**AMAZON EC2**

**7.1** **Introduction**

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

**7.2** **Features of Amazon EC2**

Amazon EC2 provides the following features:

1. Virtual computing environments, known as instances
2. Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits you need for your server (including the operating system and additional software)
3. Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
4. Secure login information for your instances using key pairs, Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as instance store volumes
5. Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes
6. Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as regions and Availability Zones
7. A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
8. Metadata, known as tags, that you can create and assign to your Amazon EC2 resources
9. Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

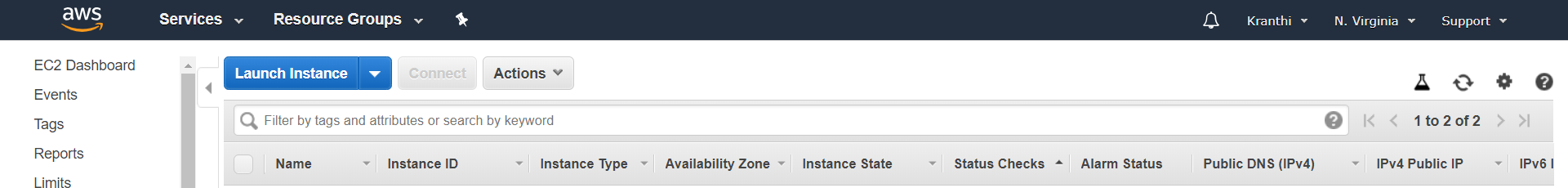
**7.3 How to launch an EC2 instance on AWS:**

Now we need to deploy this web portal on our EC2 instance to make this portal available for multiple users who wants to make use of.

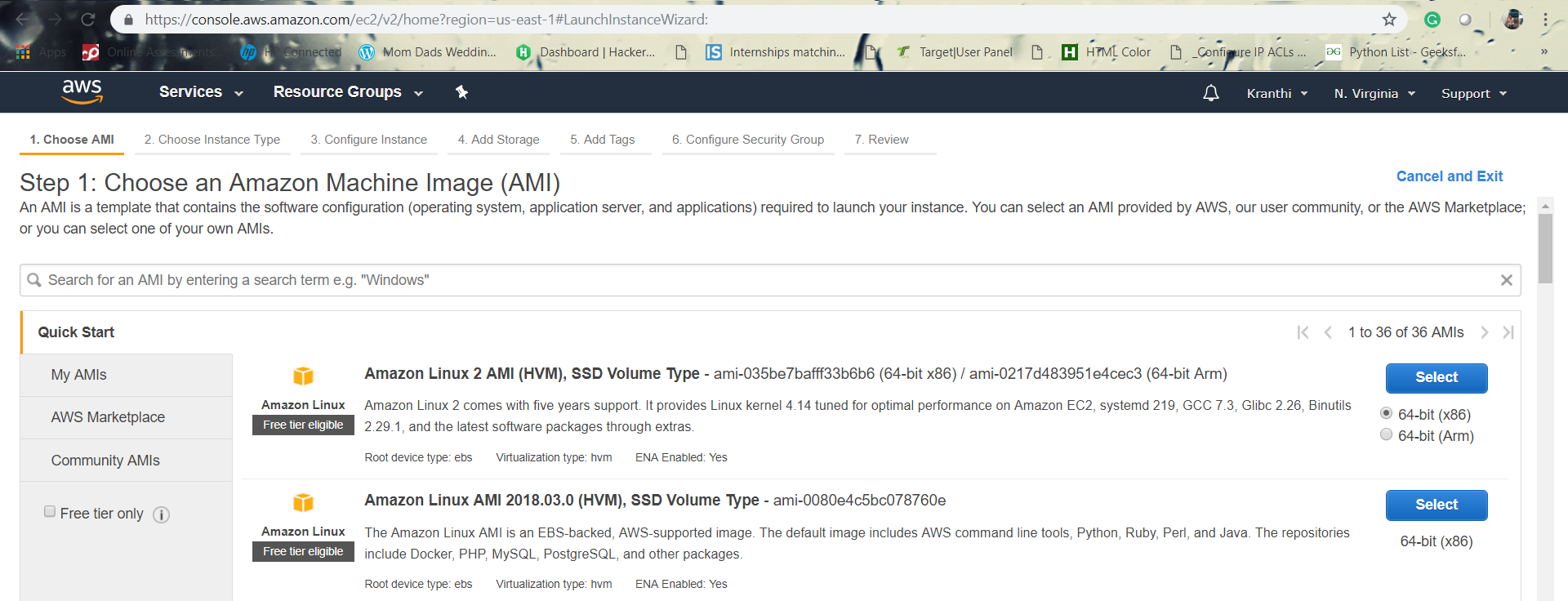
Creation of Instance in AWS Console:

Step 1: Choose the Instance option in AWS EC2.

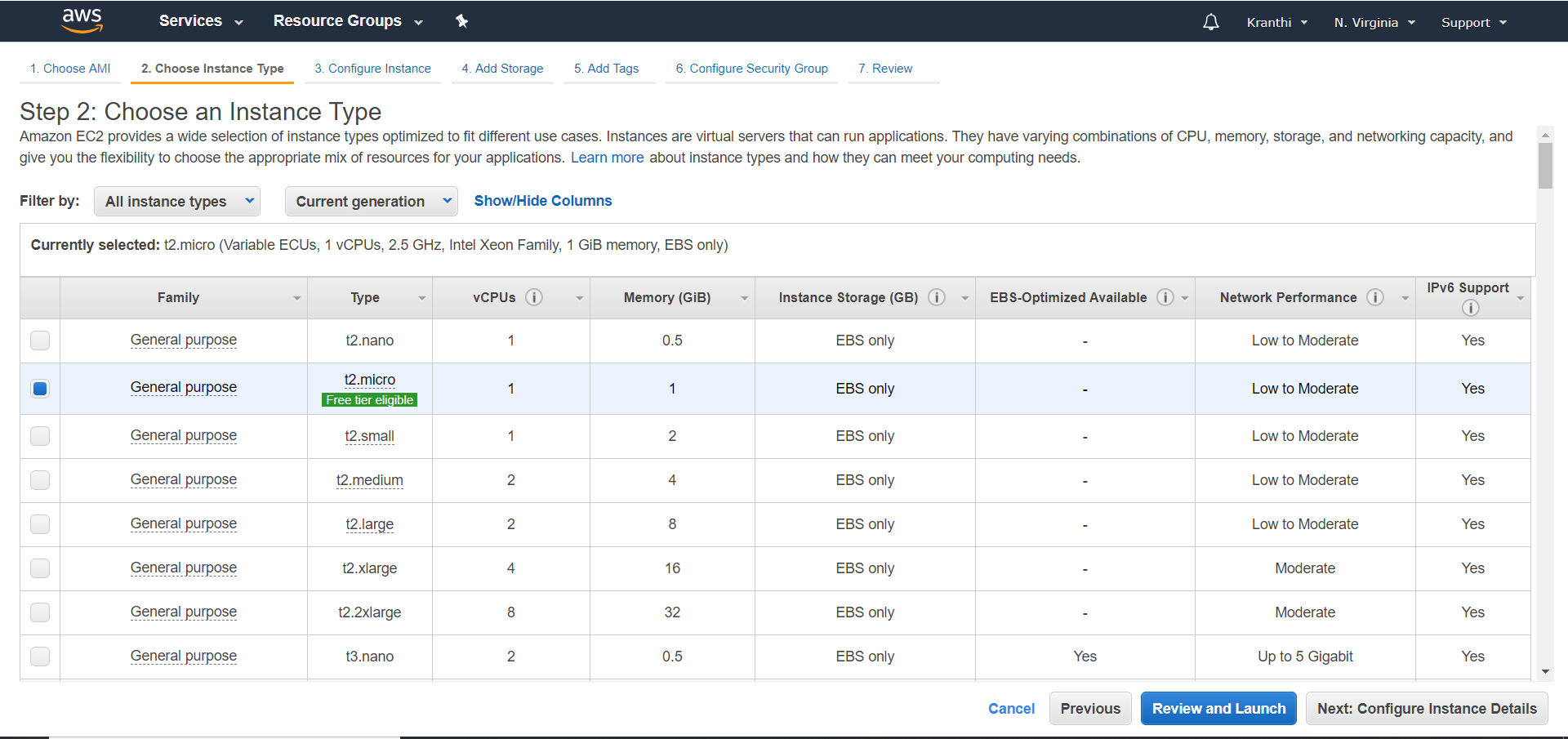
Step 2: Launch Instance that was on the top right side of the screen displayed.



Step 3: Choose an Amazon Machine Image(AMI).We choose (**Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type** - ami-0080e4c5bc078760e) free tier to launch our web portal.



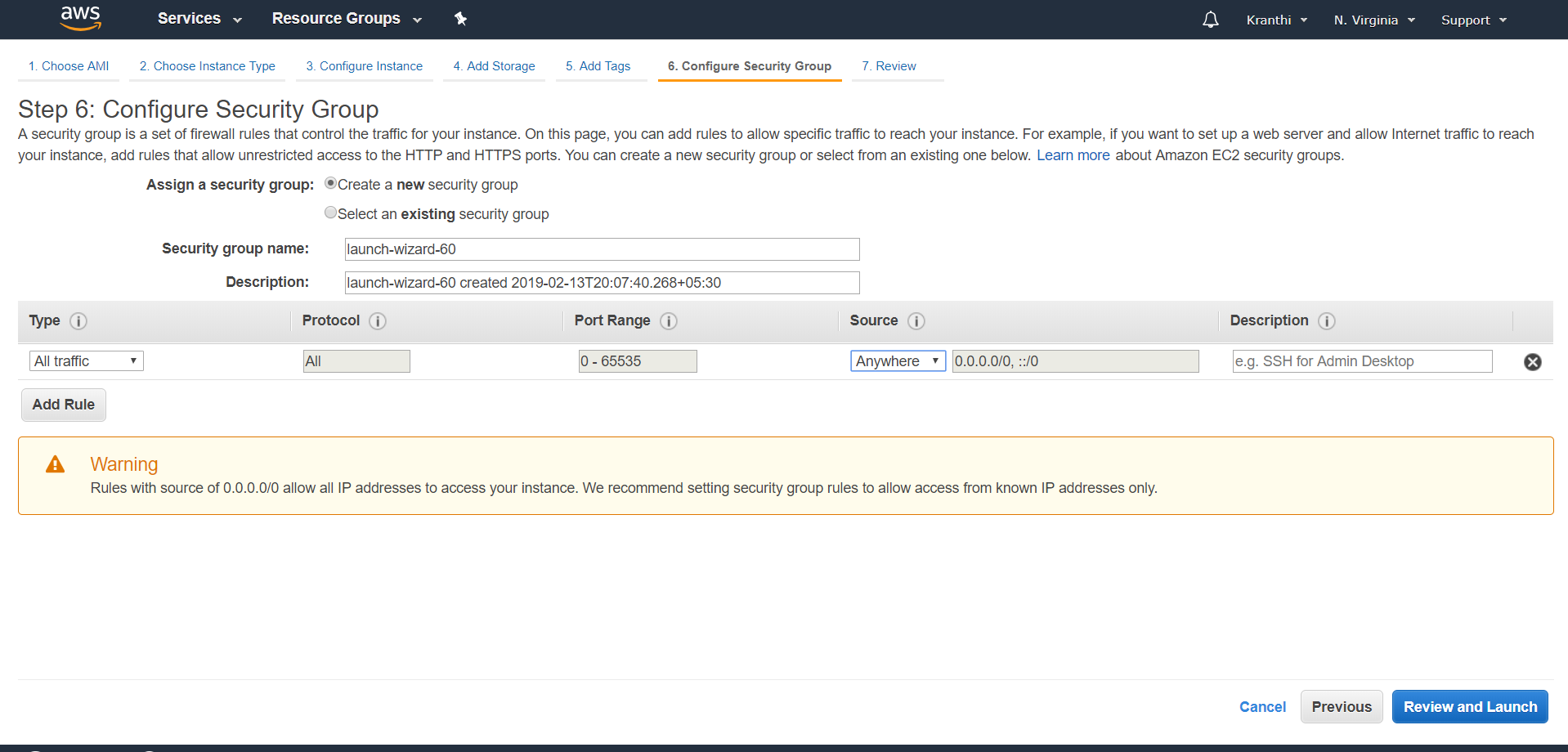
Step 4: Now choose an Instance type which fits to different use cases. Instances are Virtual Web servers that run our application. We choose (**:** t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)) for general purpose.



