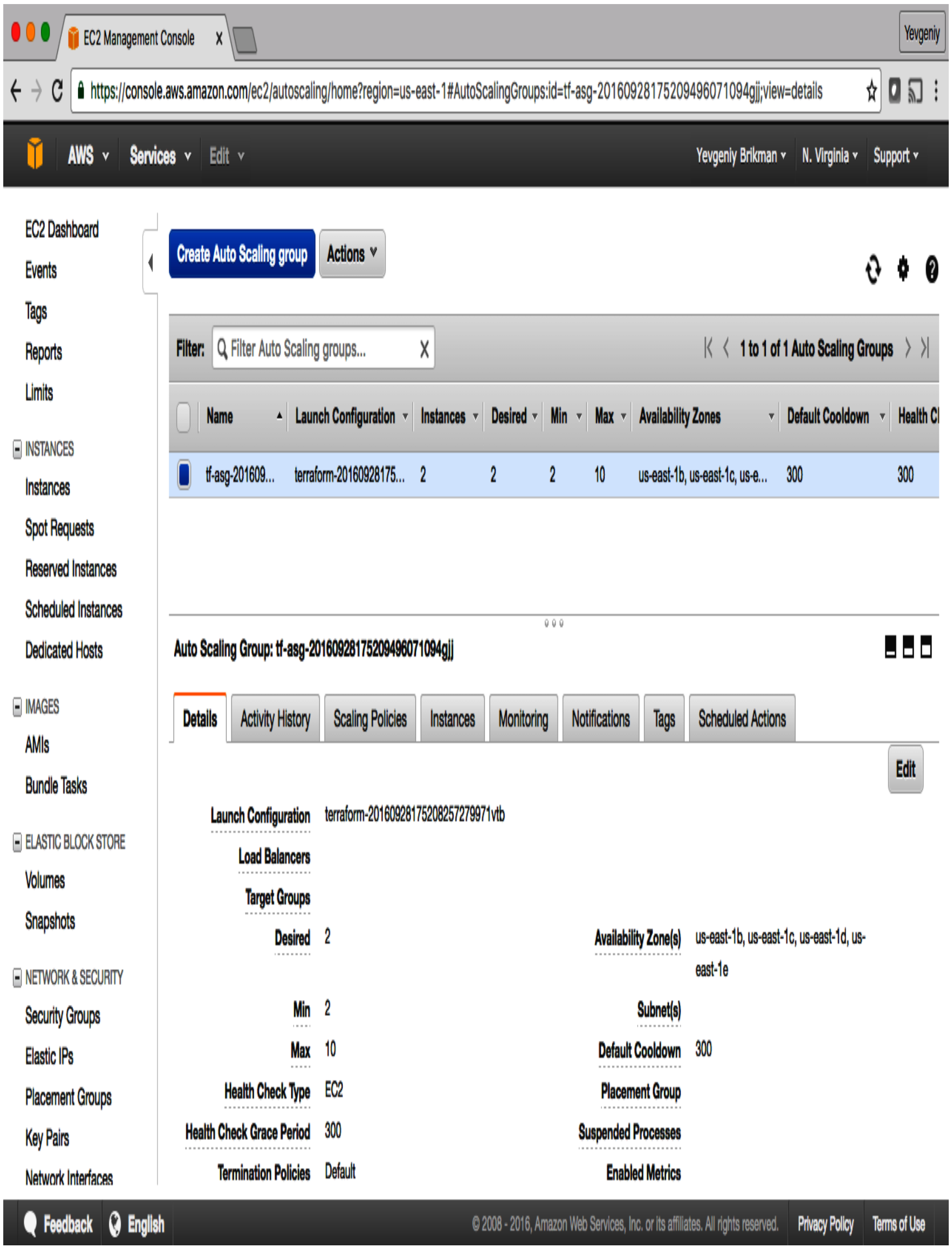
There's one last thing to do before you deploy the load balancer—replace the old `public_ip` output of the single EC2 Instance you had before with an output that shows the DNS name of the ALB:

```
output "alb_dns_name" {
  value       = aws_lb.example.dns_name
  description = "The domain name of the load balancer"
}
```

