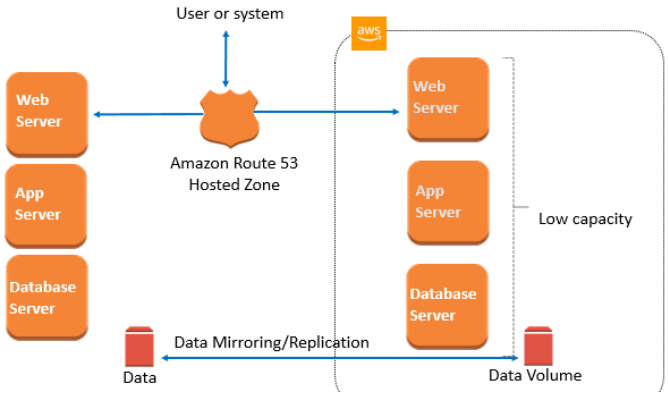
Fully Working Low-Capacity Standby Architecture

In this lab, we are going to design a Disaster Recovery architecture. AWS has recommended four different types of architectures for Reliability which includes Backup & Restore, Pilot Light, Fully Working Low-Capacity Standby and Multi-Site Active-Active. In this lab, we are going to design a **Fully Working Low-Capacity Standby** architecture for Disaster Recovery.

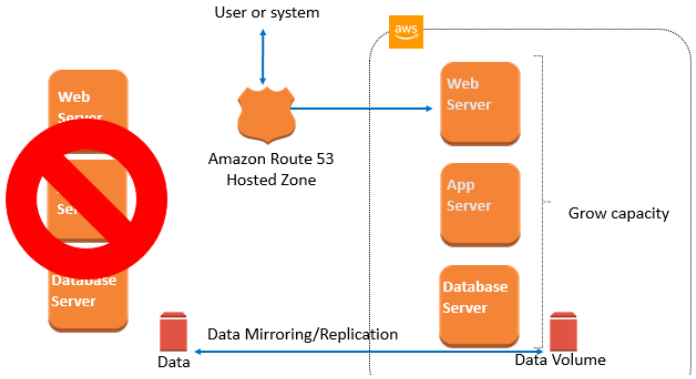
With reference to the below image, let's consider that you have an on-premise Infrastructure which includes Web Server, Application Server and Database Server. You've a similar environment on AWS but all the resources are running at low capacity. Also, there is a synchronous Data replication or mirroring is happening for the Databases from the On-premise Infrastructure to AWS EBS Volume. Lastly, the Route53 is primarily pointing towards the On-premise Infrastructure with Secondary Failover policy to the AWS environment.

Can we achieve Load Balancing in this architecture?

Fully Working Low-Capacity Standby Architecture is designed for Reliability & High Availability. Since, the resources running on the AWS environment are not scale at the Production level rather they are running at the Low Capacity, hence it is not possible to achieve Load Balancing in this case.



With reference to the below image, if your entire Infrastructure shuts down due to Power Supply failures or any Catastrophic events, then we let the AWS resources to run at the Production level. It means we are scaling out our AWS environment. As the Route53 has the secondary Failover Policy, it will now point to the AWS environment. Finally, due to the latest backups of the on-premise Databases, your application will be still Up and Running with the latest web contents.



Task 1: Inspect your AWS Environment

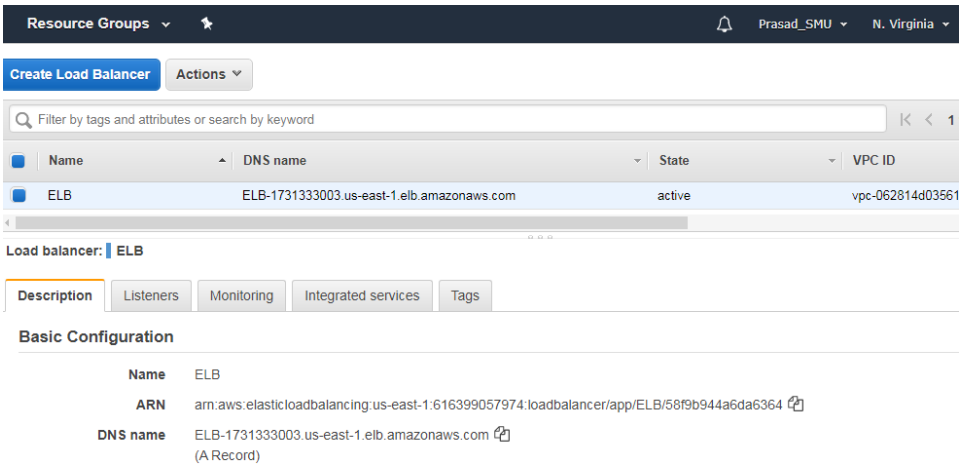
Since in the Lab 2-High Availability Across Availability Zone, we configured a Highly Available architecture across multiple Availability Zones. The entire architecture was hosted in US East (N. Virginia) (us-east-1) region. In this lab, we are going to consider this environment as the Secondary Environment.