Step 5: Configure Instance details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Step 6: Add security groups

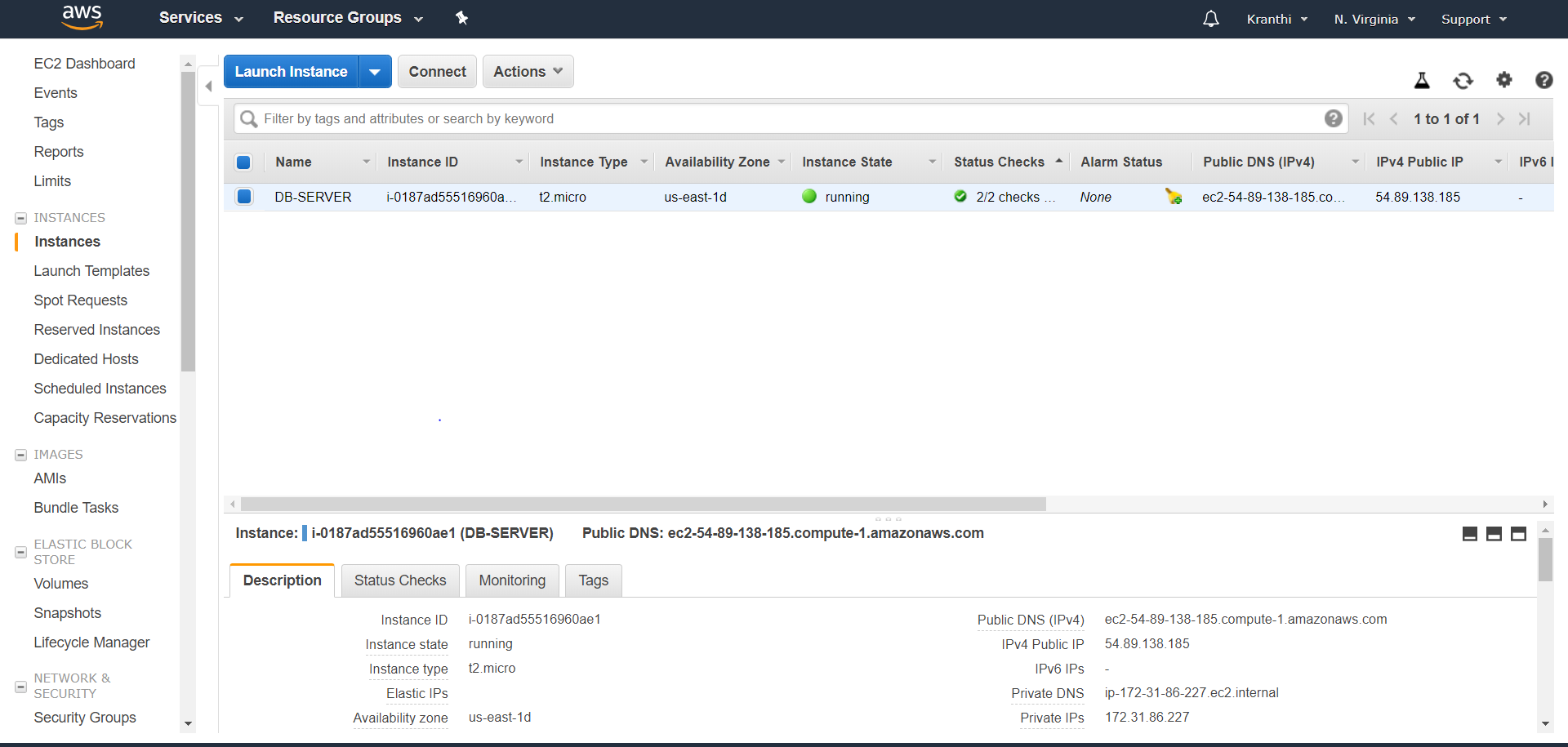


Step 7: On the **Review Instance Launch** page, choose **Launch**.

When prompted for a key pair, select **Choose an existing key pair**, then select the key pair that you created when getting set up.

Alternatively, you can create a new key pair. Select **Create a new key pair**, enter a name for the key pair, and then choose **Download Key Pair**.

This is the only chance to save the private key file, so be sure to download it. Save the private key file in a safe place. You'll need to provide the name of your key pair when launching an instance and the corresponding private key each time you connect to the instance.

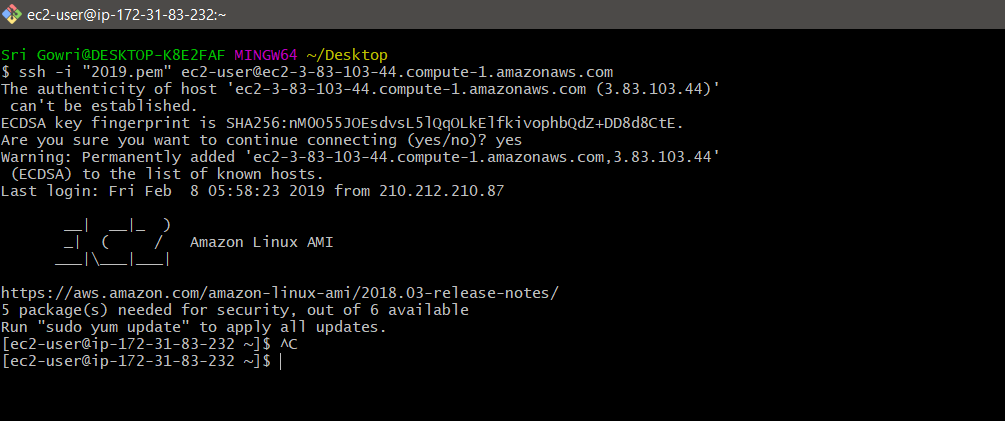


1. Now launch the web portal on the created EC2 Instance. To deploy the web portal, we need to have a little idea about LINUX.
2. Install and run Gitbash, to manage ec2 instances.
3. Install filezilla. To move our web code files from desktop to cloud EC2 instance.

**7.3.1 Execute the following commands on Gitbash console:**

1. To connect EC2 instance remotely use the below command.

$ssh -i "2019.pem" [ec2-user@ec2-3-83-103-44.compute-1.amazonaws.com](mailto:ec2-user@ec2-3-83-103-44.compute-1.amazonaws.com)



**Fig 7.1: Git Bash Console**

**7.3.2 Procedure for running scripts on server:**

Sudo su //user permissions to access the EC2

Ls

Cd Resources/

ls

chmod 700 db\_setup.sh

chmod 700 initialSetupwensiteDeploy.sh

yum install mysql-server

service mysqld start

./db\_setup.sh

mysql -u root

mysql> show databases;

[root@ip-172-31-84-75 Resources]# **ls**

db\_setup.sh initialSetupWensiteDeploy.sh User.sql

[root@ip-172-31-84-75 Resources]# **mysql atmospheredb <User.sql**

[root@ip-172-31-84-75 Resources]# **mv initialSetupWensiteDeploy.sh /home/ec2-user/**

[root@ip-172-31-84-75 Resources]# **ls**

db\_setup.sh User.sql

[root@ip-172-31-84-75 Resources]# **cd ..**

[root@ip-172-31-84-75 ec2-user]# **ls**

as.pem initialSetupWensiteDeploy.sh Resources

[root@ip-172-31-84-75 ec2-user]# **./initialSetupWensieDeploy.sh**

## Installing the NodeSource Node.js 10.x repo...

[root@ip-172-31-84-75 ec2-user]# **cd db**

[root@ip-172-31-84-75 db]# **ls**

index.js public

[root@ip-172-31-84-75 db]# **mv index.js /home/ec2-user/website-deploy/**

[root@ip-172-31-84-75 db]# **mv public /home/ec2-user/website-deploy/**

[root@ip-172-31-84-75 db]# **ls**

[root@ip-172-31-84-75 db]# **cd ..**

[root@ip-172-31-84-75 db]# **npm install express –save**

[root@ip-172-31-84-75 db]# **npm install mysql --save**

[root@ip-172-31-84-75 ec2-user]# **cd wbsite-deploy/**

[root@ip-172-31-84-75 wbsite-deploy]# **ls**

index.js node\_modules package.json package-lock.json public

[root@ip-172-31-84-75 wbsite-deploy]# **node index.js**

**ELASTIC LOAD BALANCER**

**8.1 Introduction**

Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, and Lambda functions. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones. Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault tolerant.

These are classified into 3 types, they are Application Load Balancer, Network Load Balancer, and Classic Load Balancer. In our project we have used Application Load Balancer, this operates at the request level (layer 7), routing traffic to targets – EC2 instances, containers, etc. This provides advanced request routing targeted at delivery of modern application architectures. This simplifies and improves the security of our application, by ensuring that the latest SSl/TLS ciphers and protocols are used at all times.

* 1. **Advantages**

**High availability**

Elastic Load Balancing automatically distributes traffic across multiple targets – Amazon EC2 instances, containers and IP addresses – in a single Availability Zone or multiple Availability Zones.

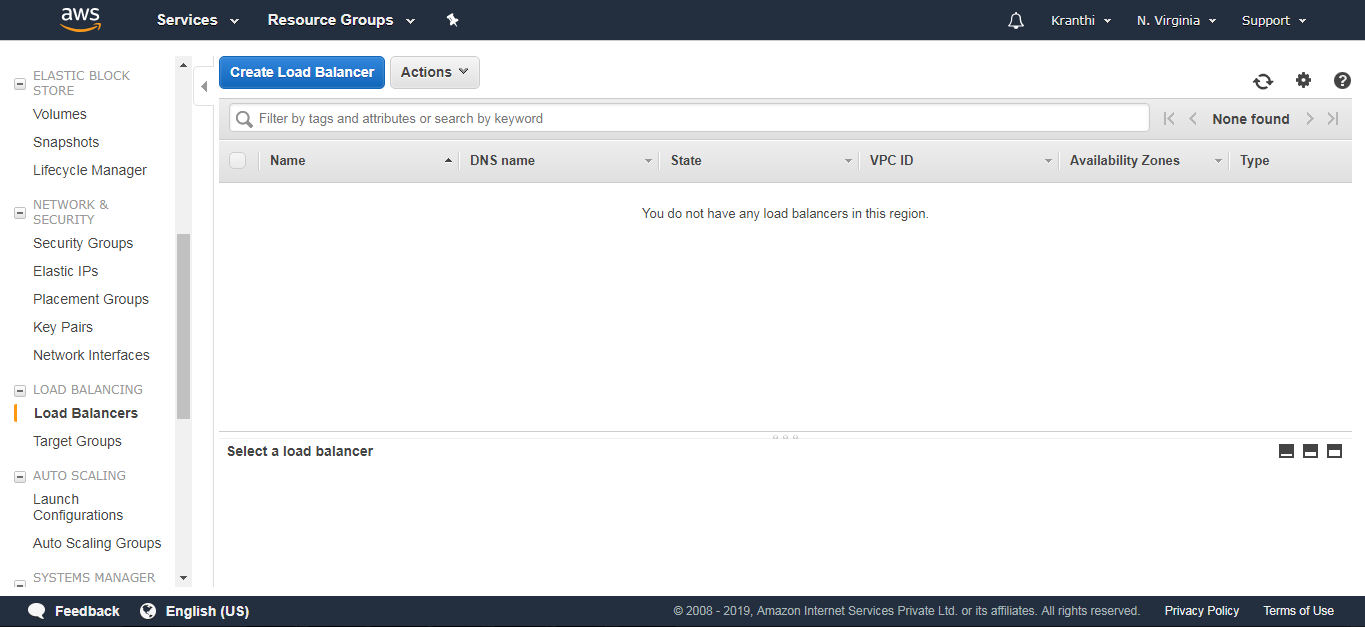
**Health Checks:**

An Application Load Balancer routes traffic only to healthy targets. With an Application Load Balancer, you get improved insight into the health of your applications in two ways: (1) health check improvements that allow you to configure detailed error codes from 200-499. The health checks allow you to monitor the health of each of your services behind the load balancer; and (2) new metrics that give insight into traffic for each of the services running on an EC2 instance.

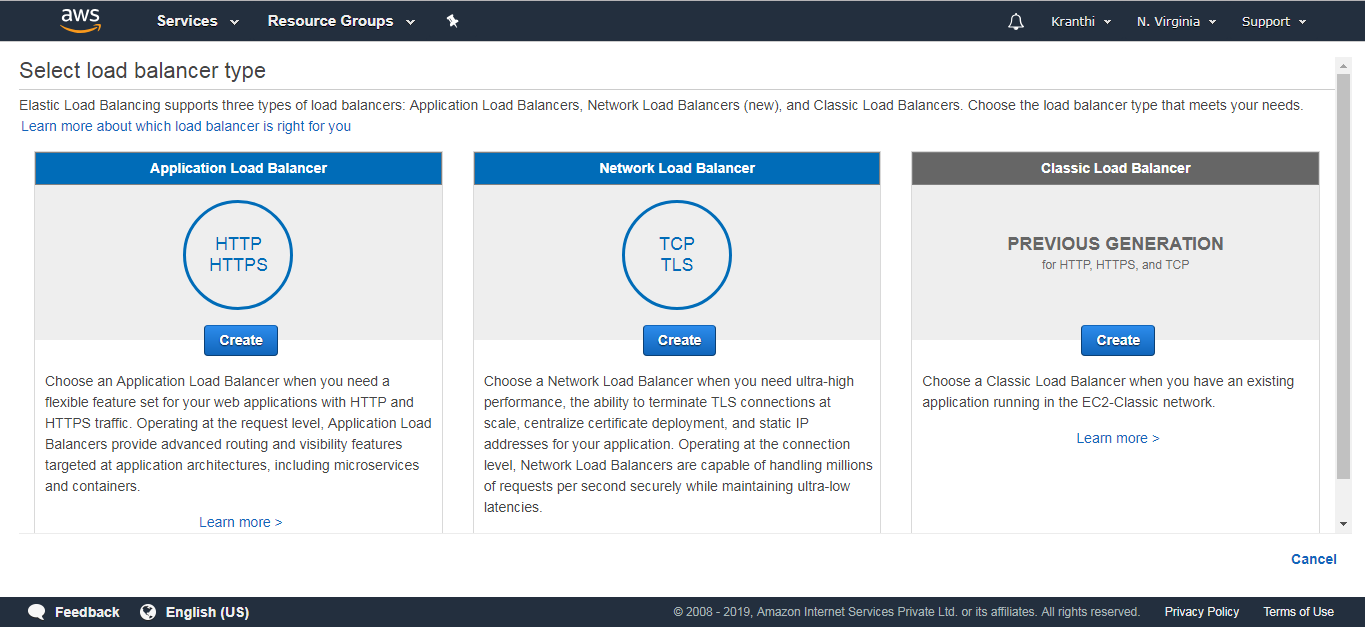
**8.3 How to add Load Balancer in AWS:**

Step 1: Login to the AWS Console.

