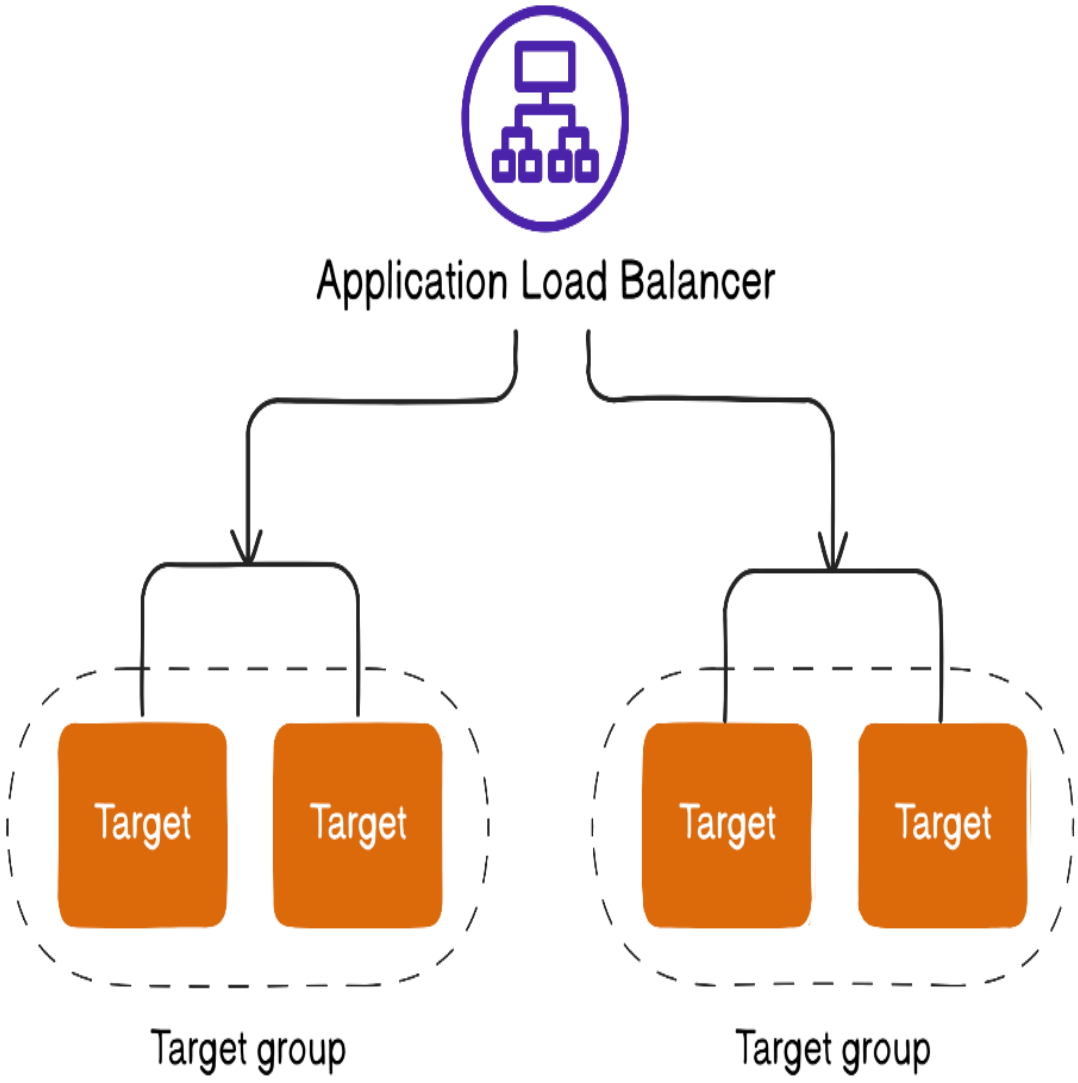
Guided Lab: Creating Your First Application Load Balancer

Description

Application Load Balancer (ALB) operates at the application layer (Layer 7 of the OSI model) and is designed to route HTTP/HTTPS traffic. But why use an ALB? While one can host an application on a single EC2 instance or vertically scale an instance for more resources, there are limits.



ALB is usually used when:

1. High availability is required: ALB can route traffic across multiple Availability Zones (AZs). If an instance in one AZ fails, ALB redirects traffic to healthy instances in other AZs.
2. Using Auto Scaling Groups (ASG) for dynamic scaling: ALB integrates seamlessly with EC2 Auto Scaling. Together, they help you create a multi-AZ environment that lets you scale out when the need arises. Instead of scaling a single instance vertically, you can simply add more instances when you need them, avoiding the fuss of resizing an EC2 instance.
3. HTTP-based routing is desired**:** ALB supports various request routing based on HTTP parameters. This enables use cases like:
    1. **Path-based routing**: Directs client requests to specific services based on the URL path. For instance, /images could route to an image server while /api goes to an API server.
    2. **Host-based routing**: Routes requests based on the domain name. Useful for hosting multiple domains on a single load balancer.
    3. **HTTP header routing:** Routes traffic based on headers, query parameters, or HTTP methods.

