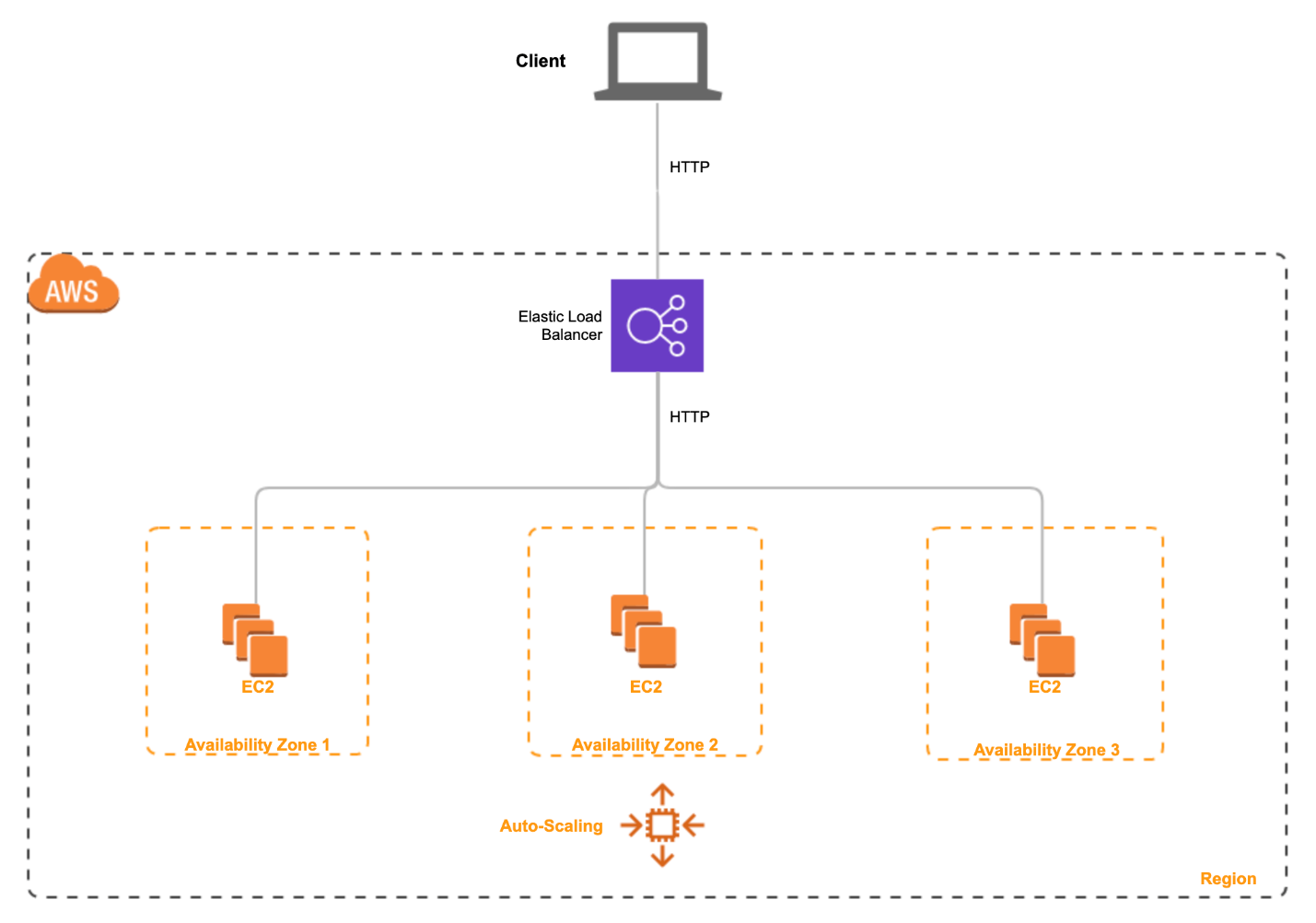
**Horizontal scaling of EC2 instances using an Auto Scaling Group with an Application Load Balancer**

Created by[Stephane Maarek | AWS Certified Cloud Practitioner,Solutions Architect,Developer](https://deloittedevelopment.udemy.com/user/stephane-maarek/)

**AWS Workspace**

**60-120 minutes**

In this lab, you will be an engineer at a company that has recently launched a video-sharing social networking service. The application server keeps getting overloaded at peak usage times. The engineering team needs to implement an Auto Scaling group (ASG) running behind an Application Load Balancer (ALB) to horizontally scale an adequate number of servers to respond to the varying workload.



Our engineering team has recently found that worker EC2 instances are not able to handle the workload for peak traffic usage for the video-sharing social networking service we recently launched. The users upload video files up to 1 GB in size to a single EC2 instance based application server which stores them on a shared EFS file system. Another set of worker EC2 instances managed via an Auto Scaling group, periodically scans the EFS share directory for new files to process and generate new videos (for thumbnails and composite visual effects) according to the video processing instructions that are uploaded alongside the raw video files.

You are tasked to work on a solution to implement an Auto Scaling group (ASG) running behind an Application Load Balancer (ALB) to manage an adequate number of worker EC2 instances to respond to the varying workload. You need to use a CPU utilization based target scaling policy to manage the worker EC2 instances.

**How you'll work**

Your project has been broken into a set of tasks. To complete these tasks, use the provided workspace. You can launch your workspace by clicking below or using the button in the top right of the screen.

Task

1-Create an Application LB

2-Create Launch Template for the EC2

3-Configure Auto Scaling groups to leverage the Launch Template to launch EC2 and use the ALB to distribute the tracfic to the provisioned EC2

4-Simulate load on the EC2 by using stress utility to demonstrate horizontal scaling for the EC2 instance via Auto Scaling group

Create an Application LB

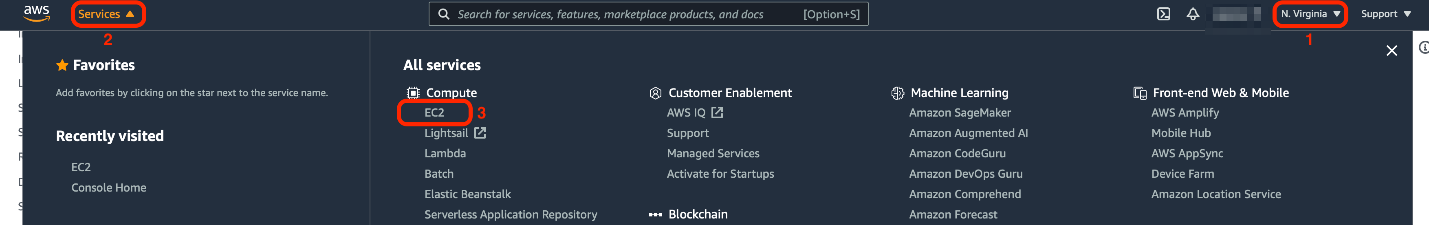
### Tasks

### 0/4

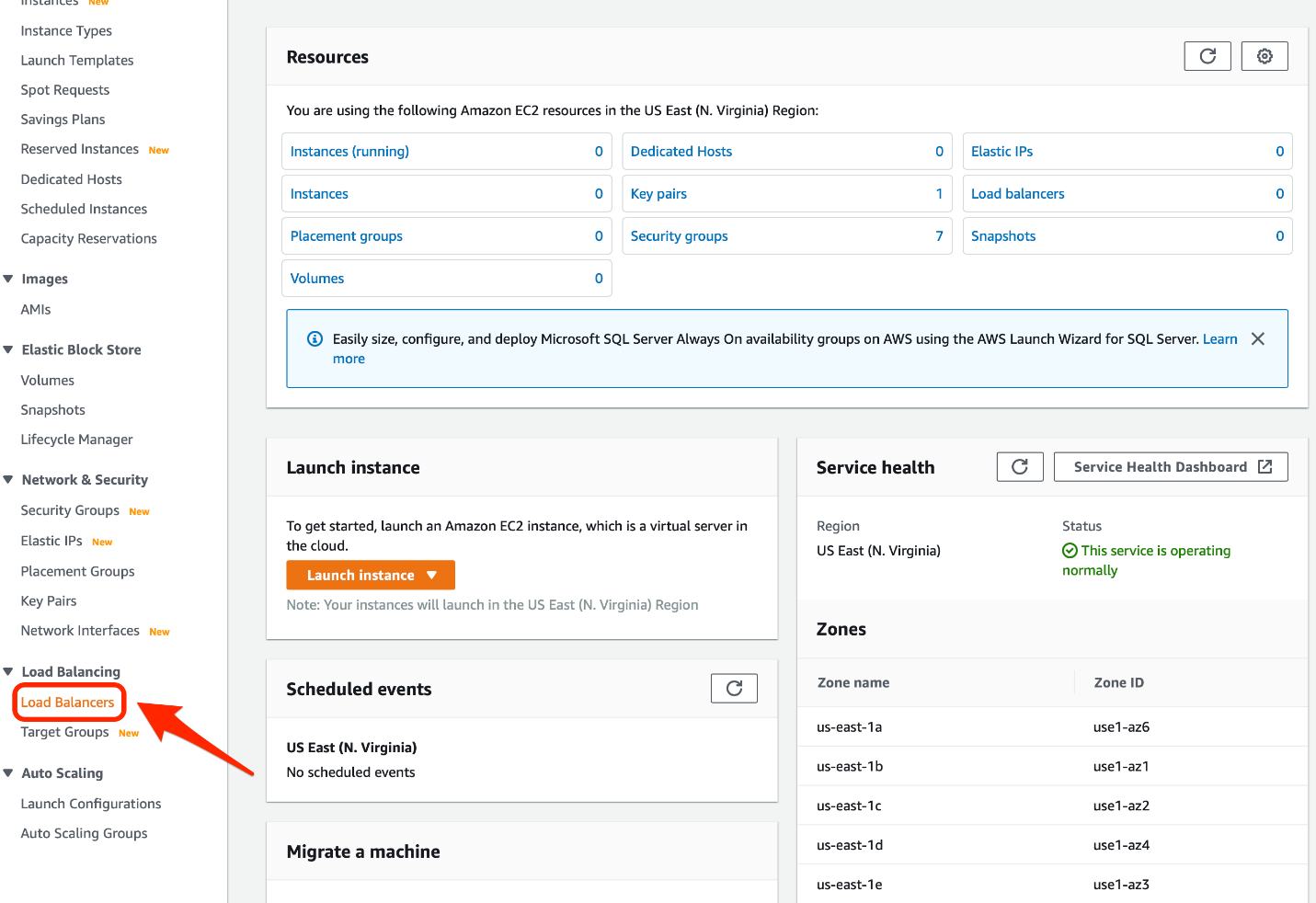
### 1. Create an Application Load Balancer (ALB)

You will create an internet-facing Application Load Balancer (ALB) that handles incoming HTTP web traffic on port 80. The ALB will be configured in the default VPC with 3 Availability Zones enabled for your ALB so that it can distribute the traffic to the targets with cross-zone load balancing.

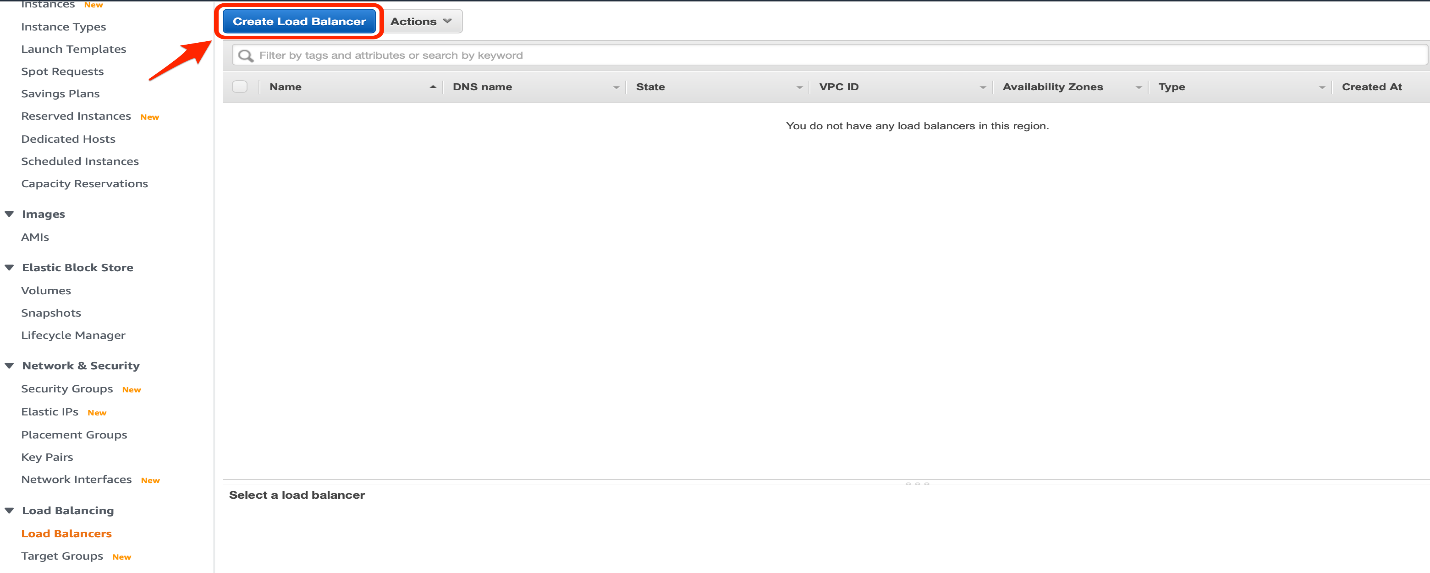
1. Make sure that you are in the N.Virginia AWS Region. Select EC2 via the services menu on the AWS Management Console.



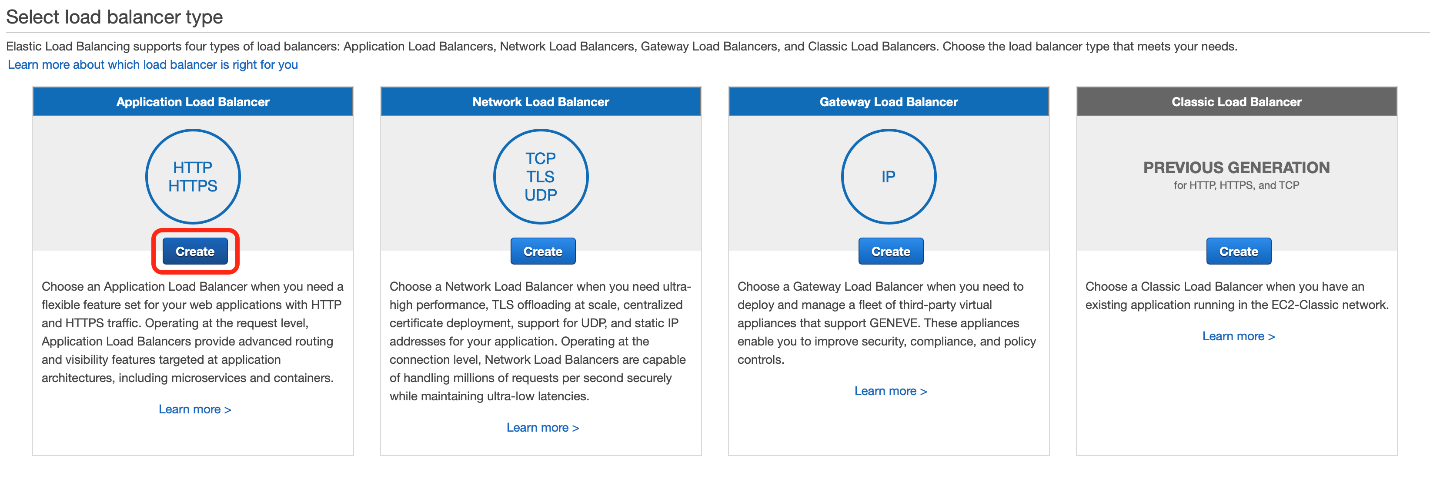
1. Select **Load Balancers** from the left sidebar



1. Click **Create Load Balancer**



1. Select **Application Load Balancer** from the available load balancer types



1. Set the configuration values for the Application Load Balancer as follows:

    Name: Lab-ALB

    Scheme: internet facing

    IP address type: ipv4

    Load Balancer Protocol: HTTP

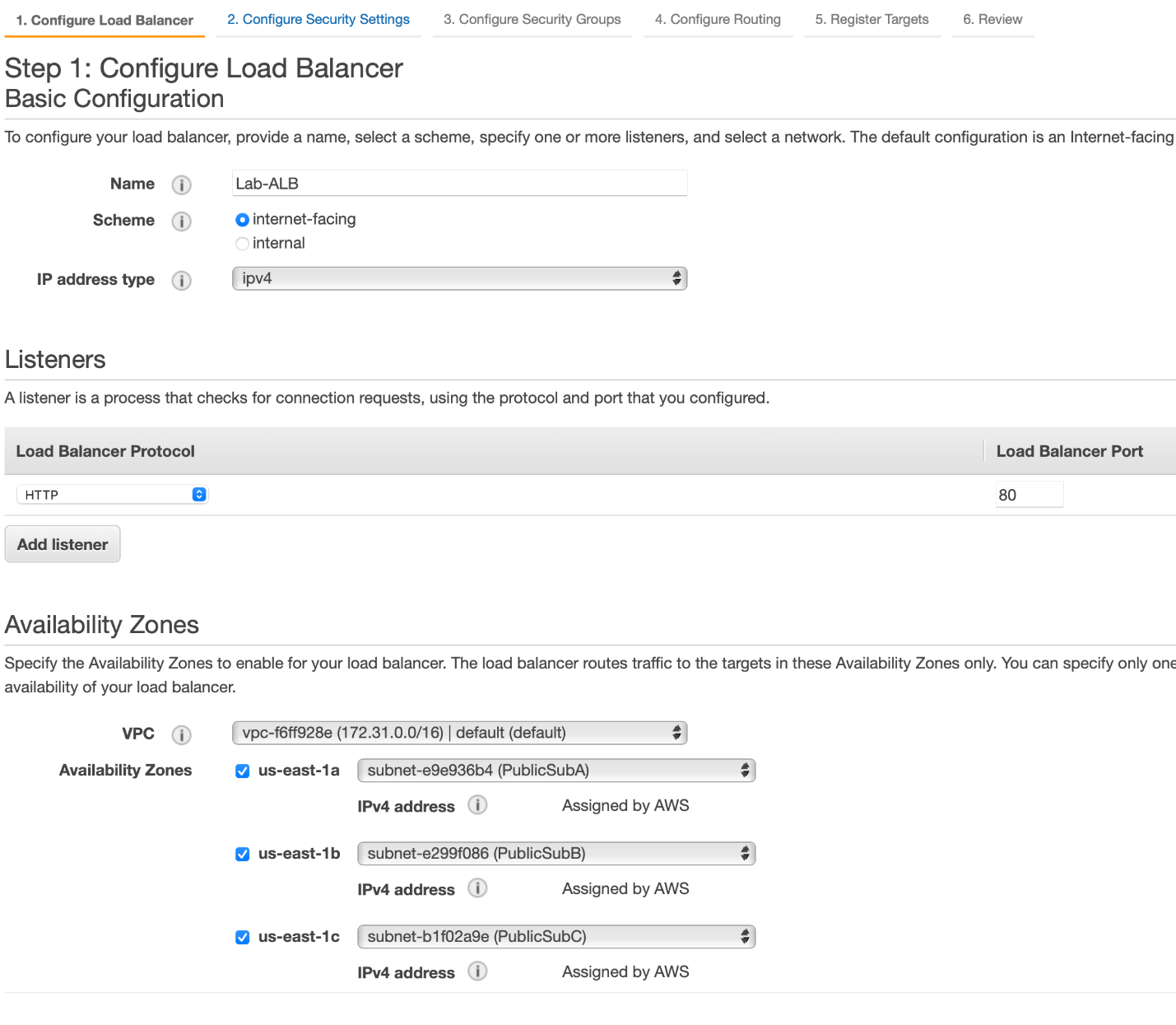
    Load Balancer Port: 80

    Select the default VPC

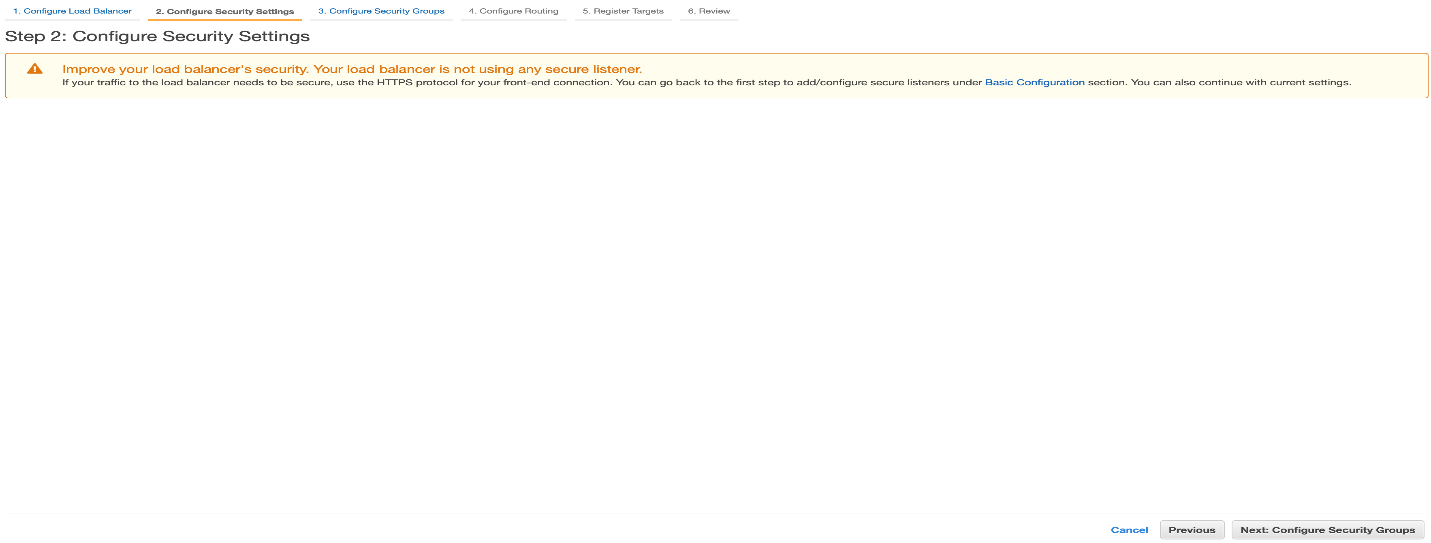
    Select the three Availability Zones as us-east-1a, us-east-1b, us-east-1c

    Leave the option for AWS Global Accelerator as unchecked

    Click Next to go to the **Configure Security Settings** section



1. Skip **Configure Security Settings** section as ALB only handles the HTTP traffic. Click Next to go to the **Configure Security Groups** section.