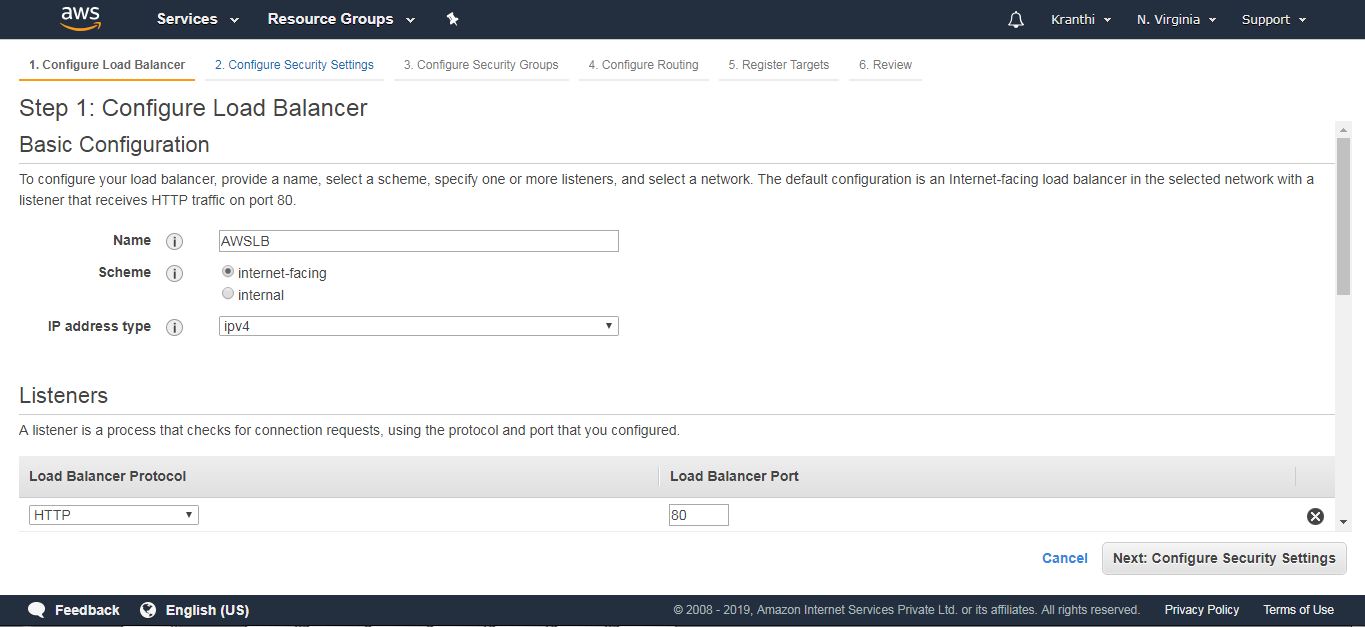
Step 2: Select EC2 Instances, there you find Load Balancer.

Step 3: Choose Create Load Balancer that appeared on the screen below.

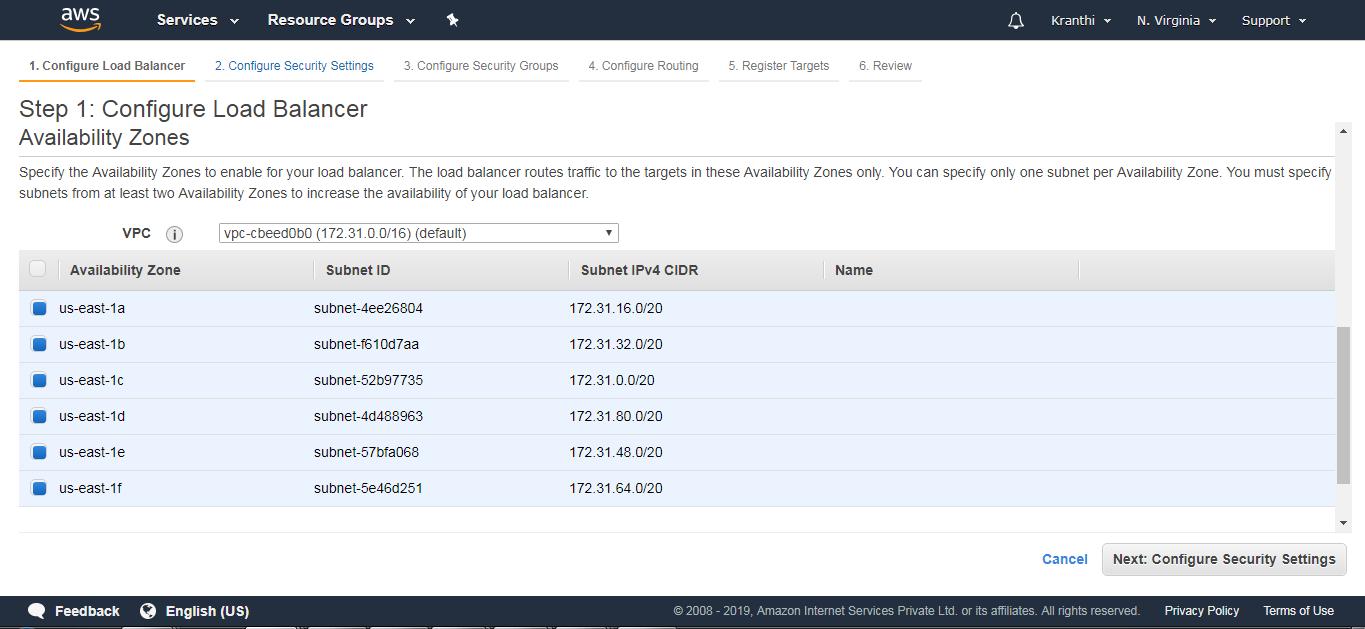
Step 4: Select Application Load Balancer and click on create.

 **Fig 8.1: Load Balancers**

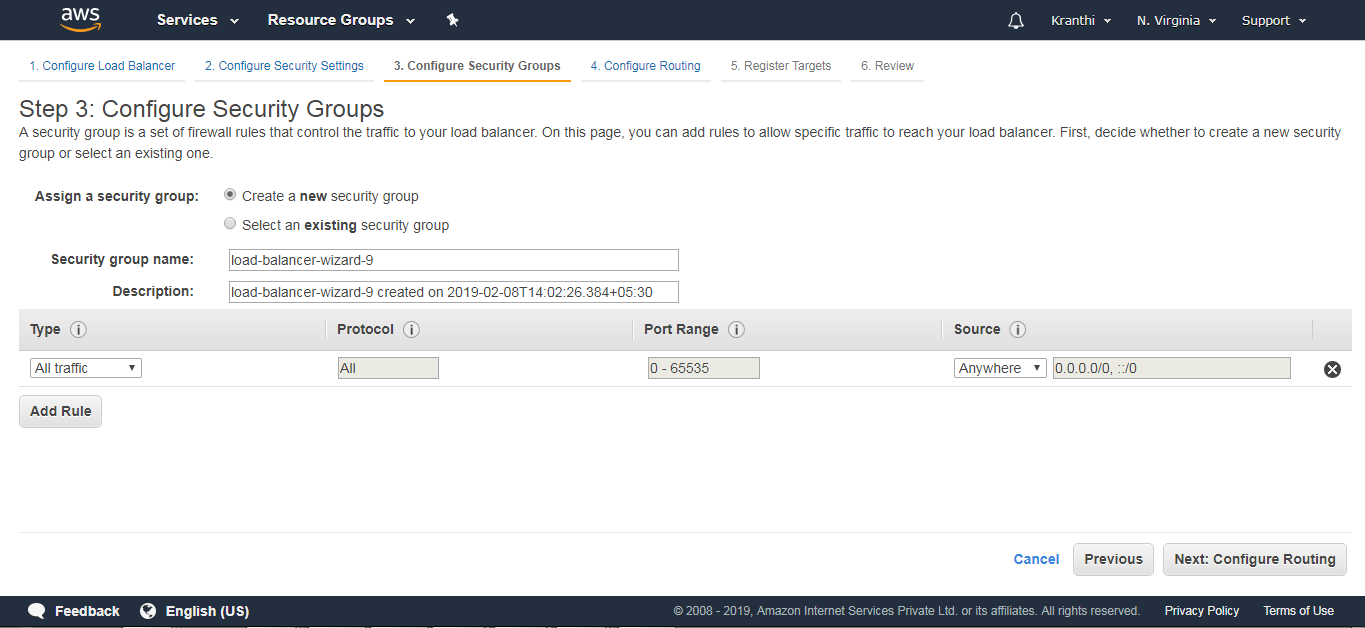
Step 5: Configure Load Balancer.



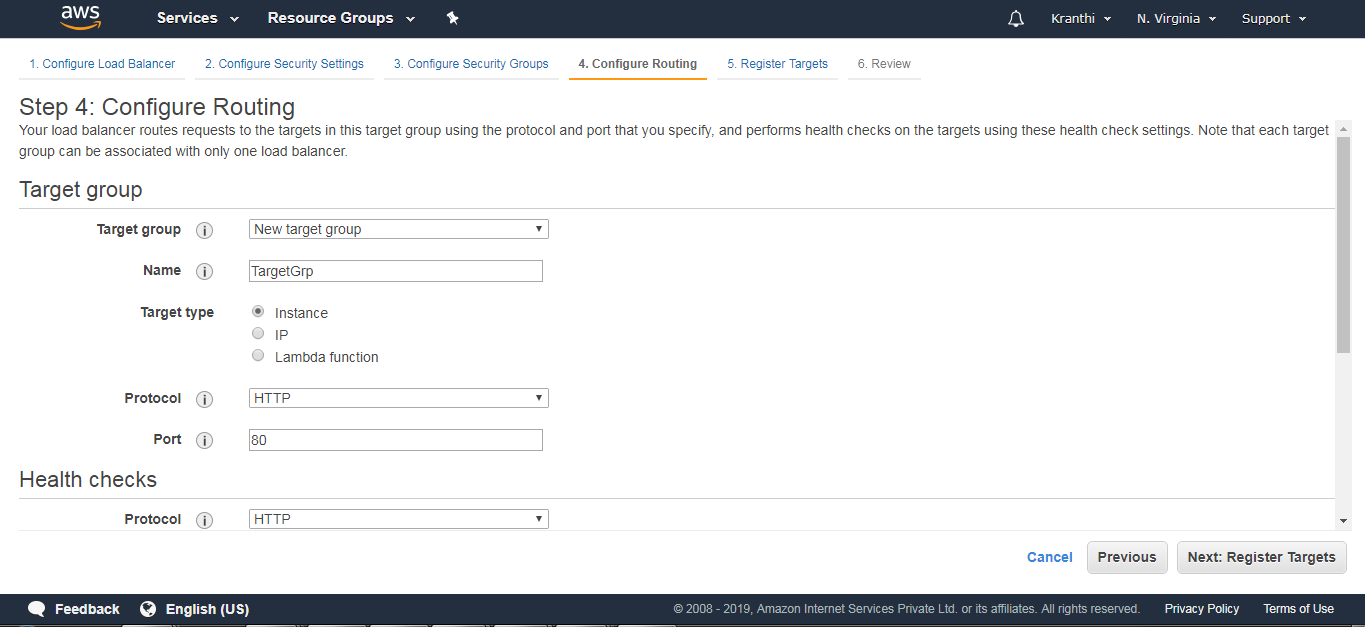
Add the availability Zones

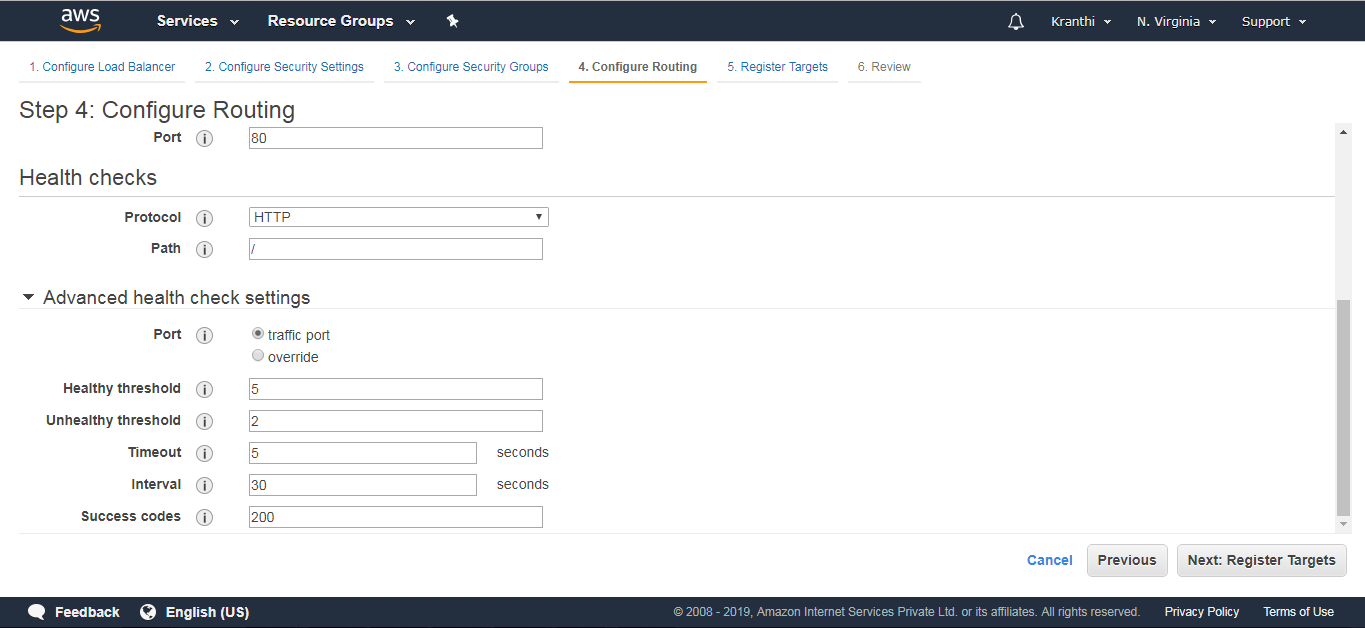


Step 6: Configure Security groups.