Prerequisite

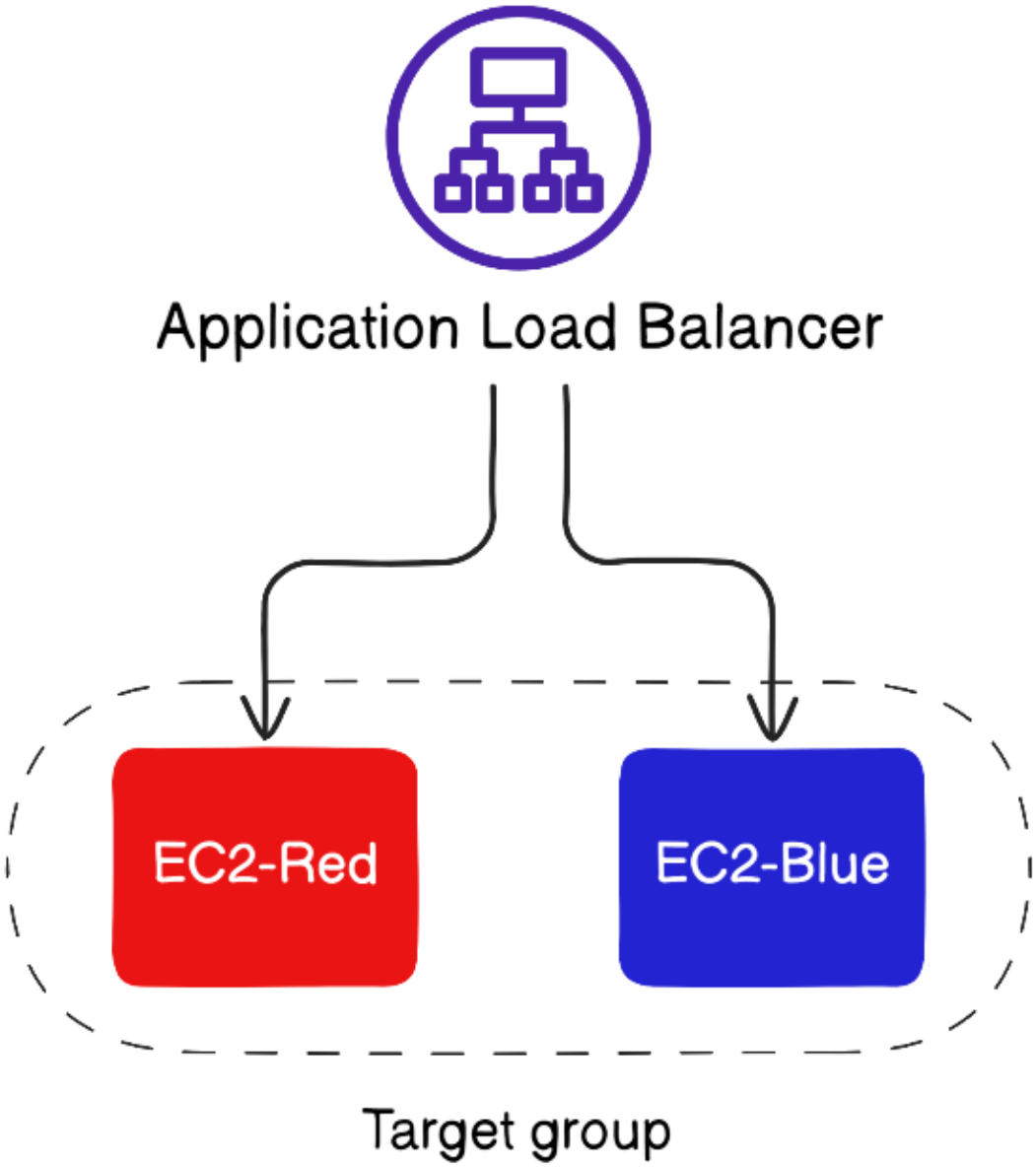
This lab assumes you have experience creating EC2 instances and are familiar with its basic components.

If you find any gaps in your knowledge, consider taking the following labs:

- Creating an Amazon EC2 instance (Linux)
- Setting up a Web server on an EC2 instance
- Launching an EC2 Instance with User Data

Objectives

In this lab, we'll set up two EC2 instances to demonstrate how an ALB distributes traffic visually. One will display a red web page, and the other a blue one. As you access the ALB, the page colors will switch, representing the load distribution between instances.

