Autoscaling and load balancing:

Q1. Differences between ELB, ALB, and NLB?

ELB:

- An ELB is a software-based load balancer which can be set up and configured in front of a collection of AWS Elastic Compute (EC2) instances. The load balancer serves as a single entry point for consumers of the EC2 instances and distributes incoming traffic across all machines available to receive requests. In addition to providing a single point of entry, the ELB also performs a vital role in improving the fault tolerance of the services which it fronts. The ELB regularly conducts a health check of all instances which have been registered with it, and only routes traffic to those machines which respond as active and healthy to the health check.

NLB:

- This is the distribution of traffic based on network variables, such as IP address and destination ports. It is layer 4 (TCP) and below and is not designed to take into consideration anything at the application layer such as content type, cookie data, custom headers, user location, or the application behavior. It is *context-less*, caring only about the network-layer information contained within the packets it is directing this way and that. This is a TCP Load Balancer only that does some NAT magic at the VPC level. It uses EIPs, so it has a static endpoint unlike ALB and CLBs (by default, contact support if this is a requirement for your CLB or ALB). Each Target can be on different ports.

ALB:

- This is the distribution of requests based on multiple variables, from the network layer to the application layer. It is context-aware and can direct requests based on any single variable as easily as it can a combination of variables. Applications are load balanced based on their peculiar behavior and not solely on server (operating system or virtualization layer) information. This is a feature filled L7 load balancer, HTTP and HTTPS listeners only. Provides the ability to route HTTP and HTTPS traffic based upon rules, host based or path based. Like an NLB, each Target can be on different ports. Even supports HTTP/2. Configurable range of health check status codes (CLB only supports 200 OK for HTTP health checks). The first difference is that the Application Load Balancer (as the name implies) works at the Application Layer (Layer 7 of the OSI model). The network load balancer works at layers 3 & 4 (network and transport layers). The network load balancer just forward requests whereas the application load balancer examines the contents of the HTTP request header to determine where to route the request. So, the application load balancer is performing content based routing.
- The other difference between the two is important because network load balancing cannot assure availability of the *application.* This is because it bases its decisions solely on network and TCP-layer variables and has no awareness of the application at all. Generally a network load balancer will determine “availability” based on the ability of a server to respond to ICMP ping, or to correctly complete the three-way TCP handshake. An application load balancer goes much deeper, and is capable of determining availability based on not only a successful HTTP GET of a particular page but also the verification that the *content* is as was expected based on the input parameters.

Q2. Differences between step scaling and target scaling.

Step Scaling:

We specify MULTIPLE thresholds Along with different responses.

Threshold A - add 1 instance when CPU Utilization is between 40% and 50%

Threshold B - add 2 instances when CPU Utilization is between 50% and 70%

Threshold C - add 3 instances when CPU Utilization is between 70% and 90%

There can be multiple thresholds

Target scaling:

AWS creates the Cloudwatch alarm for you on the basis of the target specified by you. It monitors the alarm and in the event of any alarm breach, tries to keep the metric near the target value by scaling in or scaling out. Additionally, target tracking scaling policy also adjusts to the changes in the metric due to a changing load pattern.

We can go for a target tracking policy if we are sure that our scaling metric increases/decreases in proportion of the number of instances in the auto scaling group. On the other hand, if I'd want more fine grained control of the scaling in or scaling out I'd go for step scaling say for example I've a real time websocket application behind a load balancer (with least connection algorithm) and the average CPU utilization suddenly starts jumping over 80, I'd add maybe 3-4 instances so that all new connections get routed to my new instances and the other instances

Q3. Differences between Launch configuration and launch template.

- Launch template is similar to launch configuration which usually Auto Scaling group uses to launch EC2 instances.
- However, defining a launch template instead of a launch configuration allows you to have multiple versions of a template.
- AWS recommends that we should use launch templates instead of launch configurations to ensure that we can leverage the latest features of Amazon EC2, such as T2 Unlimited instances.

Q4. Differences between EC2 health check and load balancer health check

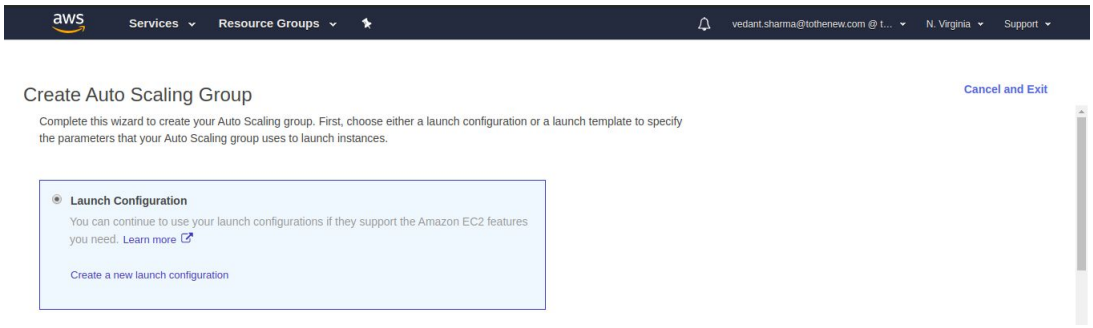
EC2 health check watches for instance availability from hypervisor and networking point of view. For example, in case of a hardware problem, the check will fail. Also, if an instance was misconfigured and doesn't respond to network requests, it will be marked as faulty.

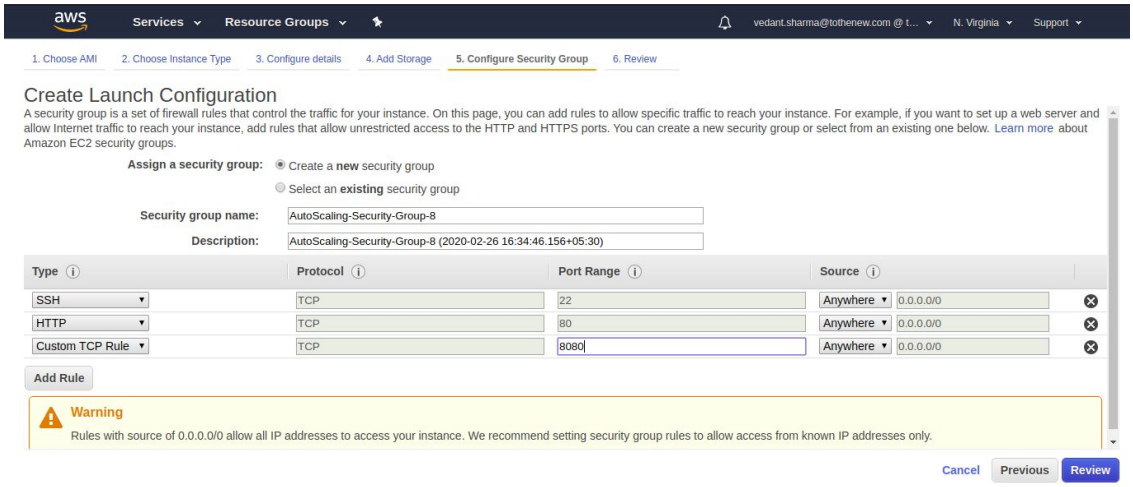
ELB health check verifies that a specified TCP port on an instance is accepting connections OR a specified web page returns 2xx code. Thus ELB health checks are a little bit smarter and verify that actual app works instead of verifying that just an instance works.

Q5. Create 2 auto-scaling groups with

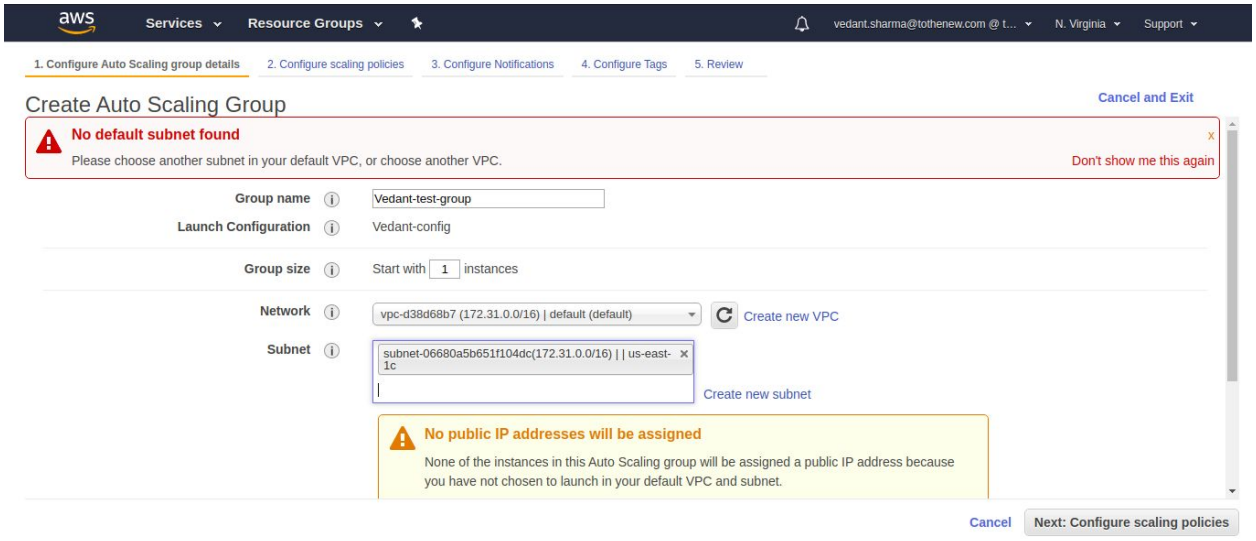
- launch configuration and
- launch template

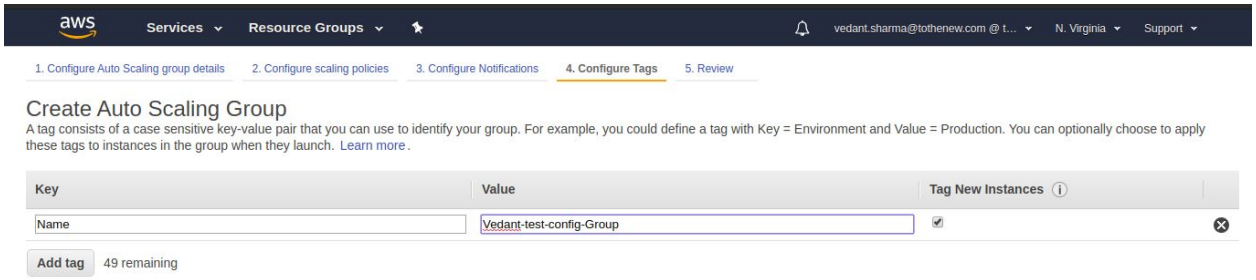
Using launch configuration:

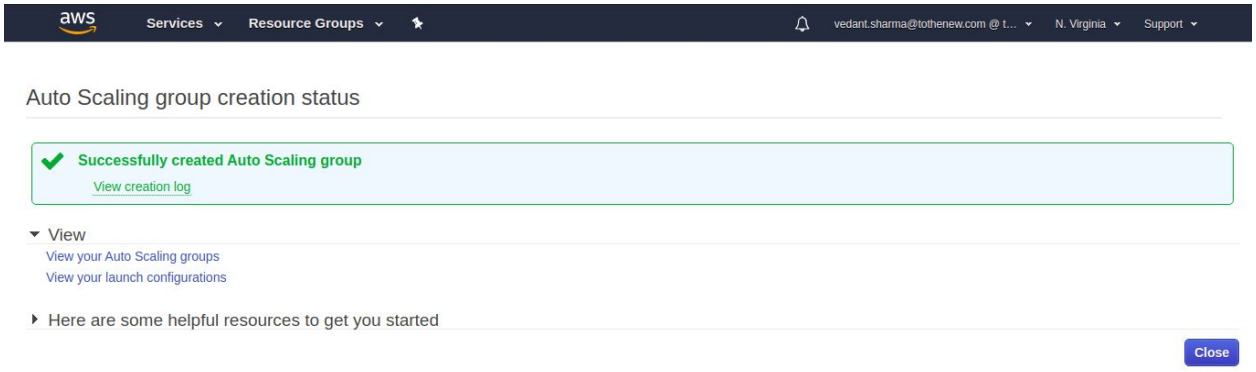




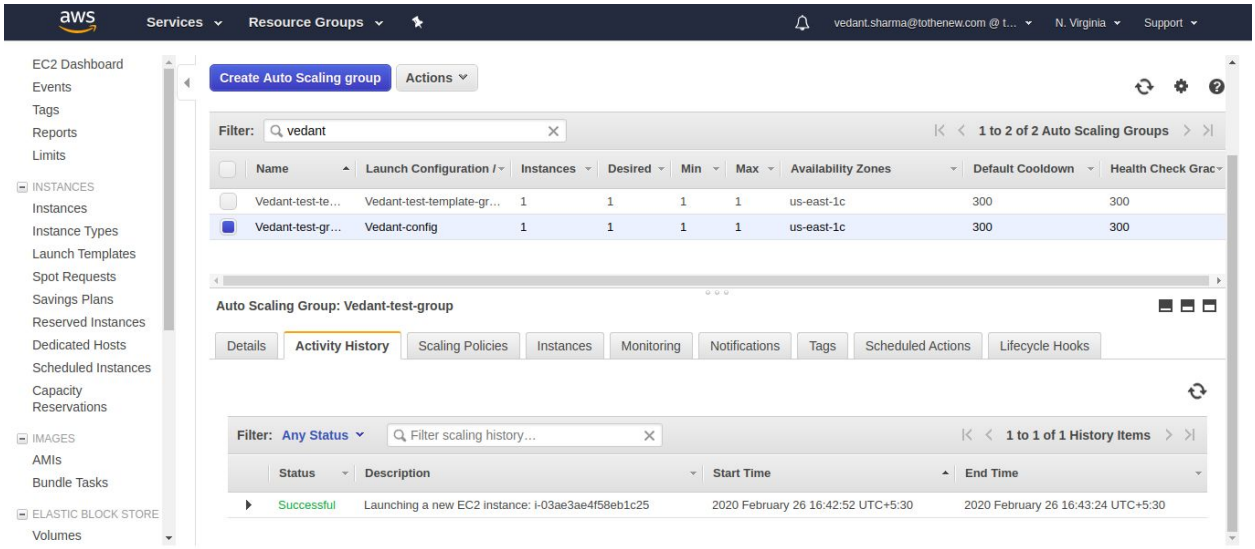
Making an autoscaling group:



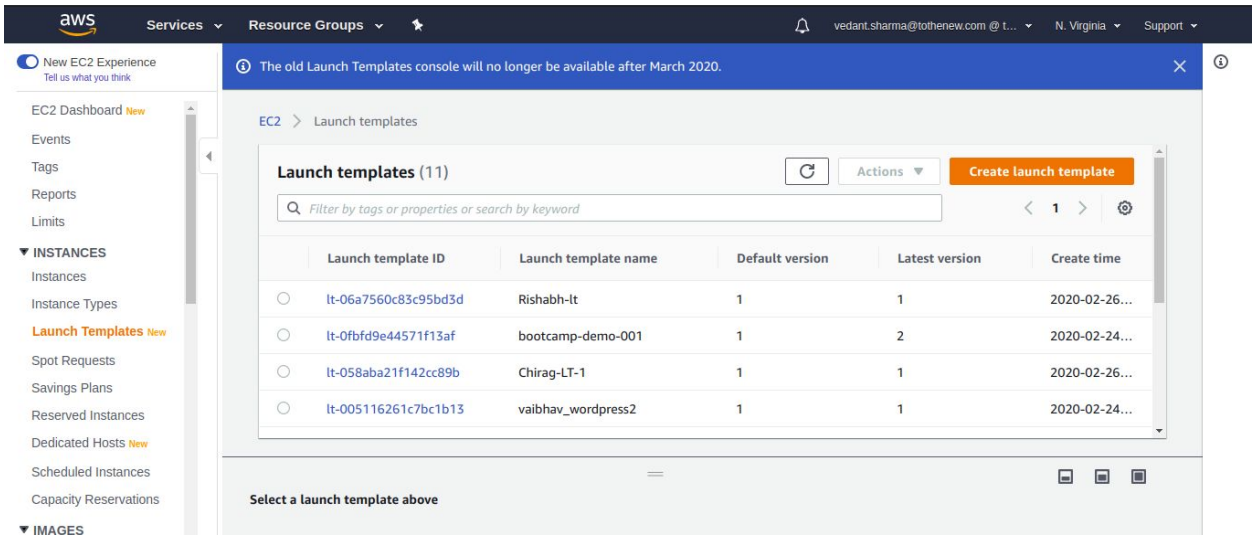


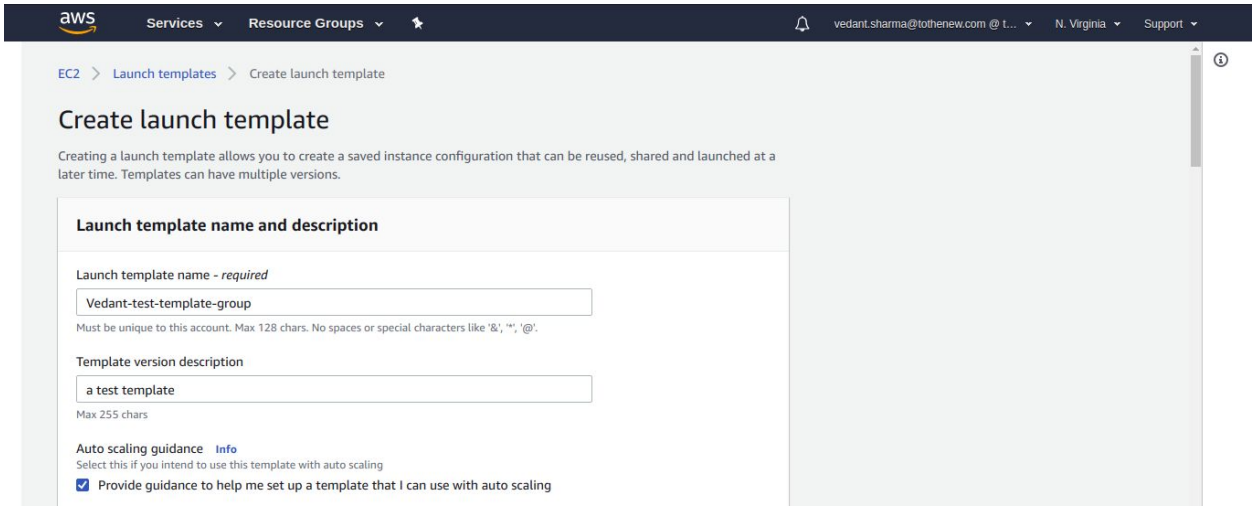


Visible instance status on the autoscaling group activity check:

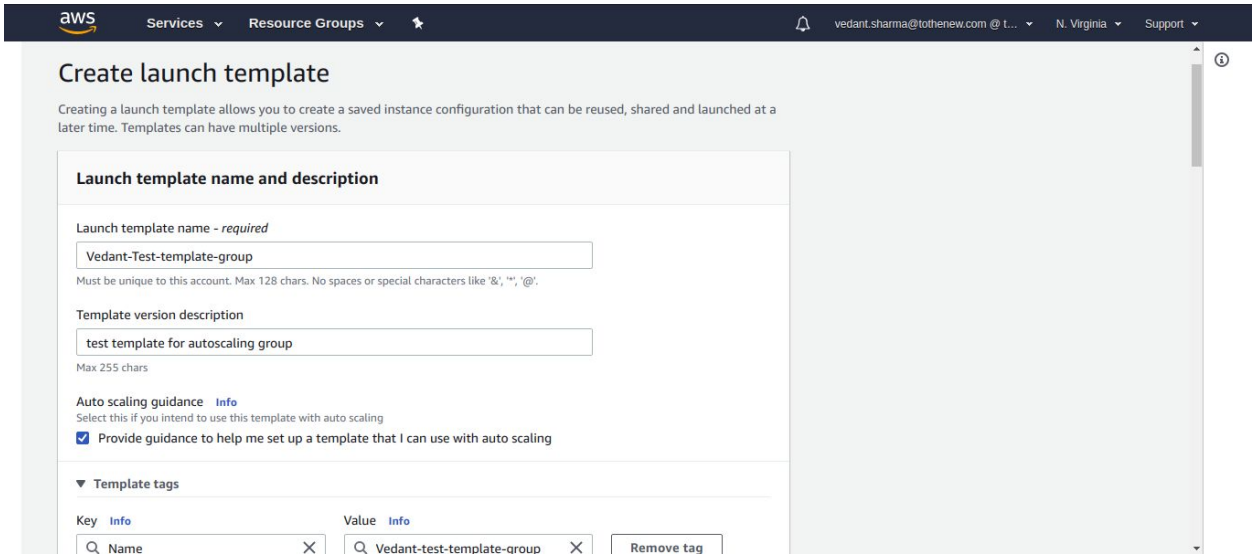


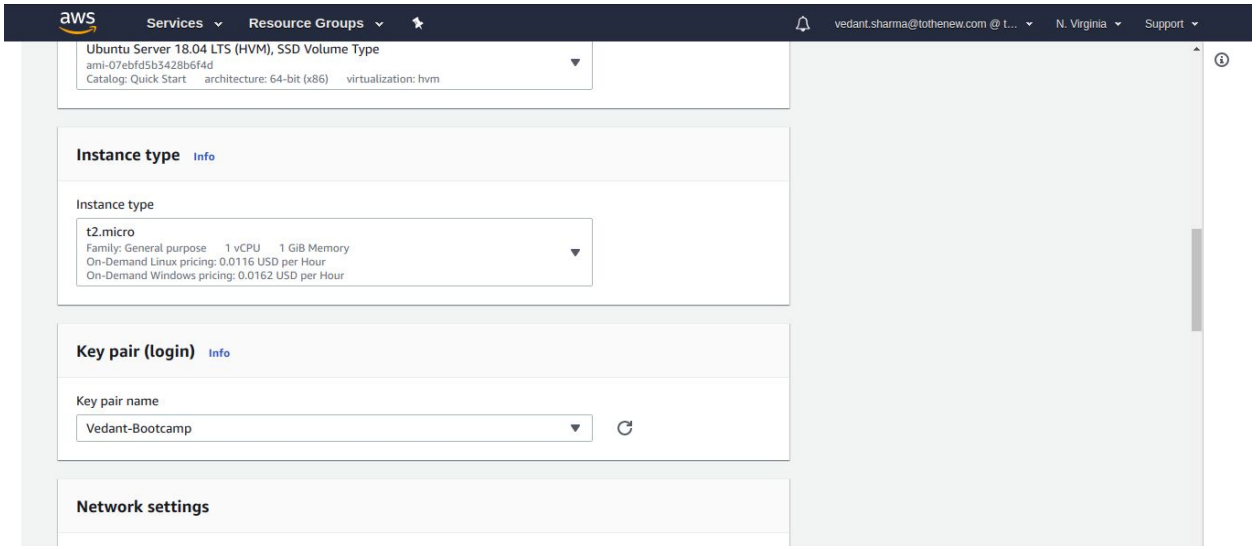
Using Launch template:

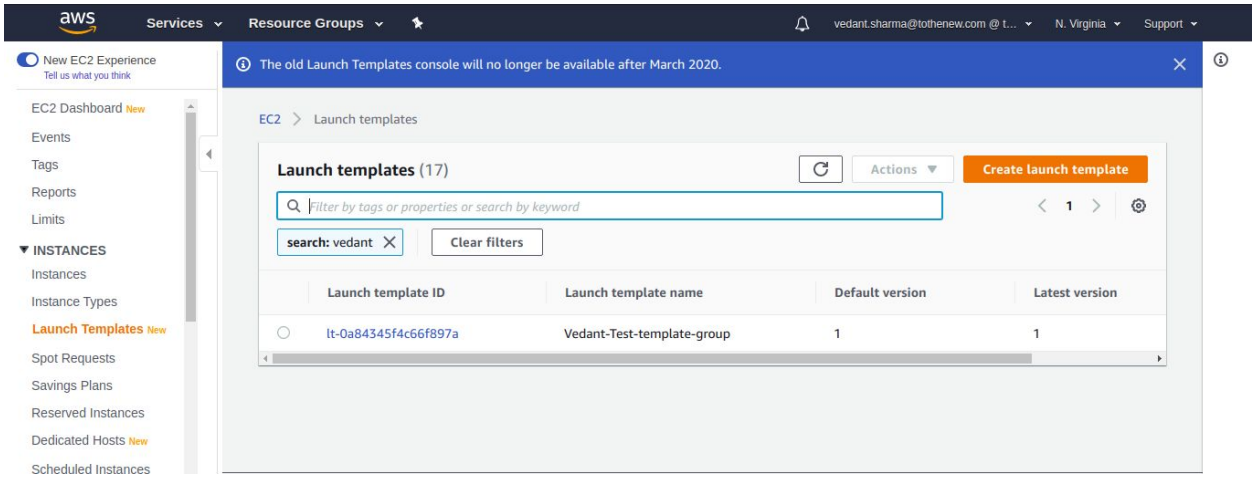




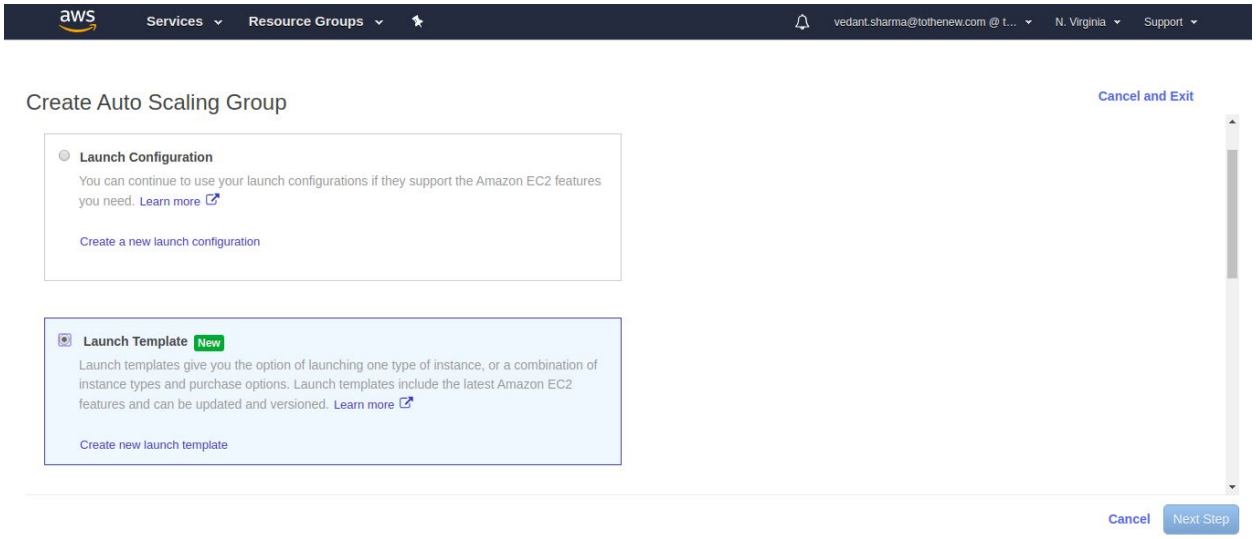
Tagging and naming the template with specifying the instance types to be involved and also specifying the keypair to be used for the login.