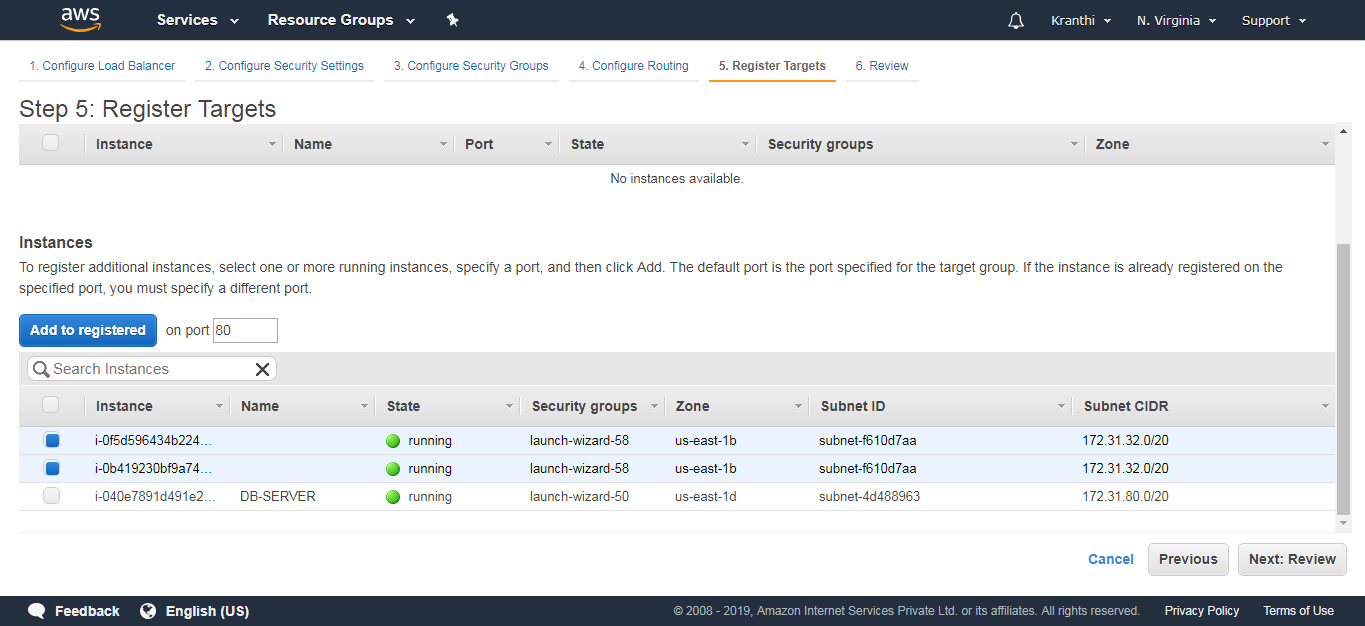


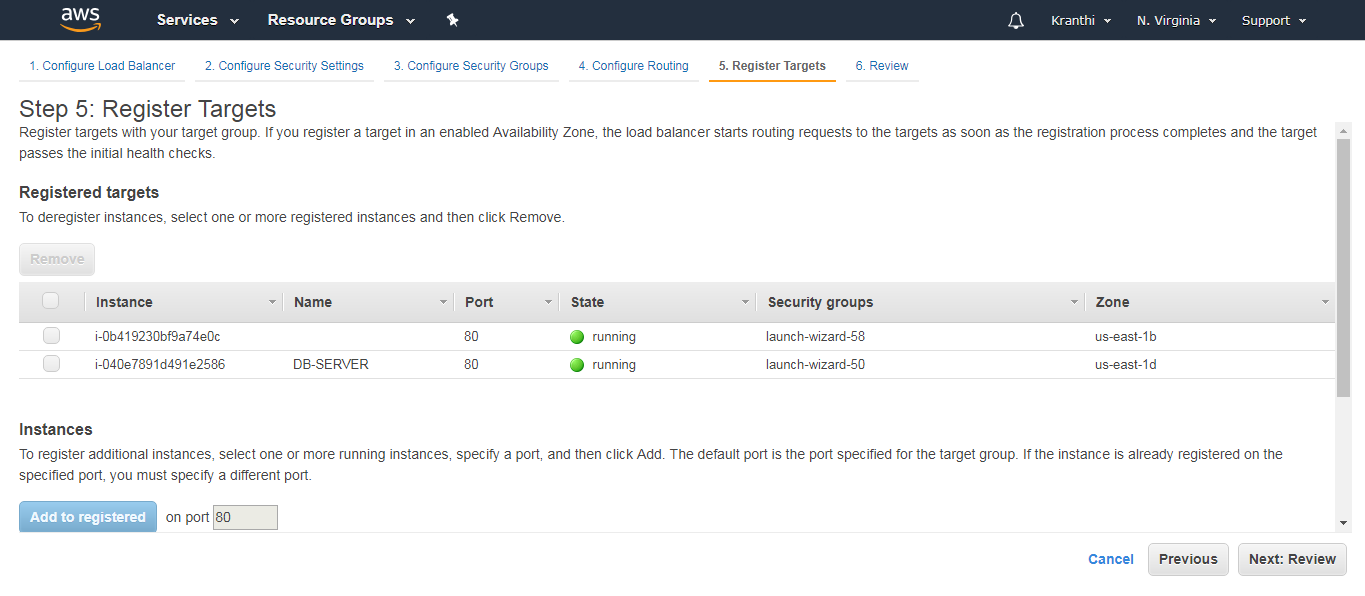
Step 7:Configure Routing



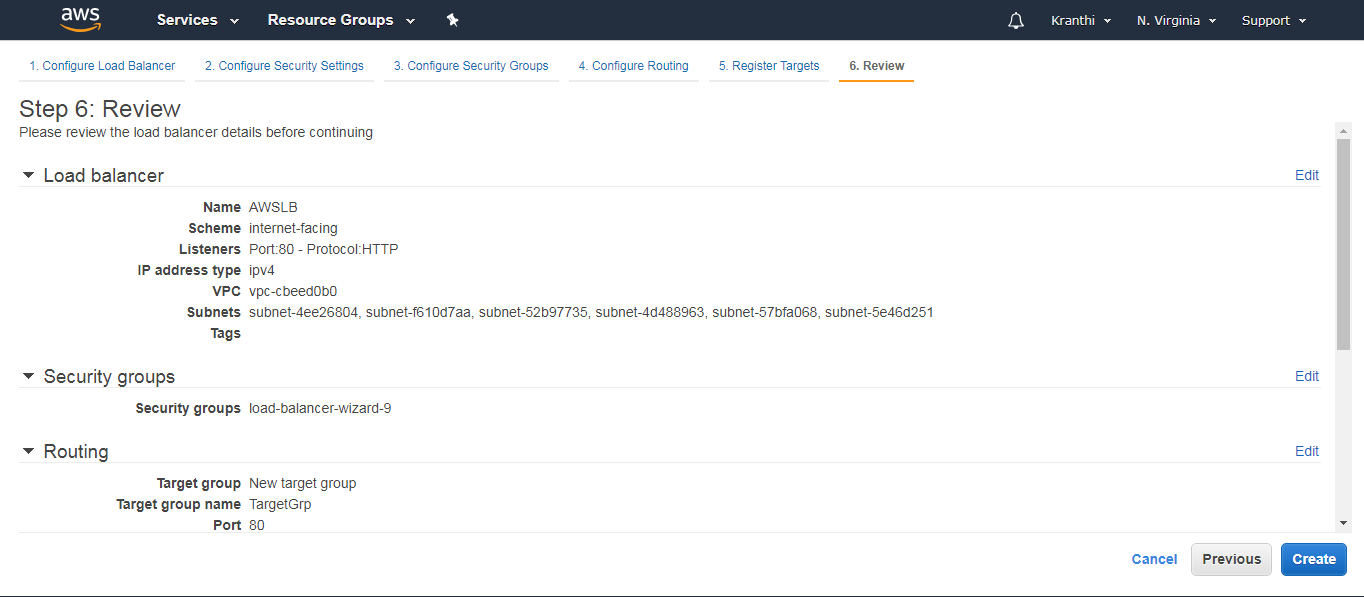


Step 8: Register Targets (nothing but EC2 instances that are created).

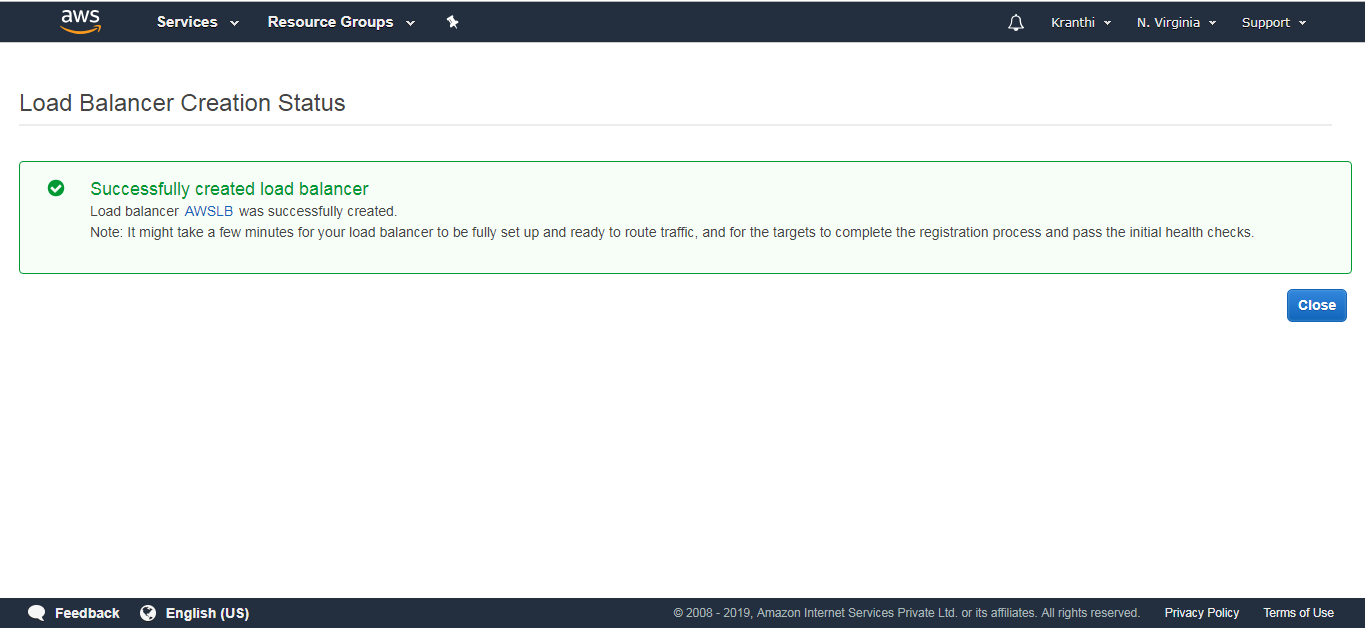




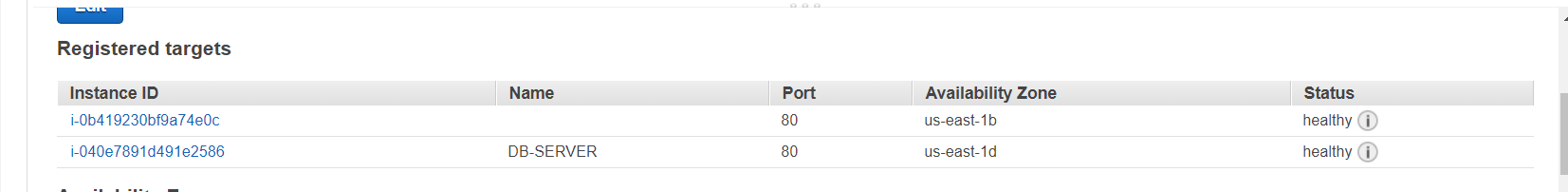
Review: View all the information of an Elastic Load Balancer.