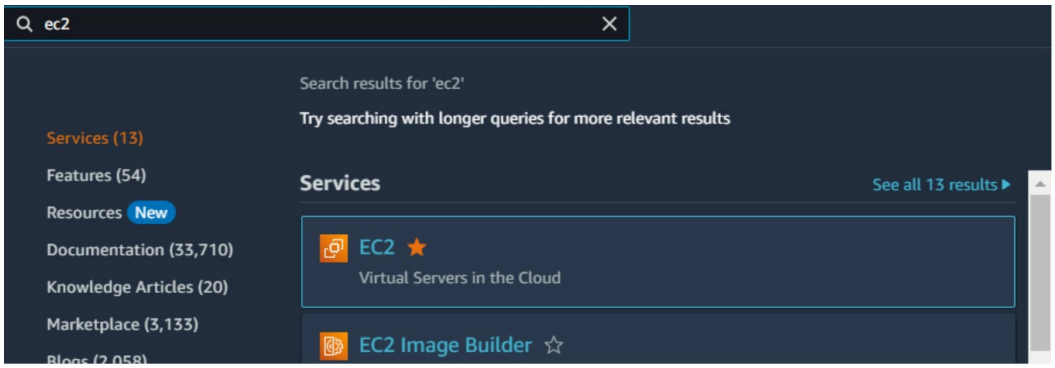


Subscribe to access AWS PlayCloud Labs

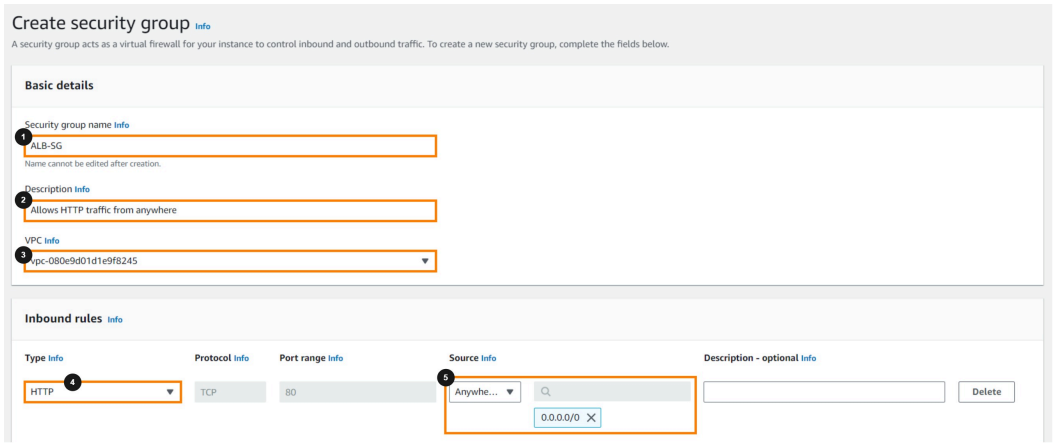
Lab Steps

Creating the EC2 instances

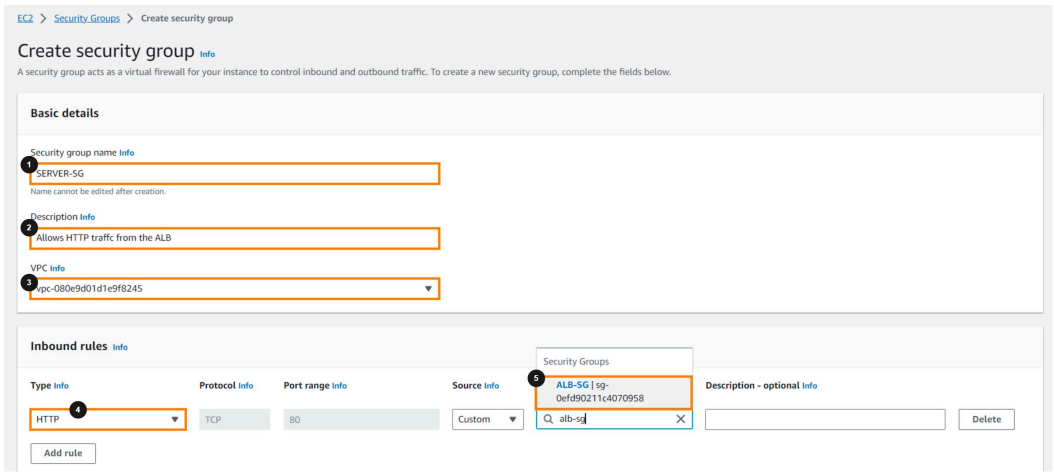
1. Search '*ec2*' in the AWS Management Console search bar. Click **EC2** on the search results.



2. On the left window, under **Network & Security**, select **Security Groups**, then click **Create security group**. Create a security group named ‘ALB-SG’ for the Application Load Balancer. Configure it with an inbound rule allowing HTTP traffic from 0.0.0.0/0.



3. Create another security group named ‘SERVER-SG’ for the EC2 instances. Configure it with an inbound rule allowing HTTP traffic from the Application Load Balancer’s security group (ALB-SG).



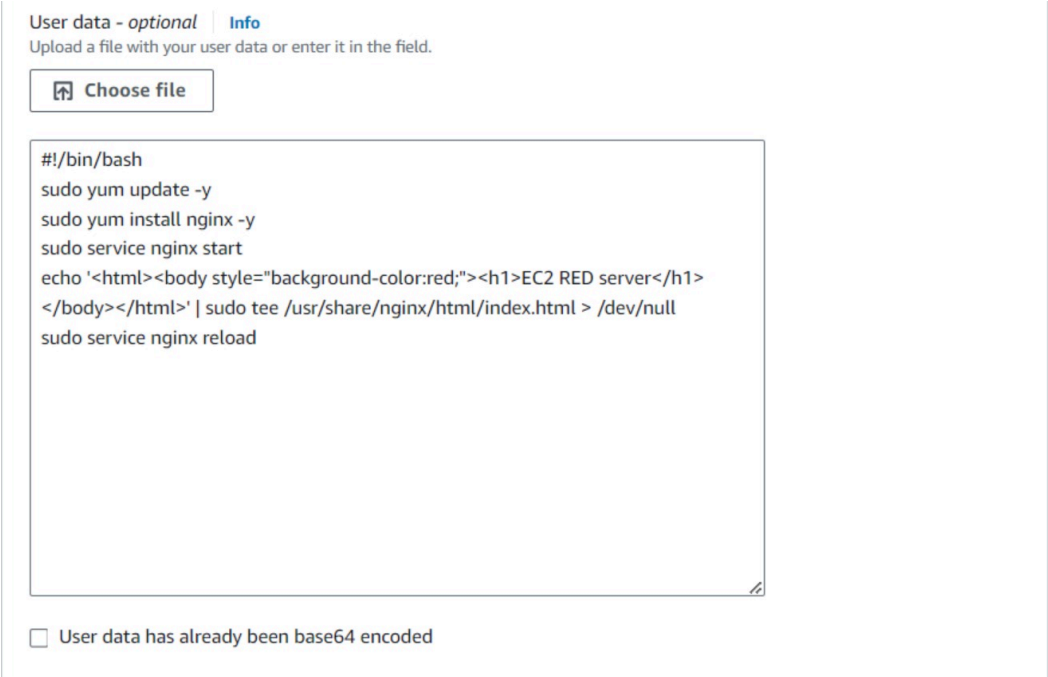
By setting up these security groups, we make sure that the application cannot be accessed directly using the EC2 instances’ public IP addresses. All incoming traffic is routed exclusively through the ALB, ensuring users interact with our application via the ALB’s endpoint only.

3. Launch two EC2 instances with the following configurations.

EC2-RED

- Name: EC2-RED
- Instance type: t2.micro
- AMI: Amazon Linux
- Key pair: We're not going to SSH into any instances in this lab, so just select the '**Proceed without key pair**' option).
- Security Group: SERVER-SG
- User data:

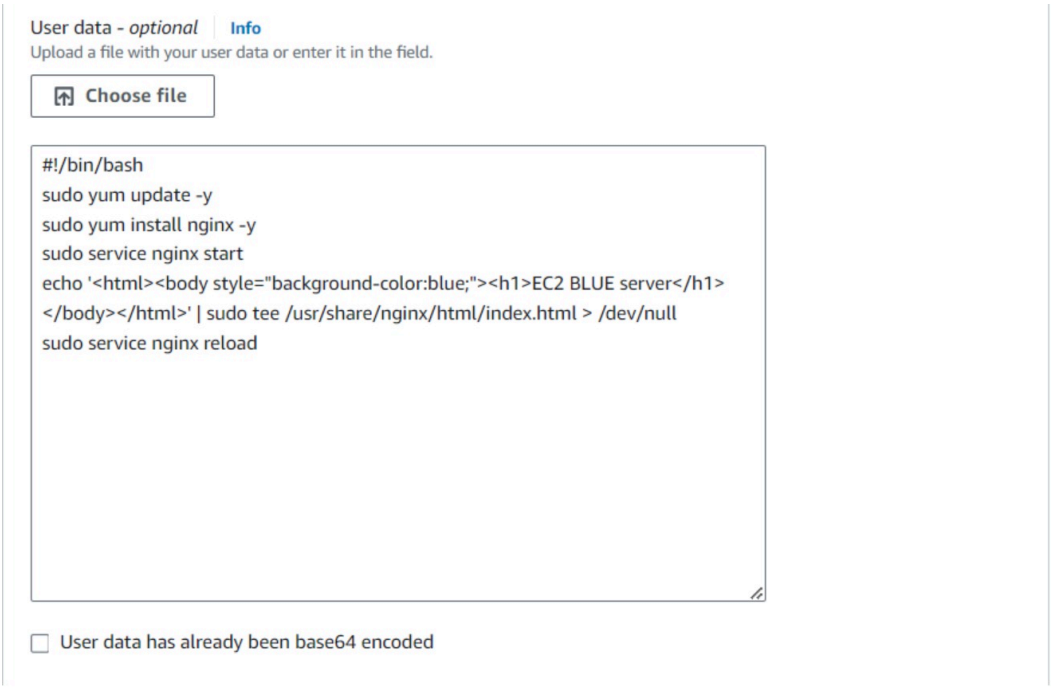
```
#!/bin/bash
sudo yum update -y
sudo yum install nginx -y
sudo service nginx start
echo '<html><body style="background-color:red;"><h1>EC2
RED server</h1></body></html>' | sudo tee
/usr/share/nginx/html/index.html > /dev/null
sudo service nginx reload
```



EC2-BLUE

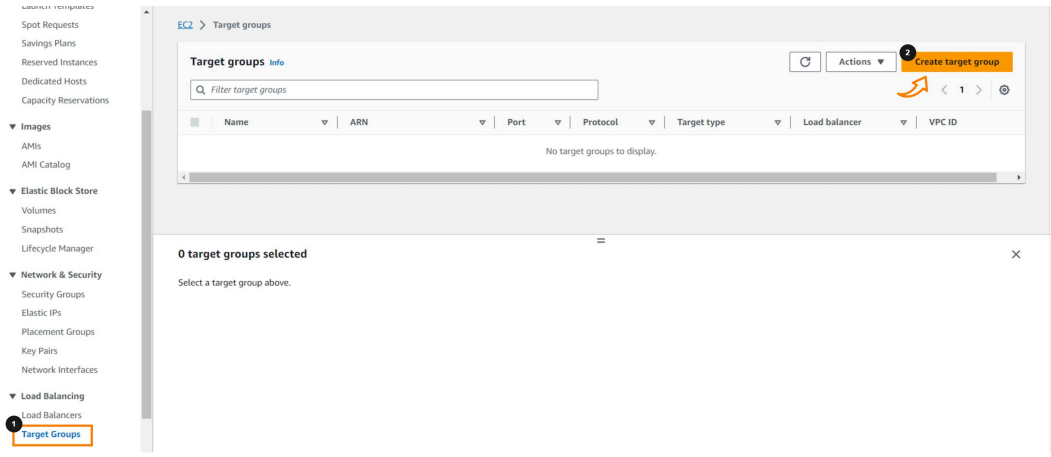
- Name: EC2-BLUE
- Instance type: t2.micro
- AMI: Amazon Linux
- Key pair: We're not going to SSH into any instances in this lab, so just select the '**Proceed without key pair**' option).
- Security Group: SERVER-SG
- User data:

```
#!/bin/bash
sudo yum update -y
sudo yum install nginx -y
sudo service nginx start
echo '<html><body style="background-color:blue;">
<h1>EC2 BLUE server</h1></body></html>' | sudo tee
/usr/share/nginx/html/index.html > /dev/null
sudo service nginx reload
```

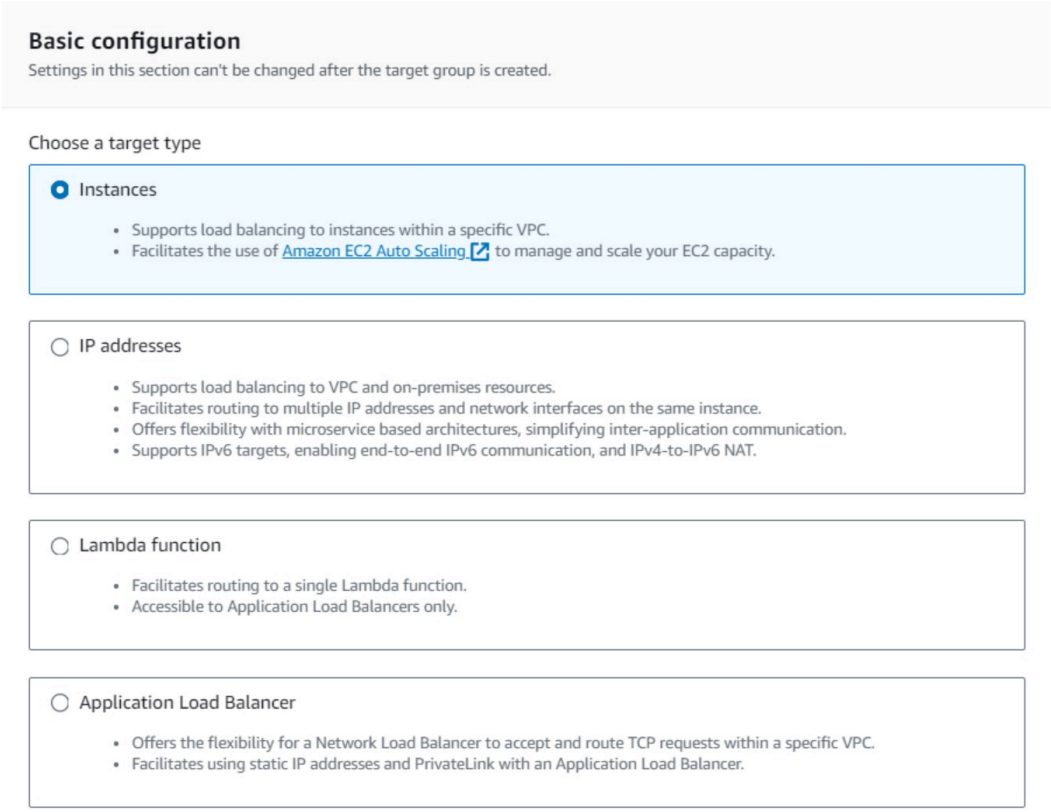


Creating the Target Group

4. On the left side of the EC2 Management Console, Under **Load Balancing**, select **Target Groups** then click **Create target group**