Login to the AWS Management Console.

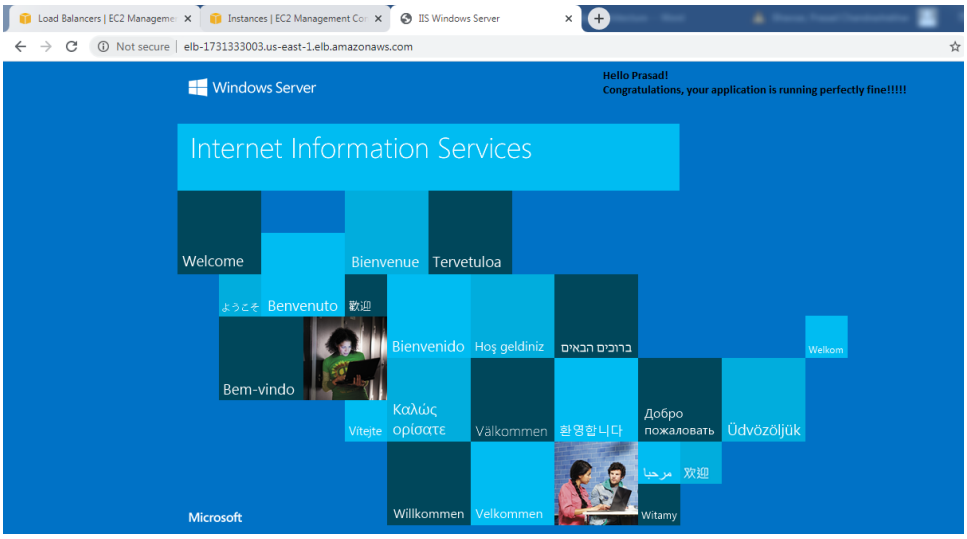
Part 1:

Select the Region as Asia Pacific (Mumbai) (ap-south-1).

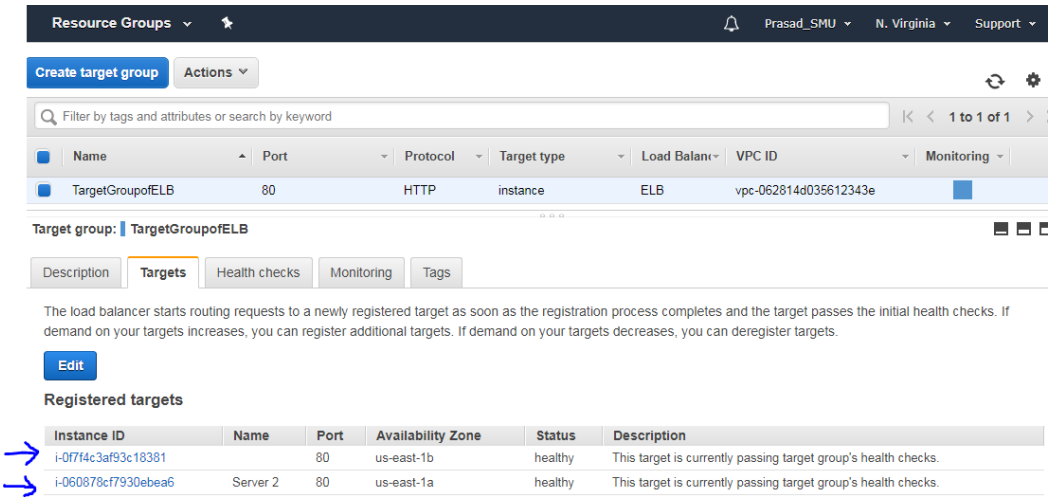
Navigate to the EC2 Service and click on the Load Balancer.



Your Load Balancer should be ACTIVE. Copy the DNS name and paste it in your browser. It should display your application's web page.