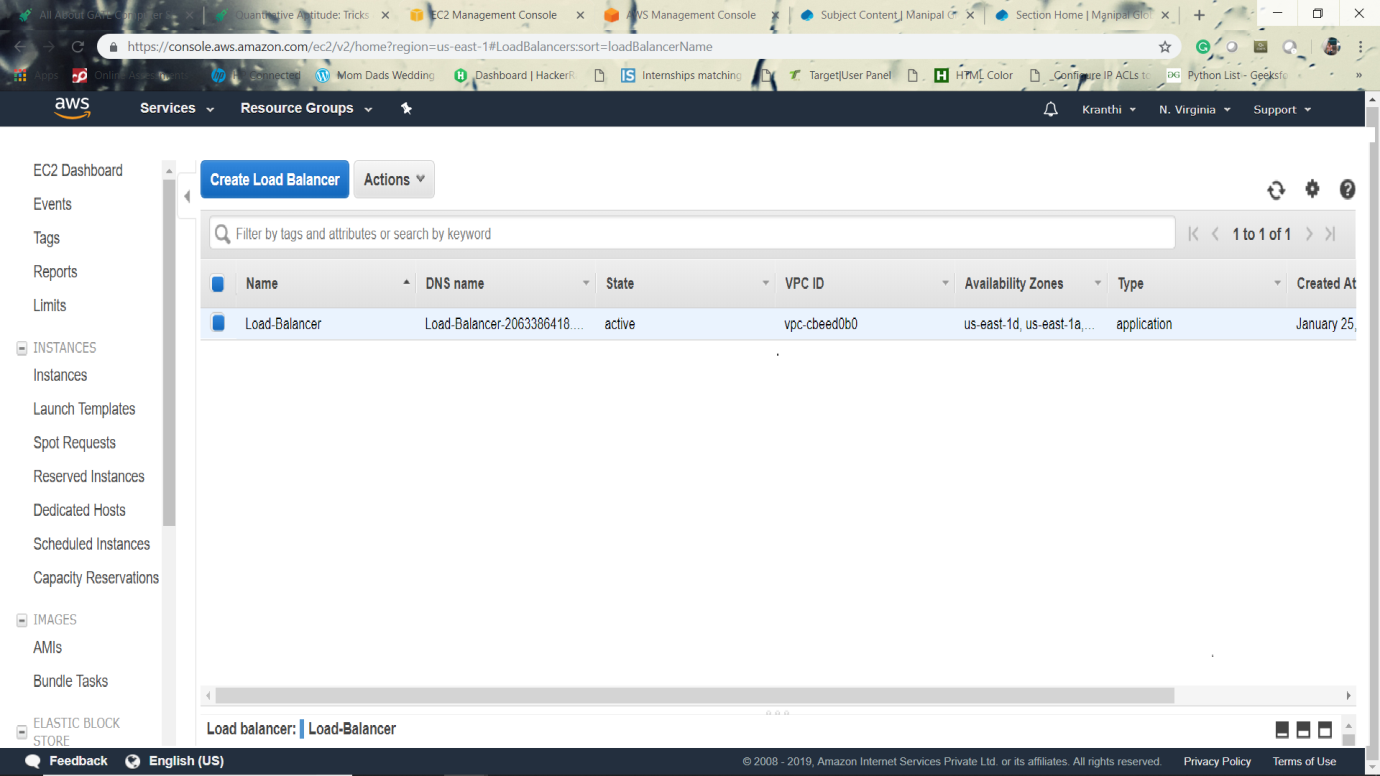
Step 9: We can now notice the notification that Load Balancer is created.



Check whether the web instances are healthy when added to a load balancer.



Step 10: This is the Load Balancer that we have created.



The web portal runs on a specified URL of Elastic Load Balancer

**AUTO SCALING**

**9.1 Introduction**

An Auto Scaling group contains a collection of Amazon EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of instance scaling and management. If you might want to increase the number of instances in that group to improve the performance of the application. Or, you can decrease the number of instances to reduce costs when demand is low. Use the Auto Scaling group to scale the number of instances automatically based on criteria that you specify. You could also maintain a fixed number of instances even if an instance becomes unhealthy. This automatic scaling and maintaining the number of instances in an Auto Scaling group is the core functionality of the Amazon EC2 Auto Scaling service.

There are two tasks to be followed for Auto-Scaling, they are

1. Create a launch template

2. Create Auto-Scaling group.

**9.2 Create a Launch Template**

**To create a launch template for an Auto Scaling group**

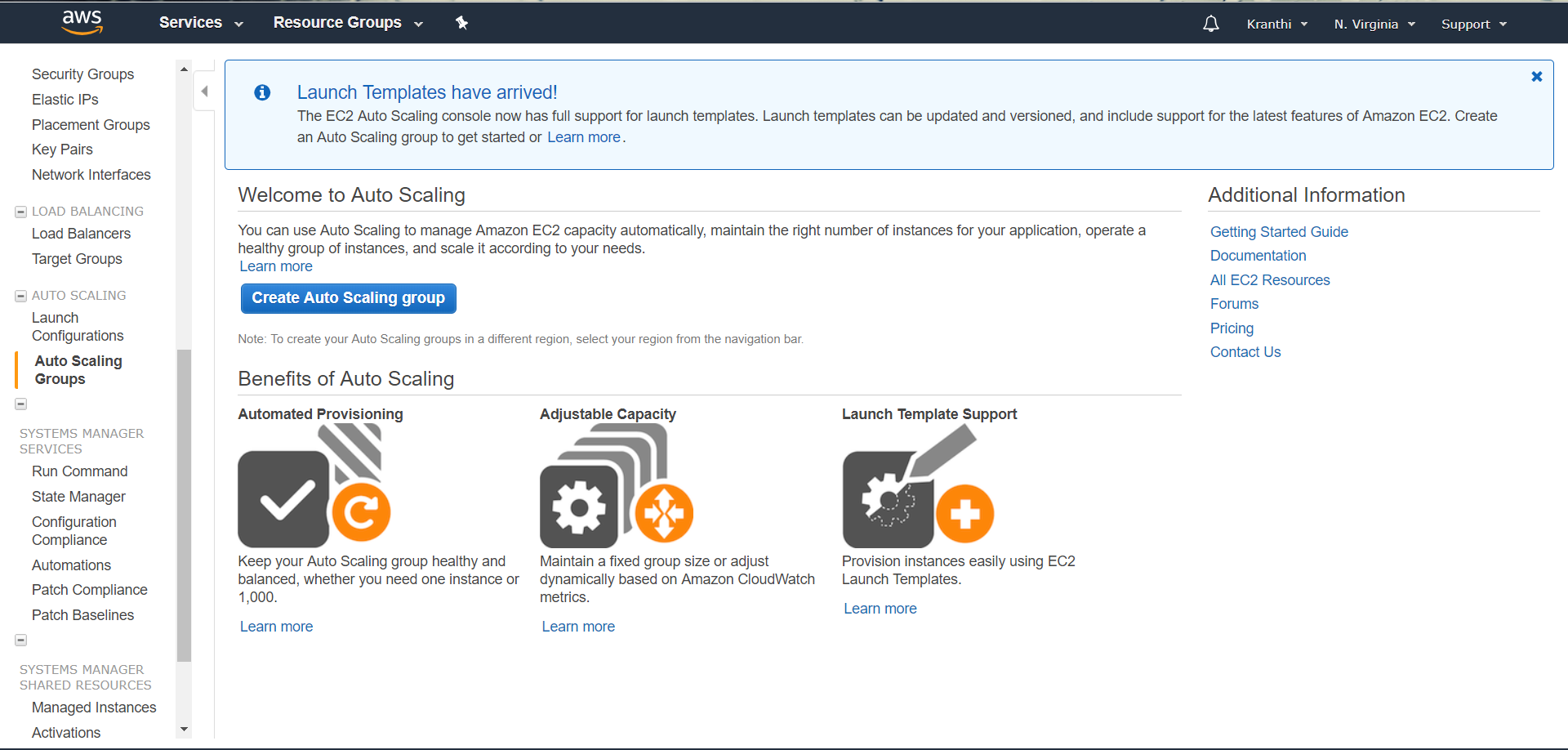
Step 1: Open the Amazon EC2 console.

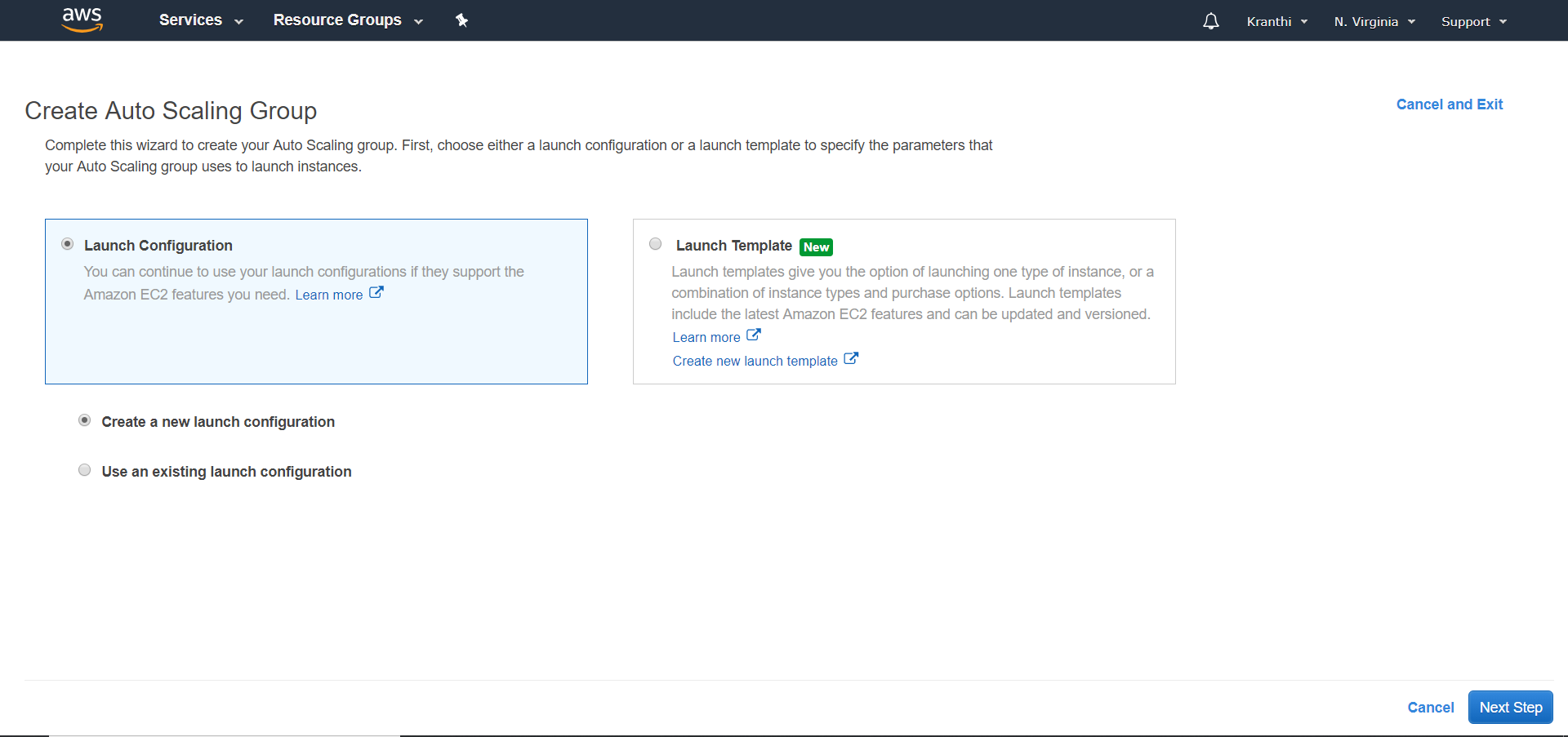
Step 2: On the navigation bar, select a region. The Amazon EC2 Auto Scaling resources that you create are tied to the region you specify.

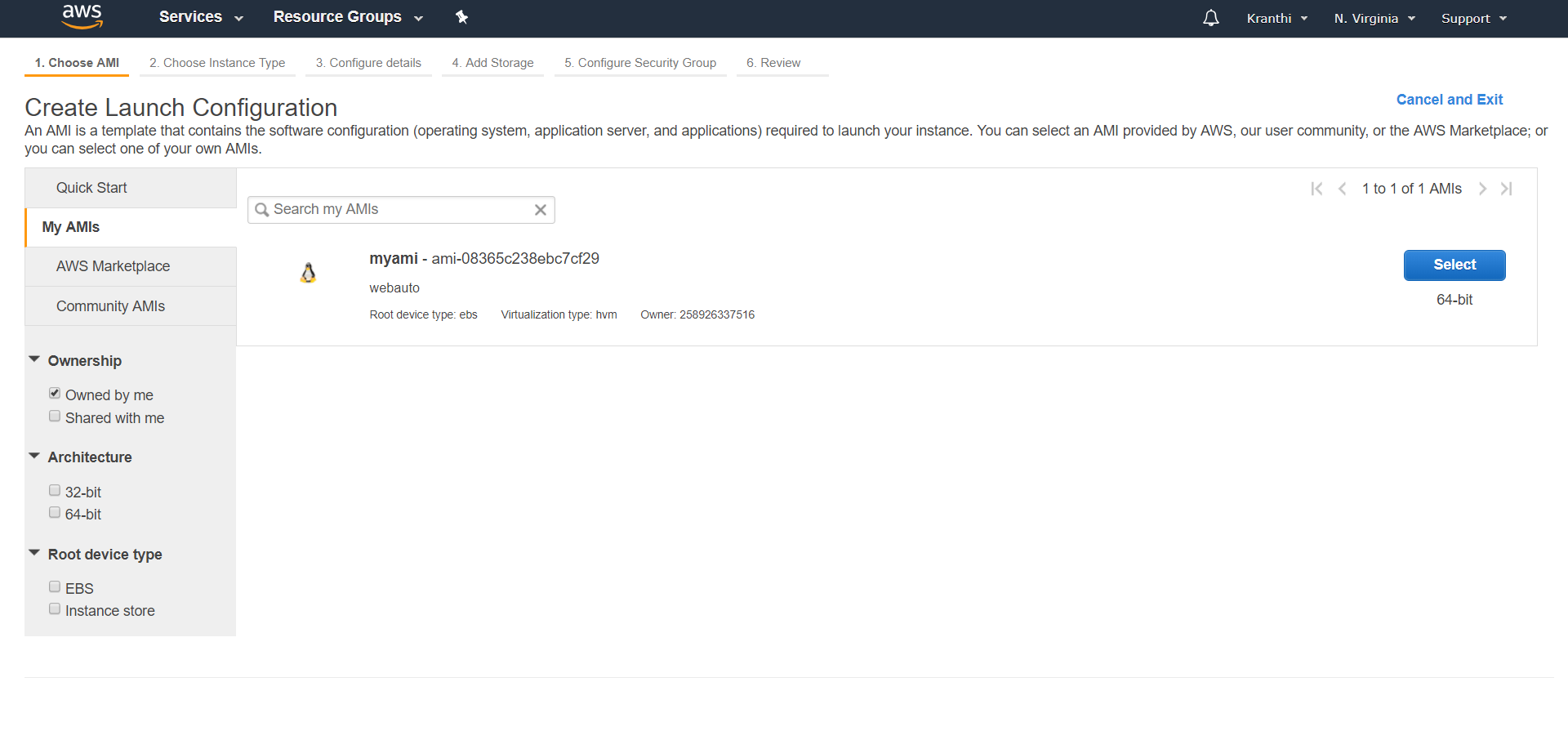
Step 3: On the navigation pane, choose Instances, Launch Templates.