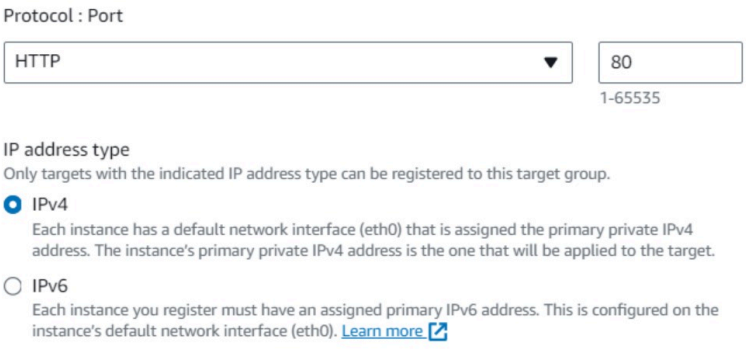


5. For **target type**, choose **Instances**.

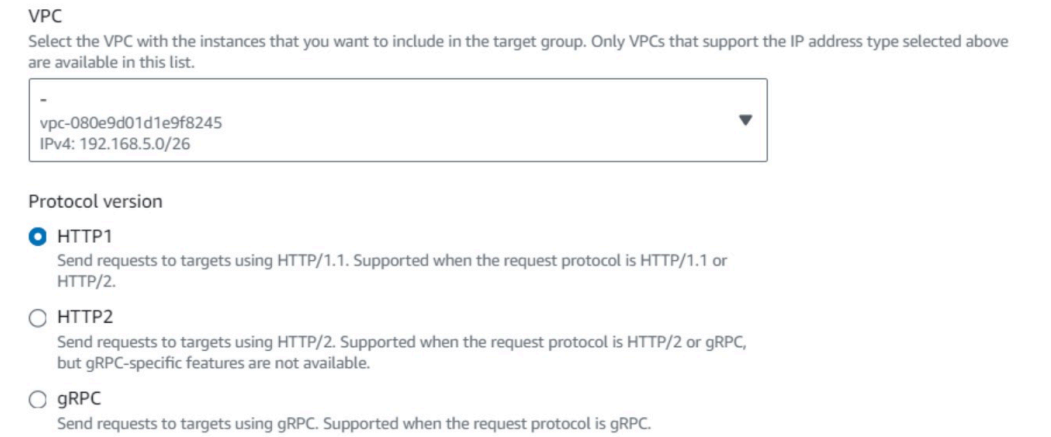


6. Enter **my-alb-target-group** as the **Target group name**.

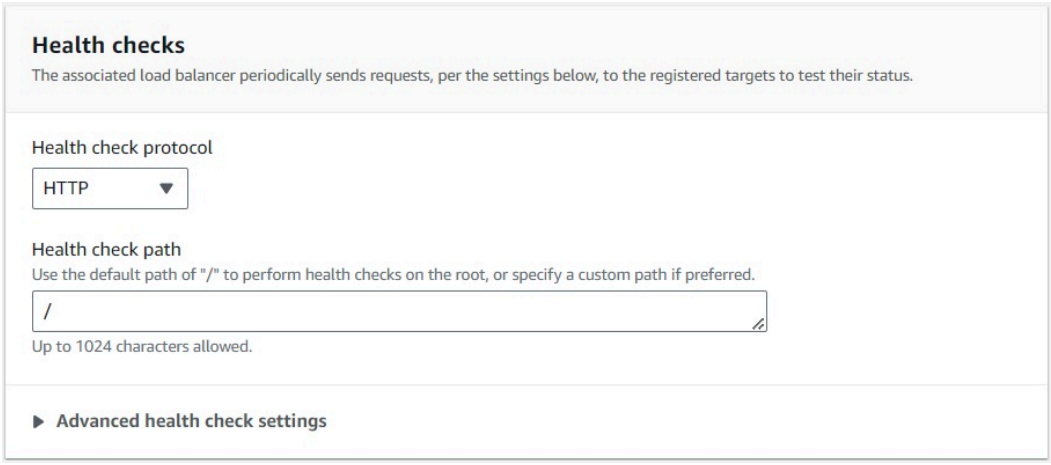
7. For **Protocol**, select **HTTP** and enter **80** for the **Port** number. Make sure the Ipv4 option is selected **IP address type.**