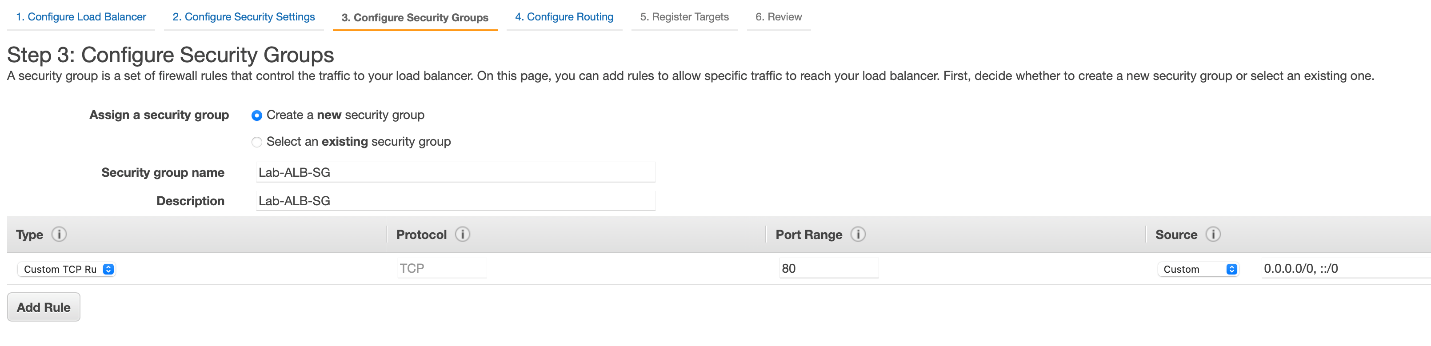


1. Create a new Security Group to be associated with the ALB, so that it allows incoming HTTP web traffic on port 80 from any source. Please fill in the values as follows:

    Security group name: Lab-ALB-SG

    Description: Lab-ALB-SG

    Then, click next to move to the **Configure Routing** section



1. Set the configuration values for the **Configure Routing** section. The ALB routes traffic to the targets defined in the target group. Please fill in the values as follows:

    Target group: New target group

    Name: Lab-TG

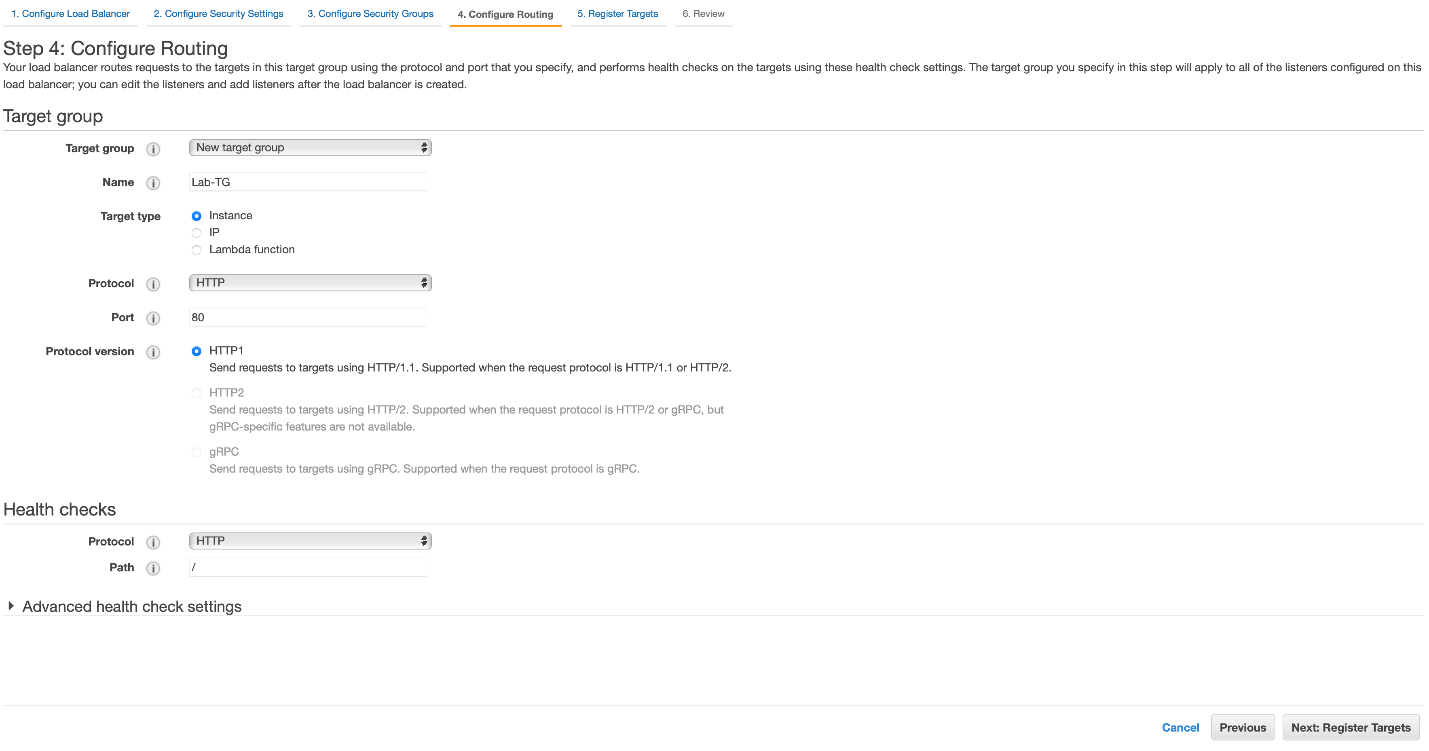
    Target type: Instance

    Protocol: HTTP

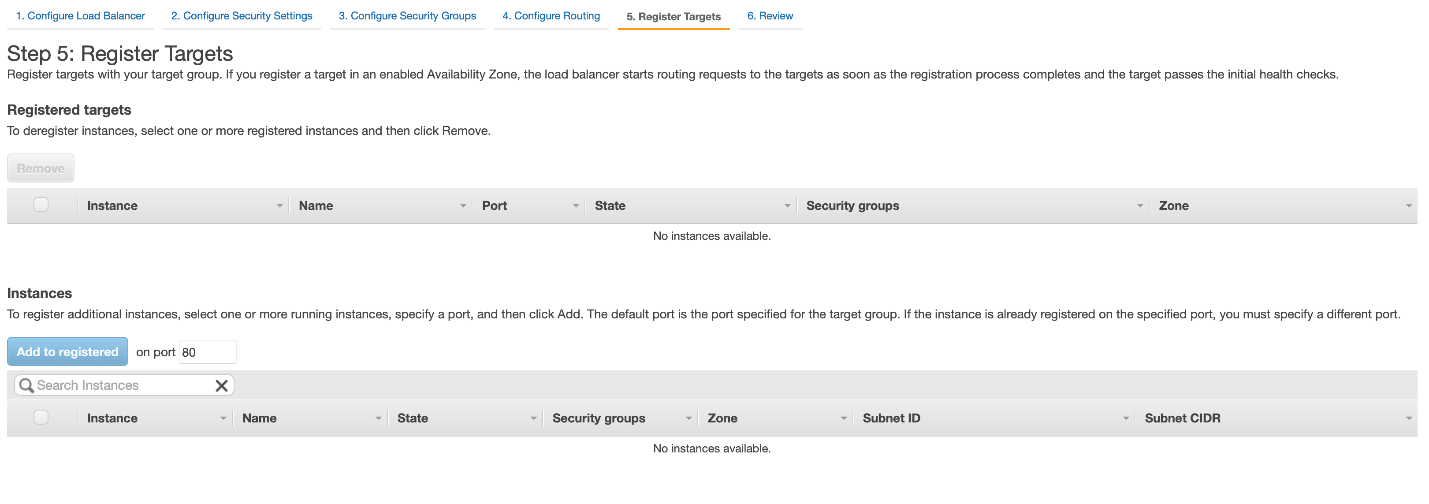
    Port: 80

    Protocol version: HTTP 1

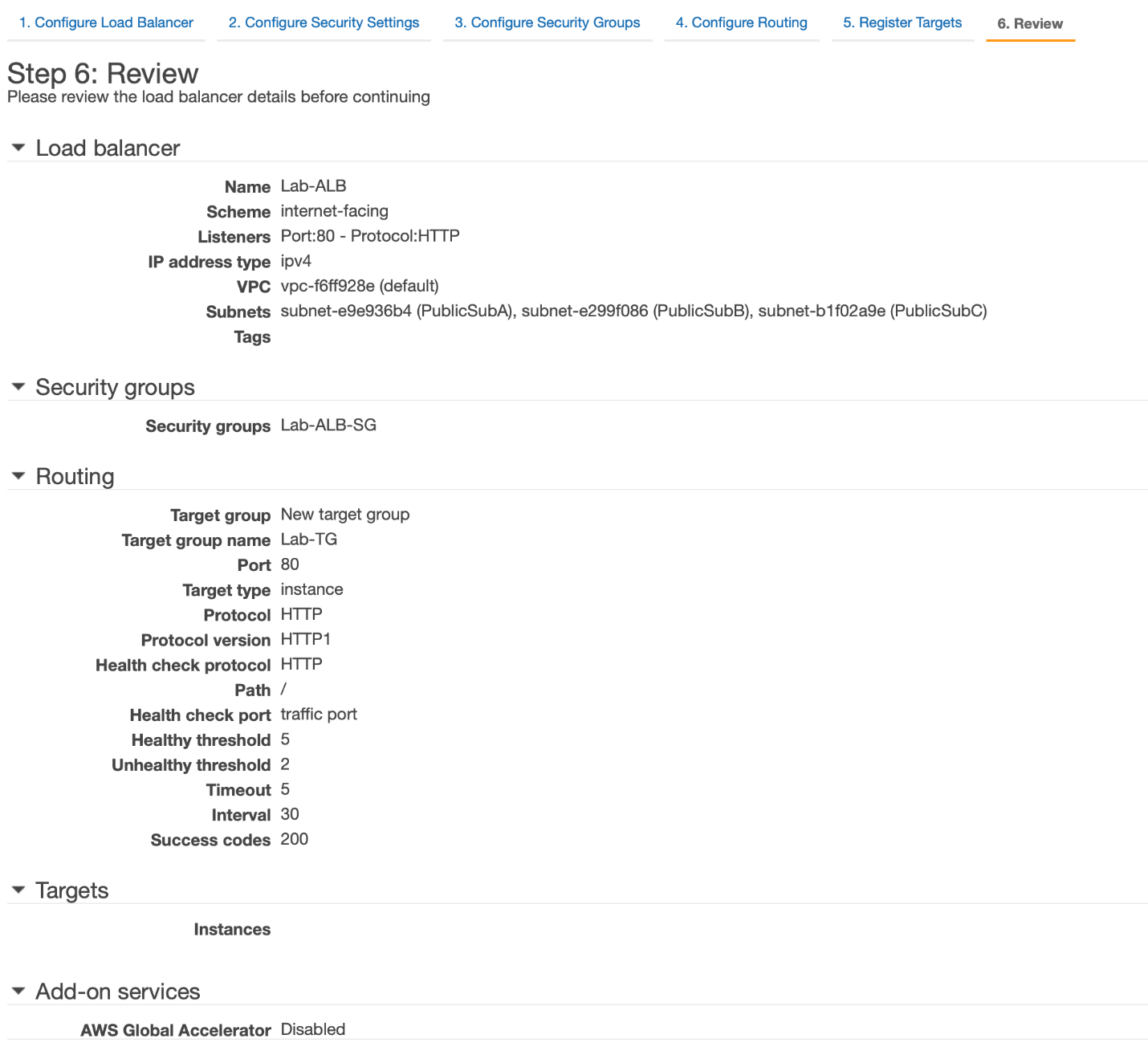
You can leave the defaults for Health Checks and Advanced health check settings. Click Next to go to the **Register Targets** section.



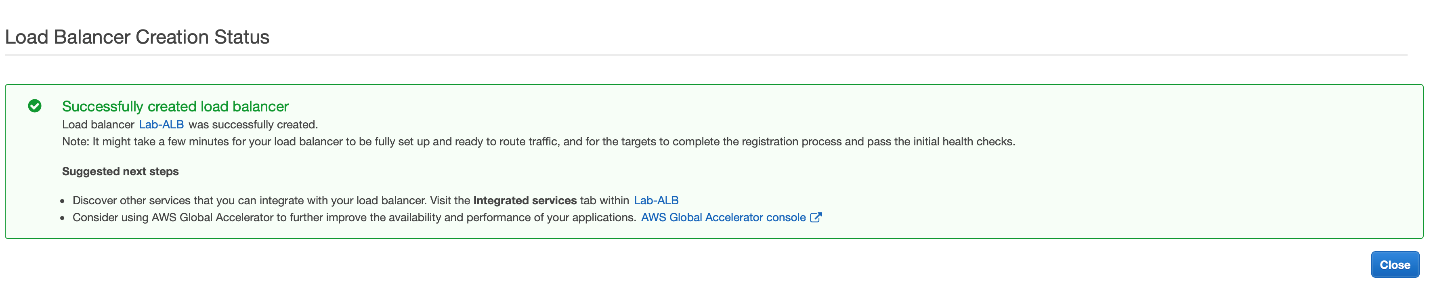
1. Skip **Register Targets** section as we would configure the ALB to route traffic via the Auto Scaling group (ASG) later in this lab. Click Next to go to the **Review** section.



1. Review the section-wise configurations for the ALB. Then click on **Create** to set up the ALB.



1. You should see that your ALB has been successfully created. Click **Close** to complete the Task 1 of this lab.



<https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html>

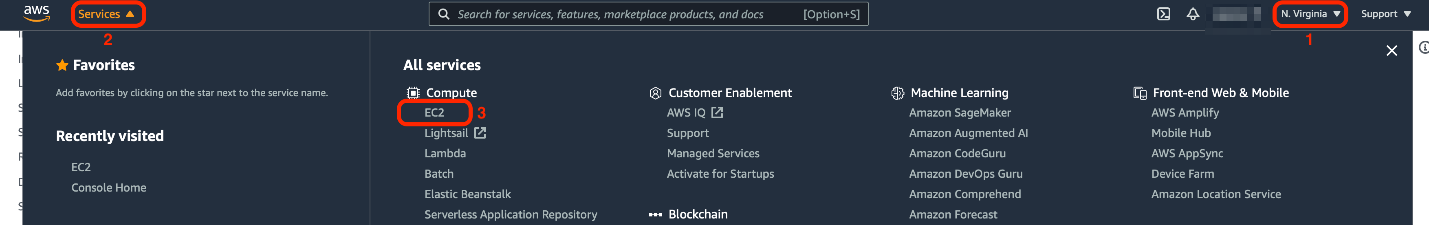
<https://aws.amazon.com/elasticloadbalancing/>

<https://aws.amazon.com/elasticloadbalancing/features/>

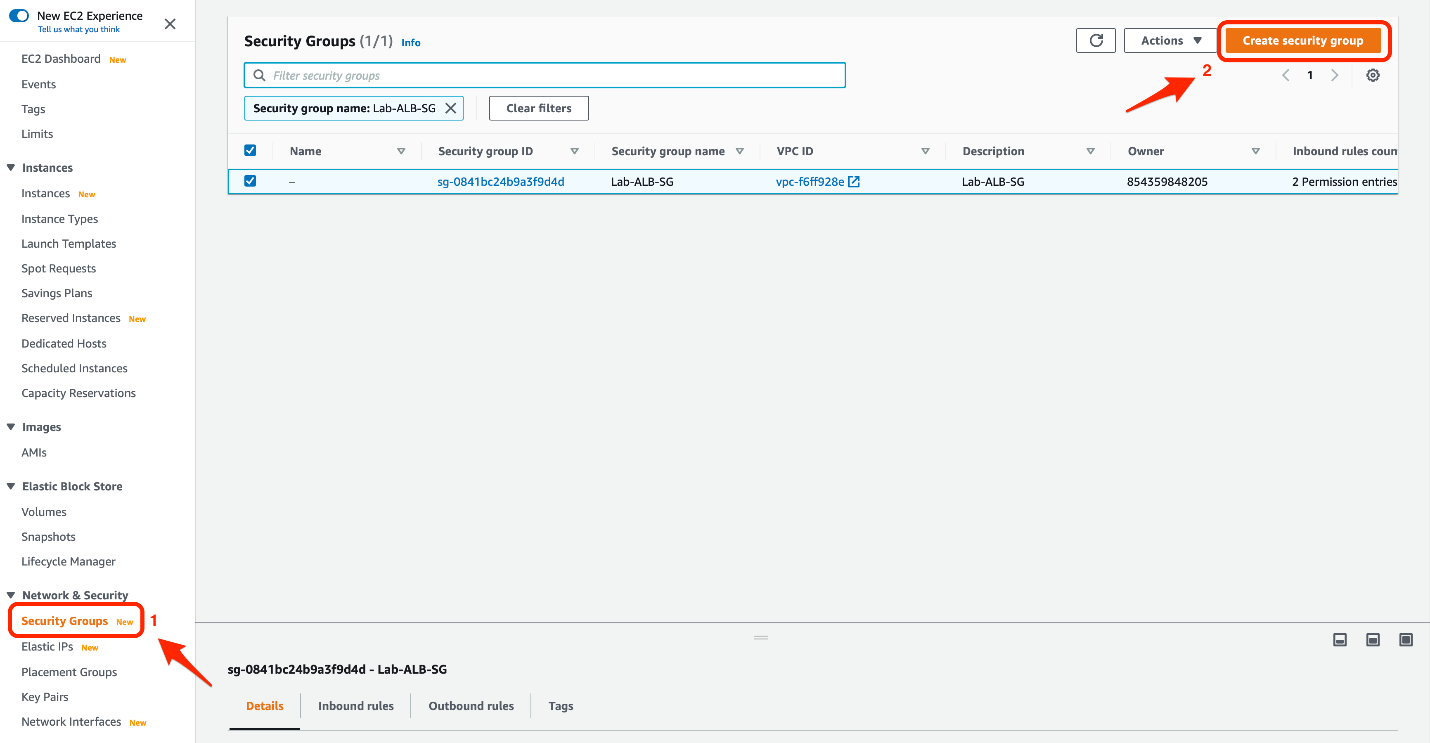
Create Launch Template for the EC2

You will use a Launch Template to automate the launch for EC2 instances via the Auto Scaling group (ASG). You will also create a new Security Group to allow incoming HTTP traffic from the ALB on port 80 as well as allow SSH access on port 22.