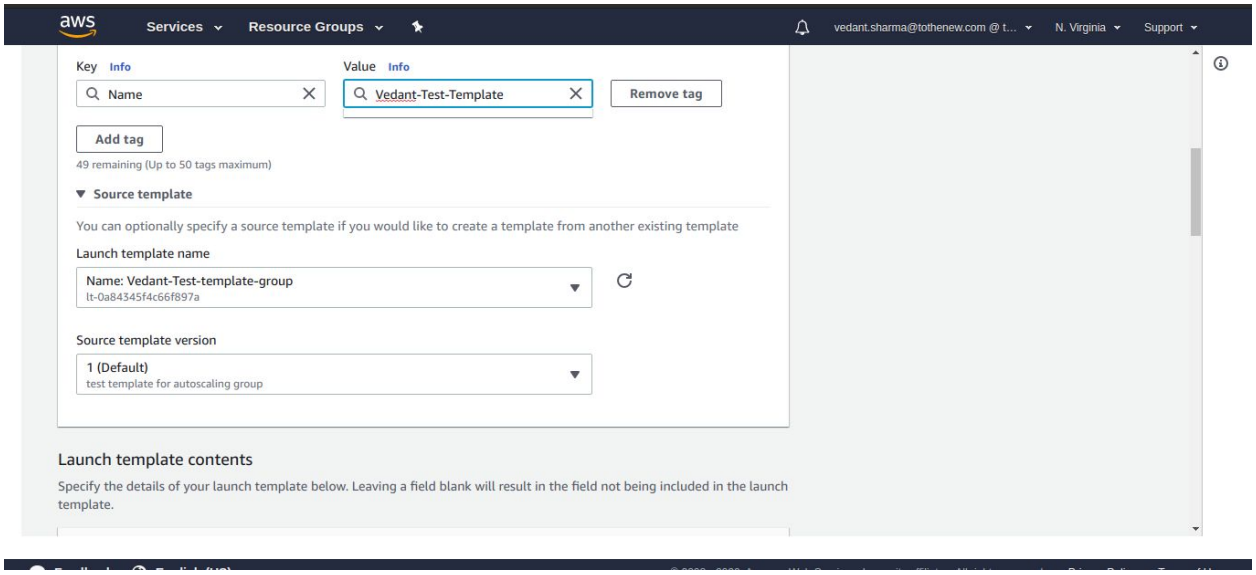




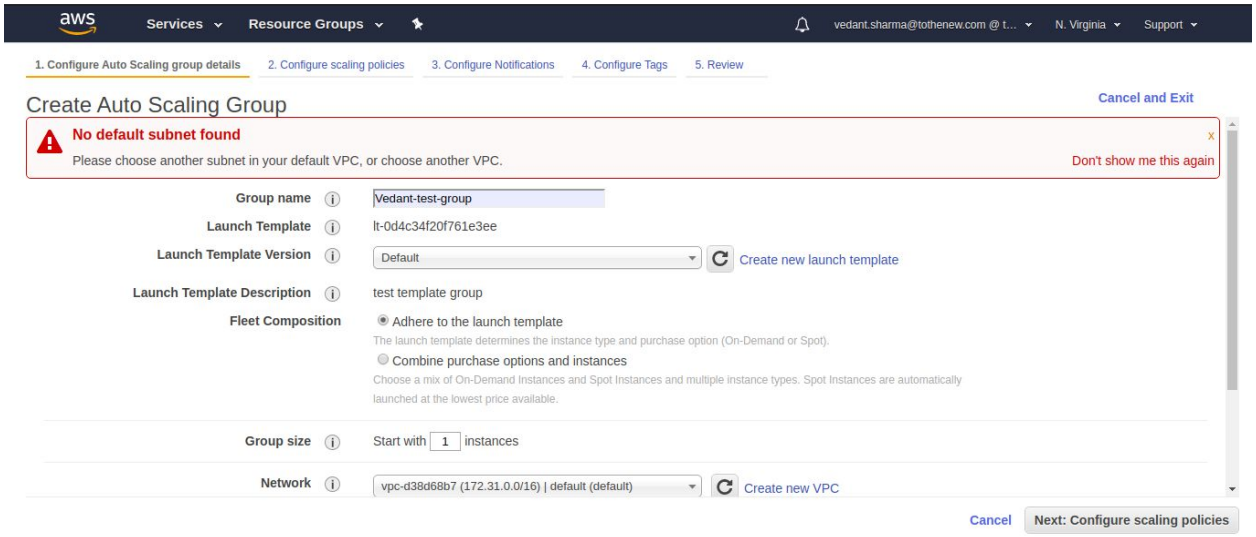
Now creating an auto scaling group using the launch template created;

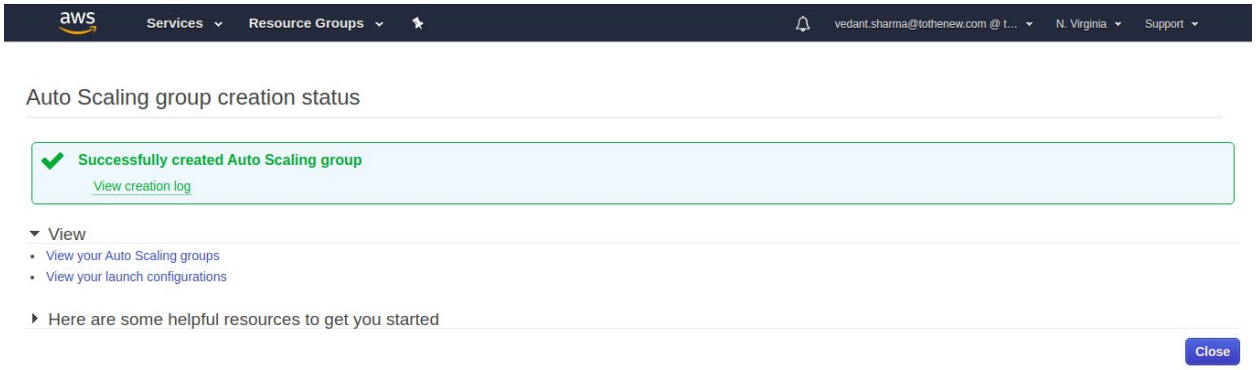


Specifying a source for the template file:

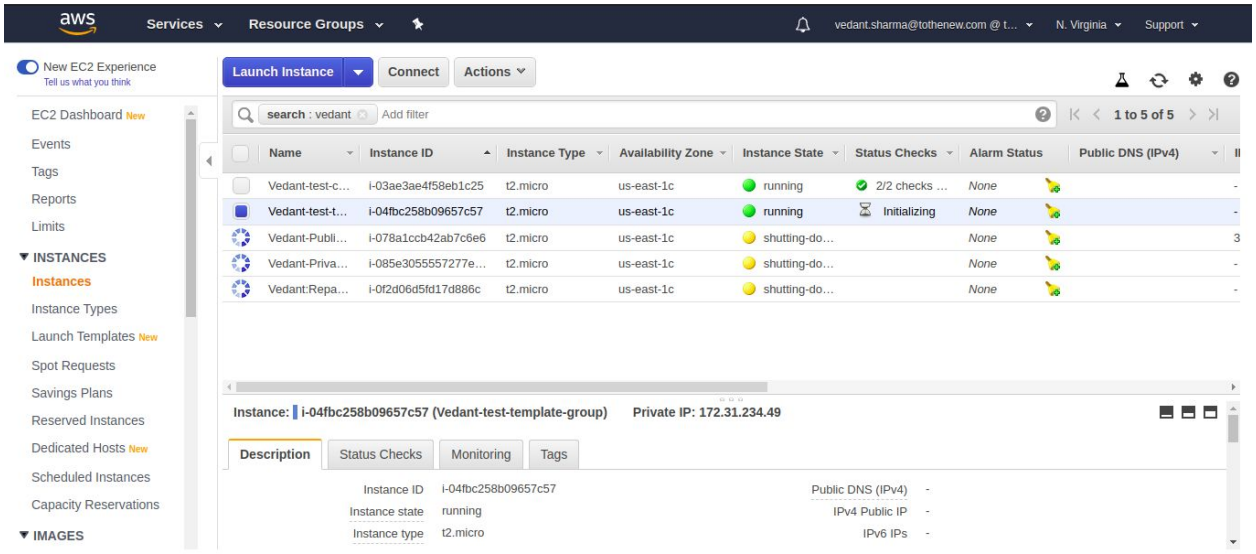


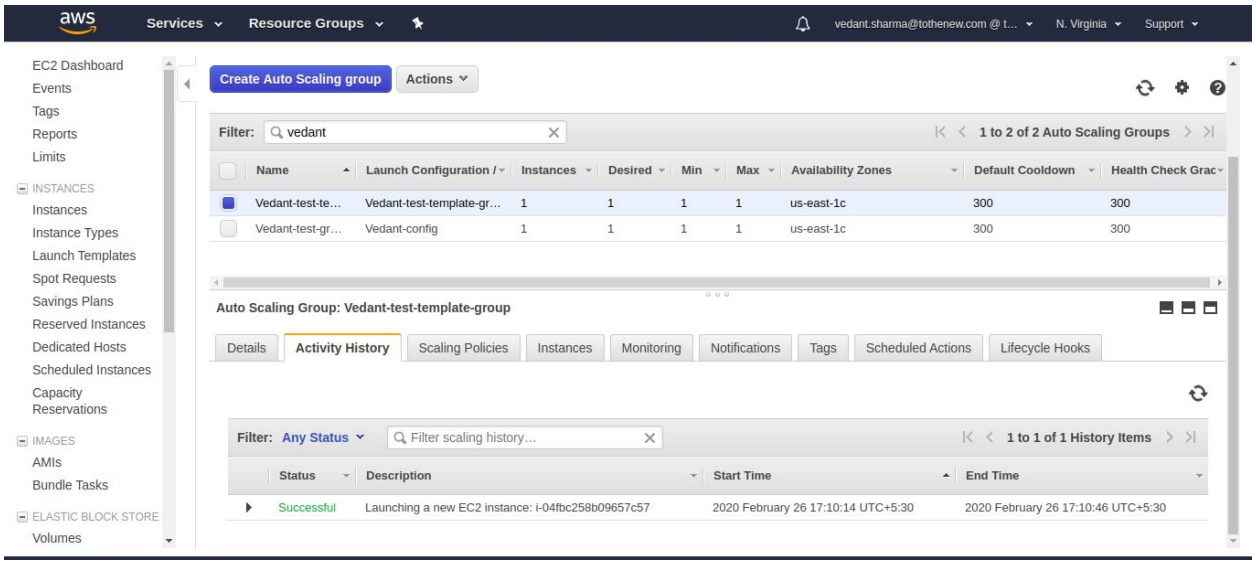
Creating a Auto scaling group by mentioning the template:





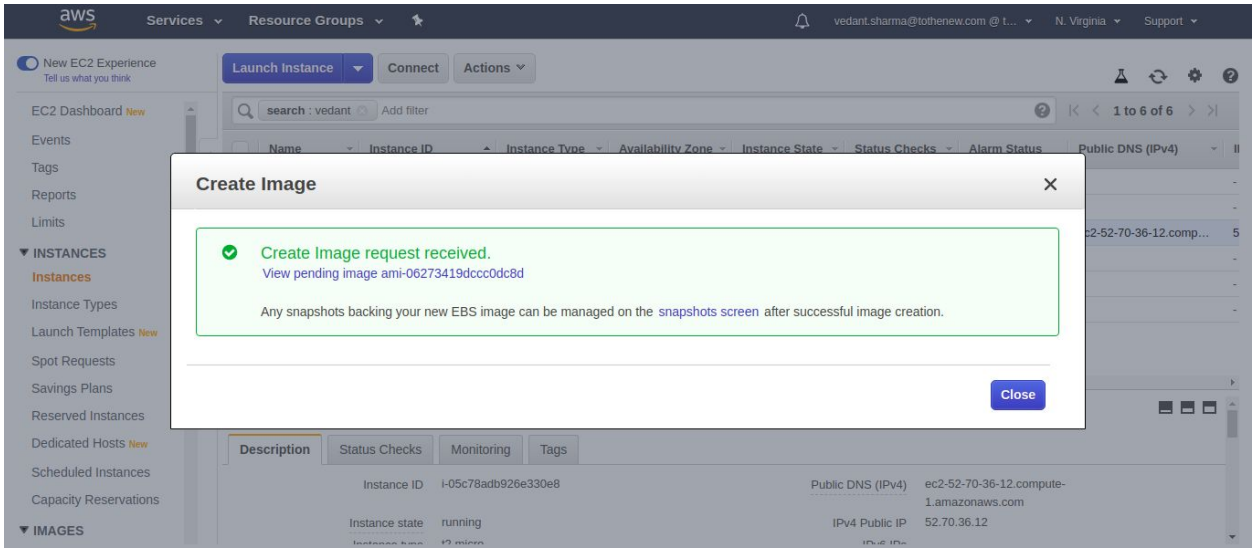
Visible launched instances:



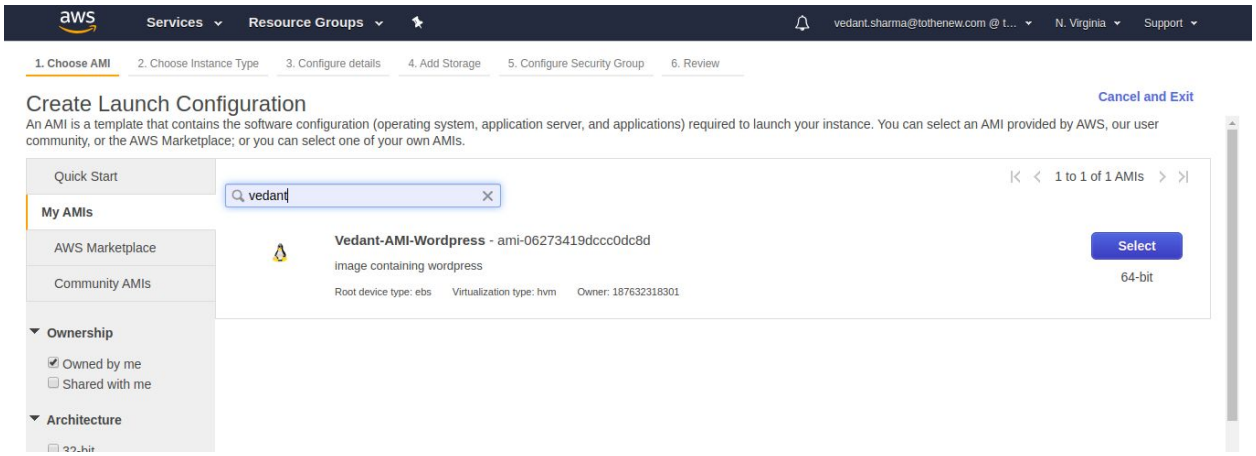


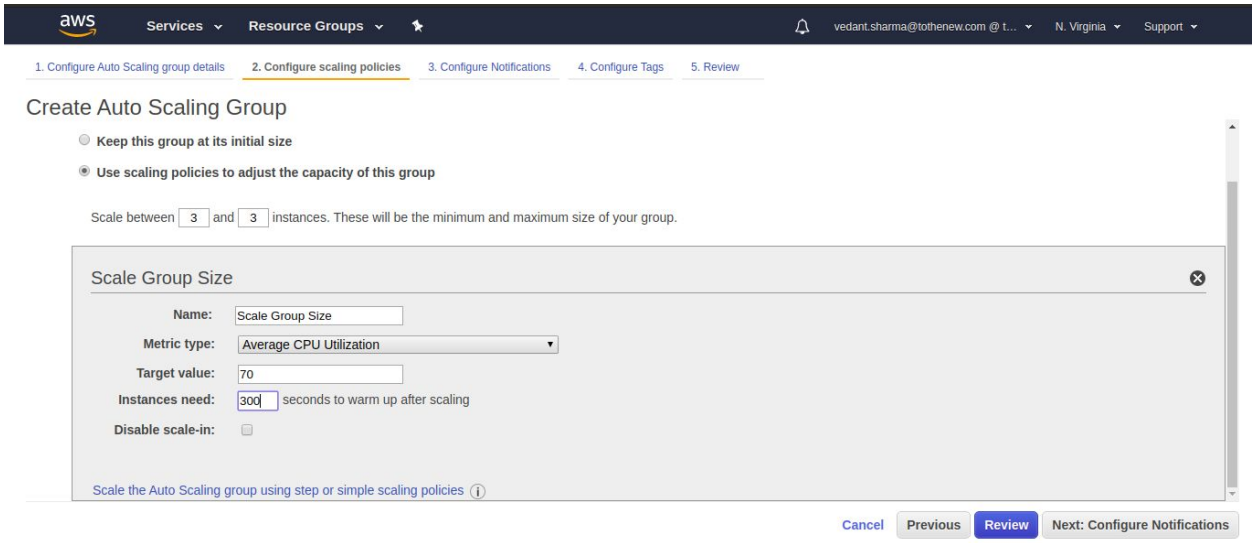
Q6. .Setup autoscaling Wordpress application with the Application load balancer. Auto-scaling should be triggered based on CPU usage of EC2 instances.

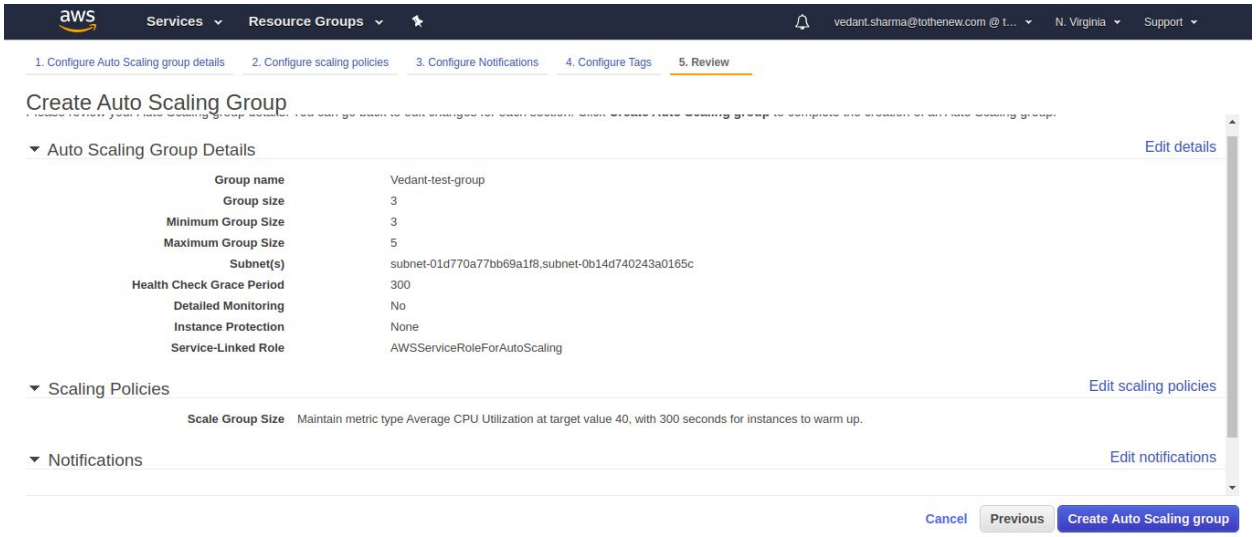
Creating an AMI of an instance with wordpress running:

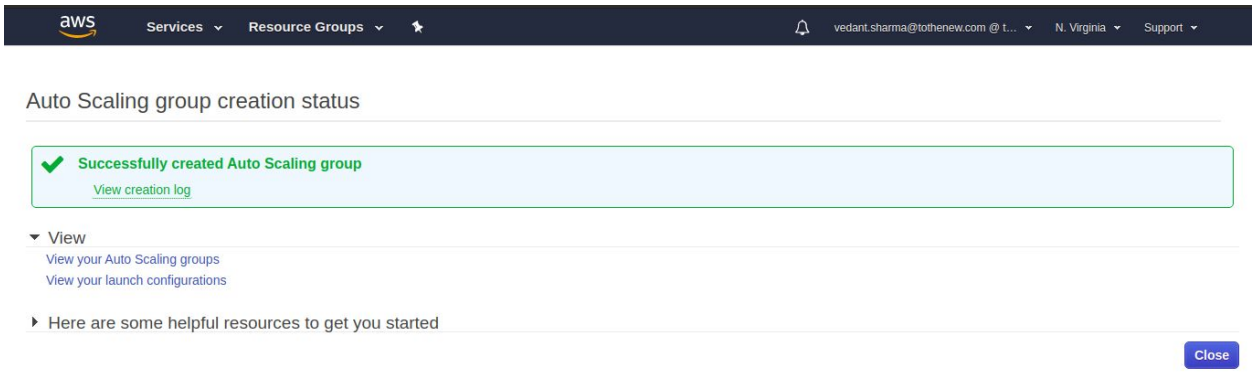


Create an auto scaling group using Ami

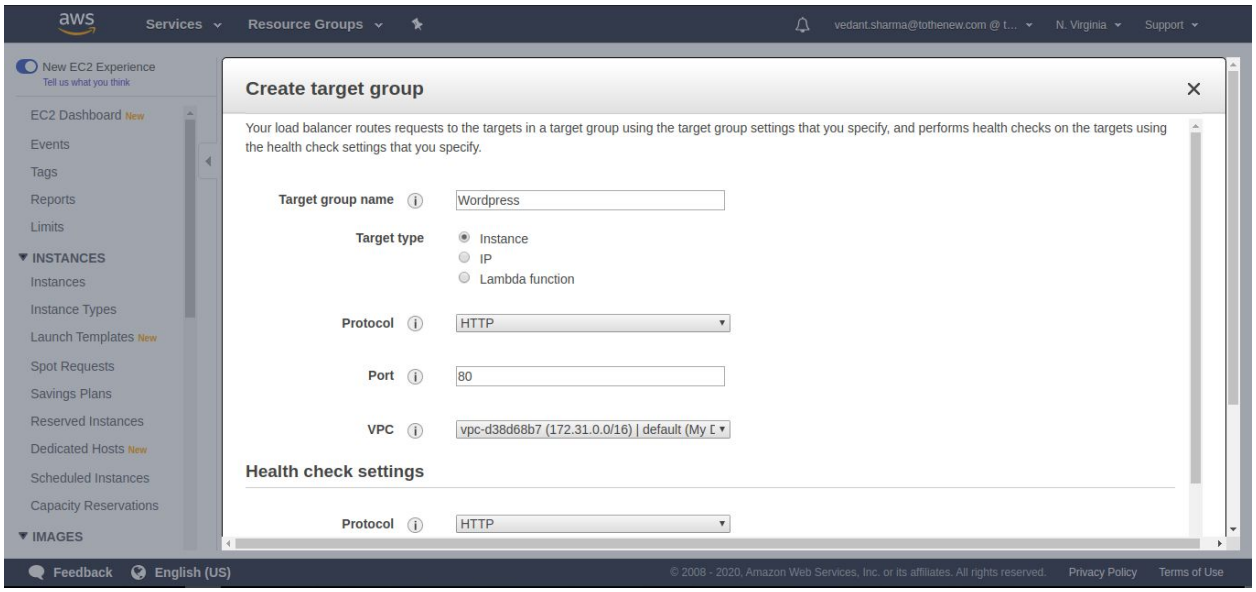


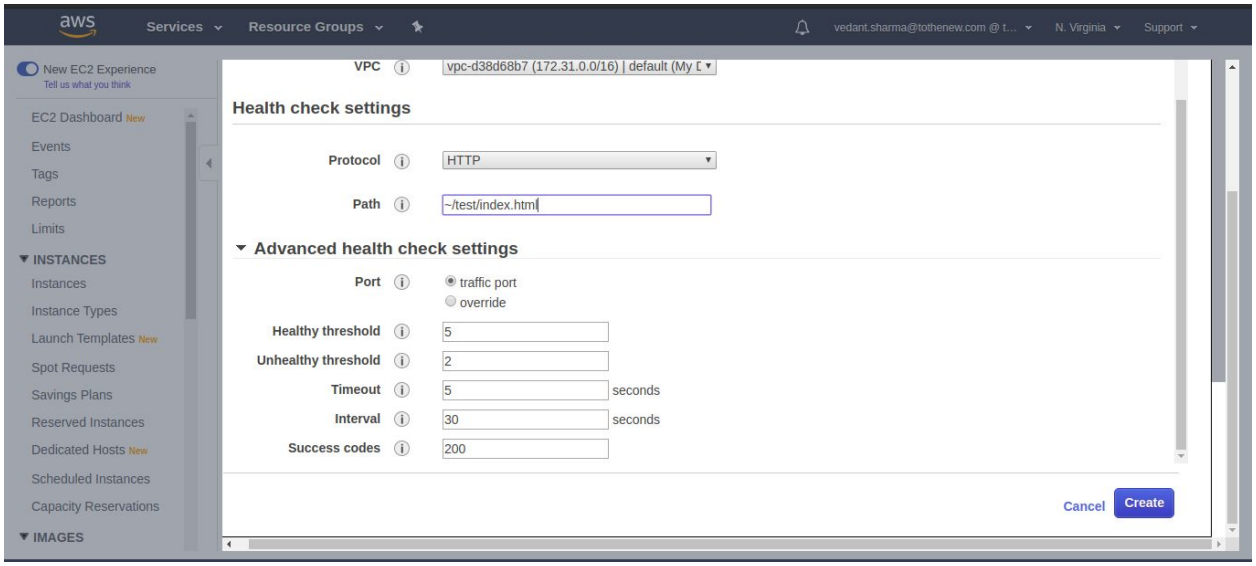




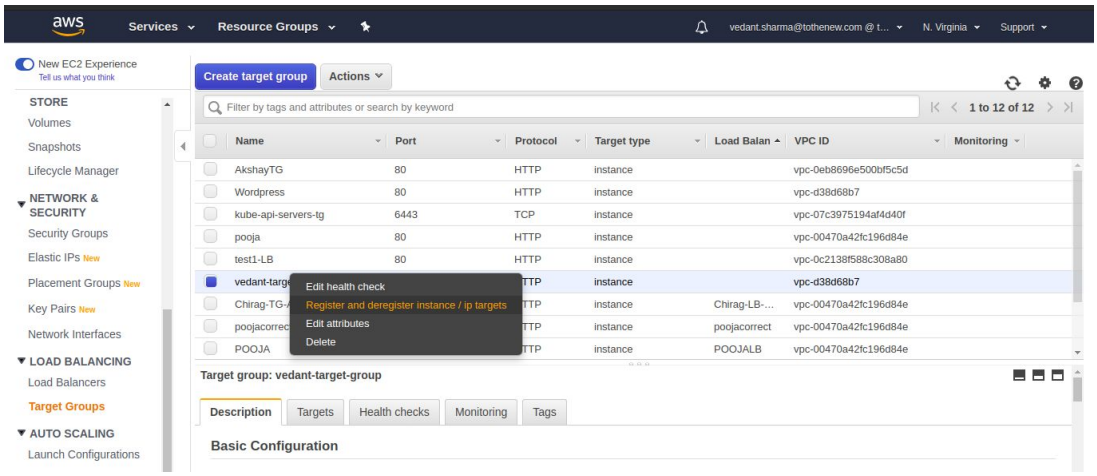


Creating a target group:

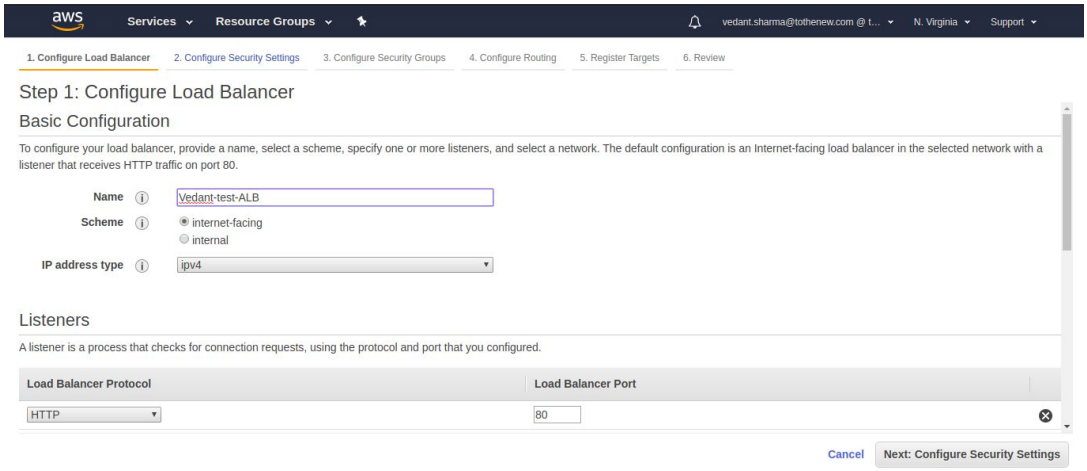




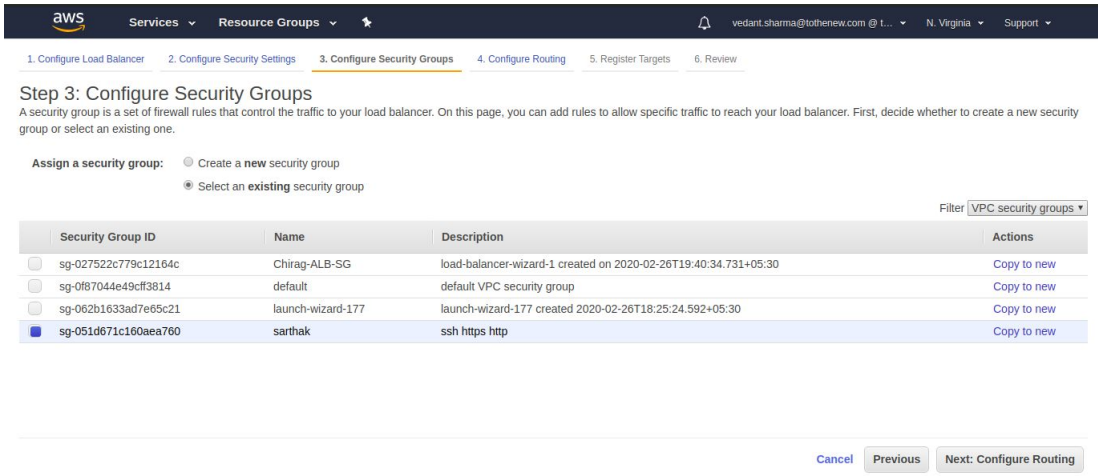
Adding the instance to the target group



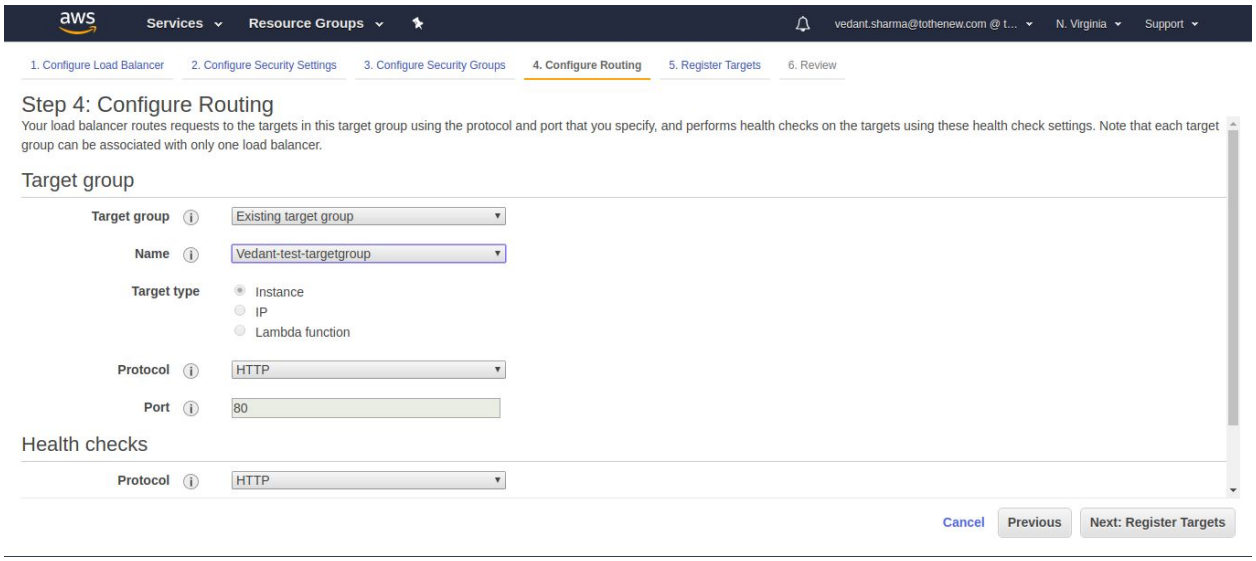
Creating an application load balancer:

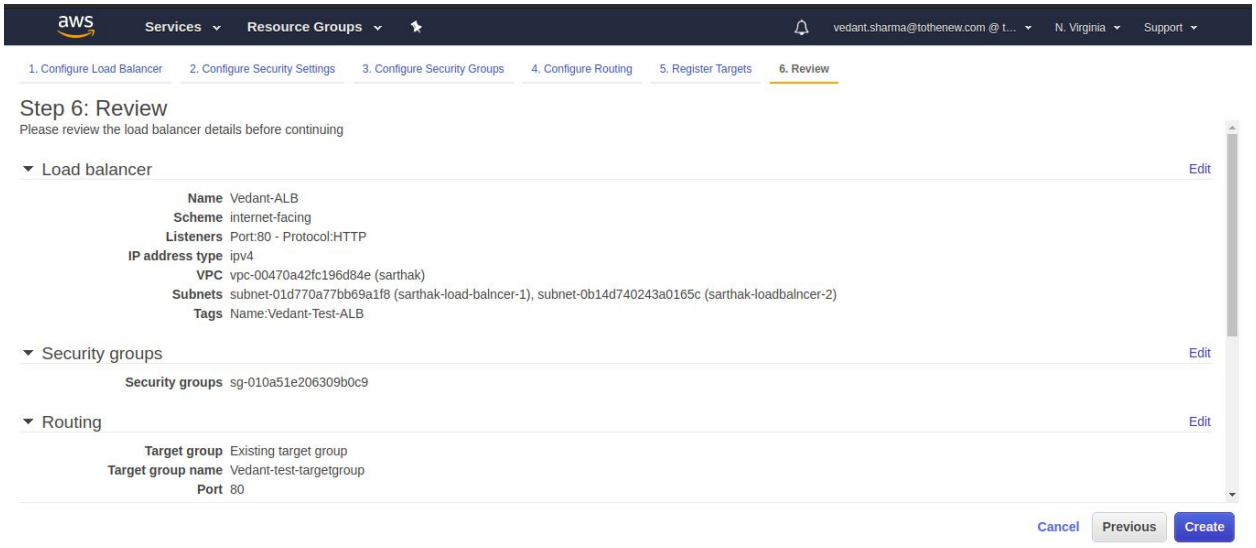


Associating with a security group:

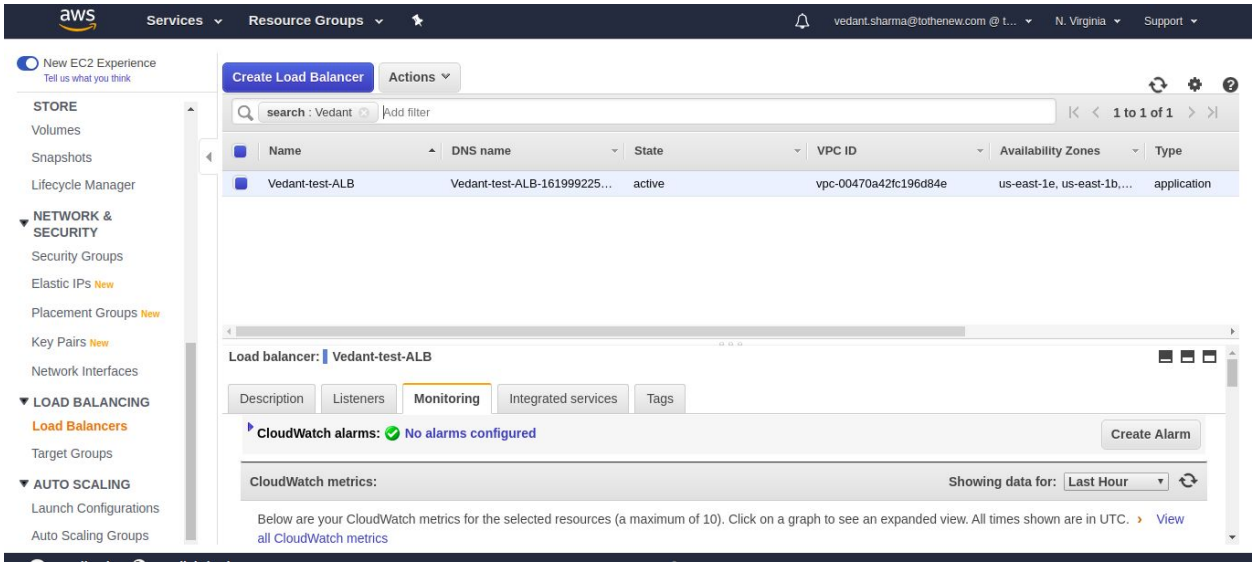


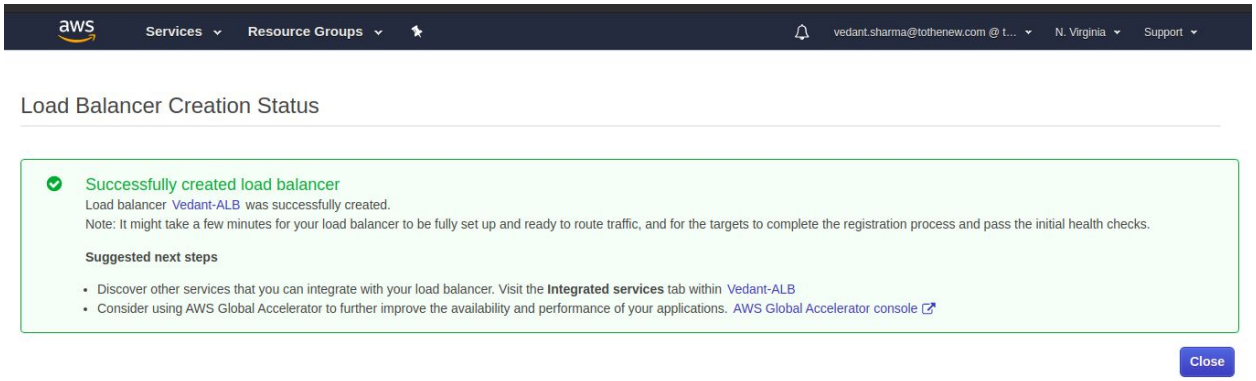
Associating target group with ALB



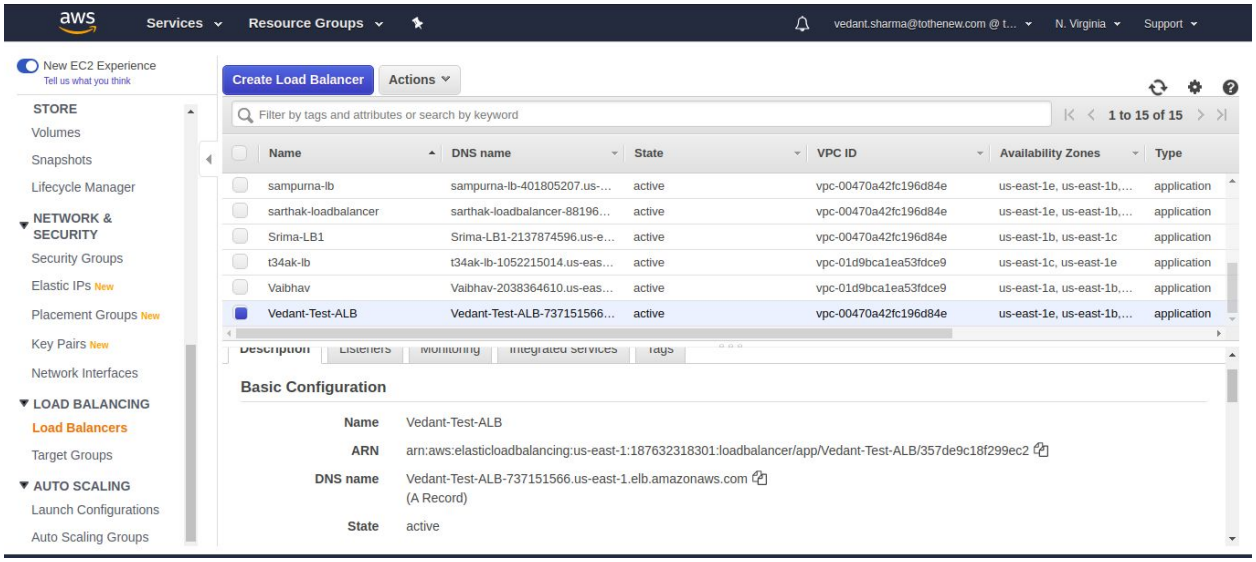


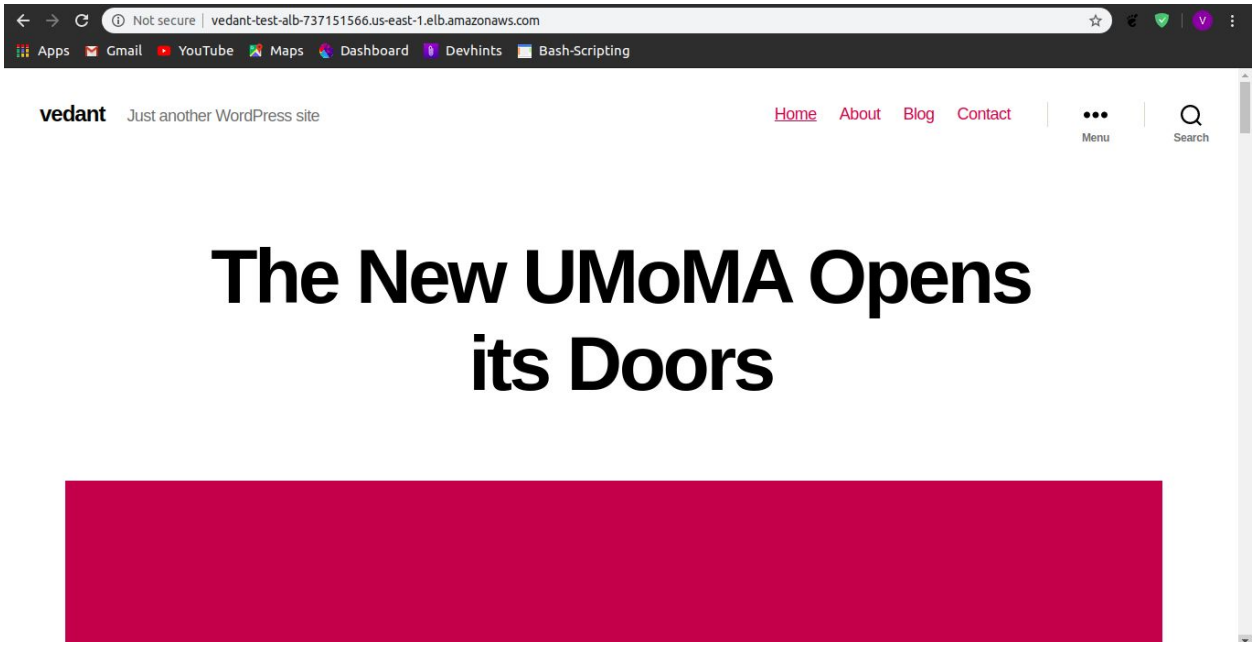
Creating an ALB





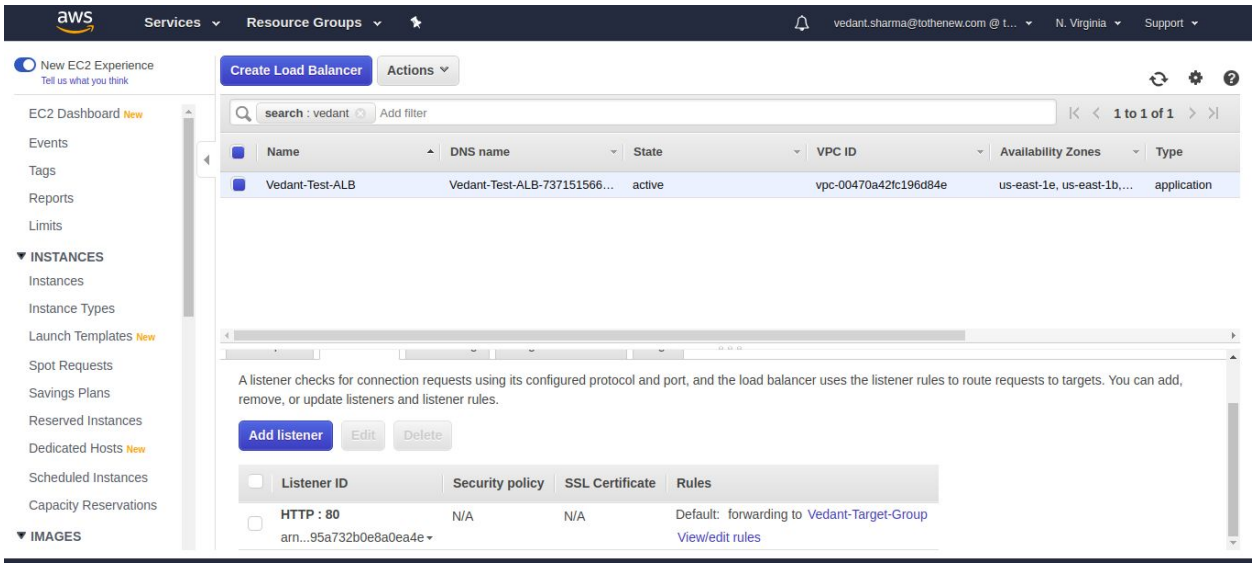
Copying the available DNS name and putting it on the browser in order to access the page



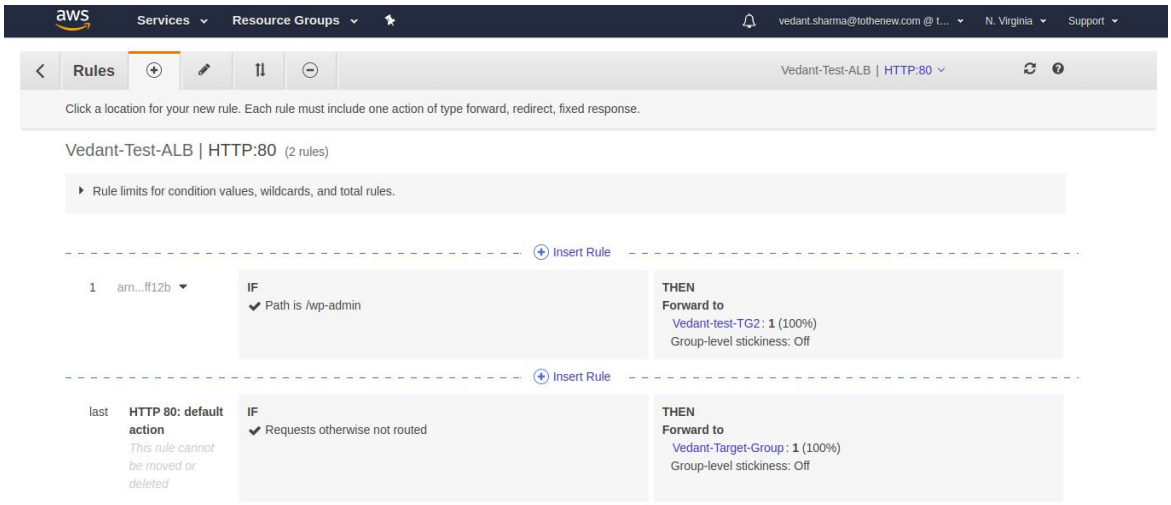


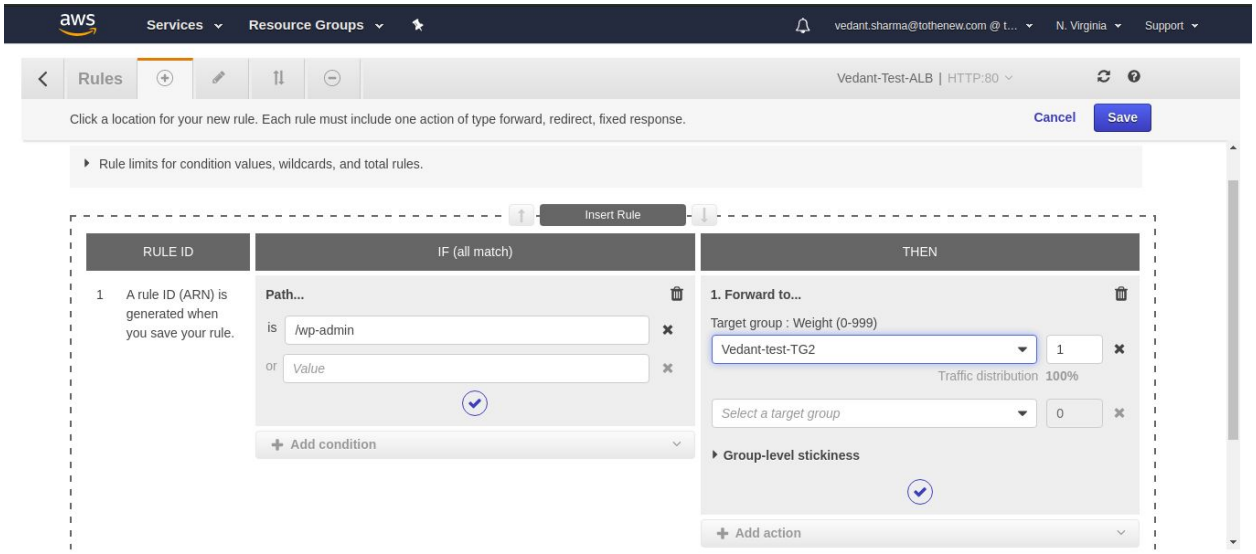
Q7. Create another Wordpress website and use the ALB created above to send traffic to this website based on the hostname