8. For **VPC**, select the default VPC. Select **HTTP1** for the **Protocol version**.

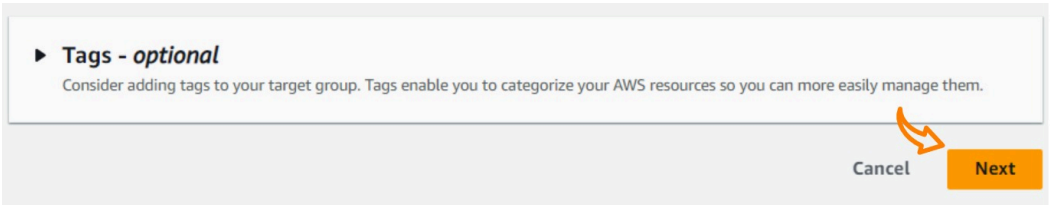


9. In the **Health checks** section, choose HTTP and leave the default '/' path as the Health check path.

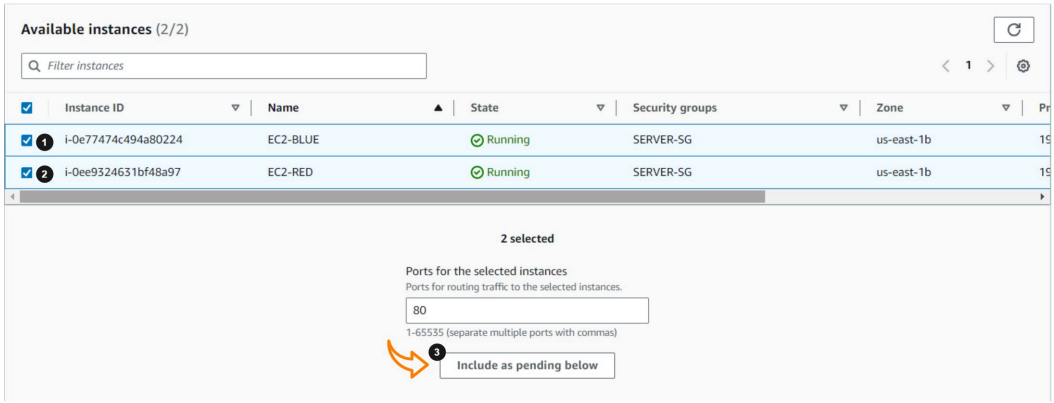


Health checks allow the ALB to ping or check your servers to see if they're okay. Think of it like a regular wellness check. It does this by trying to access a specific path, like '/', on your server. If it gets a response, it knows the server is good. If not, it thinks the server is unhealthy and stops sending it traffic. Changing the health check path to something that doesn't respond correctly can mistakenly mark healthy targets as unhealthy.

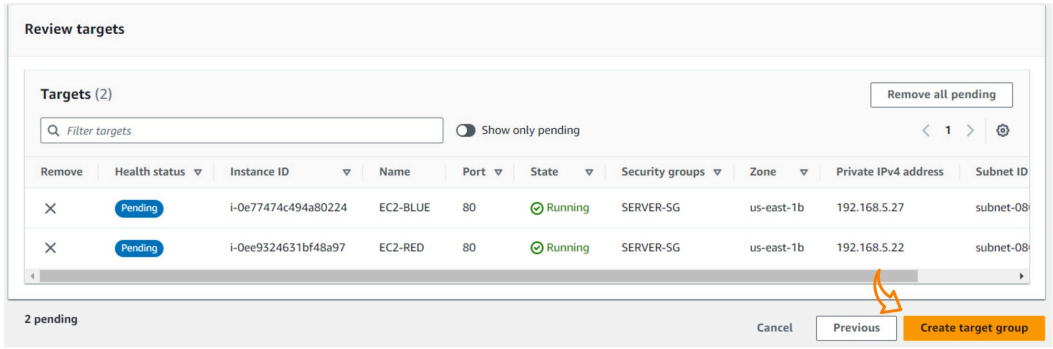
10. At the bottom of the page, click **Next.**