Run `terraform apply`, and read through the plan output. You should see that your original single EC2 Instance is being removed, and in its place, Terraform will create a launch configuration, ASG, ALB, and a security group. If the plan looks good, type **`yes`** and hit Enter. When `apply` completes, you should see the `alb_dns_name` output:

```
Outputs:
alb_dns_name = "terraform-asg-example-123.us-east-
2.elb.amazonaws.com"
```

Copy down this URL. It’ll take a couple minutes for the Instances to boot and show up as healthy in the ALB. In the meantime, you can inspect what you’ve deployed. Open up the ASG section of the EC2 console, and you should see that the ASG has been created, as shown in Figure 2-12.



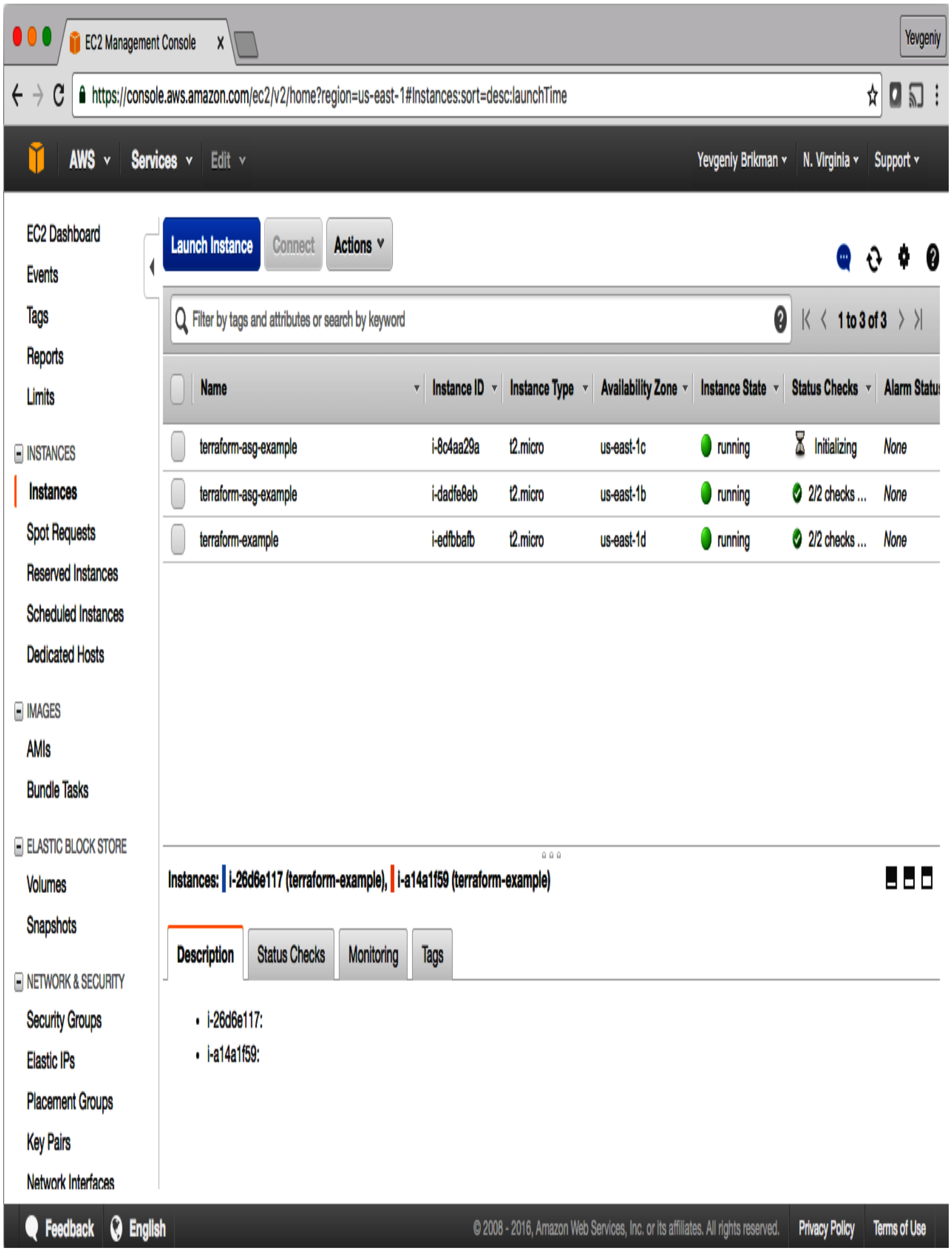
Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Cl
tf-asg-201609...	terraform-20160928175...	2	2	2	10	us-east-1b, us-east-1c, us-e...	300	300