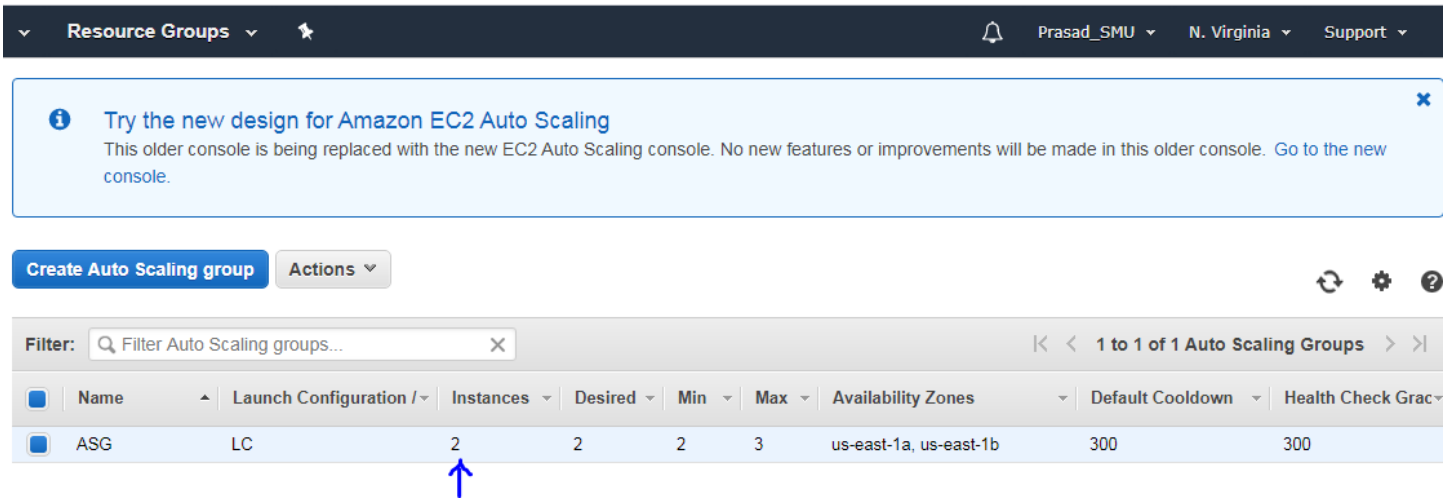


Now click on Target Group, you'll see the two EC2 Instances are running and load balancer is diverting the IIS traffic between these two EC2 Instances.



Now click on Autoscaling Group.

You'll notice that the Autoscaling has the Desired Capacity set to 2, Minimum capacity set to 2 and Maximum capacity set to 2. Minimum and Maximum Capacity is only considered while Scale-In and Scale-Out operations. As the Desired Capacity is set to 2, you noticed that there currently 2 Instances are running in Target Groups under the Load Balancer.



Keep in mind that this regional architecture (N. Virginia) is our Secondary Architecture.

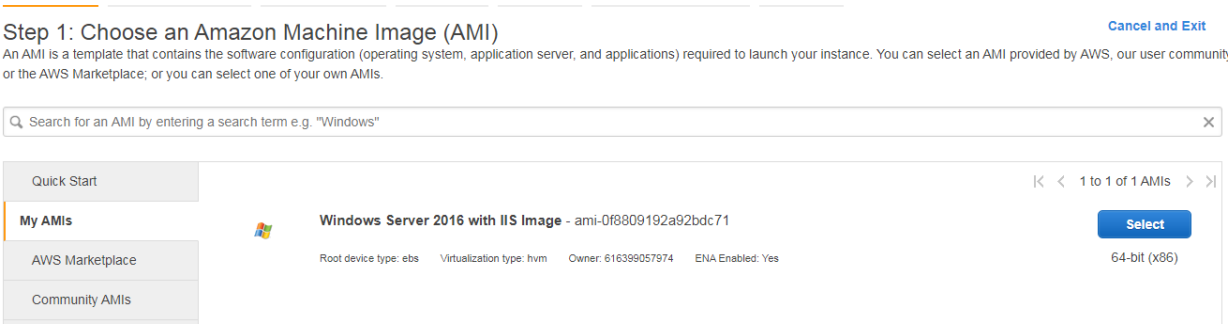
Part 2:

Change Region to Asia Pacific (Mumbai) (ap-south-1). This is going to be our PRIMARY REGION.

Navigate to EC2 Service.

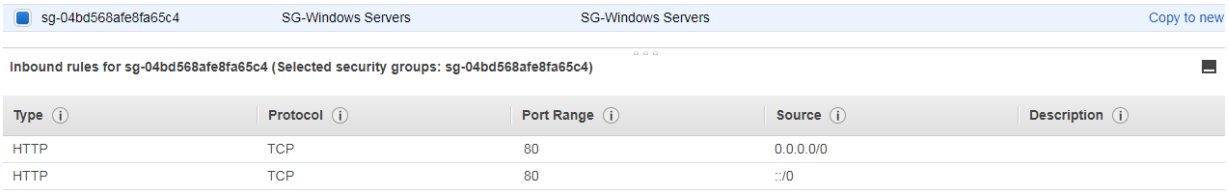
Since in the Lab 2, we've copied a template for the Windows Server 2016 IIS Server to Mumbai region. Using this template, we'll launch a new EC2 Instance now.