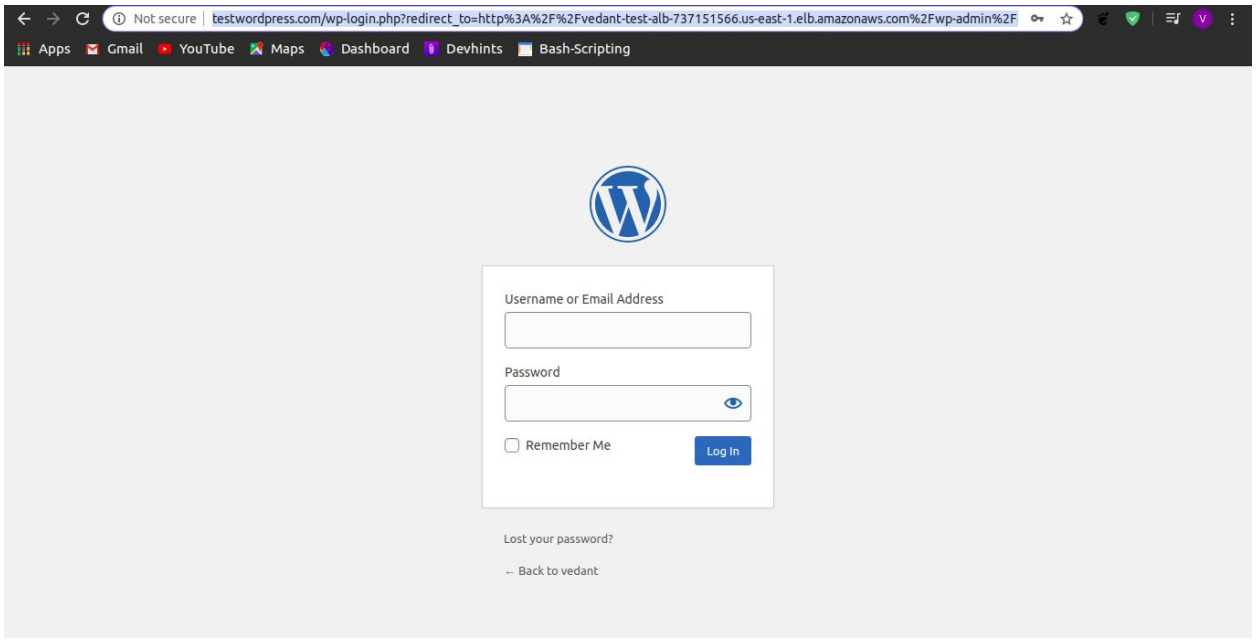
Adding rules to the load balancer for path based forwarding



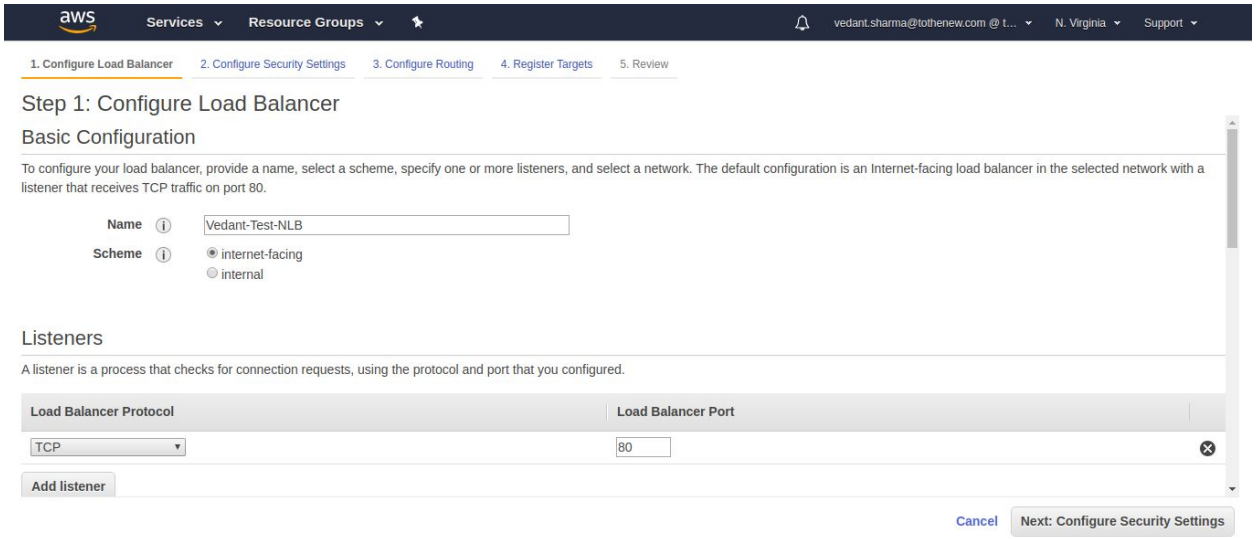
Setting /wp-admin to be forwarded to the instances running in the different target group

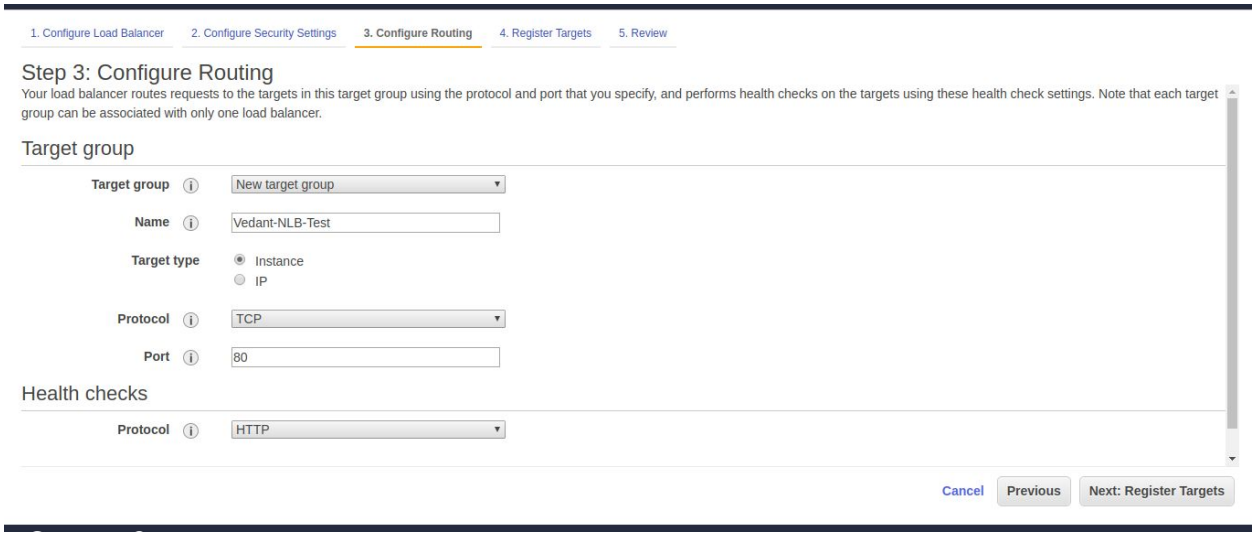


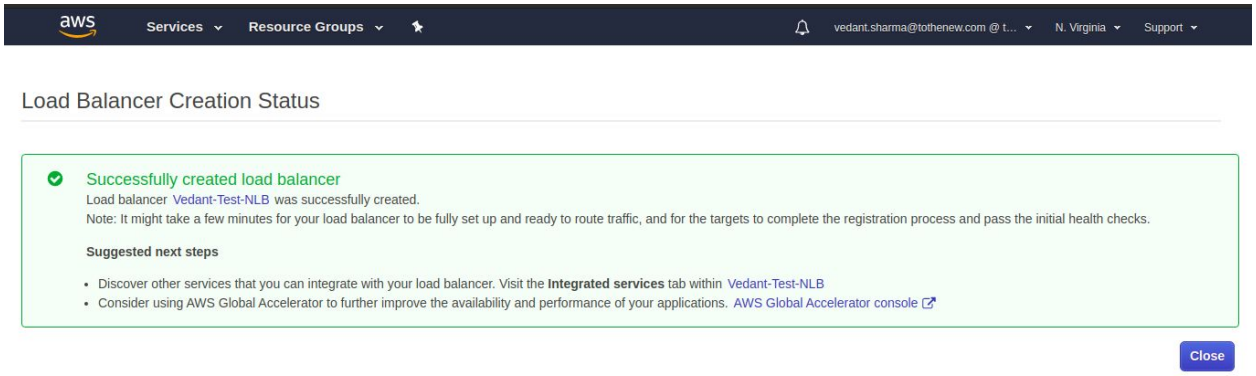

aws
Services
Resource Groups
vedant.sharma@tothenew.com @ t...
N. Virginia
Support
Rules
Vedant-Test-ALB | HTTP:80
Click a location for your new rule. Each rule must include one action of type forward, redirect, fixed response.
Cancel
Save
Rule limits for condition values, wildcards, and total rules.
Insert Rule
RULE ID
IF (all match)
THEN
1
A rule ID (ARN) is generated when you save your rule.
Path...
is
/wp-admin
or
Value
Add condition
1. Forward to...
Target group : Weight (0-999)
Vedant-test-TG2
1
Traffic distribution 100%
Select a target group
0
Group-level stickiness
Add action


Not secure
testwordpress.com/wp-login.php?redirect_to=http%3A%2F%2Fvedant-test-alb-737151566.us-east-1.elb.amazonaws.com%2Fwp-admin%2F
Apps
Gmail
YouTube
Maps
Dashboard
Devhints
Bash-Scripting
Username or Email Address
Password
Remember Me
Log In
Lost your password?
← Back to vedant

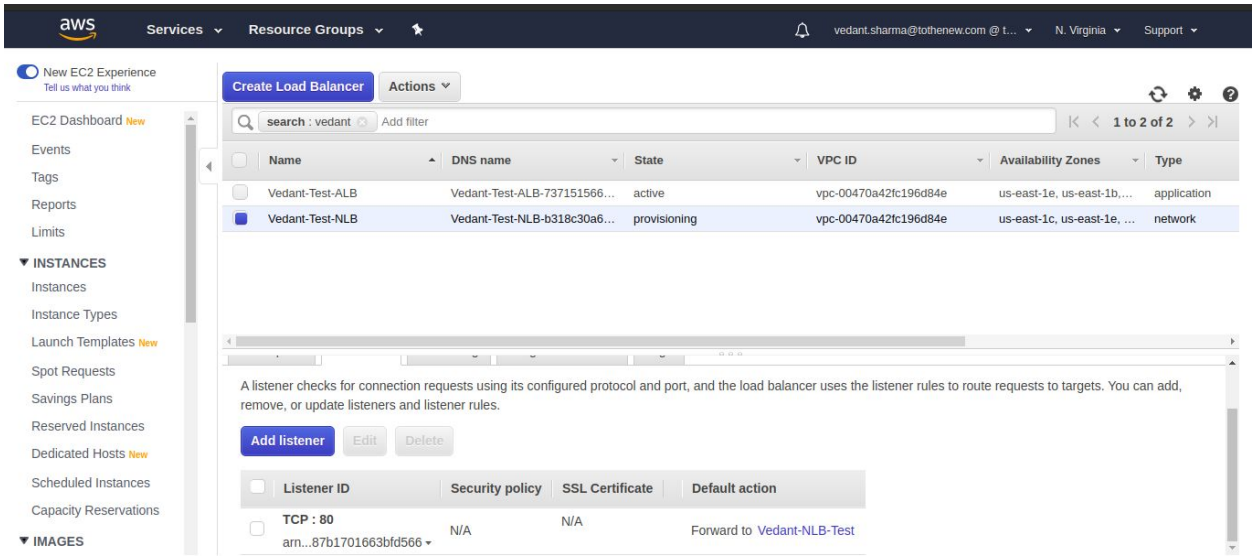
Q8. Use NLB that replaces the ALB in the above setup.