Step 4: Choose Create launch template.

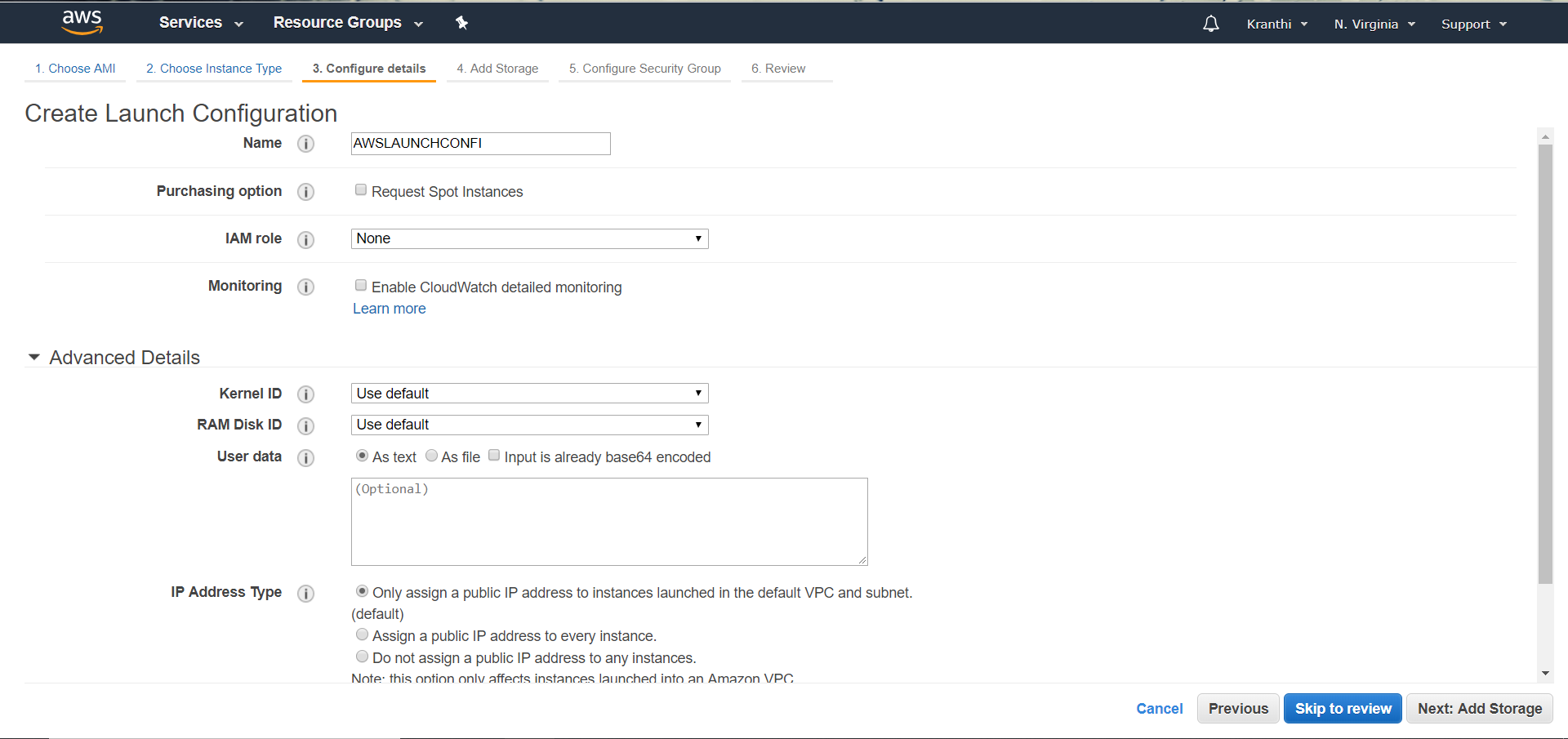
Step 5: Create Auto Scaling group.



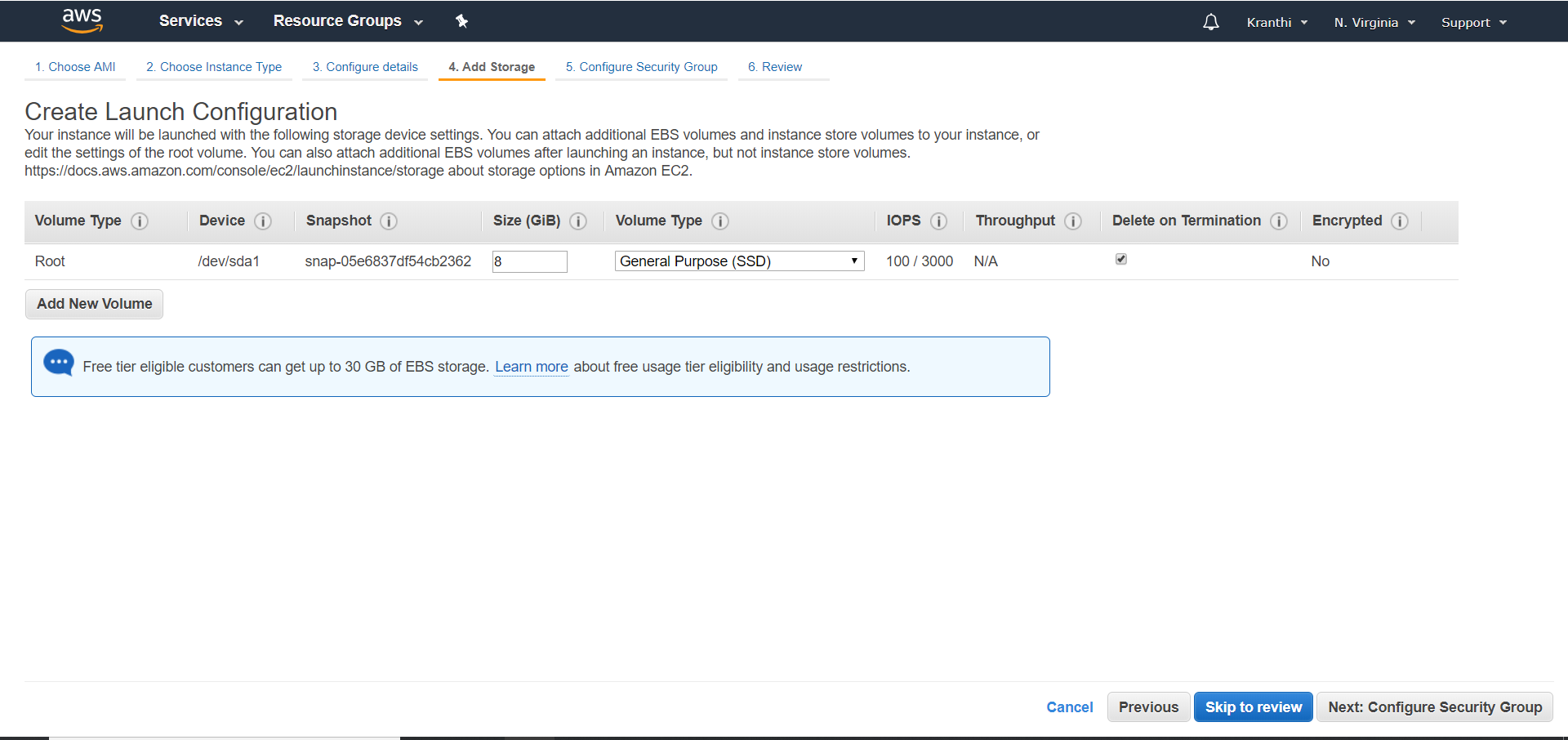
****An AMI is a template that contains the software configuration to launch the instances.

****

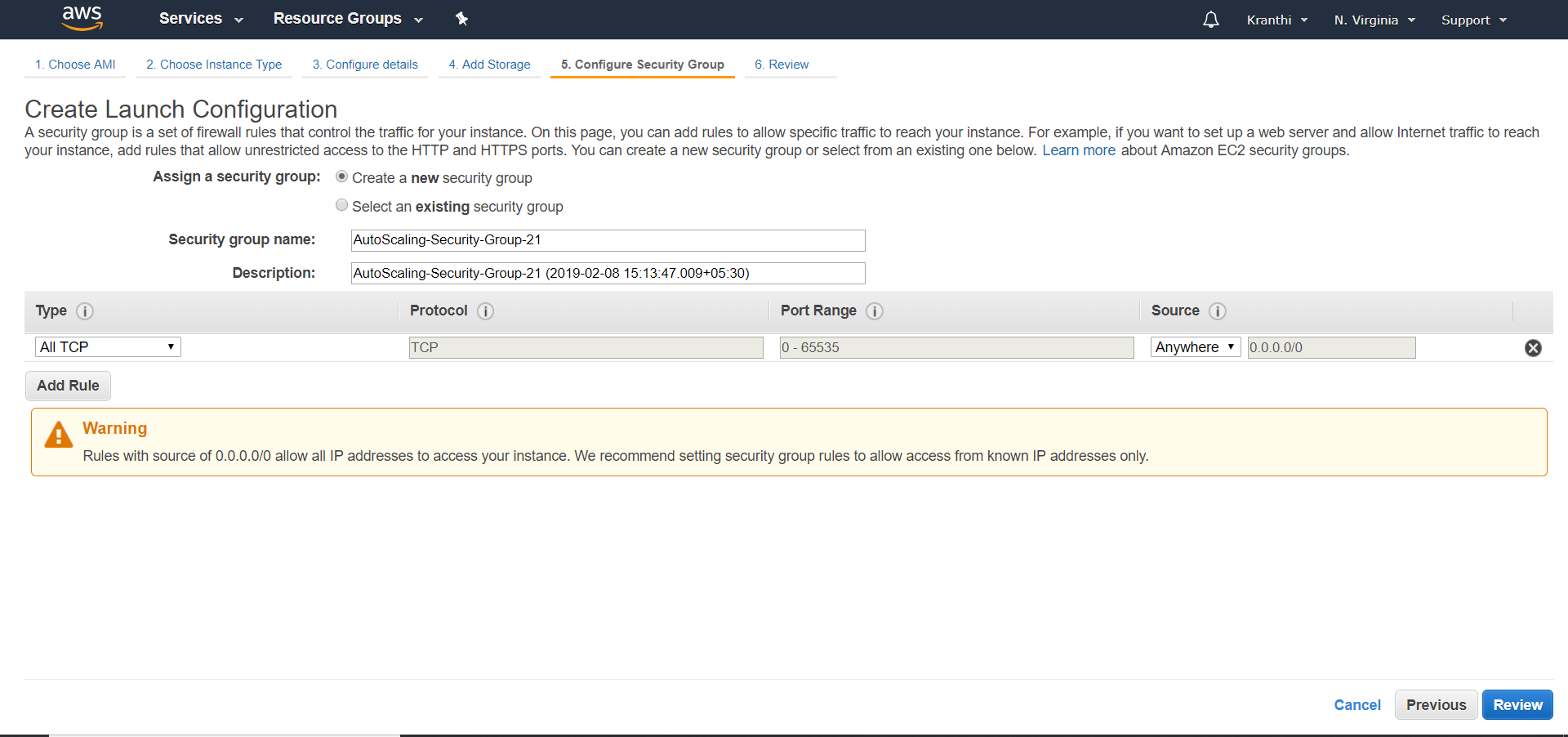
Configure the Launch configuration, set the name to display.