1. Make sure that you are in N.Virginia AWS Region. Select EC2 via the services menu on the AWS Management Console.



1. Select **Security Group** from the left sidebar and click Create **security group** to configure a new security group as follows:

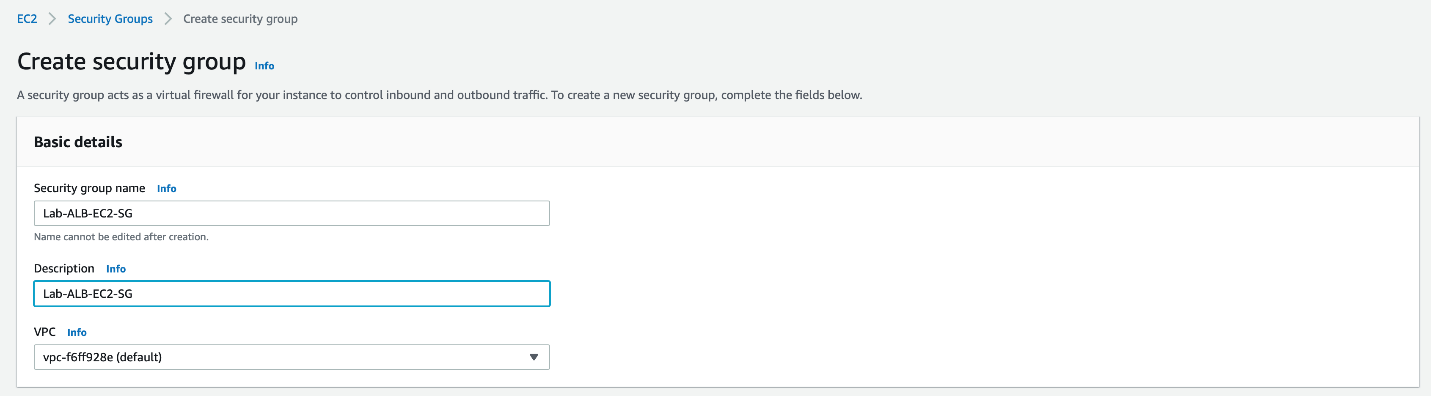


* 1. Fill in the basic details of the security group

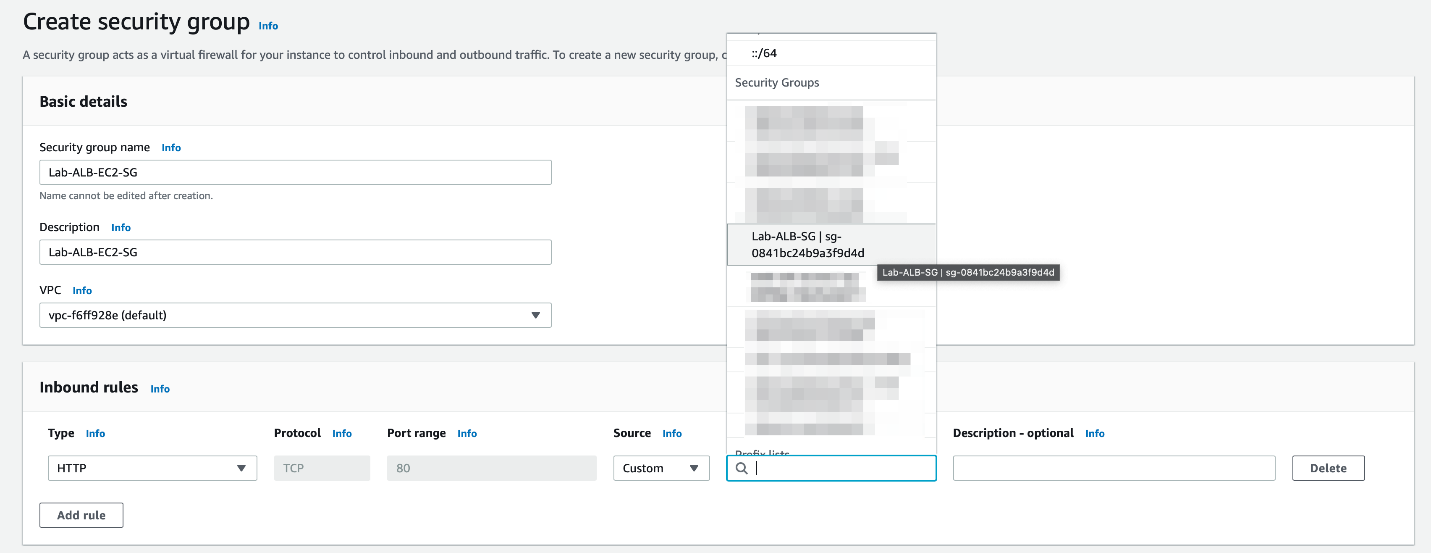
    Security group name: Lab-ALB-EC2-SG

    Description: Lab-ALB-EC2-SG

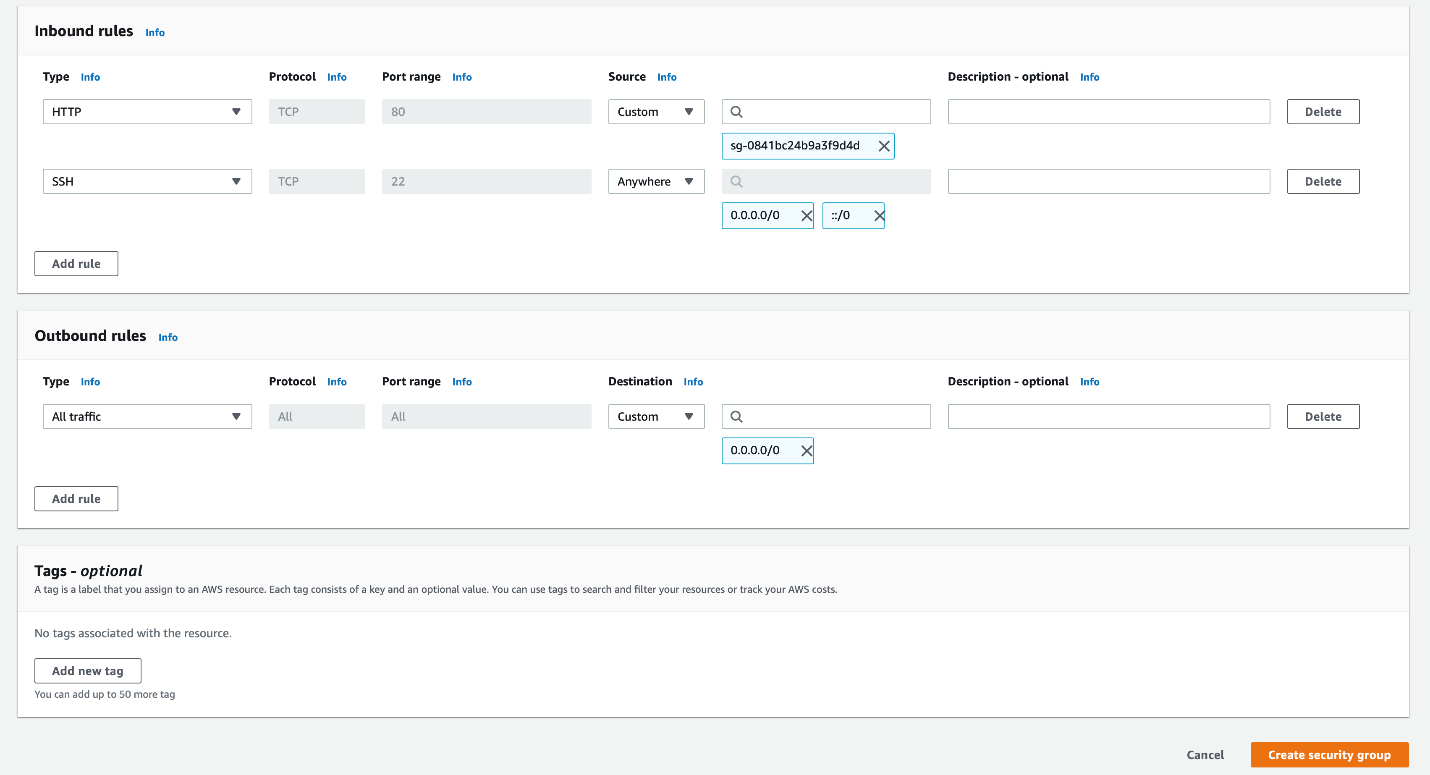
    Select the default VPC



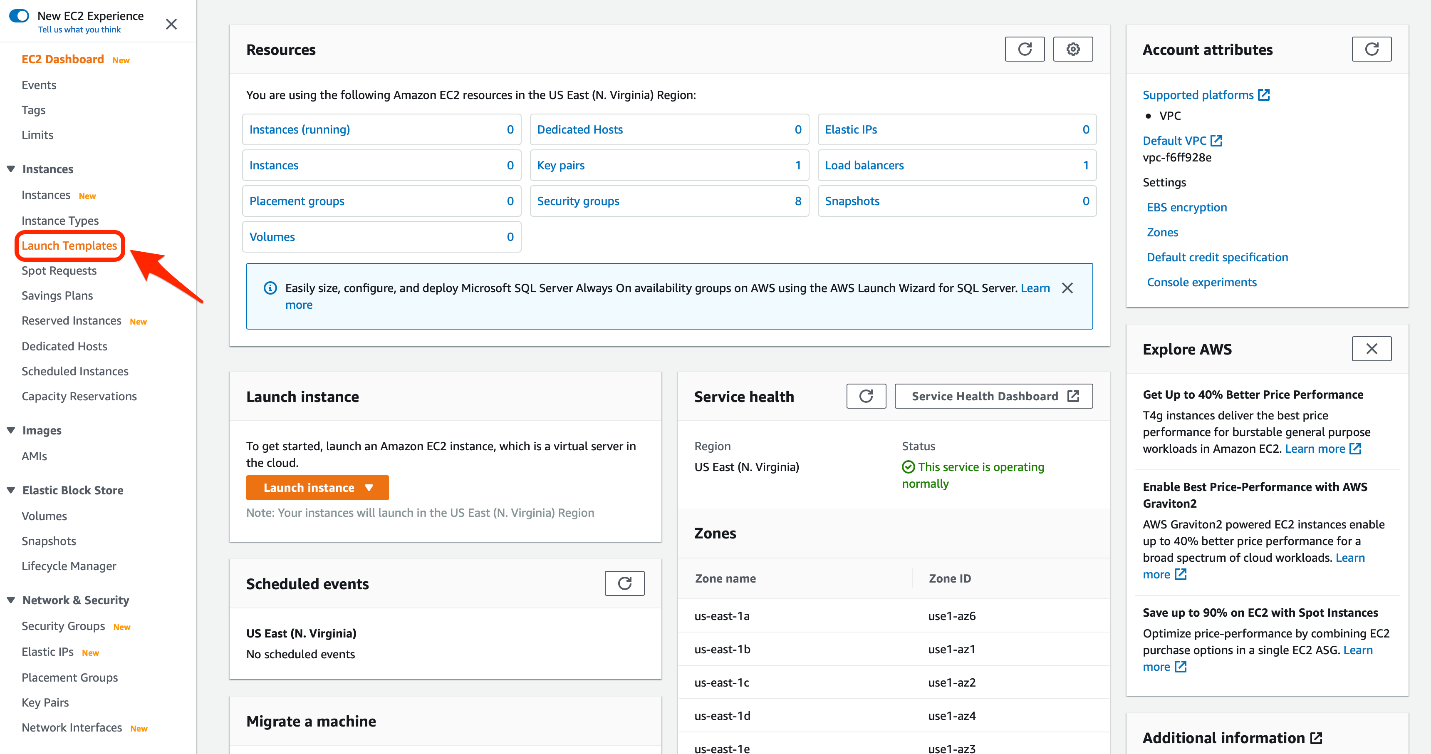
* 1. Create an inbound rule to allow incoming HTTP traffic from the ALB on port 80



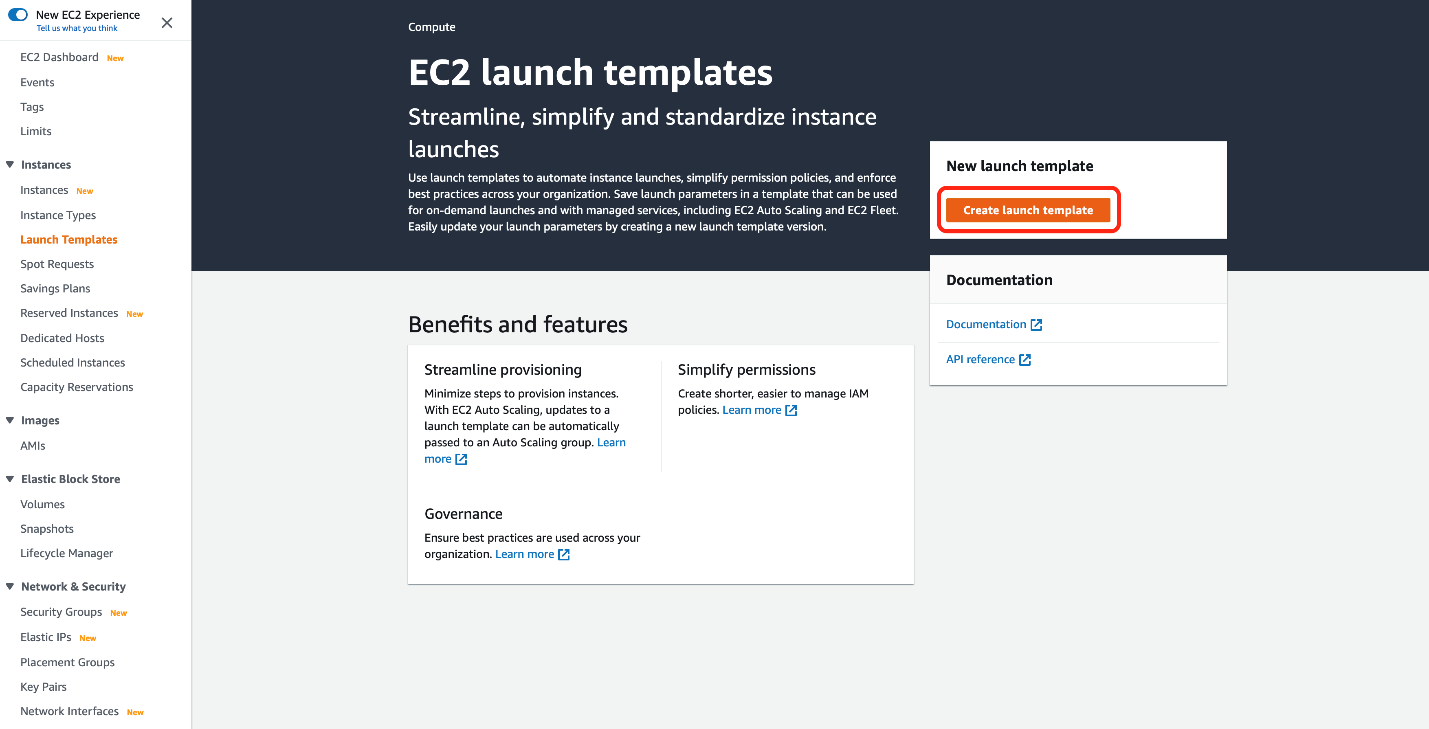
* 1. Create another inbound rule to allow SSH access on port 22. Make sure that the Security Group rules match the configuration as shown below. Click on **Create security group** to set up the security group.



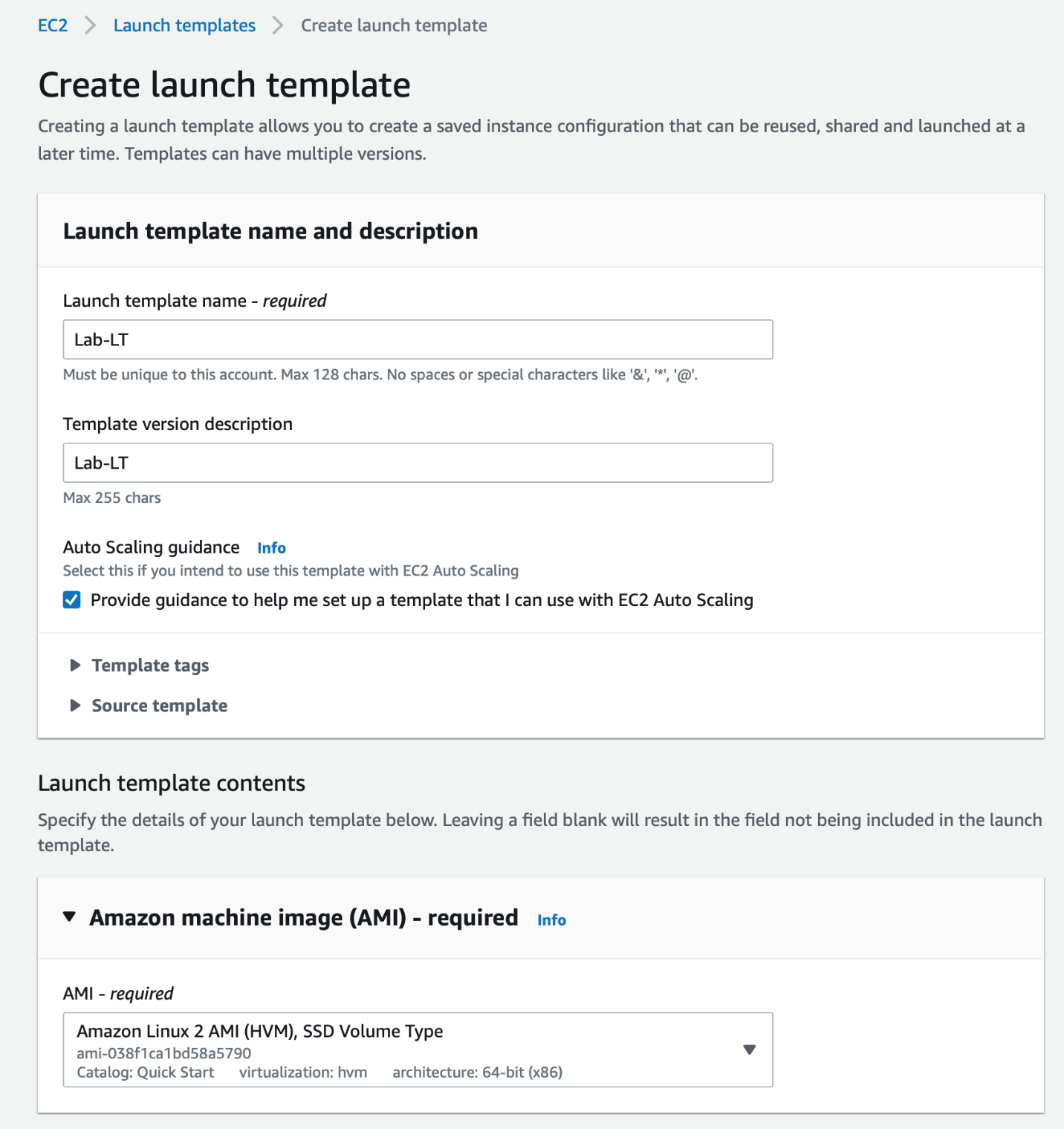
1. Select **Launch Templates** from the left sidebar