Auto Scaling Group: tf-asg-20160928175209496071094gjj

Launch Configuration	terraform-20160928175208257279971vtb		
Load Balancers			
Target Groups			
Desired	2	Availability Zone(s)	us-east-1b, us-east-1c, us-east-1d, us-east-1e
Min	2	Subnet(s)	
Max	10	Default Cooldown	300
Health Check Type	EC2	Placement Group	
Health Check Grace Period	300	Suspended Processes	
Termination Policies	Default	Enabled Metrics	

Figure 2-12. The AWS Console shows all the ASGs you've created.

If you switch over to the Instances tab, you'll see the two EC2 Instances launching, as shown in Figure 2-13.



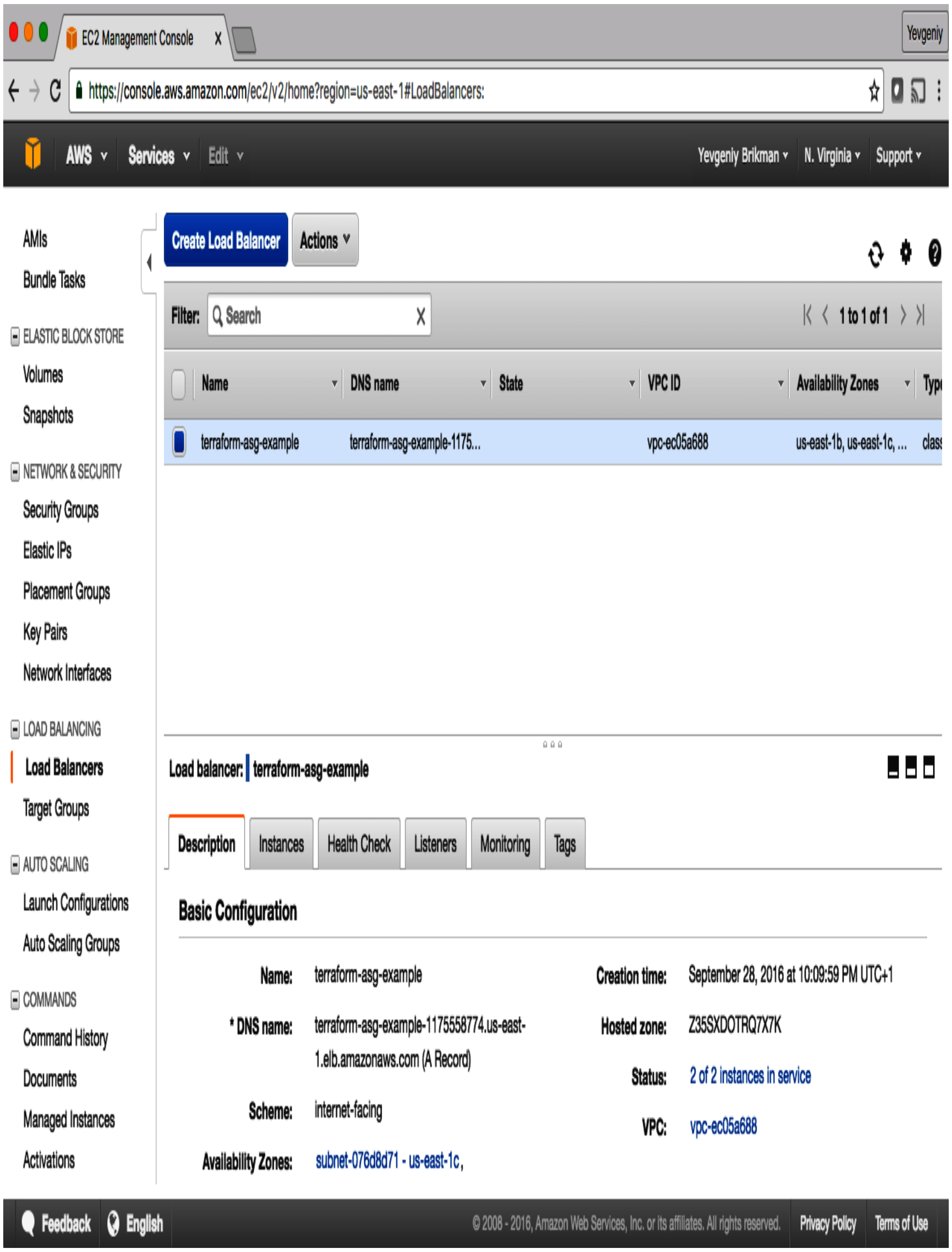
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Statu
terraform-asg-example	i-8c4aa29a	t2.micro	us-east-1c	running	Initializing	None
terraform-asg-example	i-dadfe8eb	t2.micro	us-east-1b	running	2/2 checks ...	None
terraform-example	i-edfbbafb	t2.micro	us-east-1d	running	2/2 checks ...	None