11. Select the **EC2-RED** and **EC2-BLUE** instances as targets, then click the **Include as pending below** button.

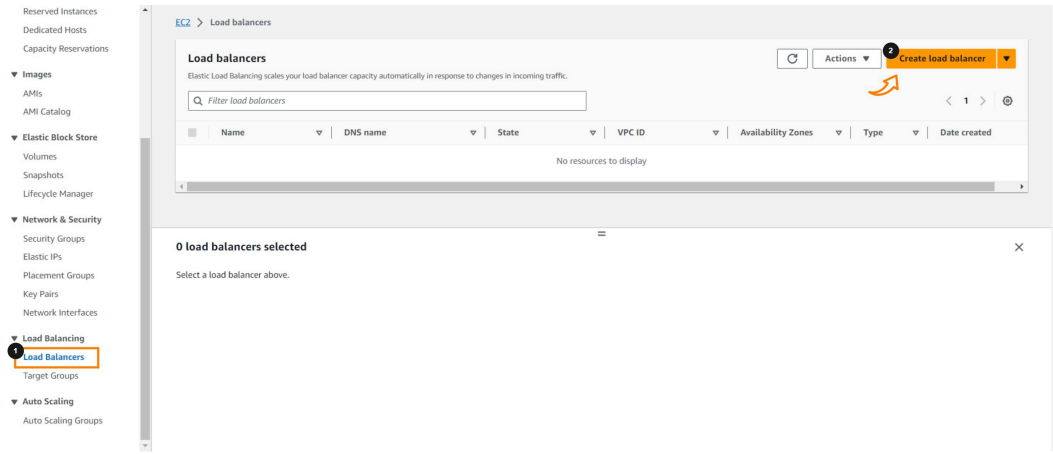


12. Click the **Create target group** button.

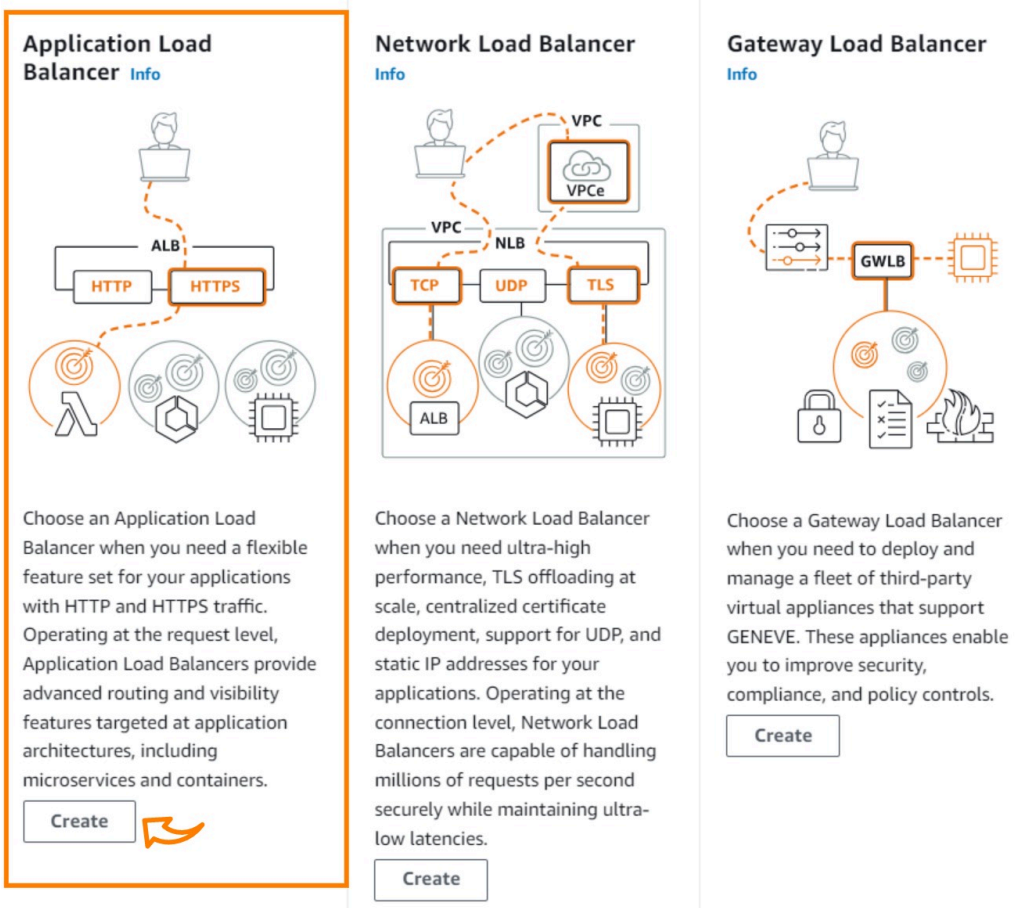


Creating the Application Load Balancer

13. Under **Load Balancing**, select **Load Balancers**, then click **Create Load Balancer**.



14. Click the **Create** button under **Application Load Balancer**.

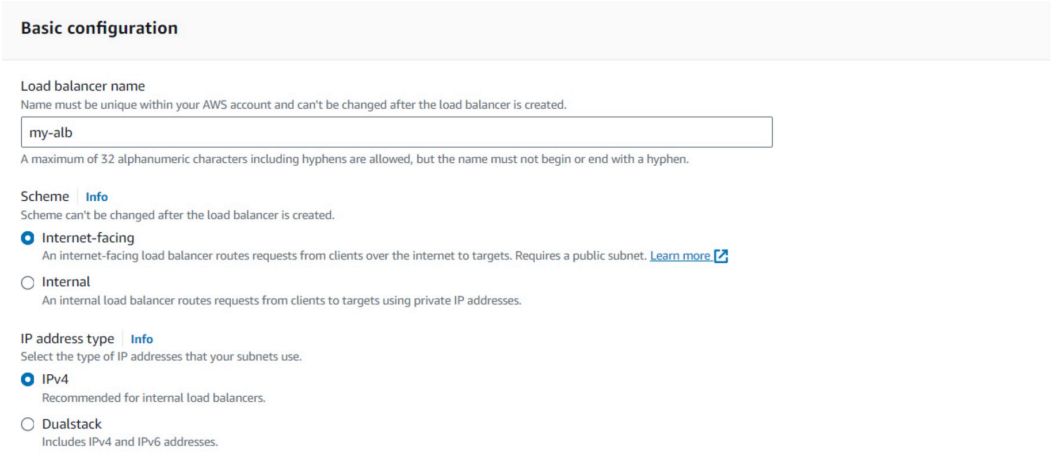


15. On **Basic configuration:**

a. Enter '*my-alb*' as the load balancer name.

b. Select **Internet-facing** for the Scheme.

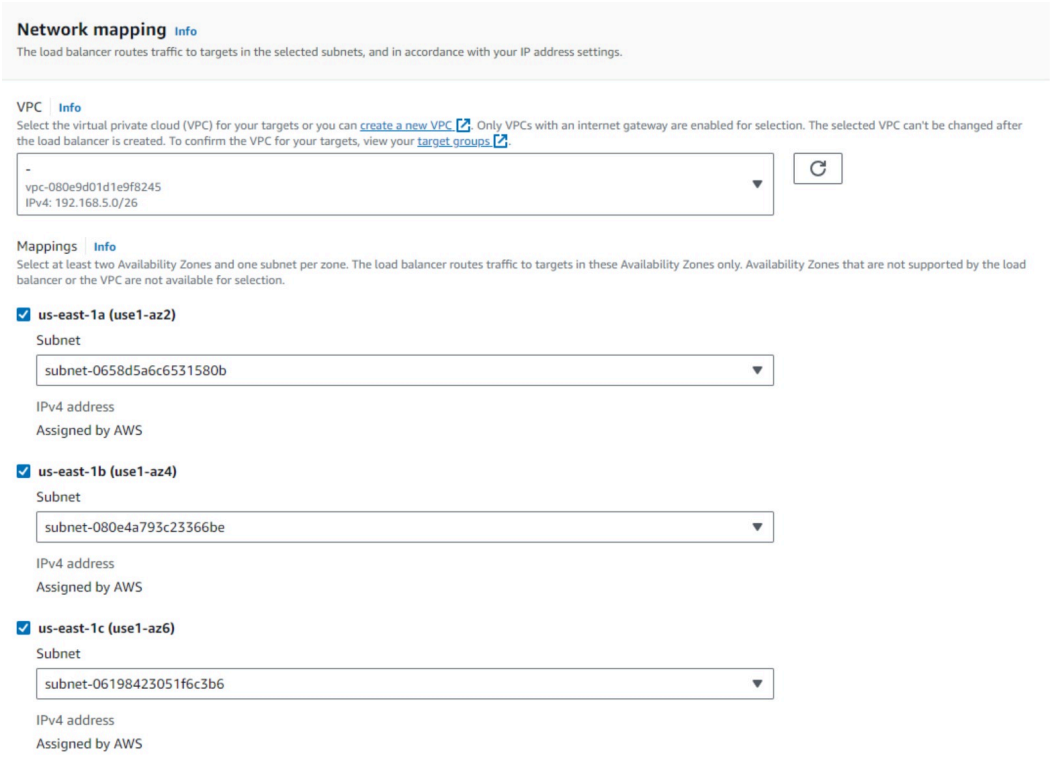
c. Select **IPv4** as the IP address type.