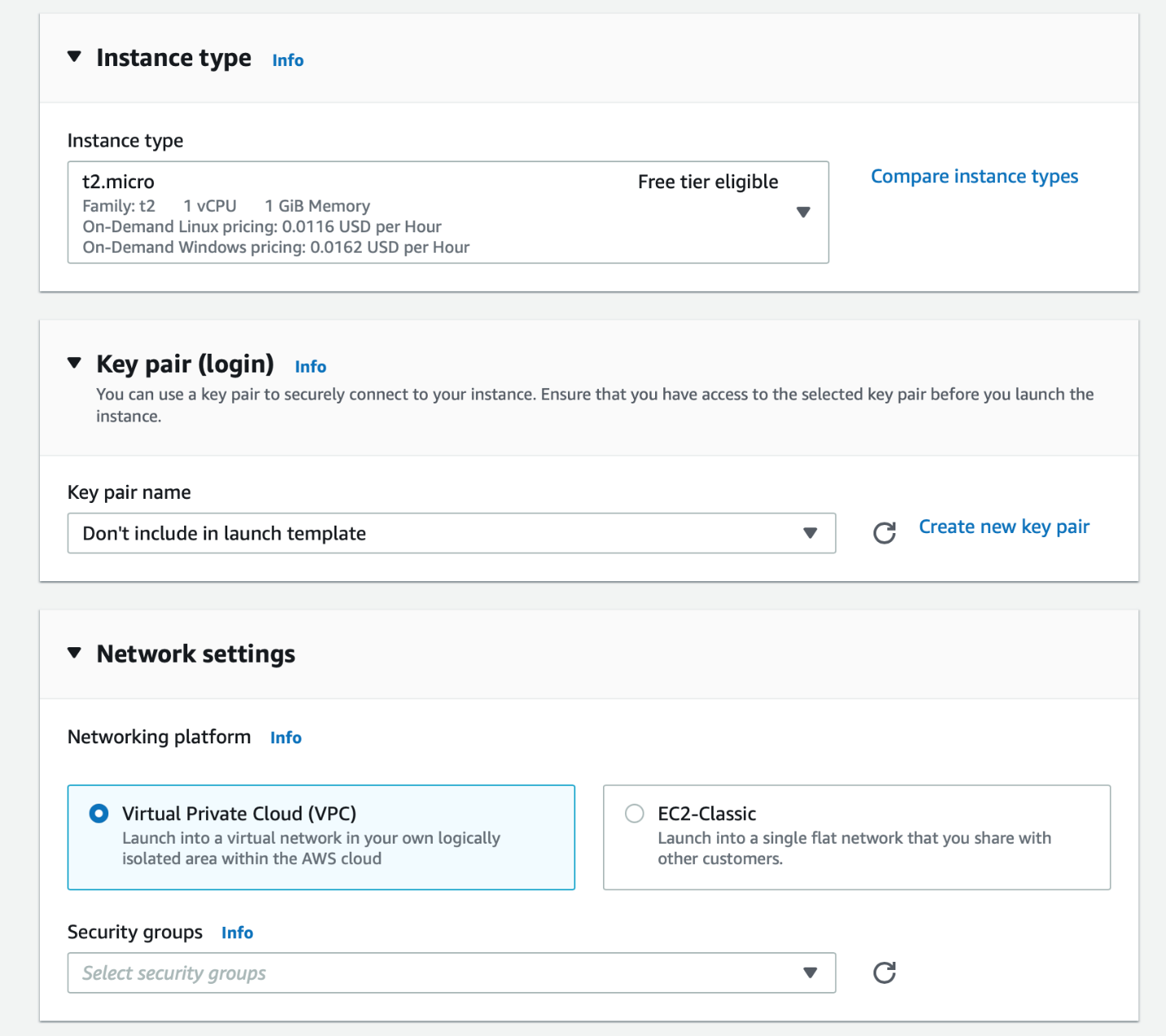
1. Click **Create launch template**



1. Set the configuration values for the Launch Template as follows:
   1. Launch template name: Lab-LT
   2. Template version description: Lab-LT
   3. Check Auto Scaling guidance
   4. Select the Amazon Linux 2 (HVM) as the AMI



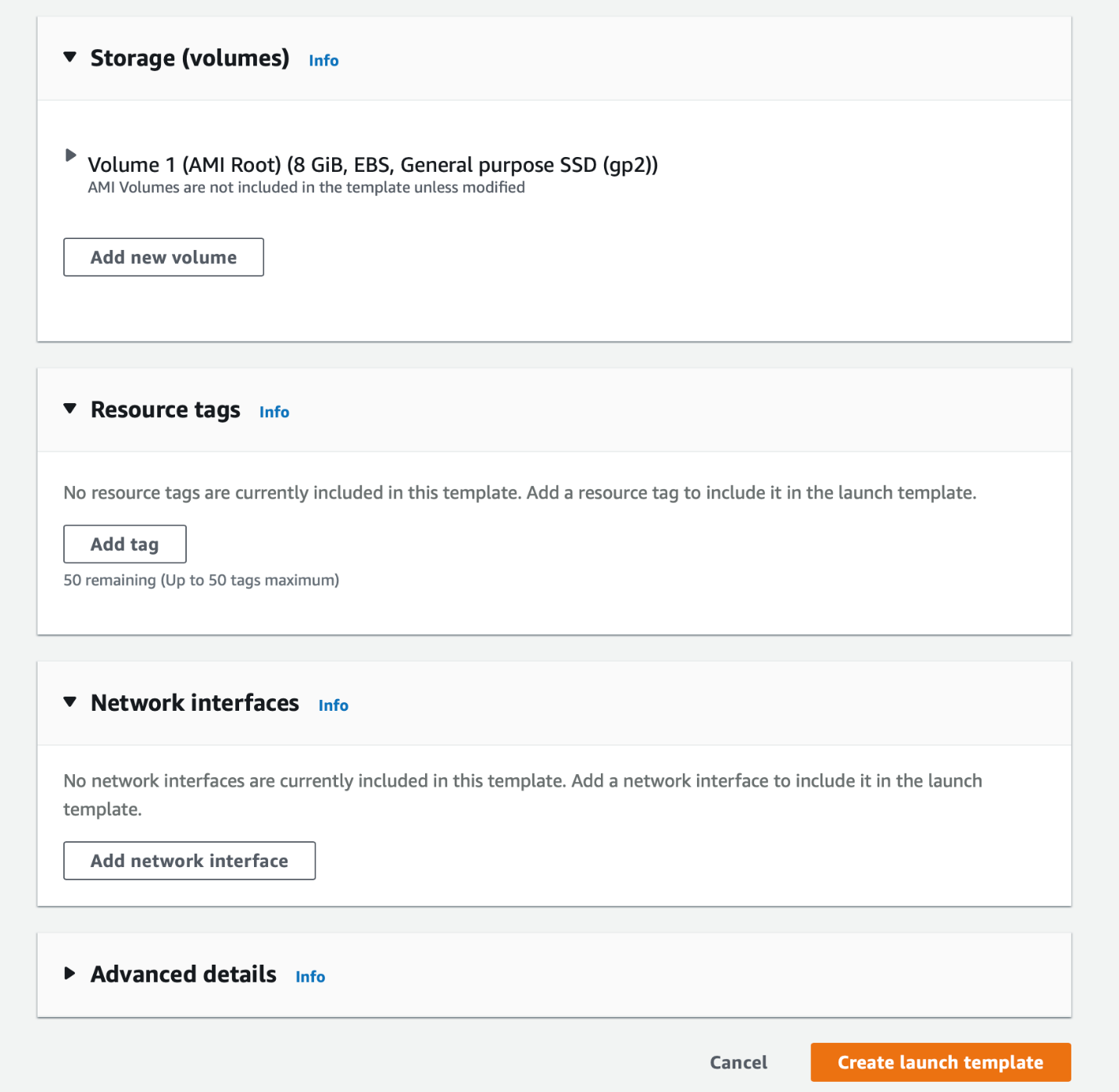
* 1. Select the instance type as t2.micro and Virtual Private Cloud as the network platform



* 1. Select the security group created in Step 2 (**Lab-ALB-EC2-SG**)



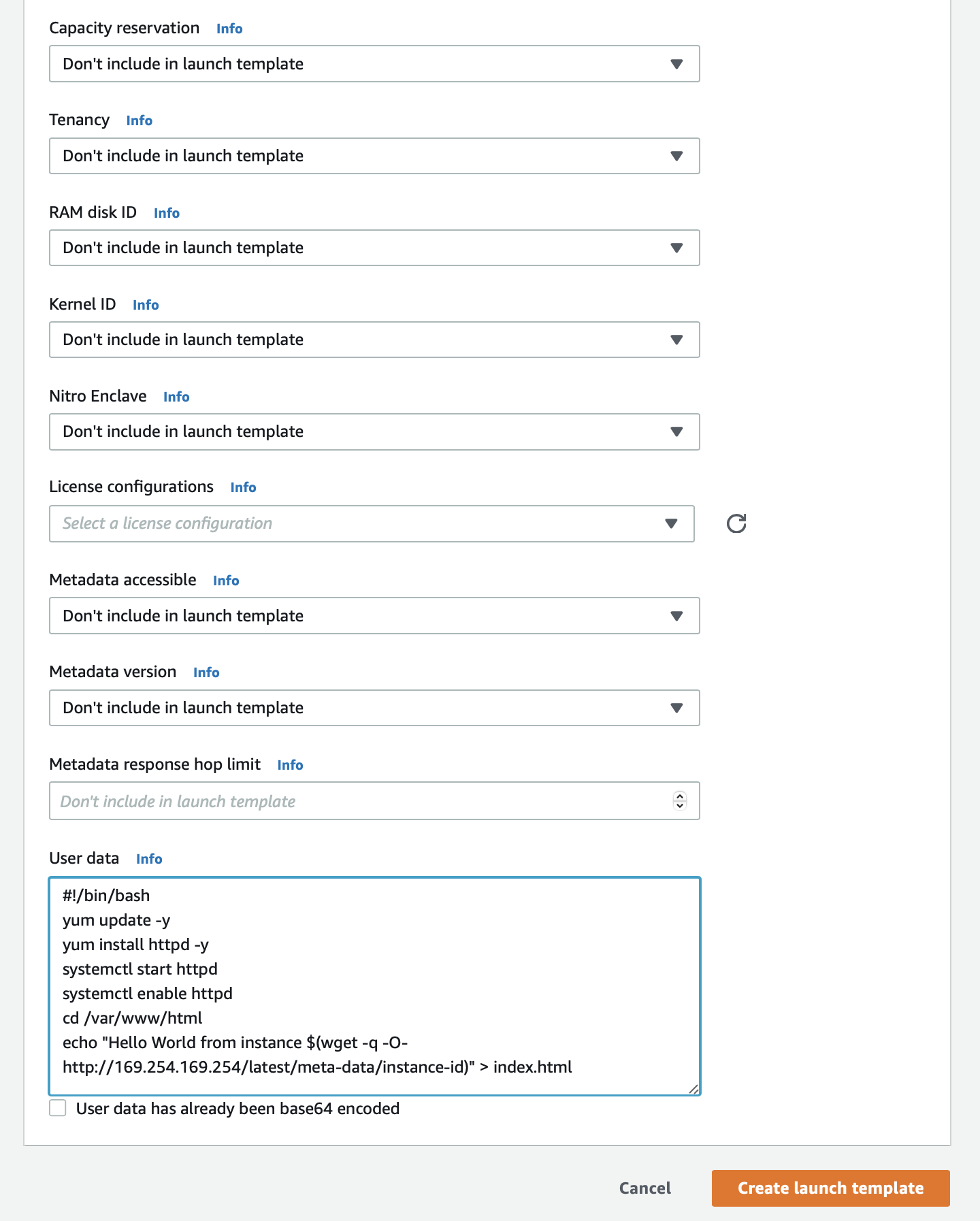
* 1. Use the default values for Storage, Resource tags and Network interfaces section



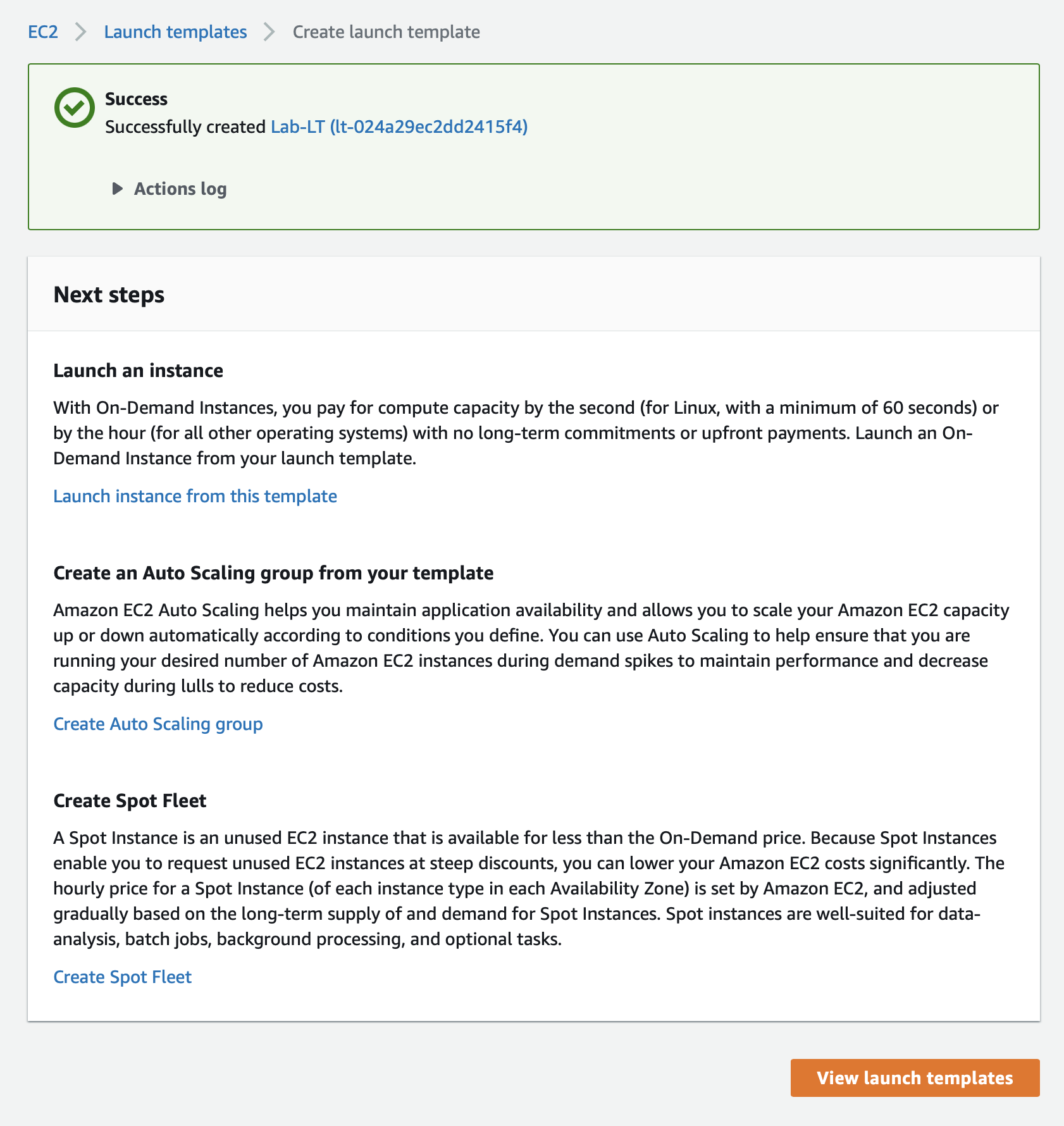
* 1. For the **Advanced details** section, fill in the **user data** as shown below.
     1. #!/bin/bash
     2. yum update -y
     3. yum install httpd -y
     4. systemctl start httpd
     5. systemctl enable httpd
     6. cd /var/www/html
     7. echo "Hello World from instance $(wget -q -O- http://169.254.169.254/latest/meta-data/instance-id)" > index.html

The user data installs and enables the Apache web server and creates an `index.html` file in `/var/www/html` directory. The html file has a simple text to help identify a given instance.

At the end, click on the **Create launch template** button to complete the set up



1. You should see that your launch template has been successfully created. Click **View launch templates** button to complete the Task 2 of this lab.



<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html>

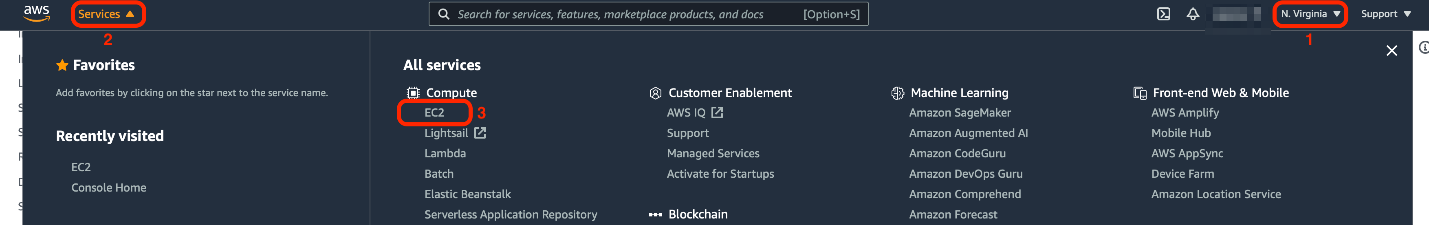
<https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-groups.html>

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html>

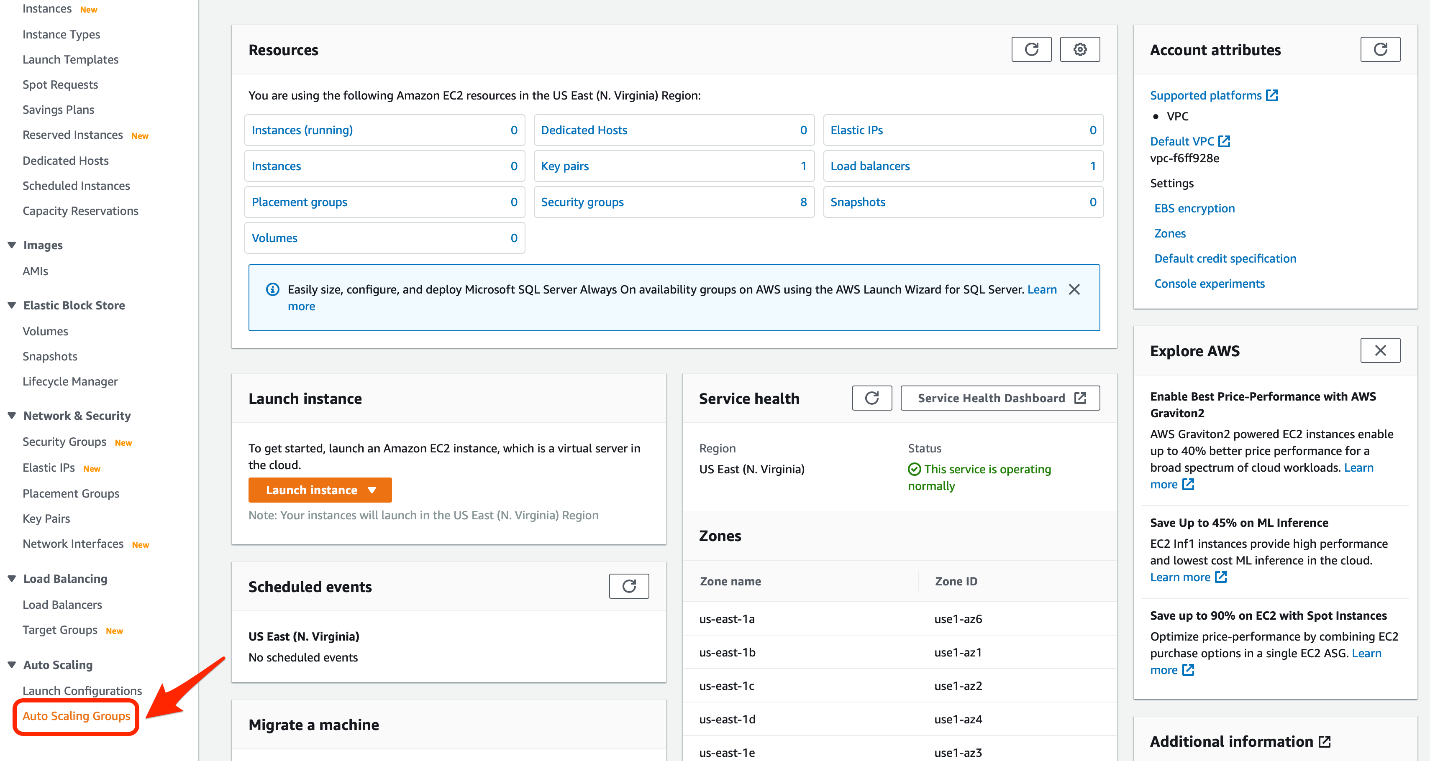
Configure Auto Scaling groups to leverage the Launch Template to launch EC2 and use the ALB to distribute the tracfic to the provisioned EC2

You will create an Auto Scaling group (ASG) to manage a fleet of EC2 instances by using a target scaling policy to maintain the fleet at a target CPU utilization of 40%. The ASG would use the Launch Template created in Task 2 to automate the launch for EC2 instances. The Application Load Balancer (created in Task 1) would distribute the traffic to the provisioned EC2 instances via this ASG.