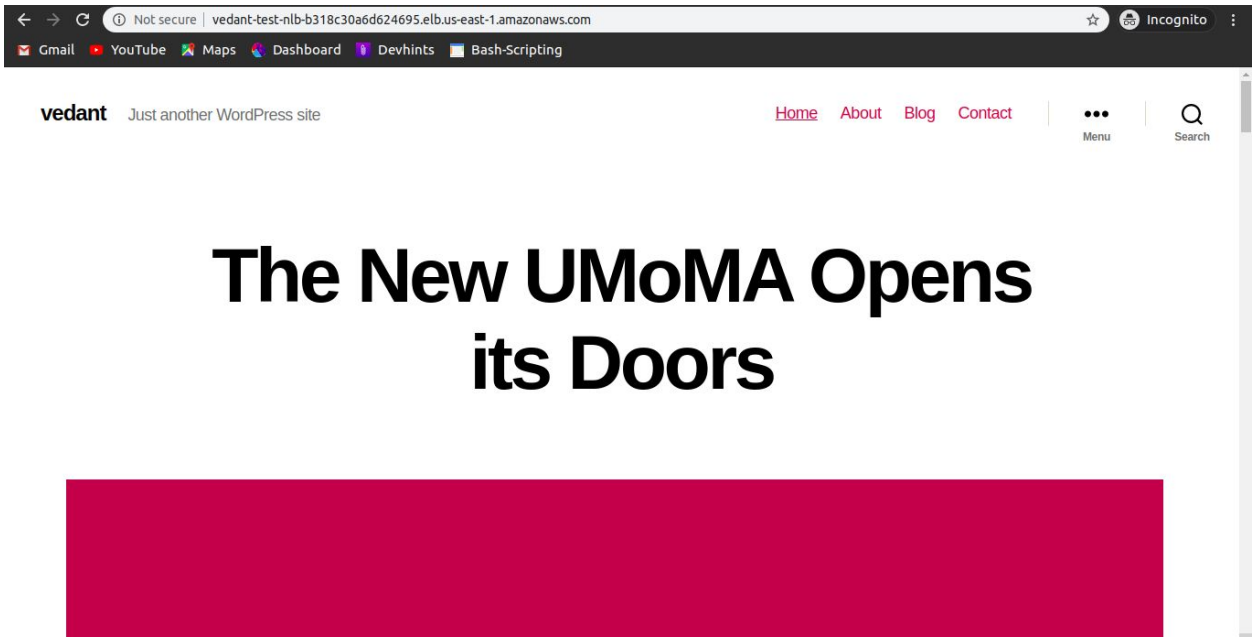




NLB using a different target group:

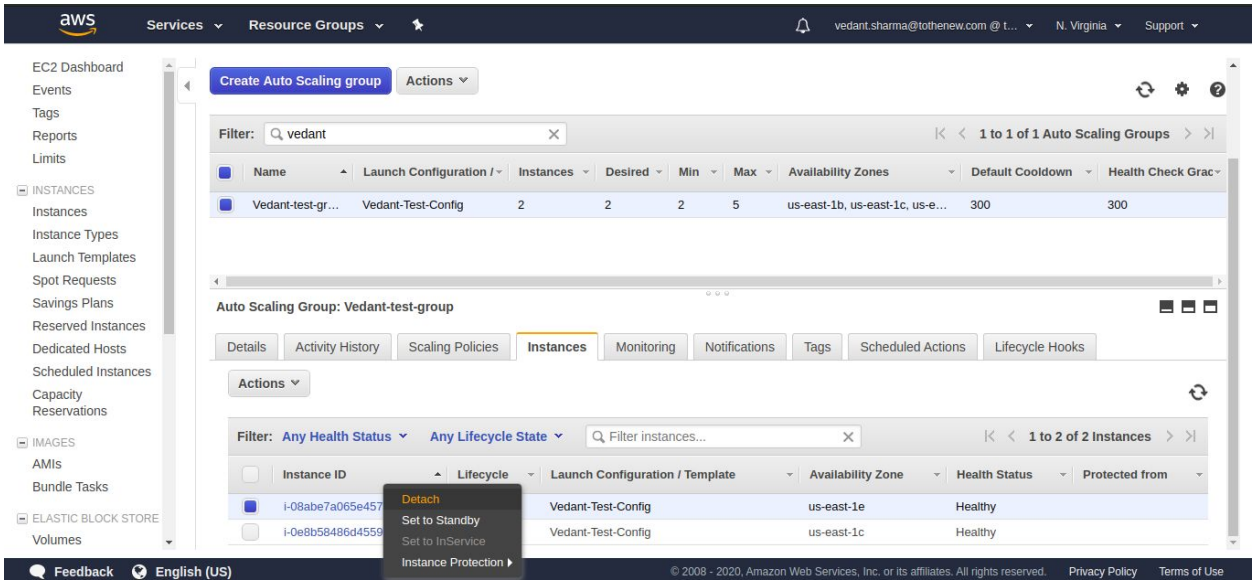


Wordpress website up using NLB

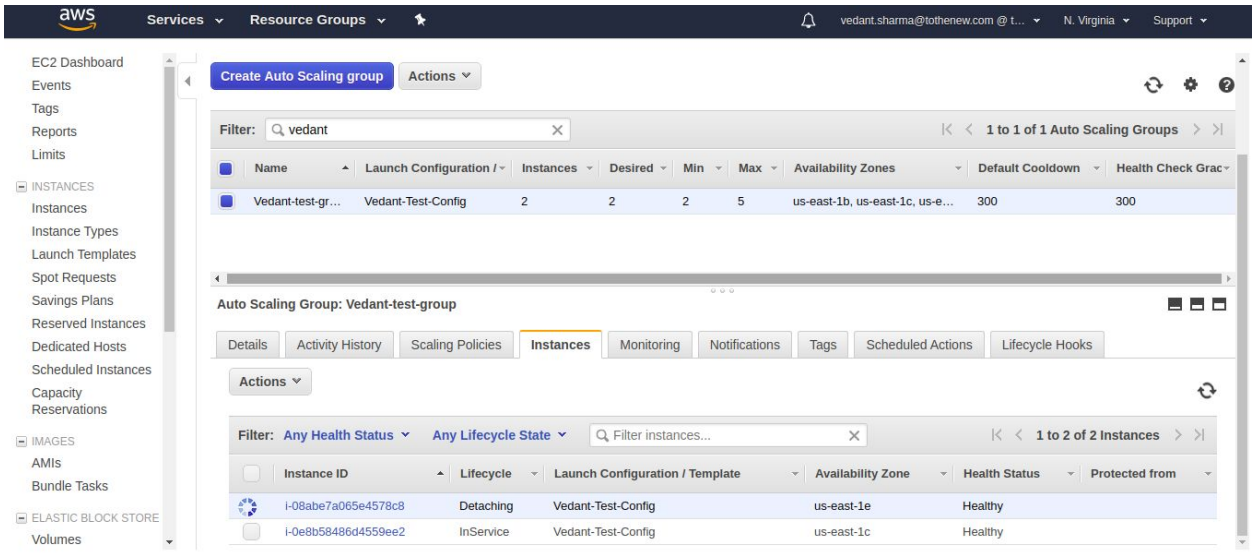


Q9. Take an instance out of the ASG

Editing the min number of instances required before detaching the instance:

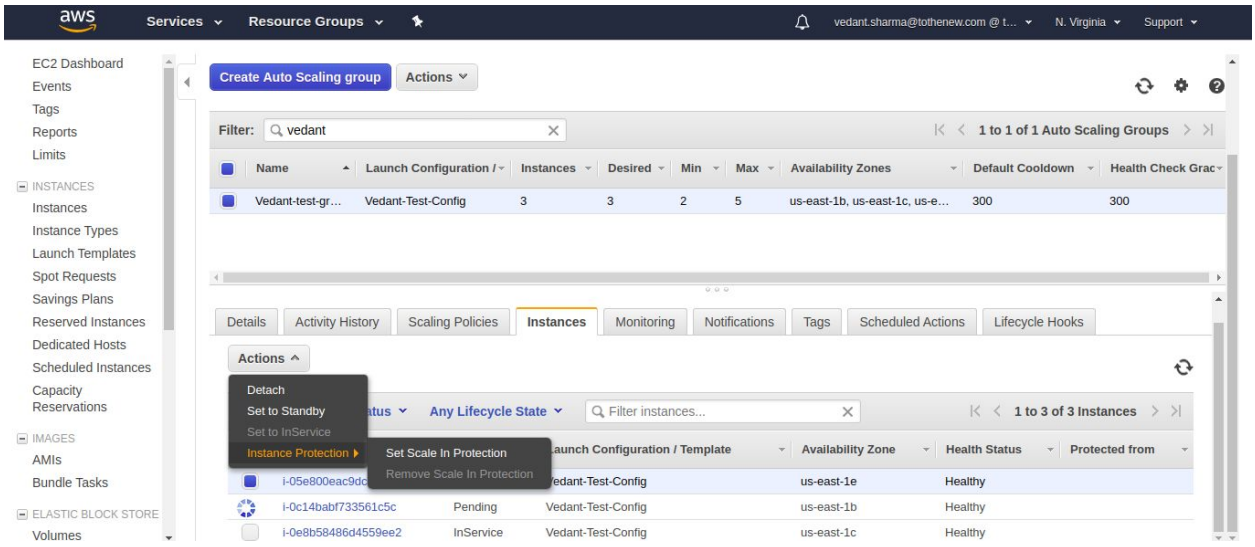


Detaching



Q10. Put scale-in protection on an instance in the ASG.

Selecting the instance under the autoscaling group and under actions tag selecting instance protection and then set scale in protection

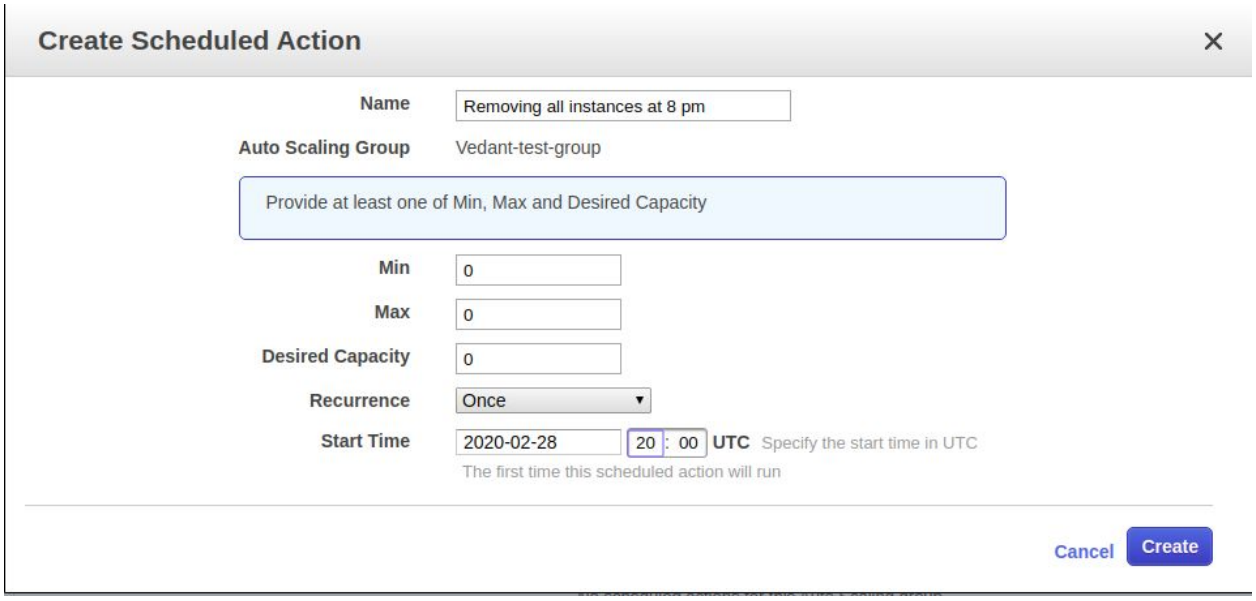


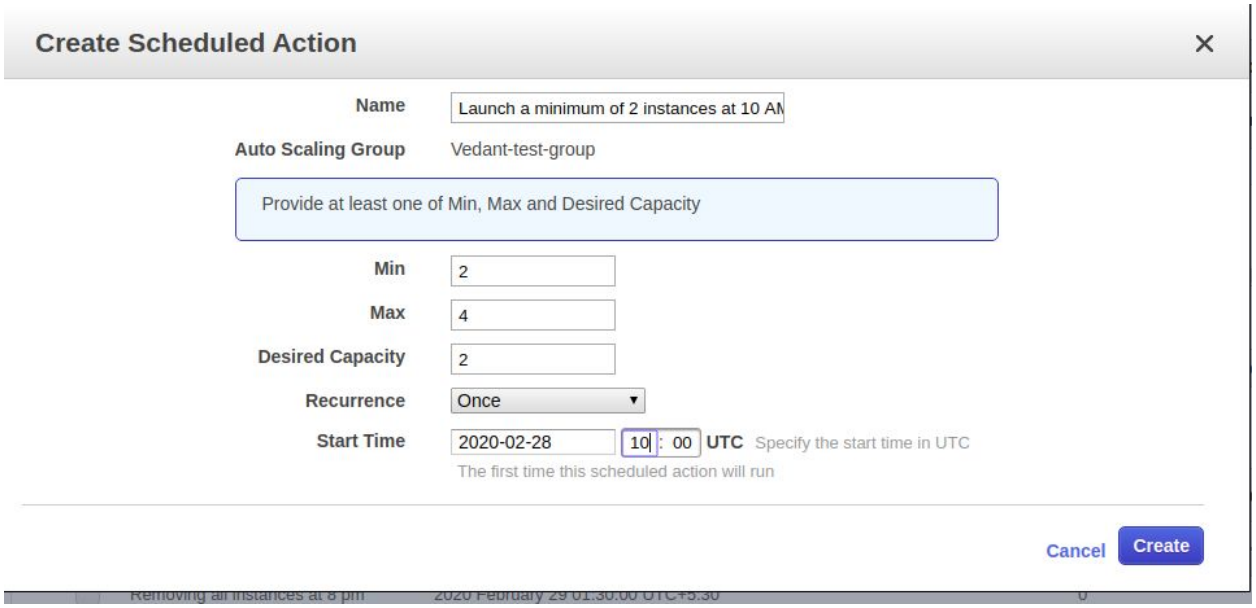
Protected from scale in:

Q11. Put Schedules in ASG to:

- Remove all instances of the ASG at 8 PM
- Launch a minimum of 2 instances at 10 AM

Creating a Schedule Action




Create Scheduled Action
Name
Launch a minimum of 2 instances at 10 AM
Auto Scaling Group
Vedant-test-group
Provide at least one of Min, Max and Desired Capacity
Min
2
Max
4
Desired Capacity
2
Recurrence
Once
Start Time
2020-02-28
10 : 00
UTC
Specify the start time in UTC
The first time this scheduled action will run
Cancel
Create