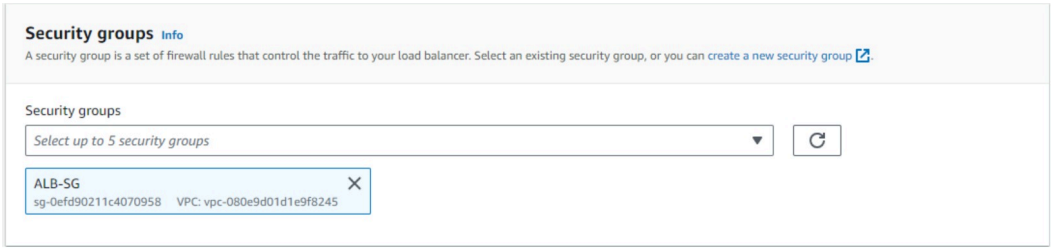
16. For **Network mapping**, select the default VPC. For **Mappings**, select all the checkboxes.

Each AZ you select is where the ALB will deploy a node to handle incoming traffic. These nodes in different AZs provide redundancy and ensure high availability for your application. Even though our

main goal in this lab isn't high availability and our EC2 instances won't be deployed across different AZs, please check all the available AZ checkboxes. Doing so mirrors a setup you'd use for a multi-AZ architecture.



17. For **Security groups**, click the dropdown menu and select **ALB-SG**



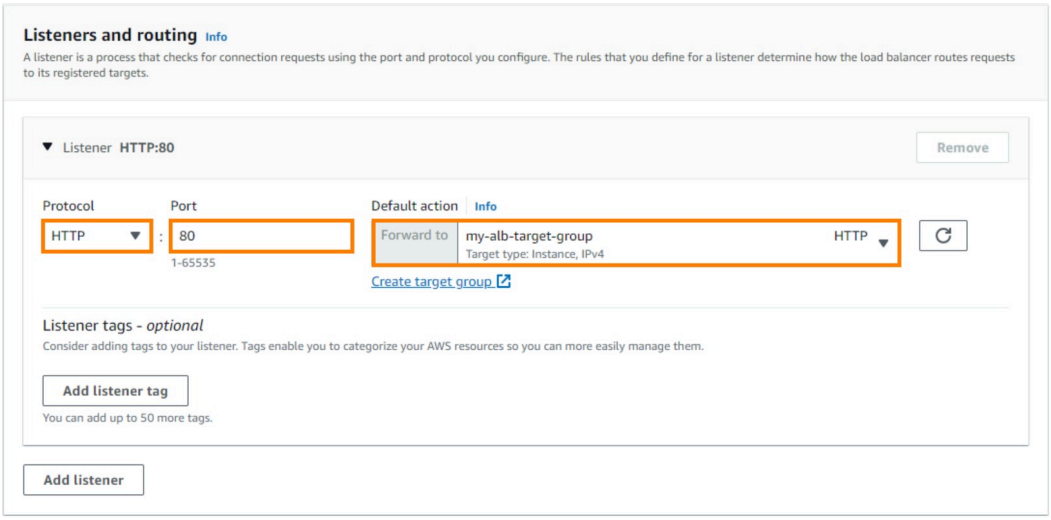
18. On **Listeners and routing:**

a. Select HTTP for Protocol

b. Set 80 for Port

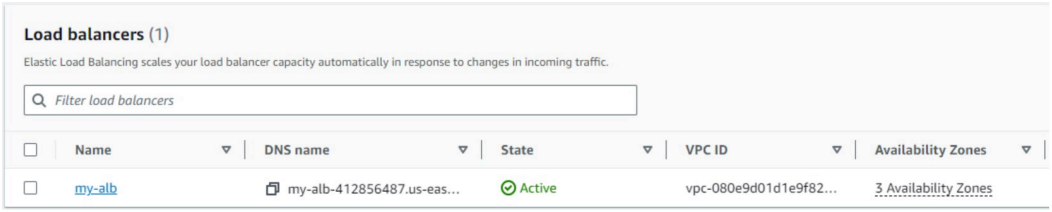
c. Click the **Select a target group** dropdown menu, and click **my-alb-target-group**.

d. Scroll down to the bottom page and click **Create load balancer**.

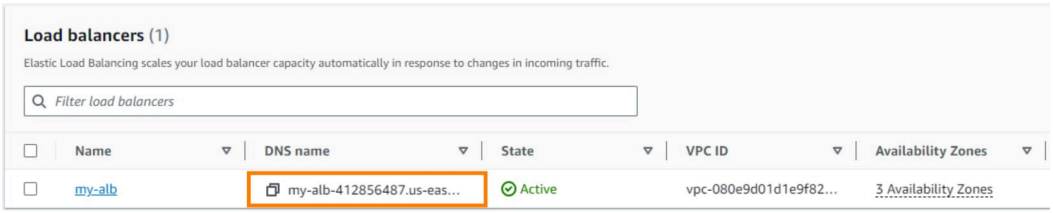


19. After you've created the load balancer, a confirmation will appear at the top of the page. Click on **View Load Balancer** to navigate to your ALB.

On the ALB dashboard, you'll see the status of your newly created ALB. Ensure that its state changes to **Active** before proceeding.



20. Once active, copy your ALB's DNS Name. This is the endpoint through which the ALB will route traffic to your instances.



21. Enter your ALB's DNS Name into your browser. You should see a color-coded page (red or blue). Hit refresh several times to see the load balancing in action (switching between colors). This demonstrates the ALB distributing traffic across your instances. Note that due to browser caching and the nature of load balancing, the switch might not always be immediate.

Notice how the responses consistently alternate between red and blue? This is because the default routing mechanism for ALB is 'round-robin'. When there are two servers, as we have now, each load balancer node receives approximately 50% of the traffic. The round-robin algorithm ensures incoming traffic is distributed evenly between the servers, cycling from one to the next in order.

You've successfully set up an Application Load Balancer and observed how it distributes incoming traffic across two instances. This foundational knowledge of ALB is crucial in scaling web applications. As you delve deeper into AWS and its services, you'll appreciate the flexibility and robustness that tools like the ALB offer. Well done on completing this lab!