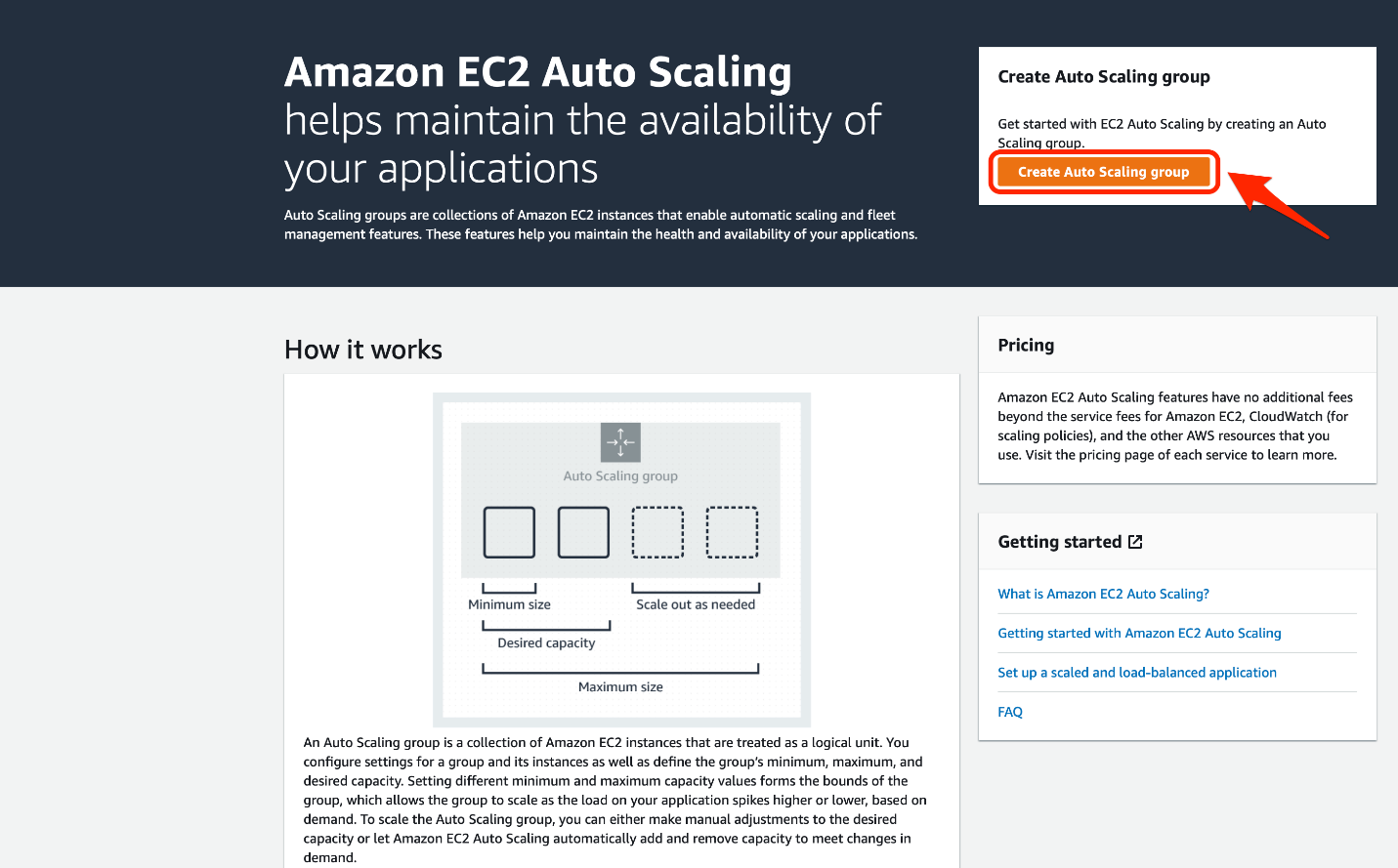
1. Make sure that you are in N.Virginia AWS Region. Select EC2 via the services menu on the AWS Management Console.



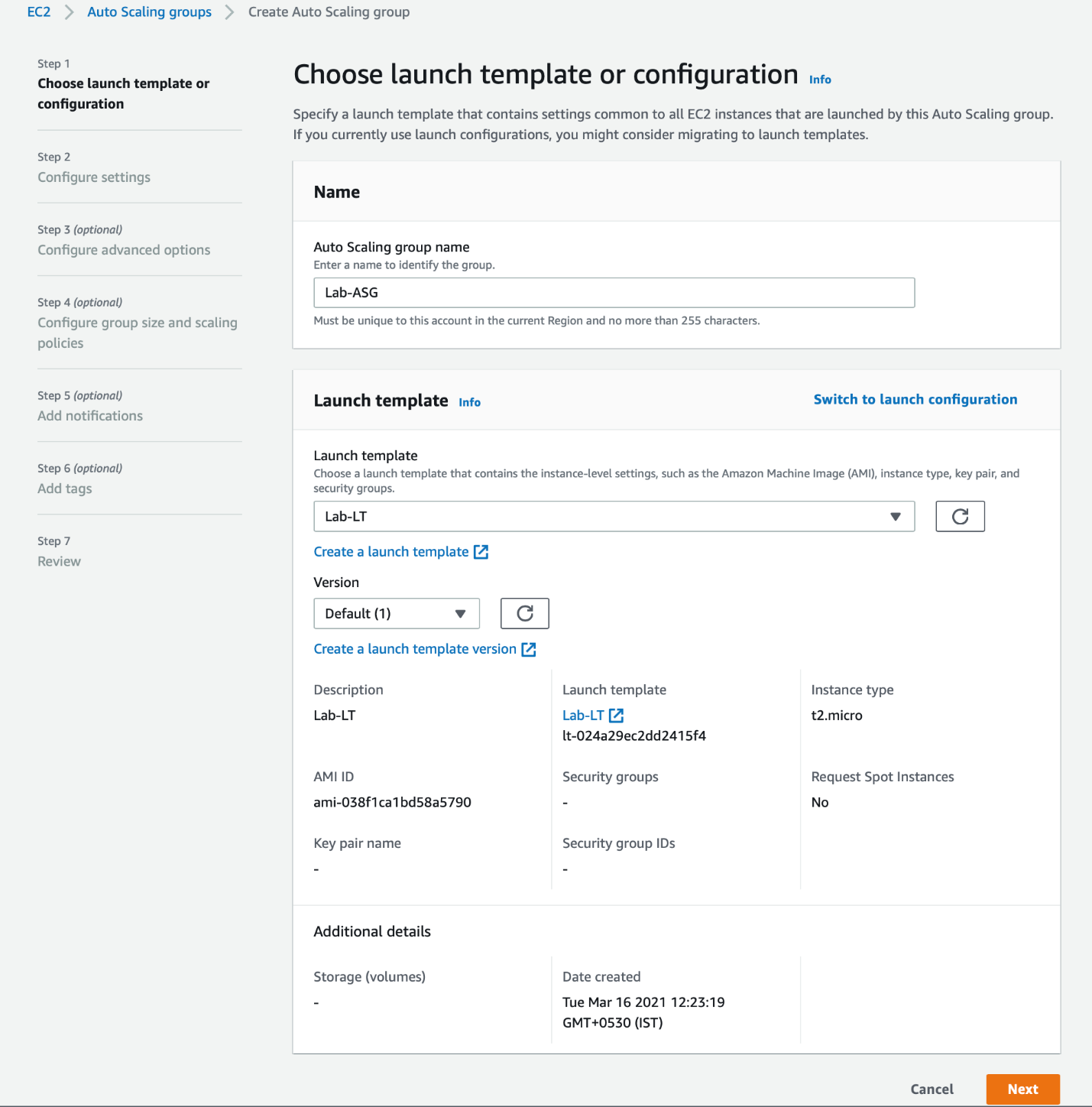
1. Select **Auto Scaling Groups** from the left sidebar



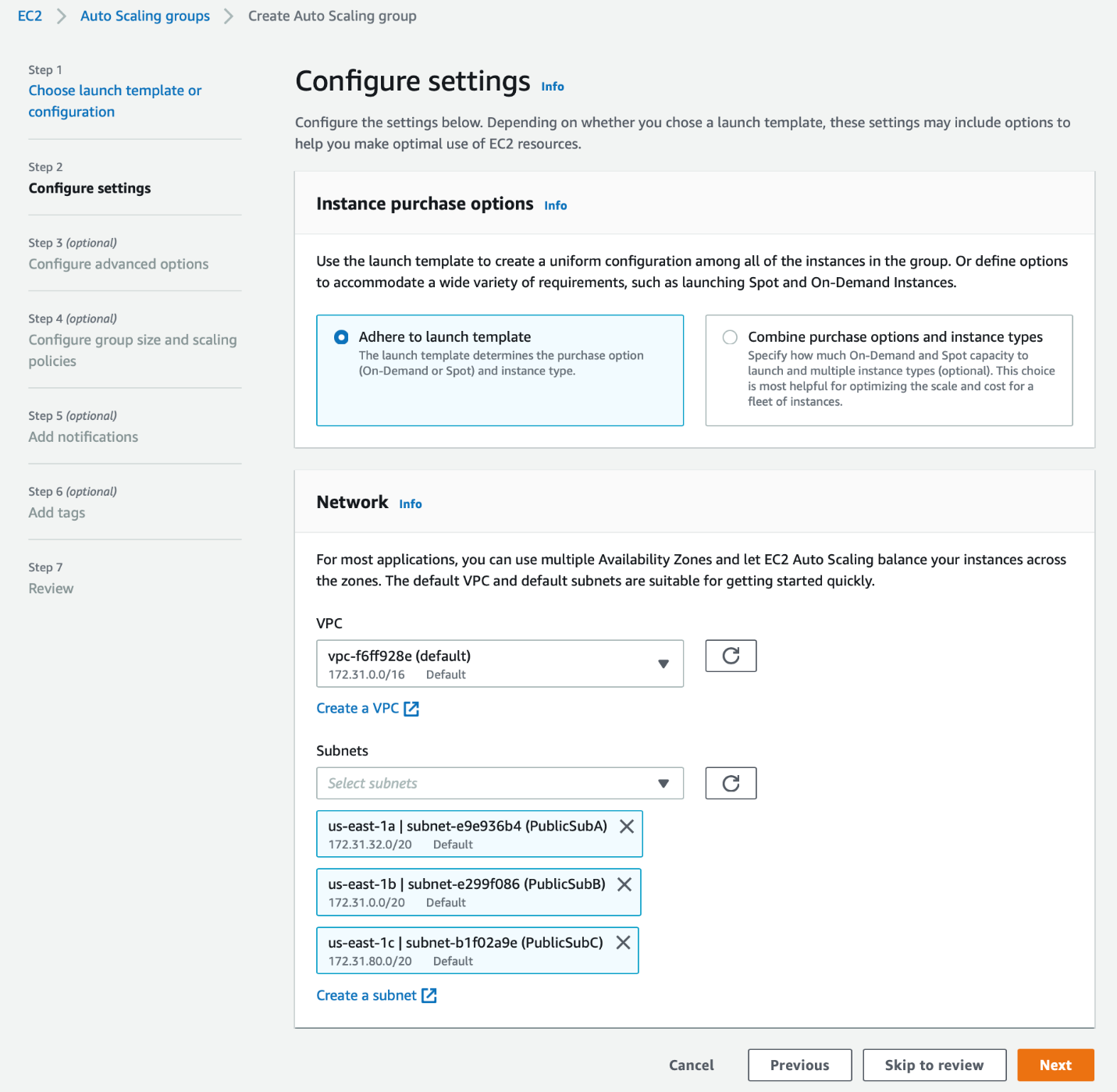
1. Click **Create Auto Scaling group**



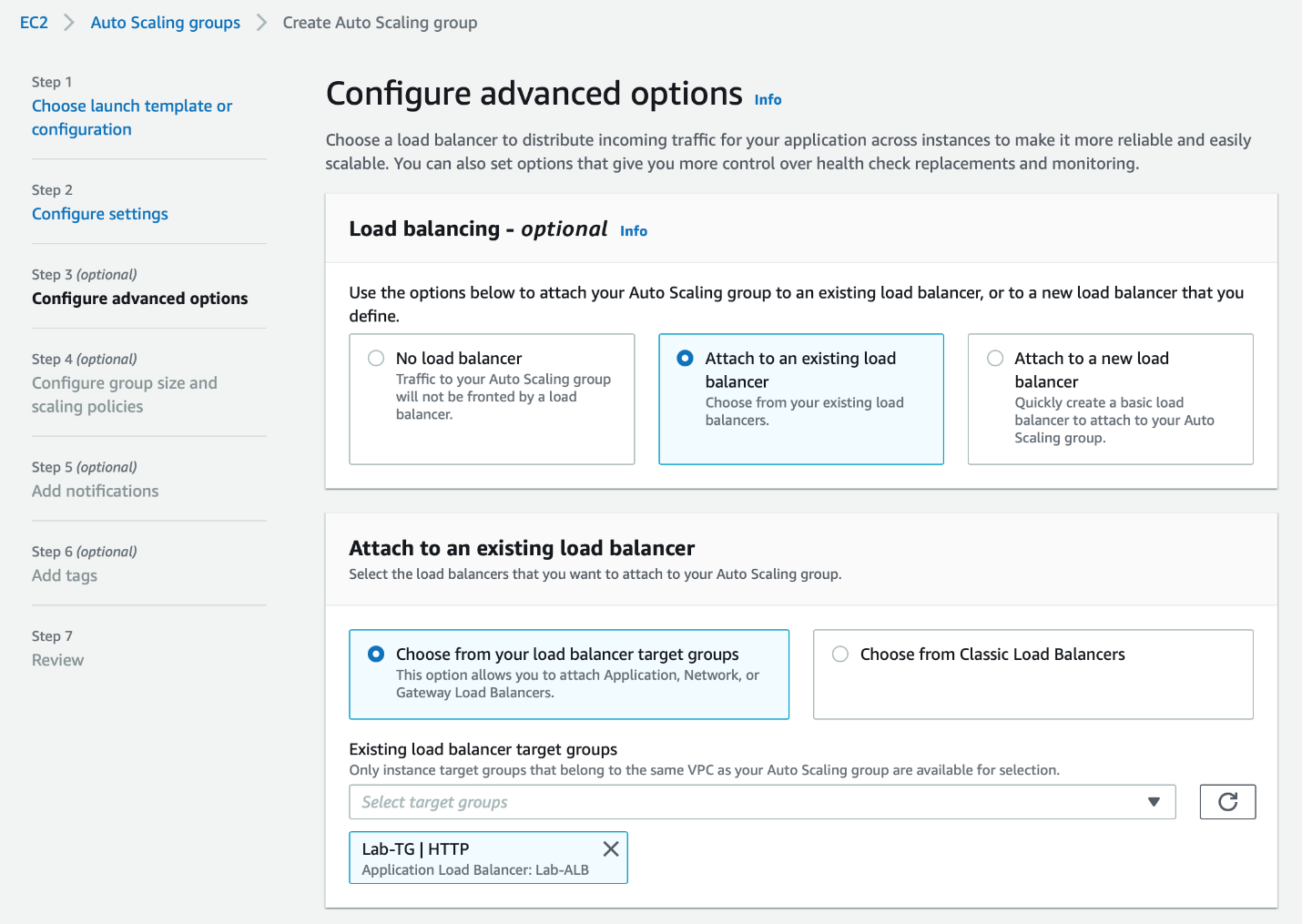
1. Enter a name and select a Launch Template for the Auto Scaling group as follows:
   1. Auto Scaling group name: Lab-ASG
   2. Choose the Launch Template (Lab-LT) created in Task 2 and click on Next



1. Configure the settings to **Adhere to launch template** and select the default VPC and the subnets should correspond to these three Availability Zones (AZs) - us-east-1a, us-east-1b, us-east-1c (make sure that these AZs are same as the ones selected in Task 1 while creating the Application Load Balancer).



1. Configure the advanced options for Application Load Balancer and health checks as follows:
   1. Attach the existing Application Load Balancer (created in Task 1) and choose the given Load Balancer target groups



* 1. Use the default values for the health checks

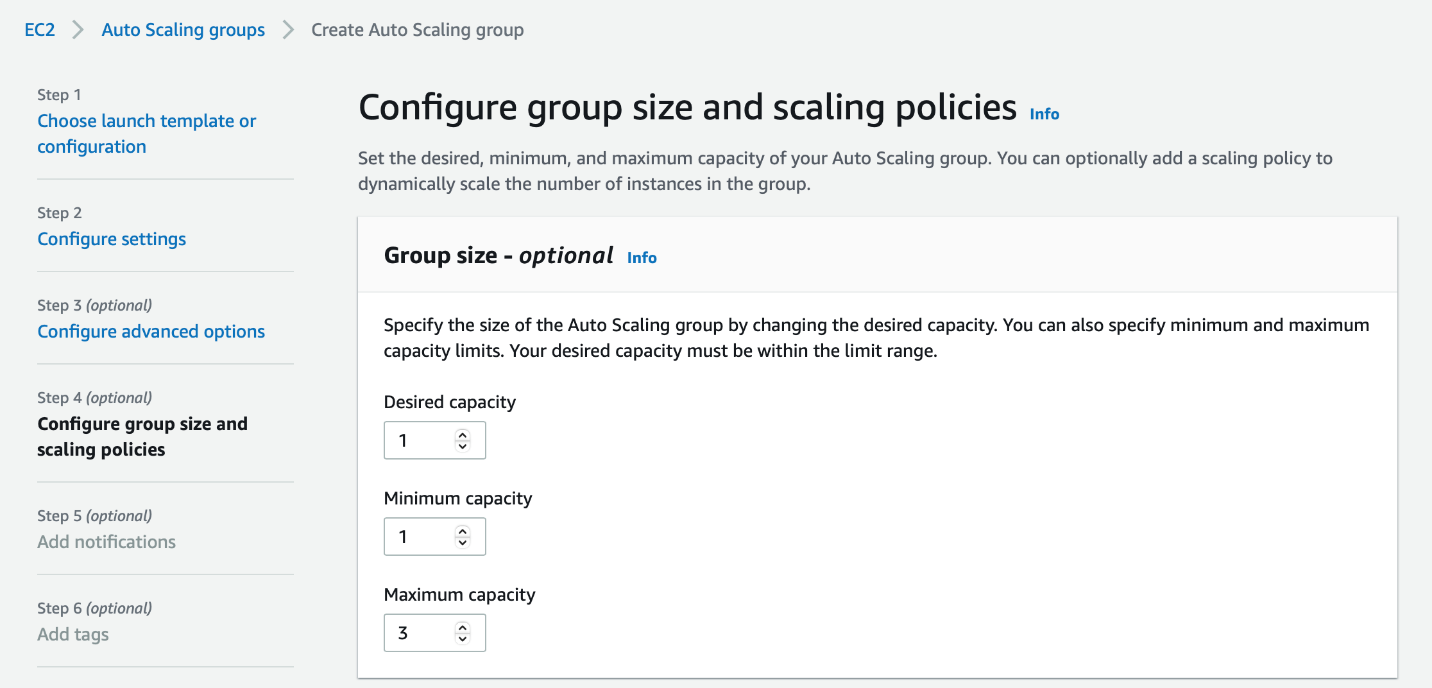


1. Configure the group size and scaling policies as follows:
   1. Set the desired, minimum and maximum capacity of your Auto Scaling group like so:

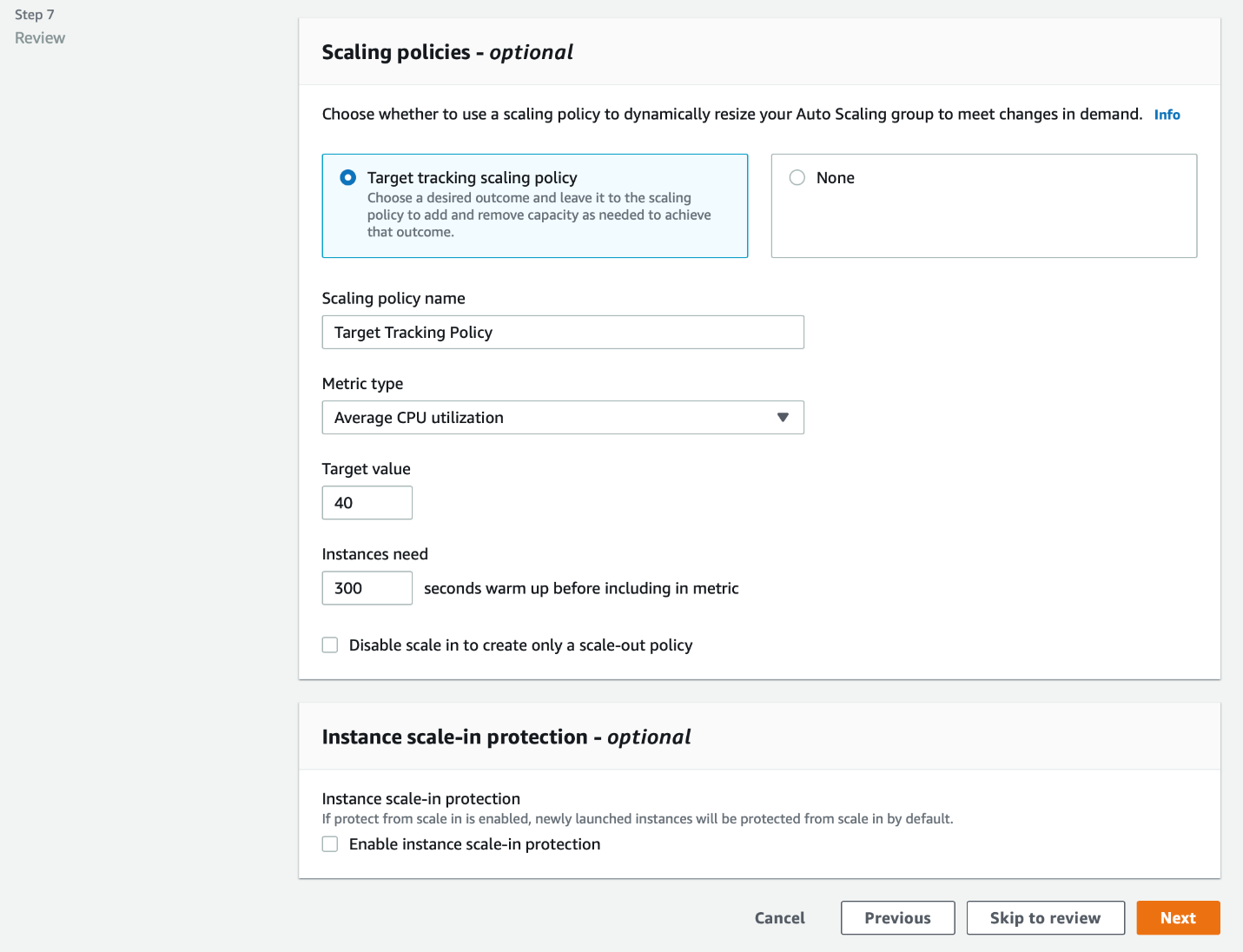
    Set **Desired capacity** to 1,

    Set **Minimum capacity** to 1

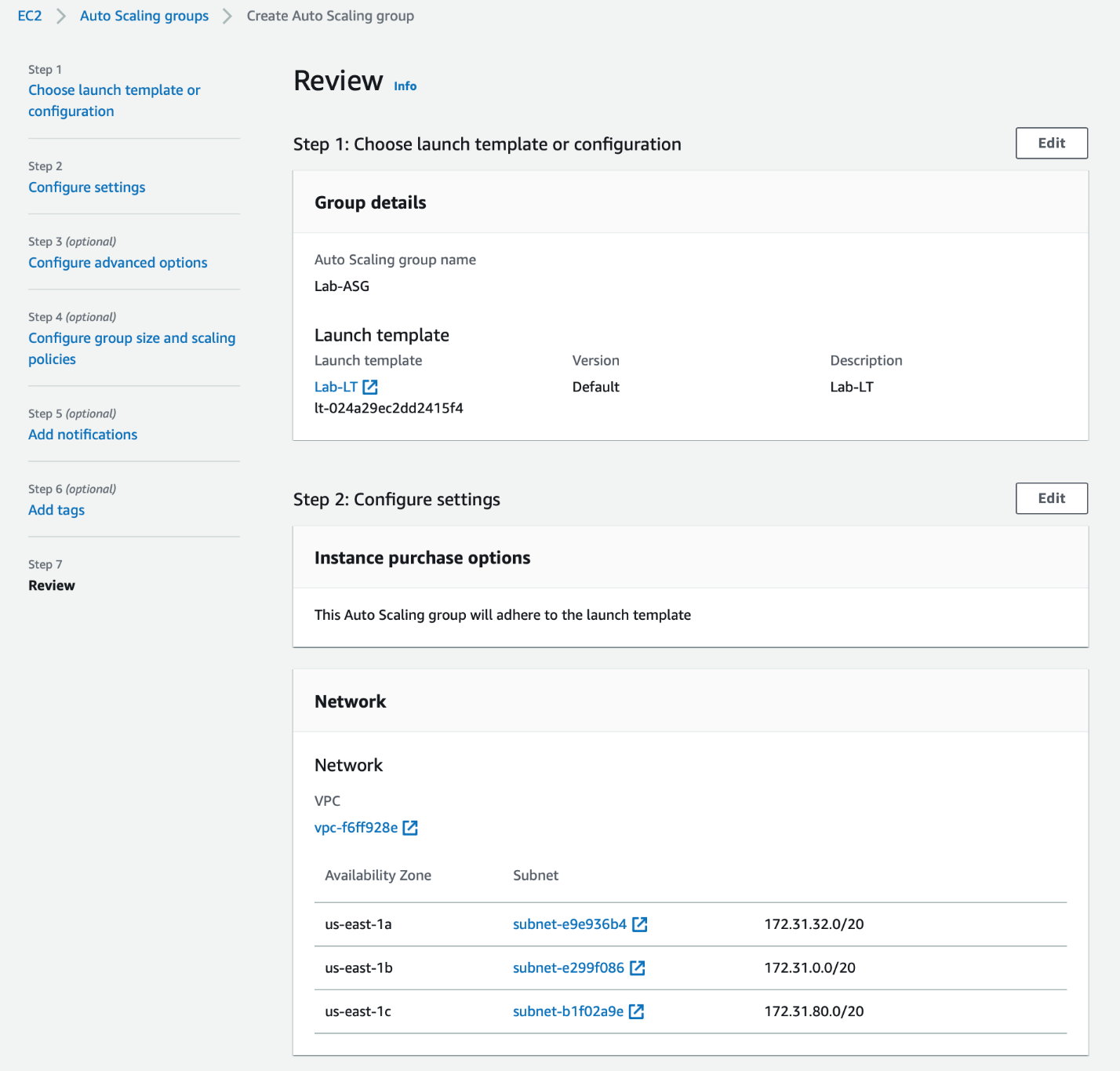
    Set **Maximum capacity** to 3

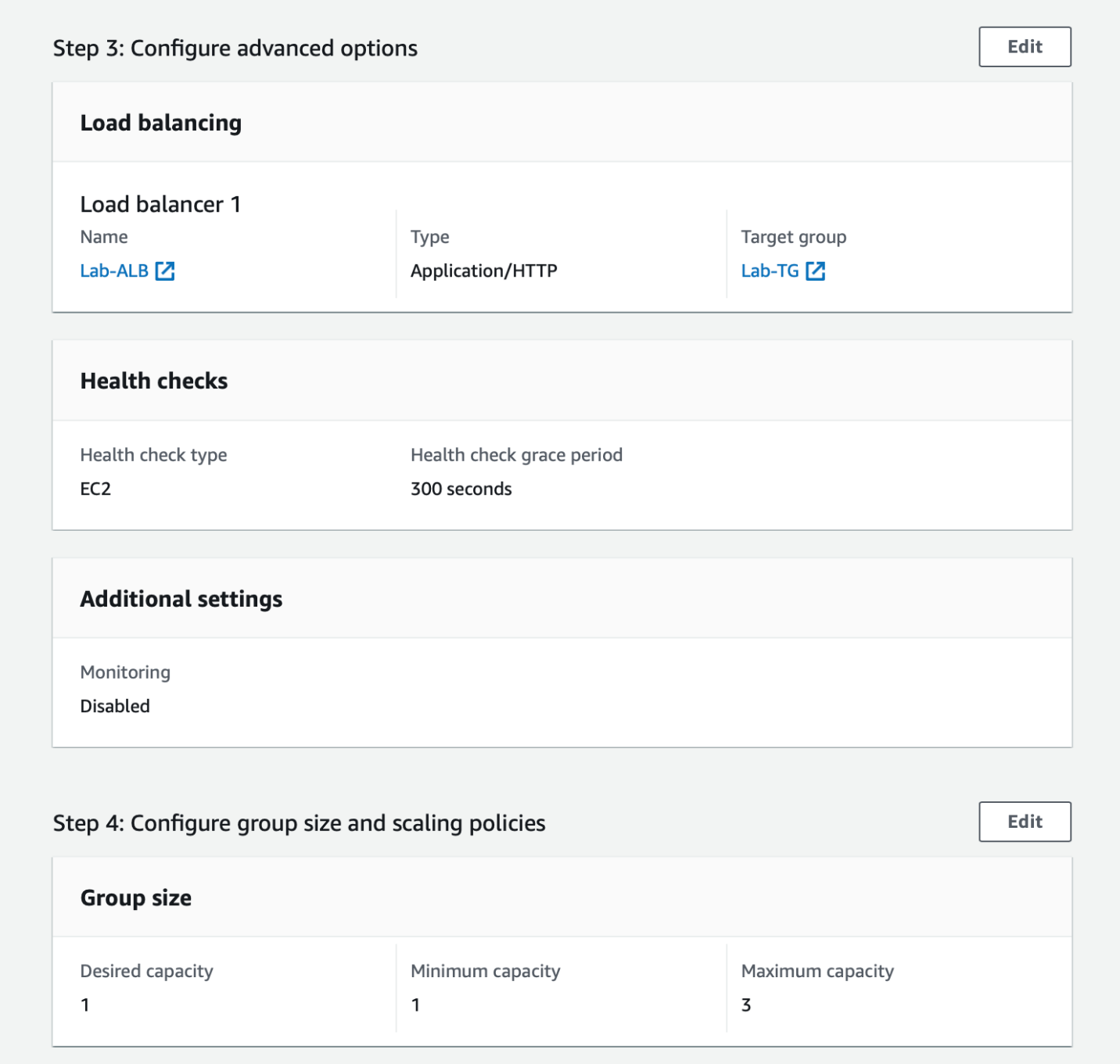


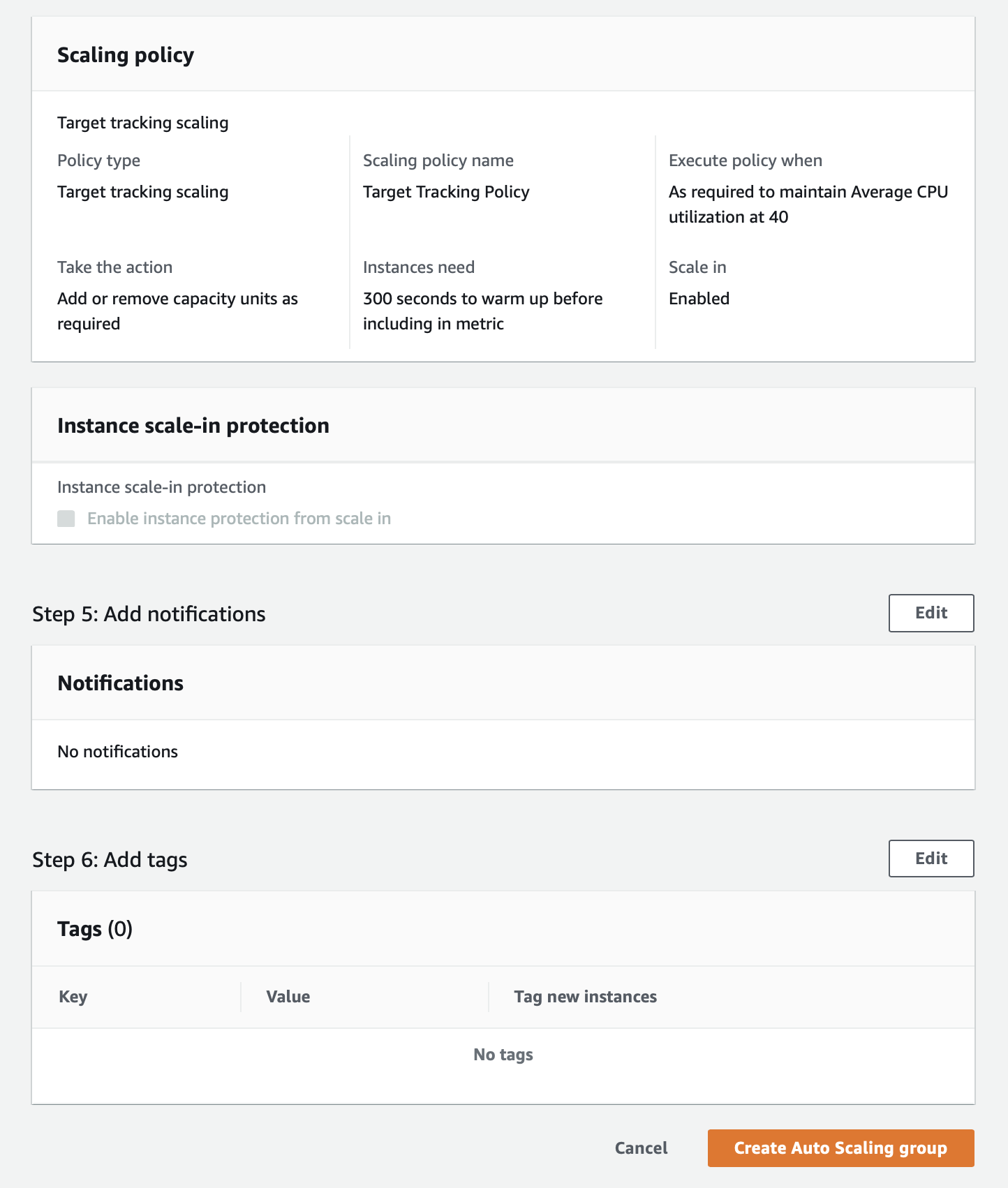
* 1. Configure scaling policy to use a **Target tracking scaling policy** with **Average CPU Utilization** as the **Metric Type**. Set the **Target value** as **40** and the instance warm-up time as **300** seconds. Leave the other values for scale-in as unchecked.



1. Review the section-wise configurations for the Auto Scaling group. Then click on **Create Auto Scaling group** to complete this step.







<https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html>

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-groups.html>

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/launch-templates.html>

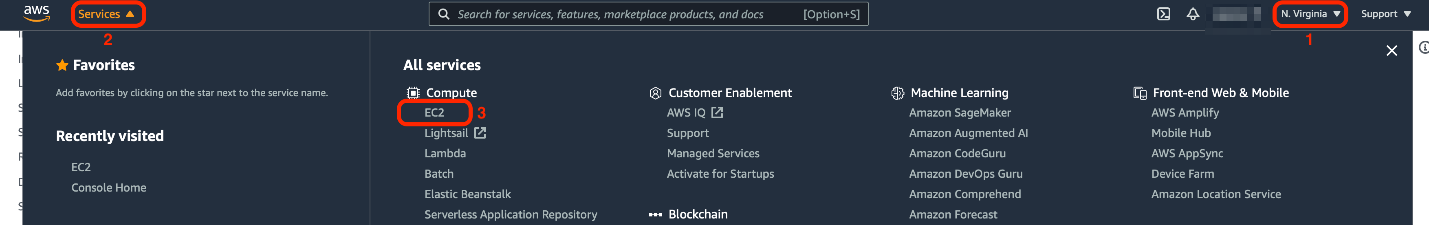
<https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scale-based-on-demand.html>

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scaling-target-tracking.html>

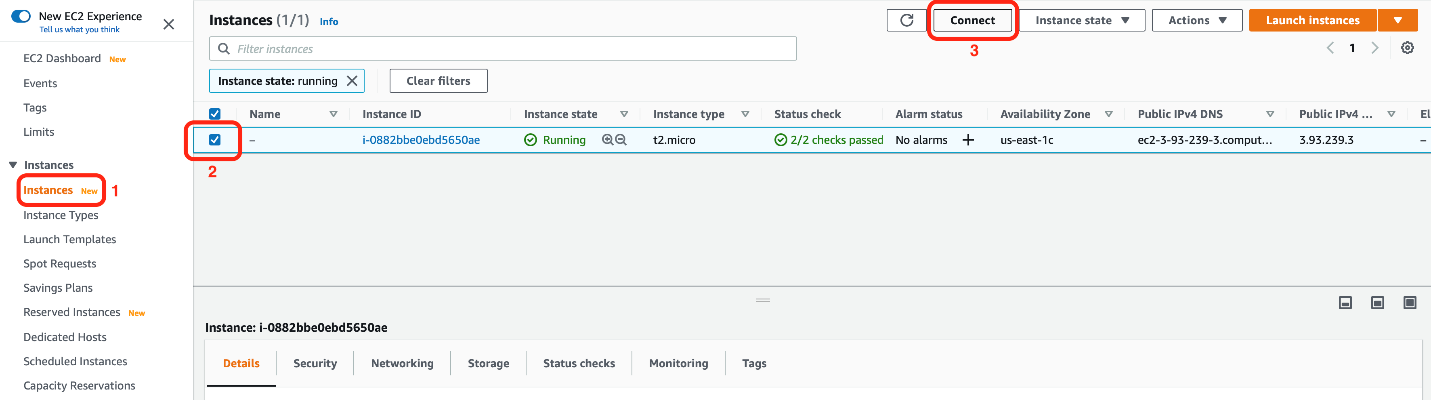
Simulate load on the EC2 by using stress utility to demonstrate horizontal scaling for the EC2 instance via Auto Scaling group

You will use the **stress** utility to simulate CPU load on the worker EC2 instances managed by the Auto Scaling group. The increased CPU utilization would trigger the target tracking scaling policy to increase the number of instances to three (maximum capacity set in Task 3). This demonstrates horizontal scaling for the EC2 instances to handle the increased workload.