****

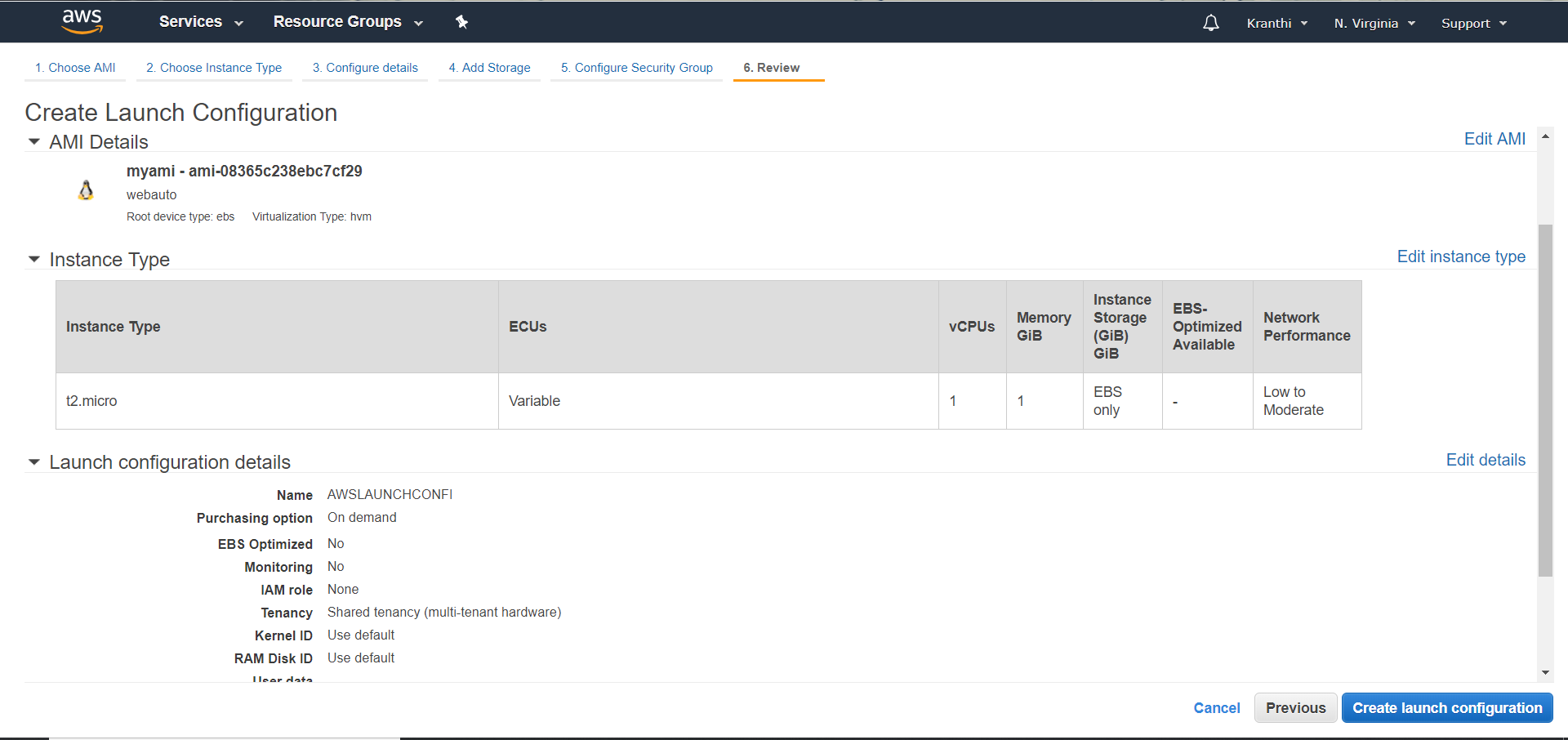
Storage settings to your instance.

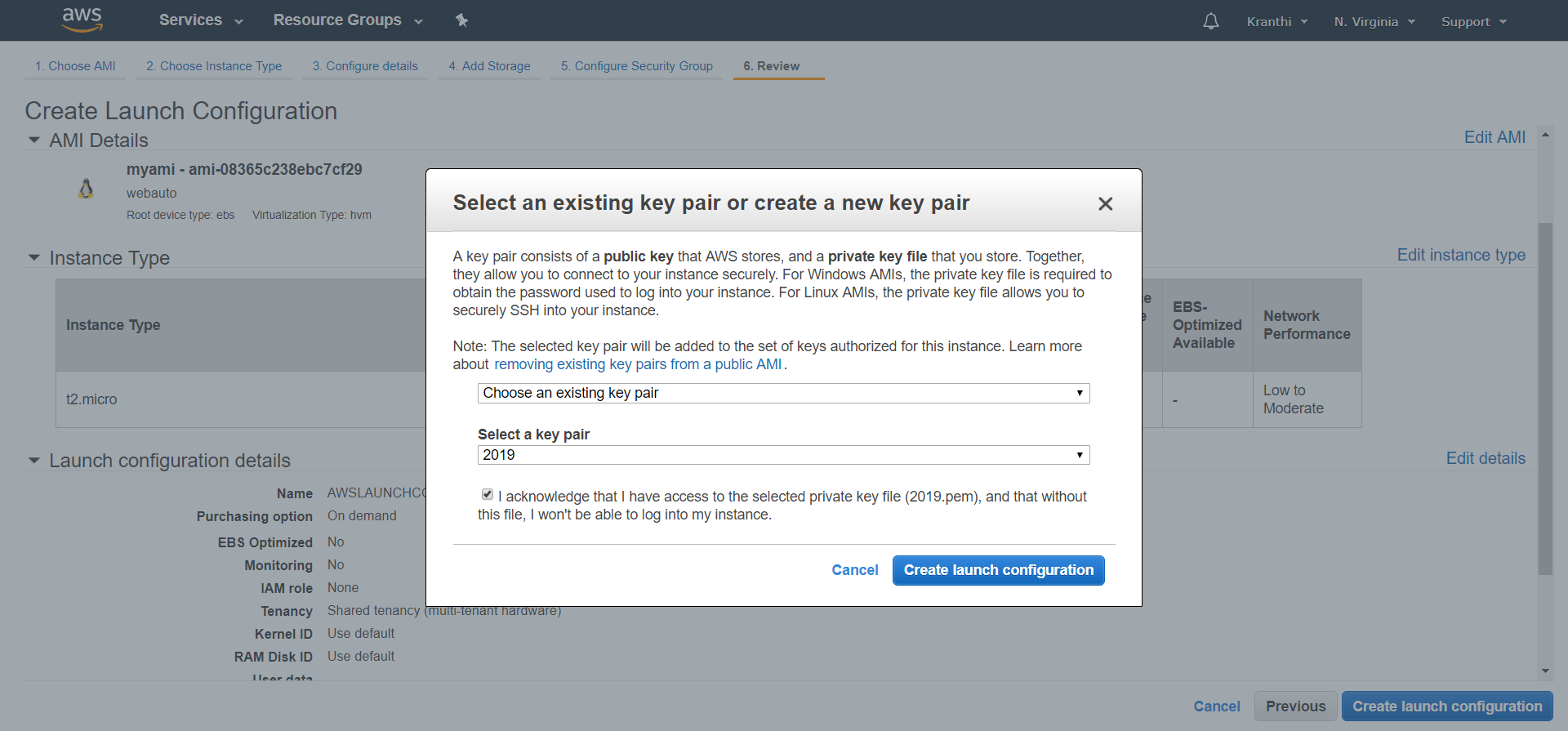
****

Configure security group and add own rules.

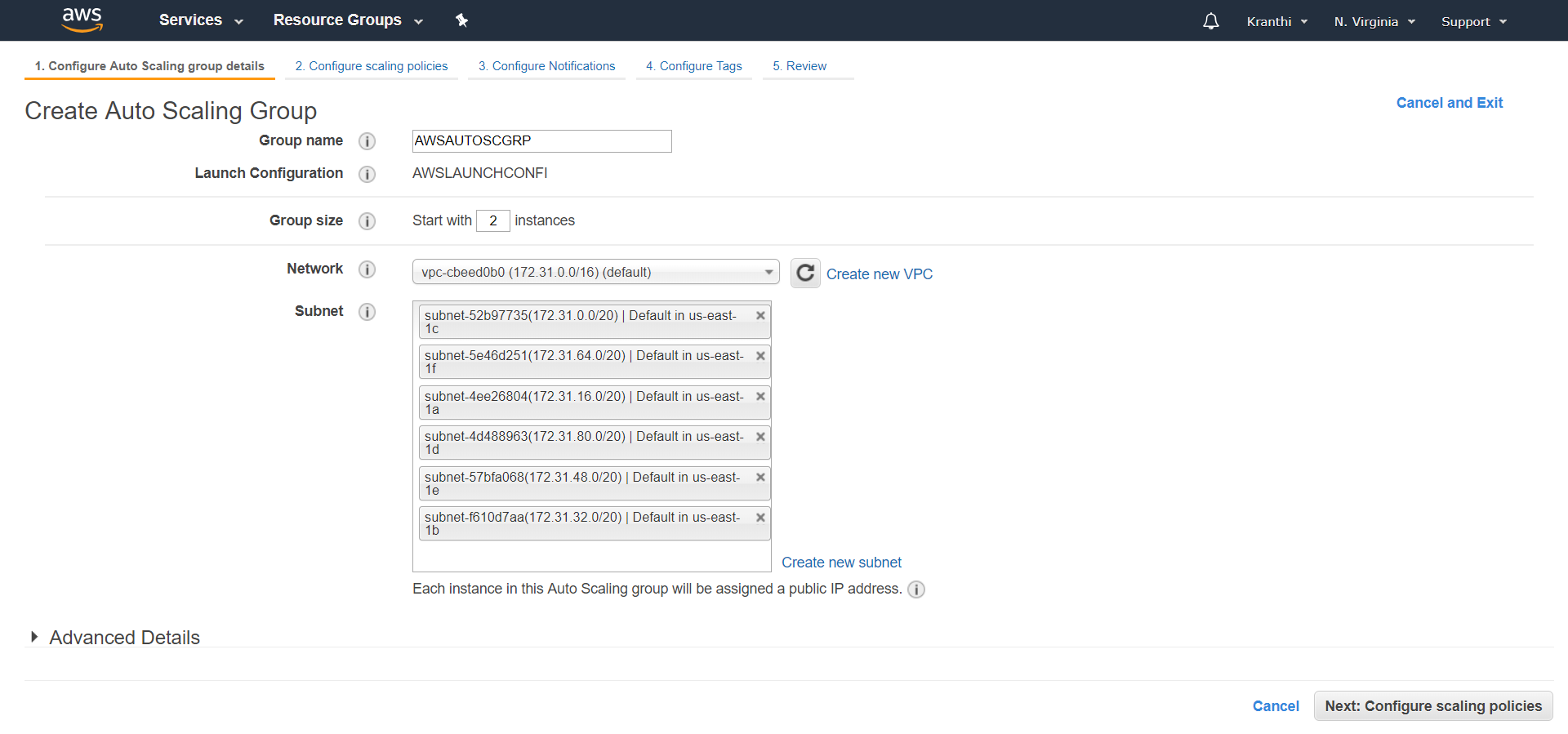
****

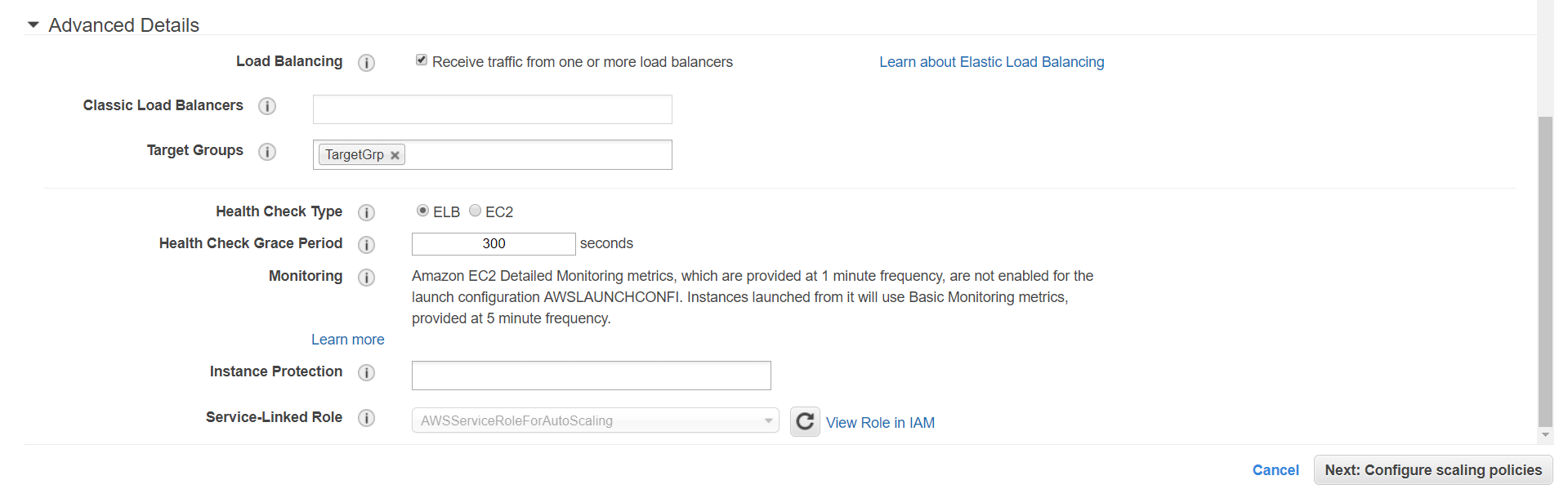
Review gives the total information about AMI.