Instances: i-26d6e117 (terraform-example), i-a14a1f59 (terraform-example)

- i-26d6e117:
- i-a14a1f59:

Figure 2-13. The EC2 Instances in the ASG are launching.

If you click the Load Balancers tab, you'll see your ALB, as shown in Figure 2-14.



Name	DNS name	State	VPC ID	Availability Zones	Typ
terraform-asg-example	terraform-asg-example-1175...		vpc-ec05a688	us-east-1b, us-east-1c, ...	clas

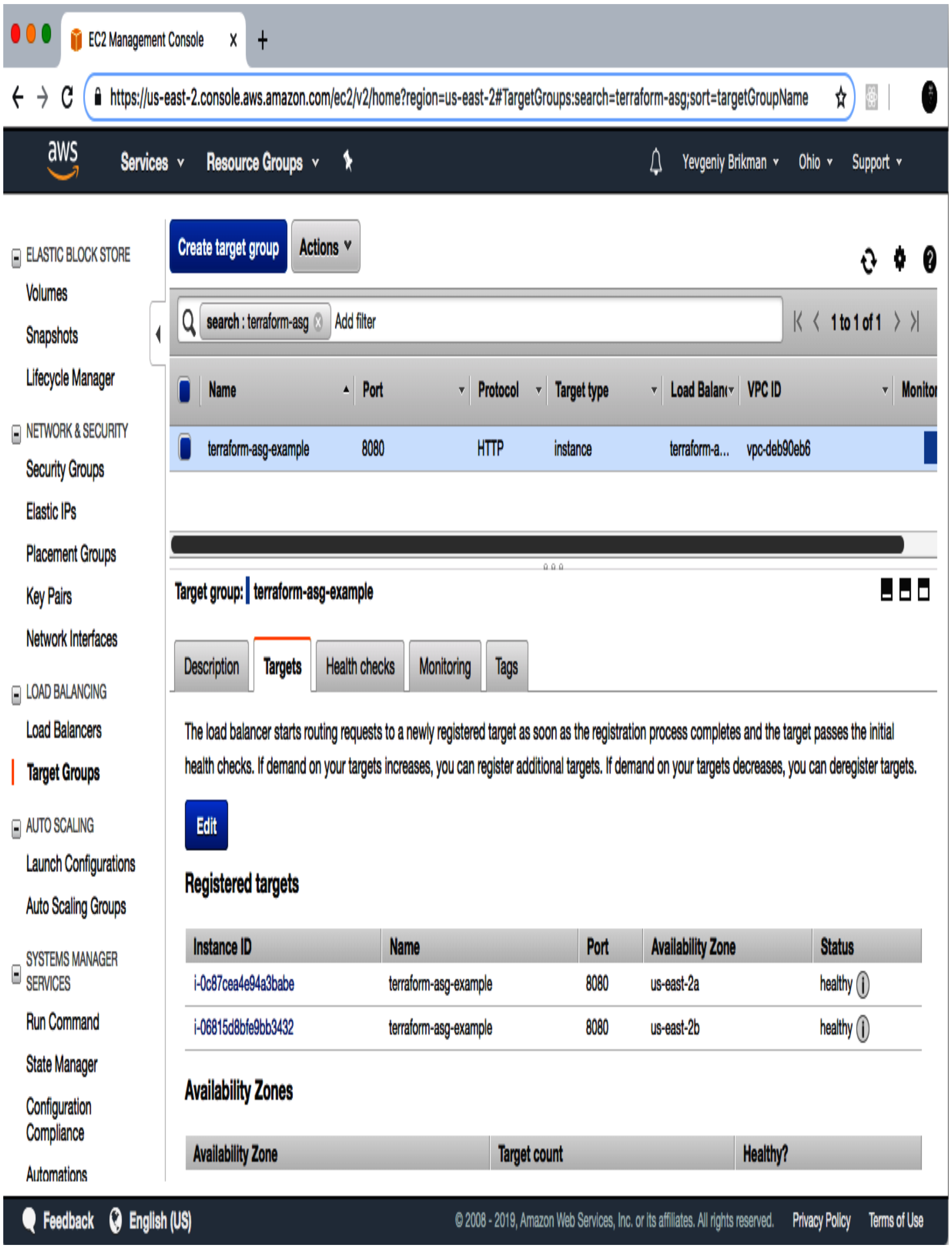
Load balancer: terraform-asg-example

Basic Configuration

Name:	terraform-asg-example	Creation time:	September 28, 2016 at 10:09:59 PM UTC+1
* DNS name:	terraform-asg-example-1175558774.us-east-1.elb.amazonaws.com (A Record)	Hosted zone:	Z35SXDOTRQ7X7K
		Status:	2 of 2 instances in service
Scheme:	internet-facing	VPC:	vpc-ec05a688
Availability Zones:	subnet-076d8d71 - us-east-1c,		

Figure 2-14. The AWS Console shows all the ALBs you've created.

Finally, if you click the Target Groups tab, you can find your target group, as shown in Figure 2-15.



Name	Port	Protocol	Target type	Load Balanc	VPC ID	Monitor
terraform-asg-example	8080	HTTP	instance	terraform-a...	vpc-deb90eb6	

Target group: terraform-asg-example

The load balancer starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks. If demand on your targets increases, you can register additional targets. If demand on your targets decreases, you can deregister targets.

Registered targets

Instance ID	Name	Port	Availability Zone	Status
i-0c87cea4e94a3babe	terraform-asg-example	8080	us-east-2a	healthy
i-06815d8bfe9bb3432	terraform-asg-example	8080	us-east-2b	healthy

Availability Zones

Availability Zone	Target count	Healthy?

Figure 2-15. The AWS Console shows all the target groups you've created.

If you click your target group and find the Targets tab in the bottom half of the screen, you can see your Instances registering with the target group and