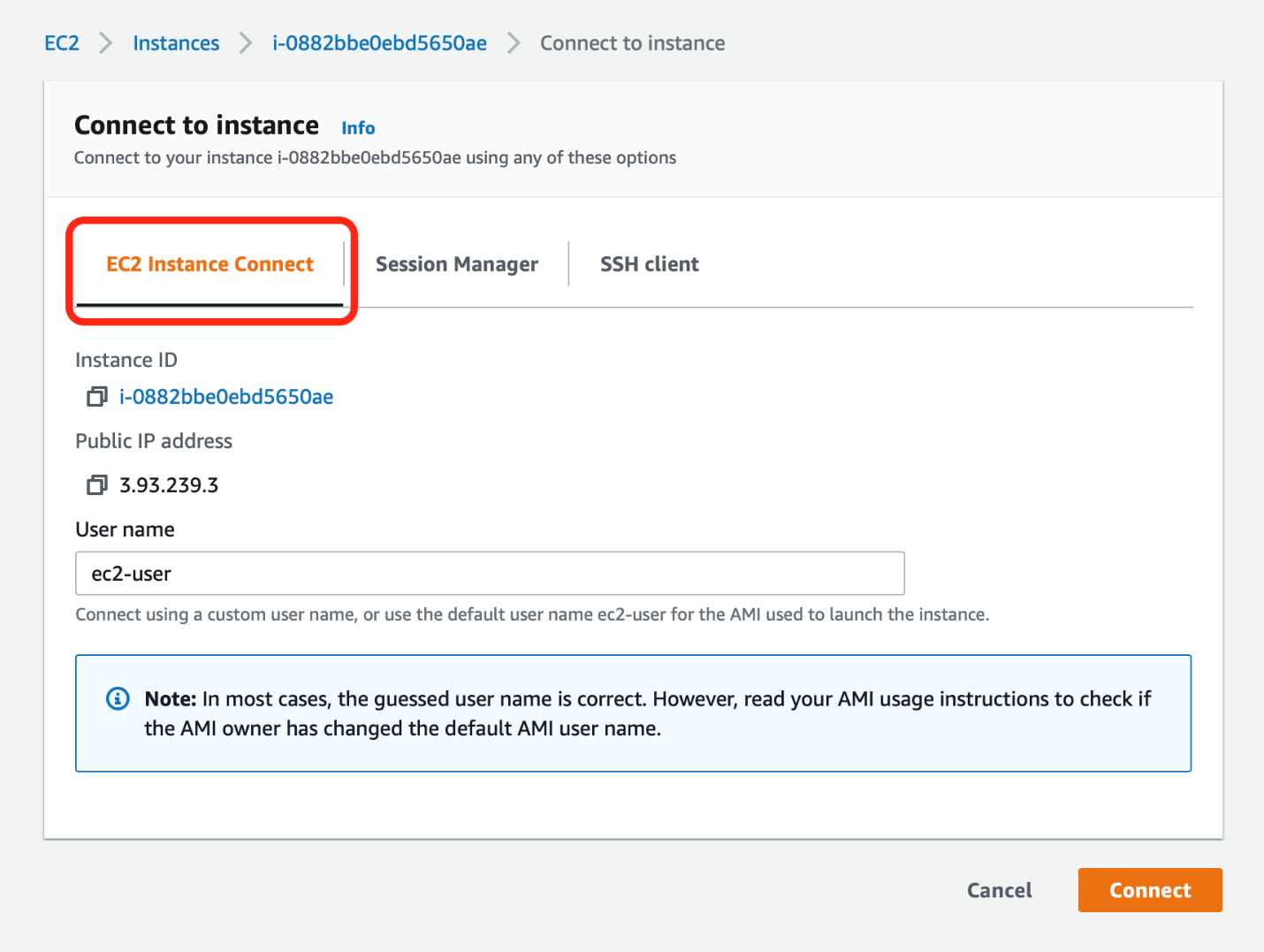
1. Make sure that you are in N.Virginia AWS Region. Select EC2 via the services menu on the AWS Management Console



1. Select **Instances** from the left sidebar, select the instance created by the Auto Scaling group in Task 3 and click on **Connect**

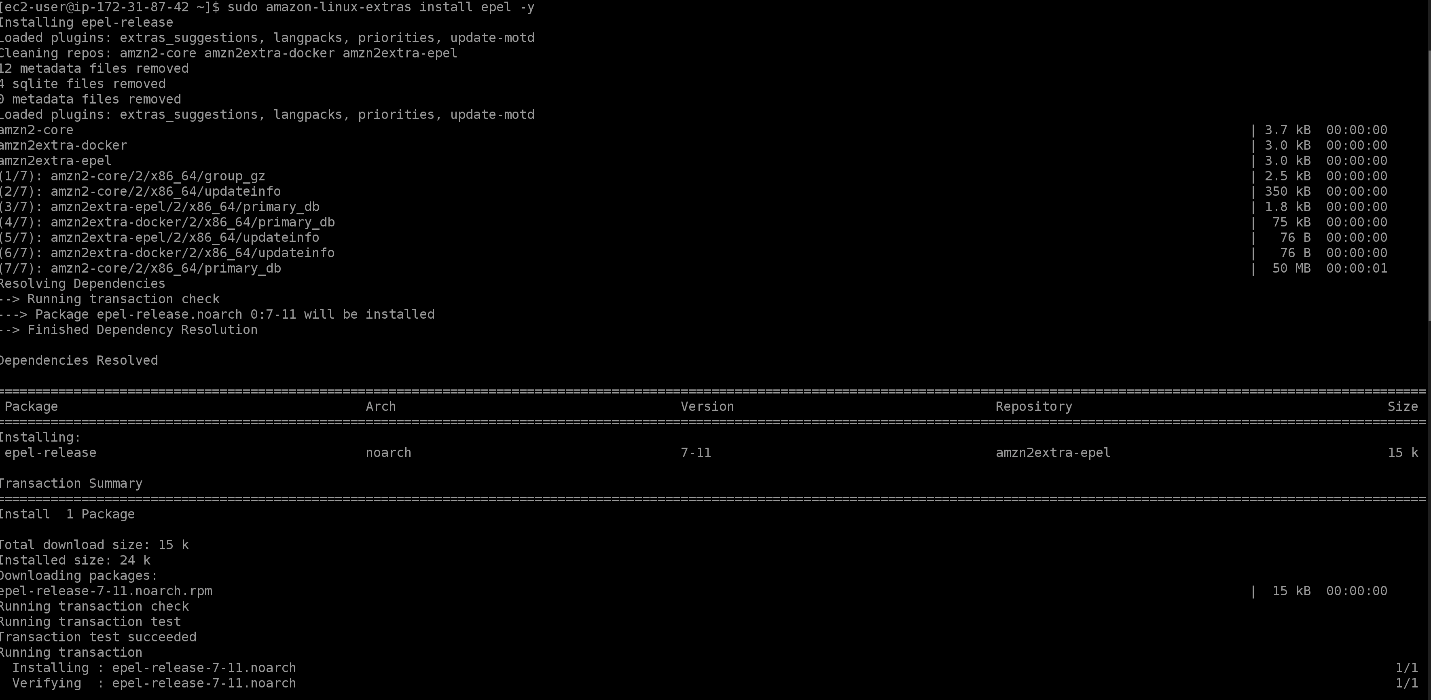


1. Use the **EC2 Instance Connect** option to connect to the selected instance

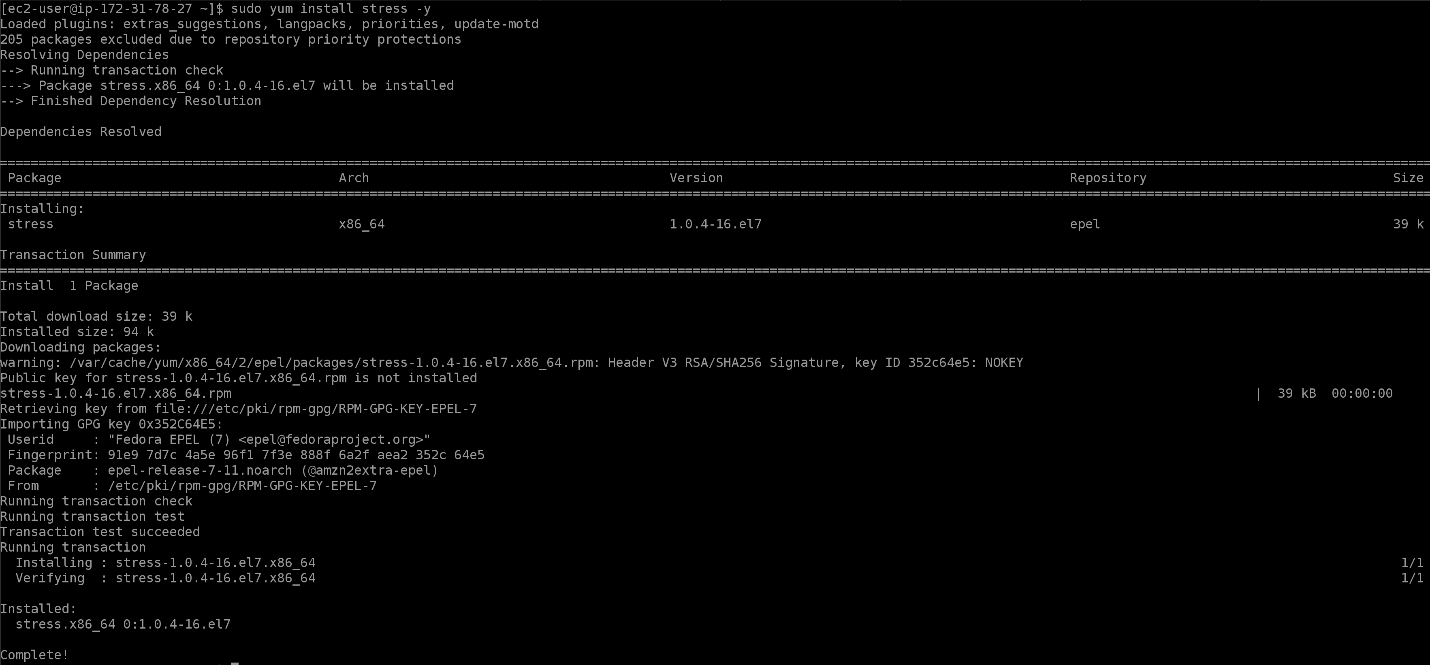


1. Run the following commands to install the **stress** utility on the EC2 instance

sudo amazon-linux-extras install epel -y



sudo yum install stress -y

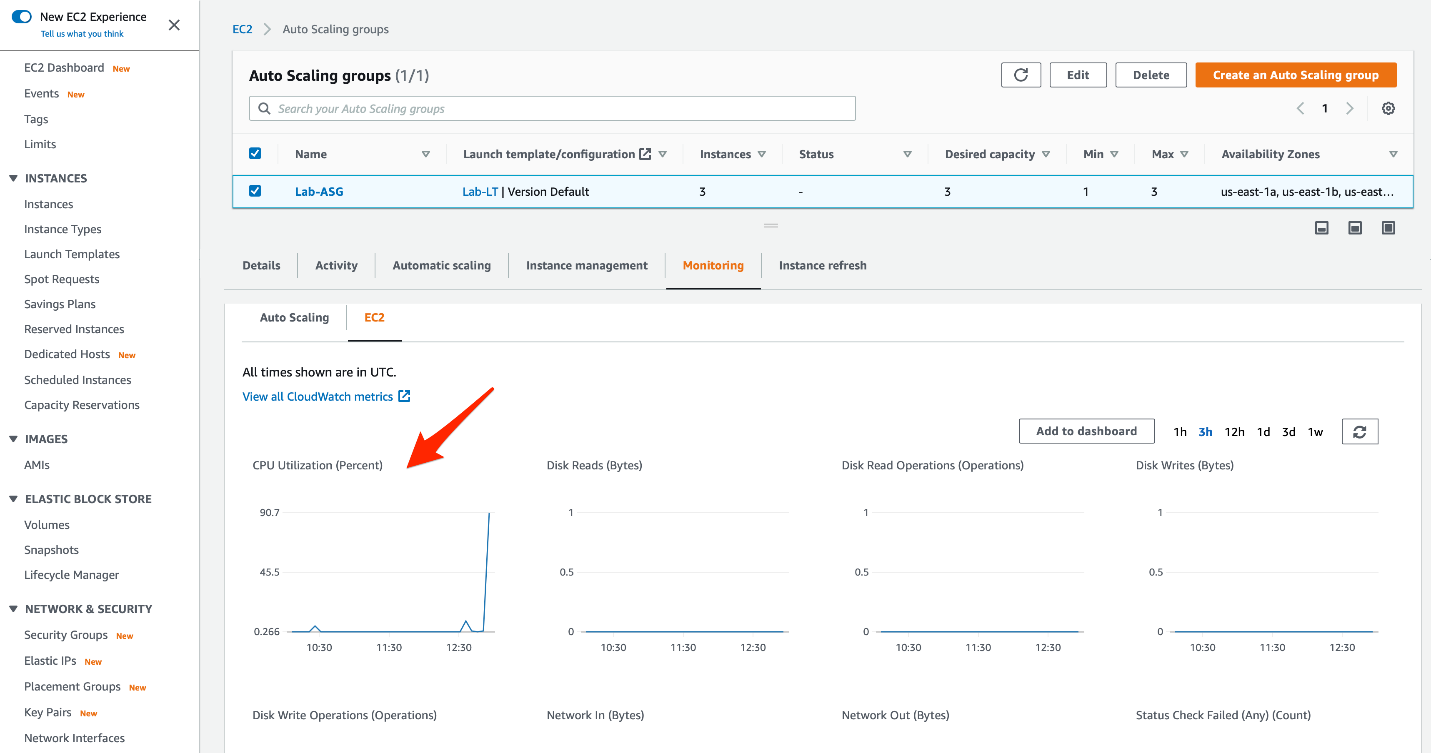


1. Invoke the stress test on the instance using the following command:

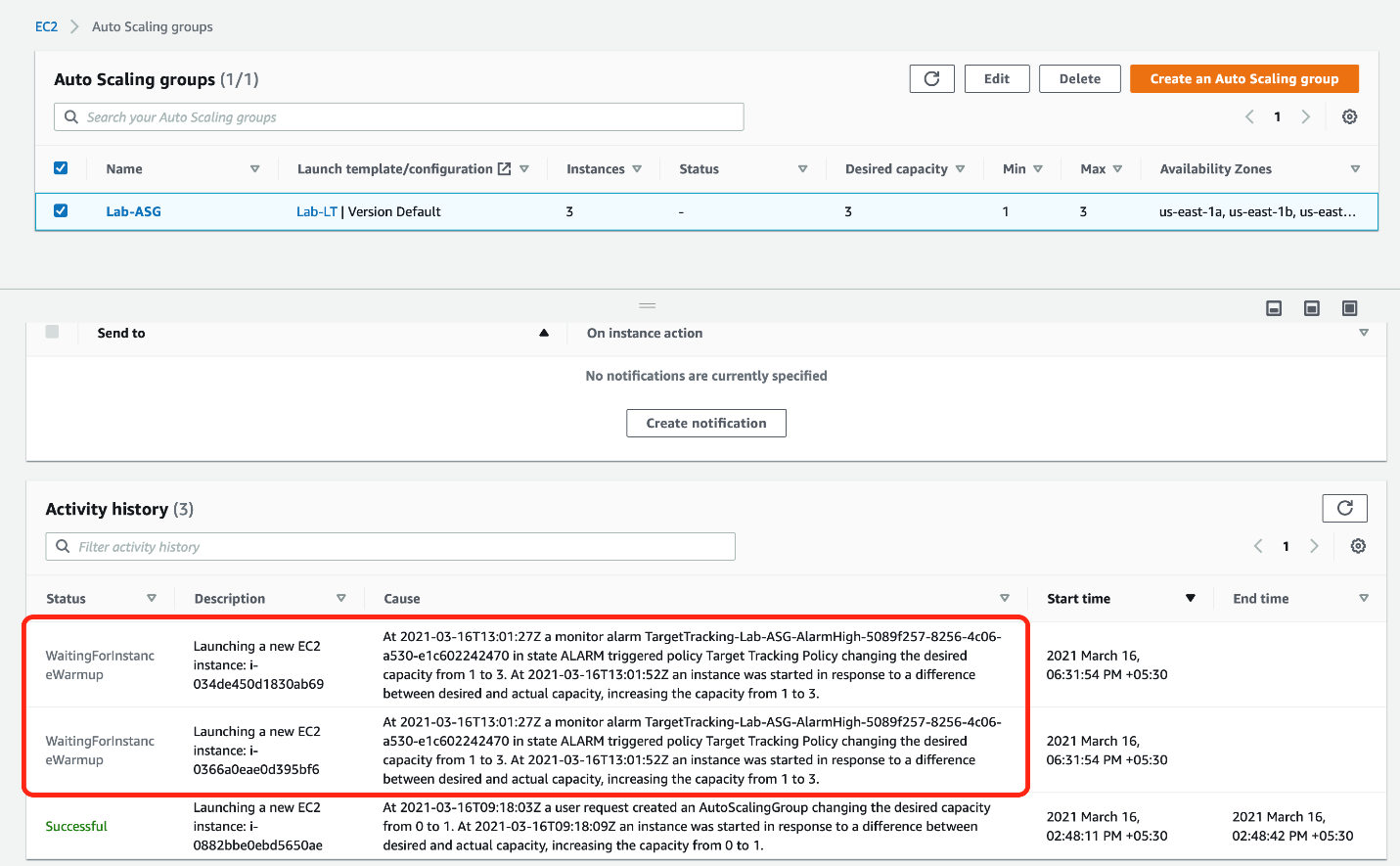
stress --cpu 8 --timeout 300



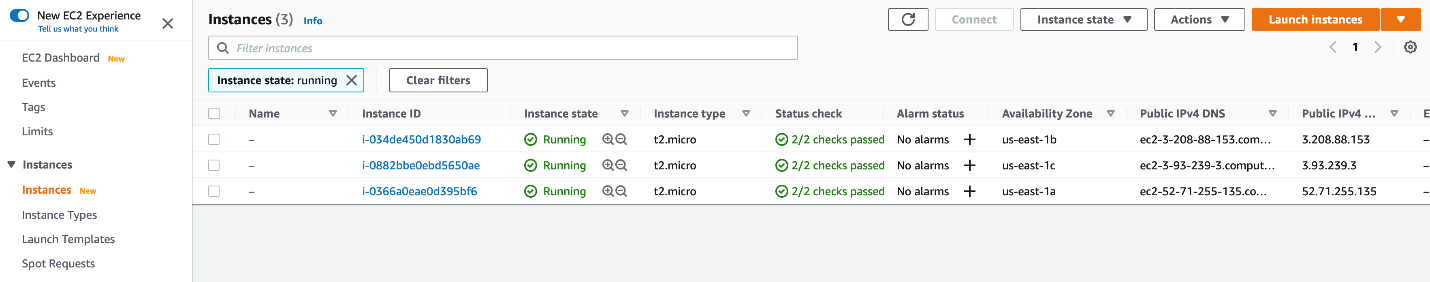
1. The instance sees a spike in the **CPU Utilization** metric which shows up on the **EC2** tab of the **Monitoring** section of the Auto Scaling group within a timespan of 5-10 minutes.



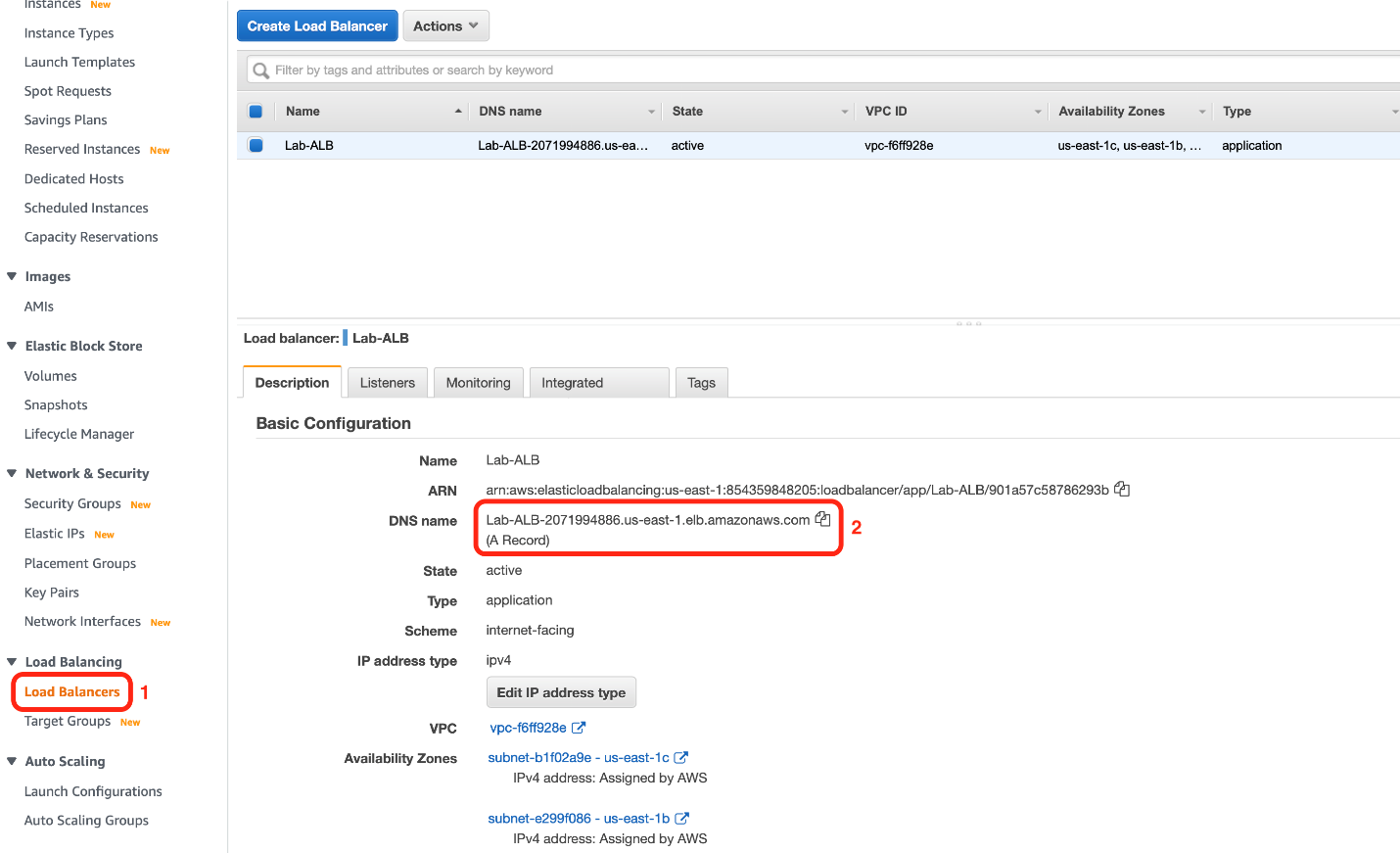
1. The spike in the CPU utilization triggers the target tracking scaling policy as the monitoring alarm breaches the target value of 40% (set in Task 3). This results in the launch of two additional EC2 instances as seen in the **Activity** tab of the Auto Scaling group.