****

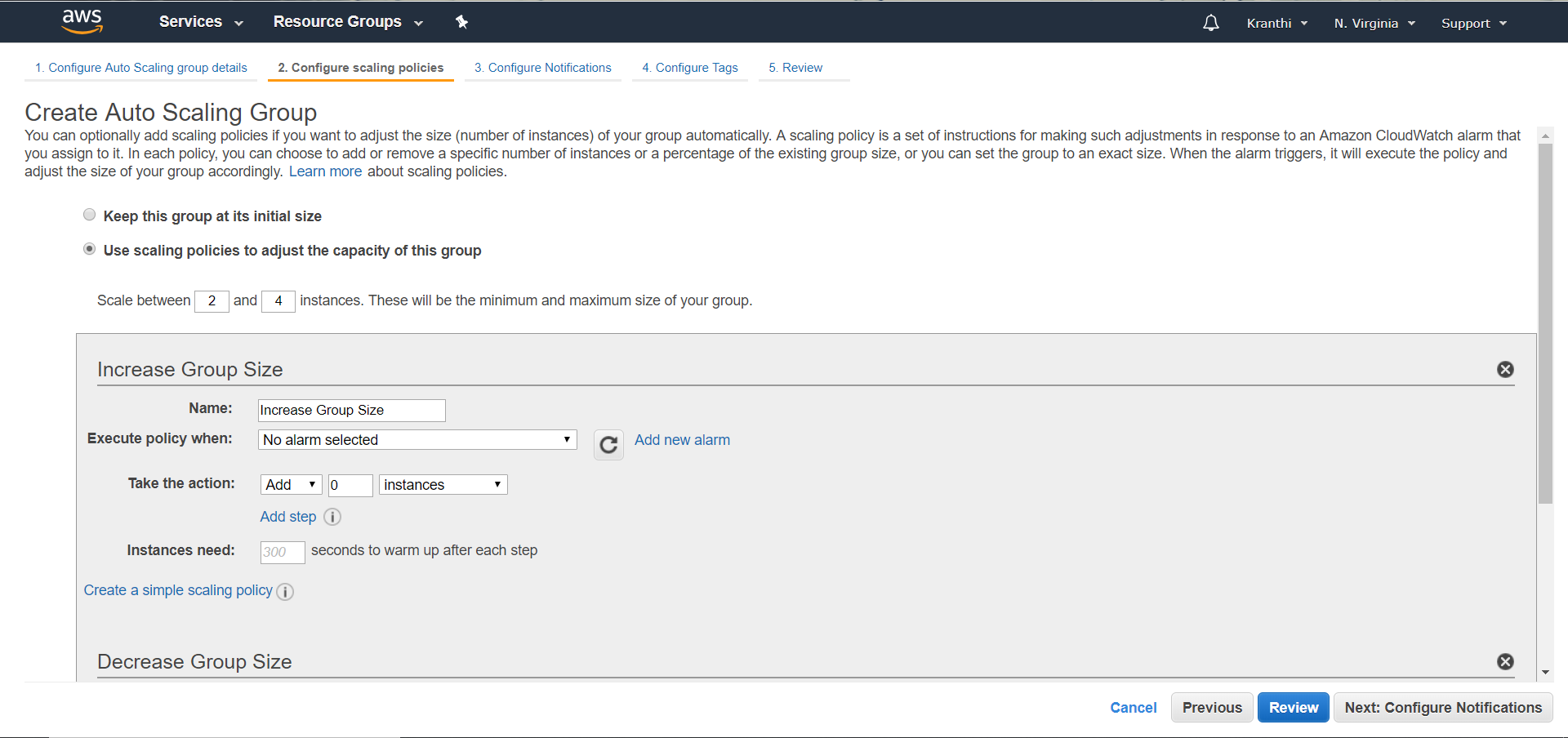
****

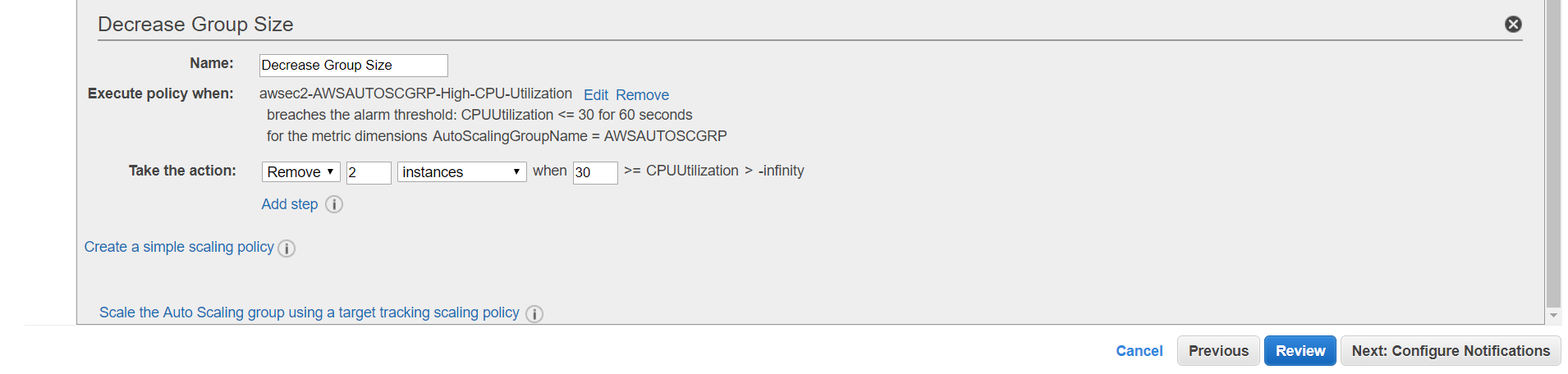
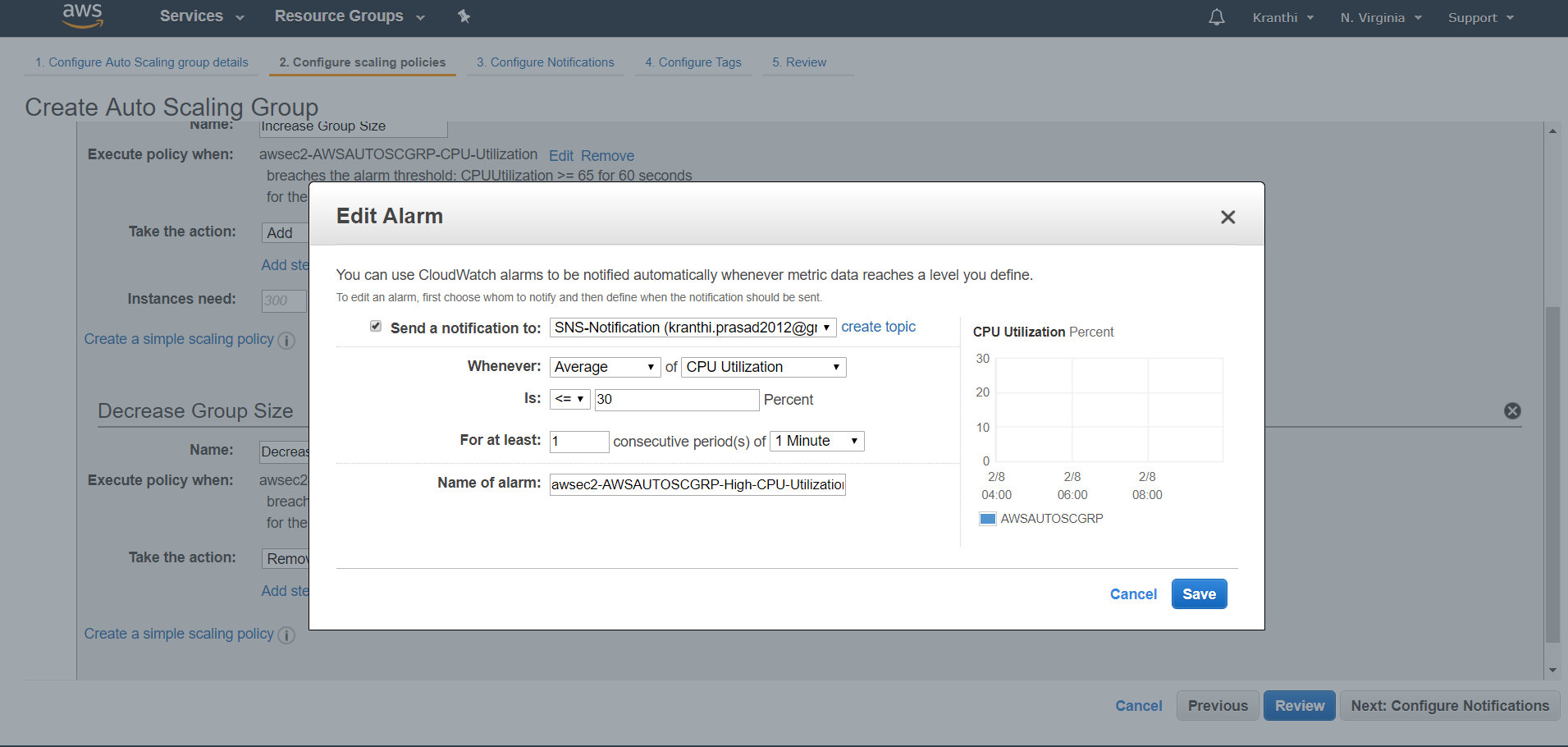
**9.3 Create Auto-Scaling groups after the launch configurations.**

****

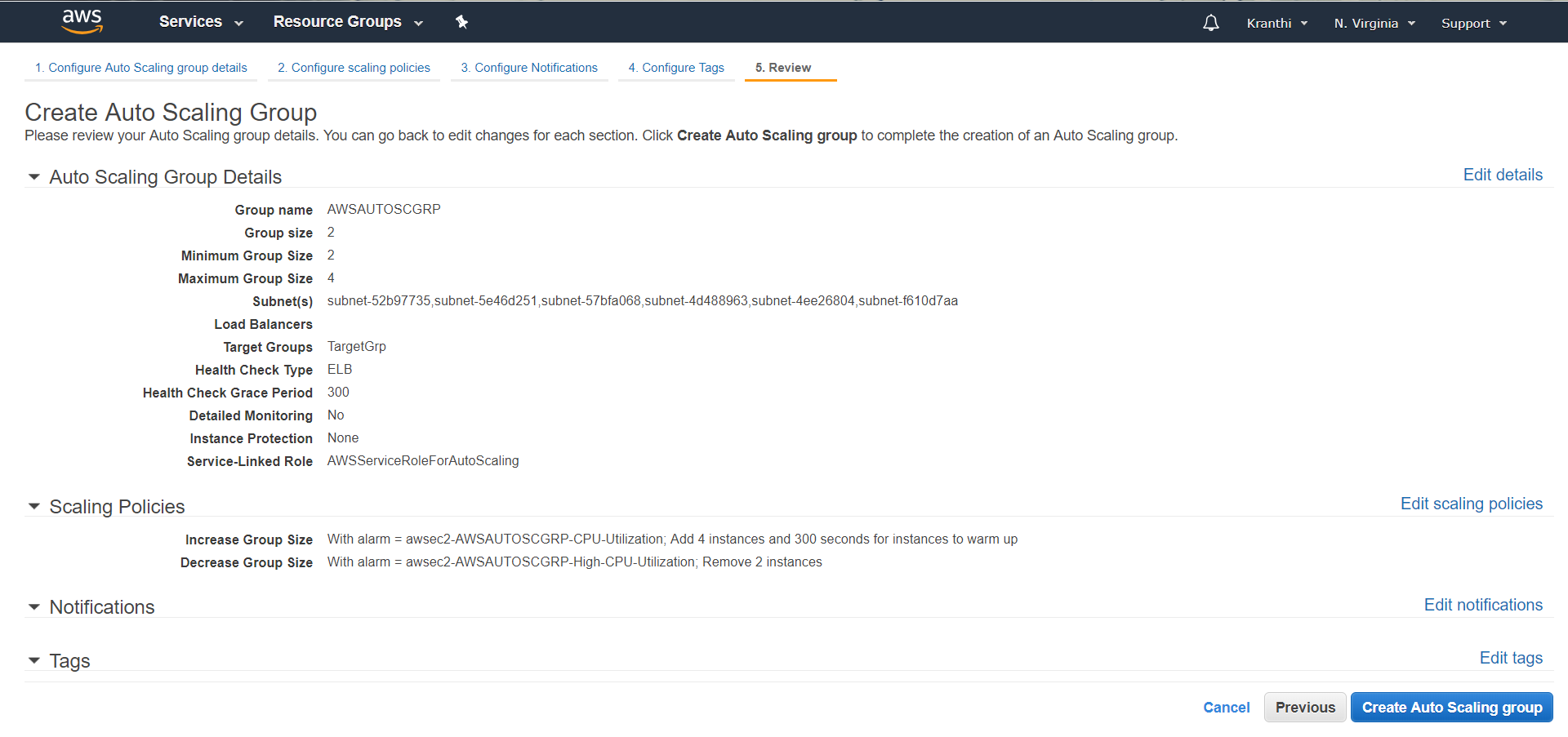
****

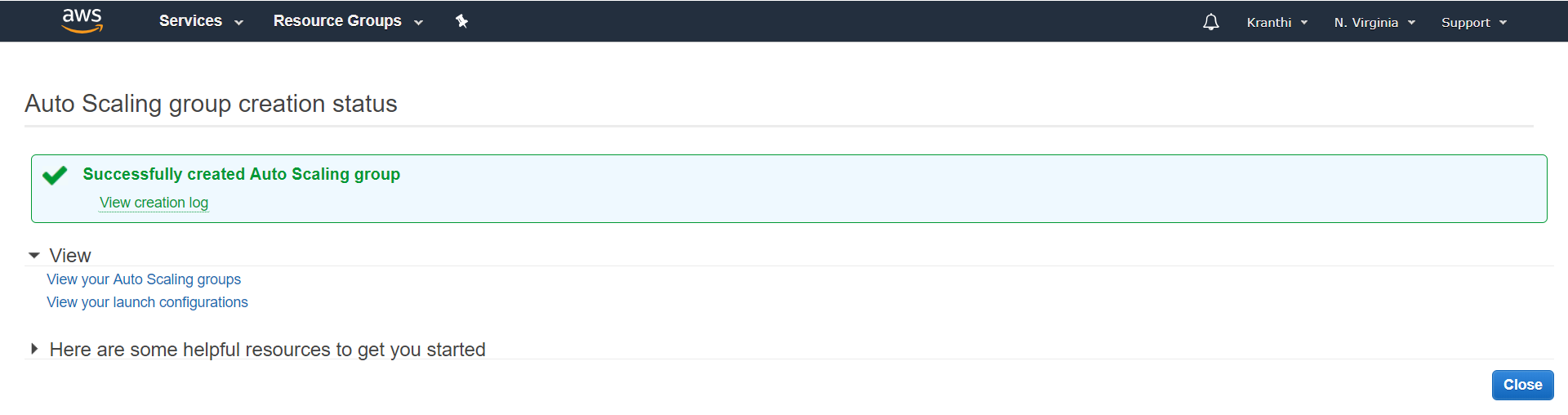
In this Auto-Scaling group, we need write how many instances we need to add when the avg-C.P.U percentages goes high or decrease.

****

****

Reviews shows whatever we have added to the auto scaling group.

****

****

The complete adding of auto-scaling was finished and shows the added one.