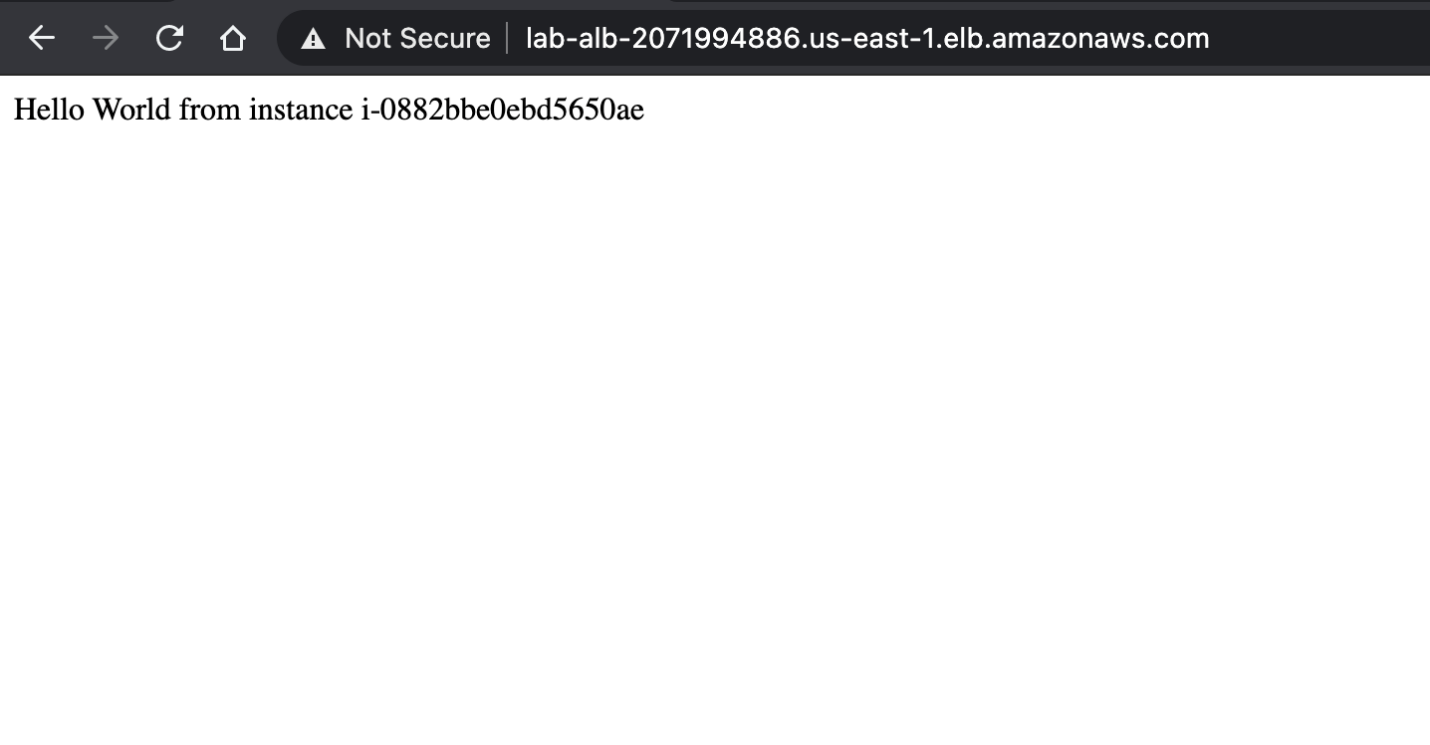
1. Three EC2 instances can be also be seen under the **Running** instance state



1. At this point, the Application Load Balancer (ALB) distributes the load evenly across these three EC2 instances. You can copy the DNS name of the ALB from the AWS console.



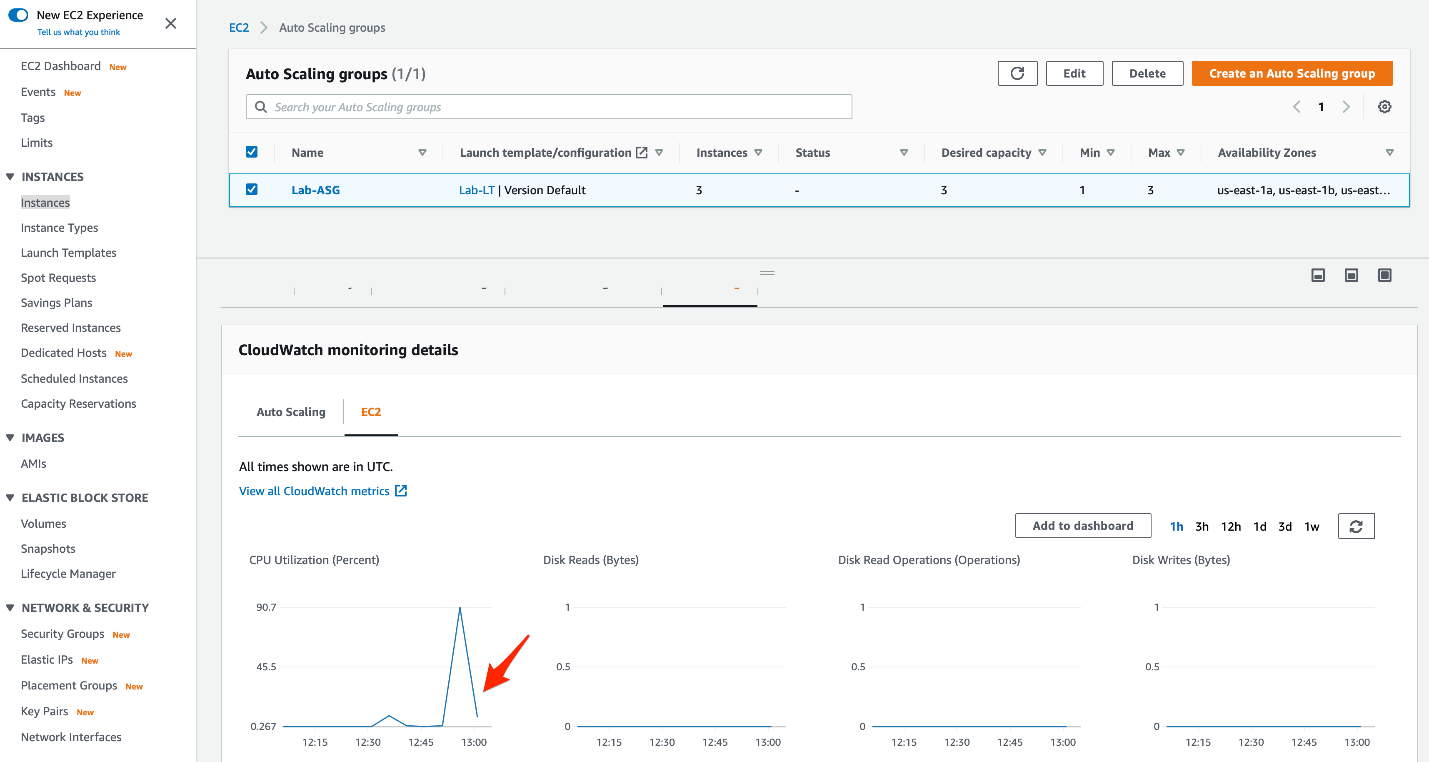
Paste the DNS name in a browser and you will see the instance ID in the welcome message. This was configured by setting EC2 user data in Task 2. If you refresh the browser, you will notice that the traffic is being evenly routed to the underlying instances and the instance IDs keep changing with each request.



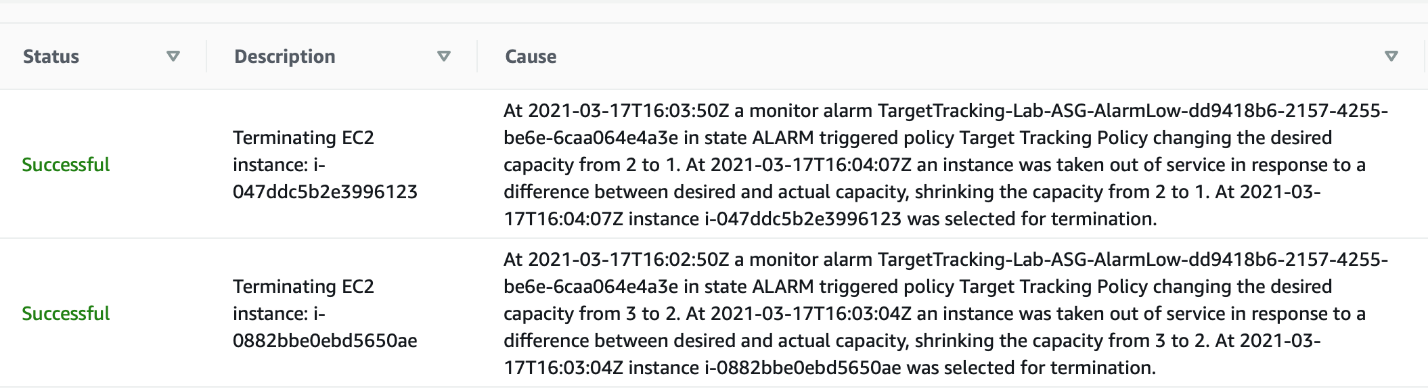




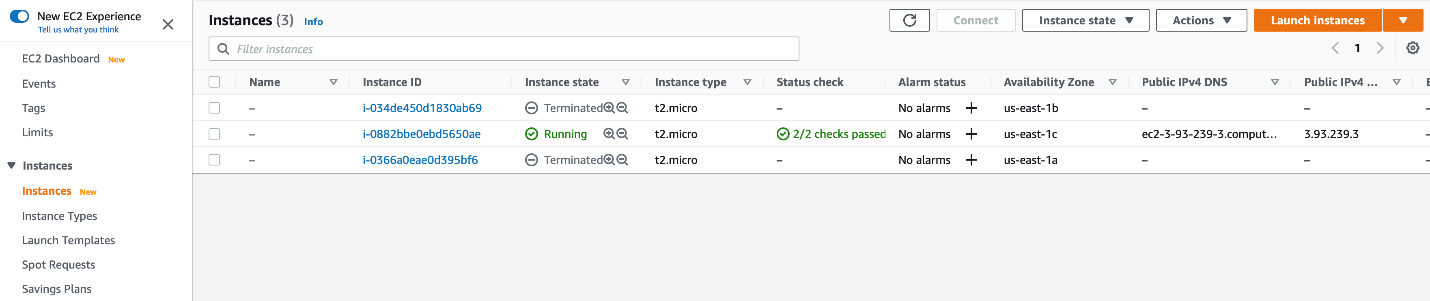
1. Once the stress test completes after the timeout of 300 seconds, the CPU Utilization again falls back to the normal level



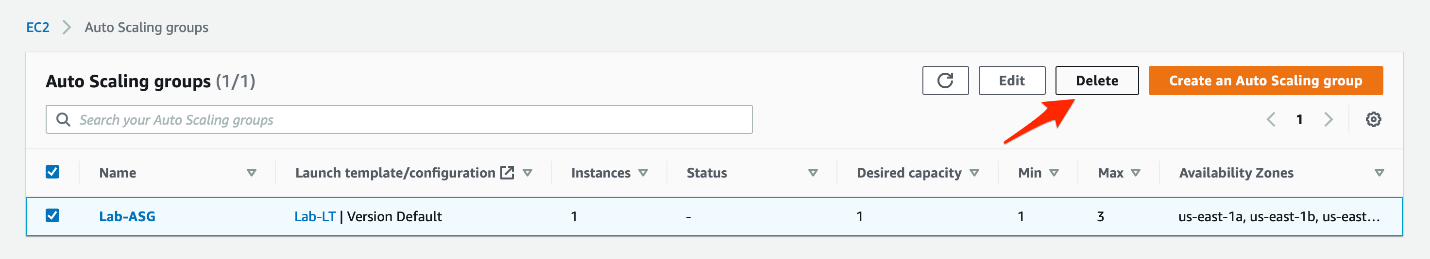
1. Once the CPU Utilization returns back to normal, the two extra EC2 instances are also terminated which can be seen in the **Activity** tab of the Auto Scaling group



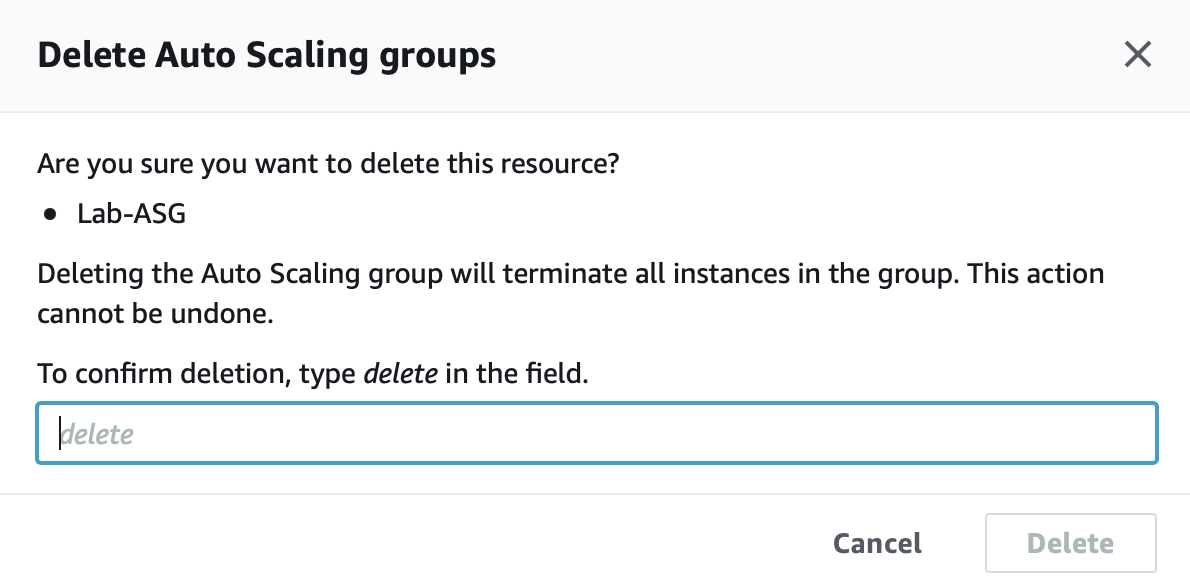
1. Two of the EC2 instances can also be seen under the **Terminated** instance state thereby demonstrating the horizontal scaling of the EC2 instances under changing workloads.



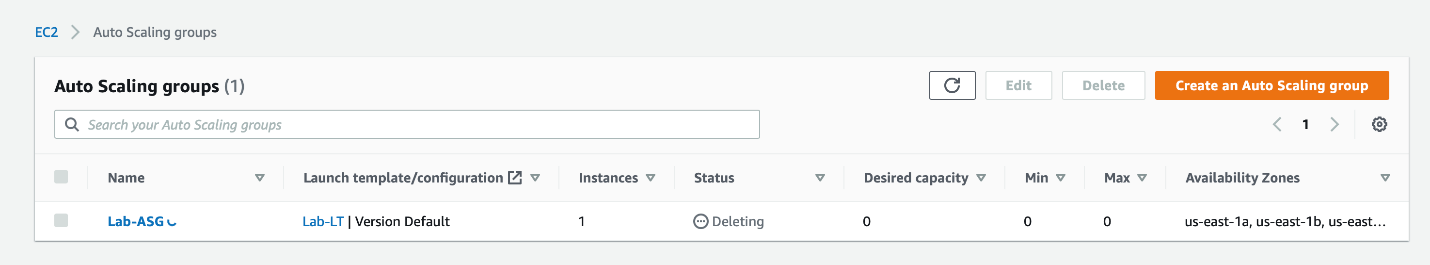
1. Lets clean up the resources created for this lab.
   1. Select the Auto Scaling group (Lab-ASG) and click on **delete**.



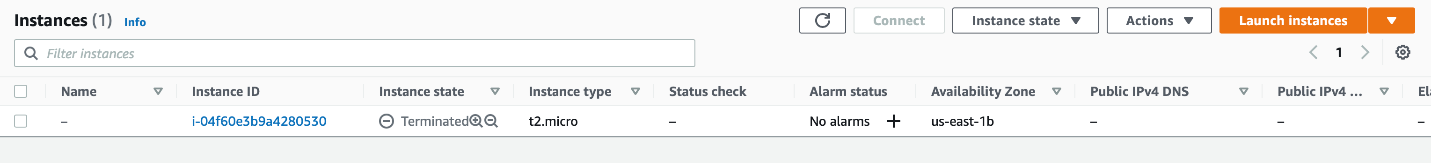
* 1. Confirm that you want to delete the Auto Scaling group by entering the word delete



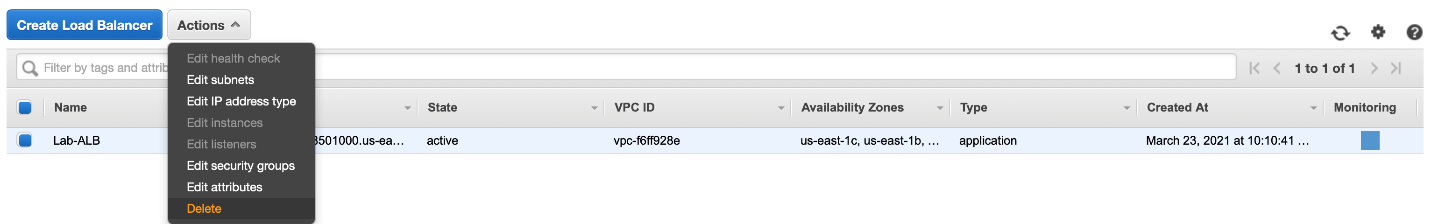
* 1. You should see that the ASG is in the *Deleting*status



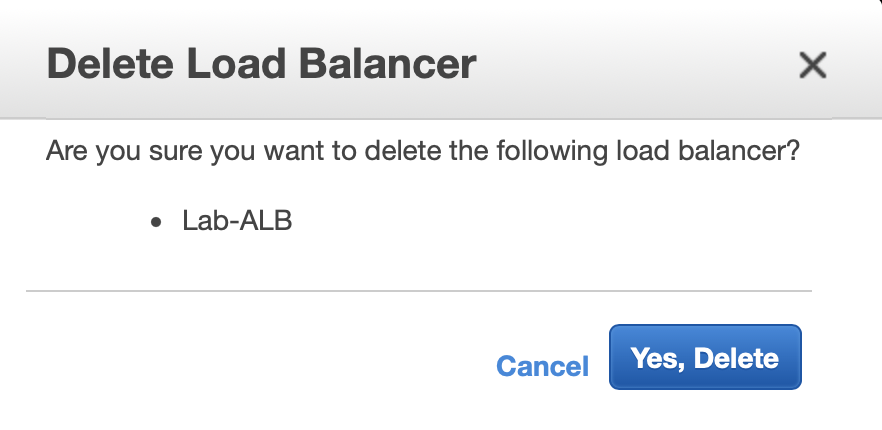
* 1. Within a few minutes, you should also see that the instance is in the *Terminated* state



* 1. Select the Application Load Balancer (Lab-ALB), click on **Actions** and then **Delete**.



* 1. Confirm that you want to delete the Application Load Balancer



1. Thank you for cleaning up the resources and congratulations for successfully completing this lab!

<https://www.tecmint.com/linux-cpu-load-stress-test-with-stress-ng-tool/> (how to impose high CPU load and stress test on linux using ‘Stress-ng’ Tool