Click on Launch Instance, then click on My AMIs and select the below AMI.

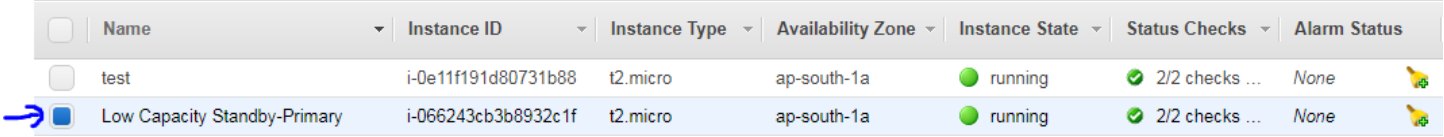


Make sure to select the Network as Custom VPC & Subnet as Public Subnet and Select the Auto-assign Public IP as Enable.

Select the Security Group if you've created for the Windows Server 2016 IIS Web Servers. Else click Next.

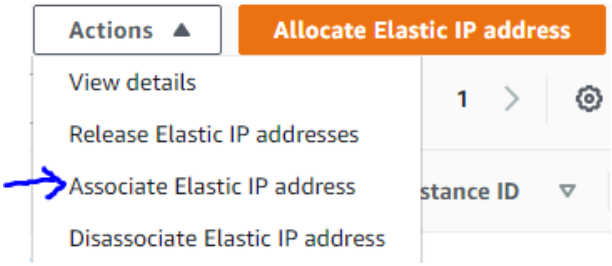


Choose an existing Key Pair and Click on launch EC2 Instance. Name the Instance as per your accordance. Instance has been launched successfully.

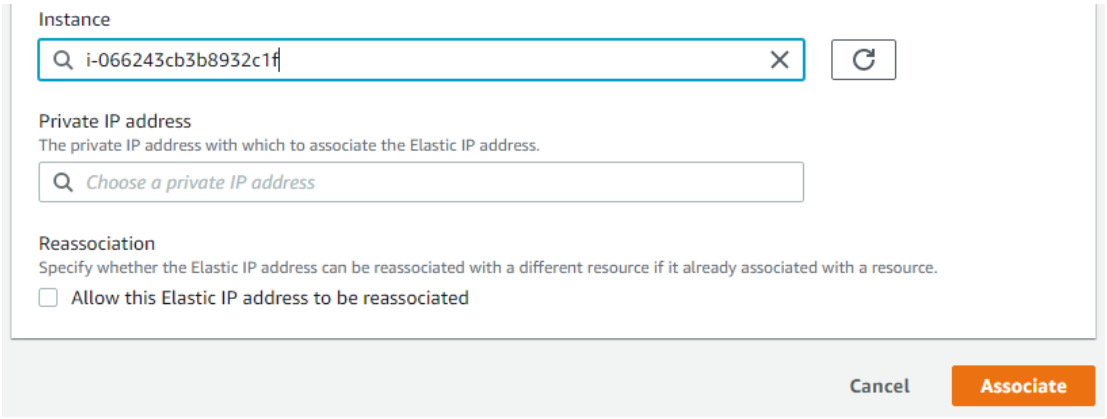


Now click on Elastic IPs (EIP) and click on Allocate Elastic IP Address. Select the Amazon Pool of IPv4 Addresses and click on allocate.

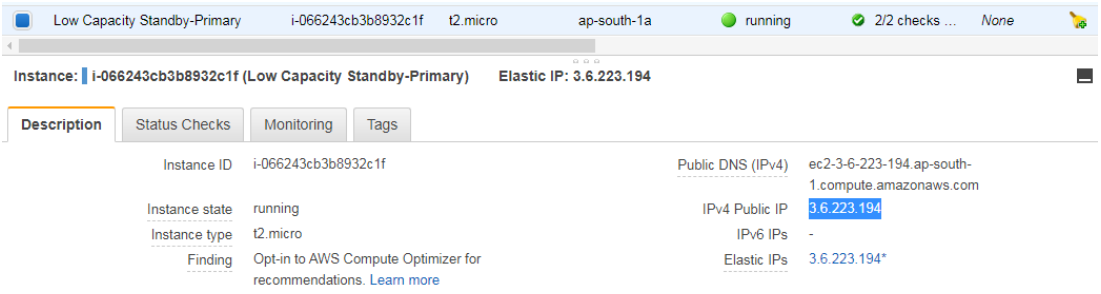
Select the Elastic IP Address and click on Actions and click on Associate Elastic IP Address.



Select the Instance from the drop-down menu to whom you want to allocate this Elastic IP Address and click on Allocate.

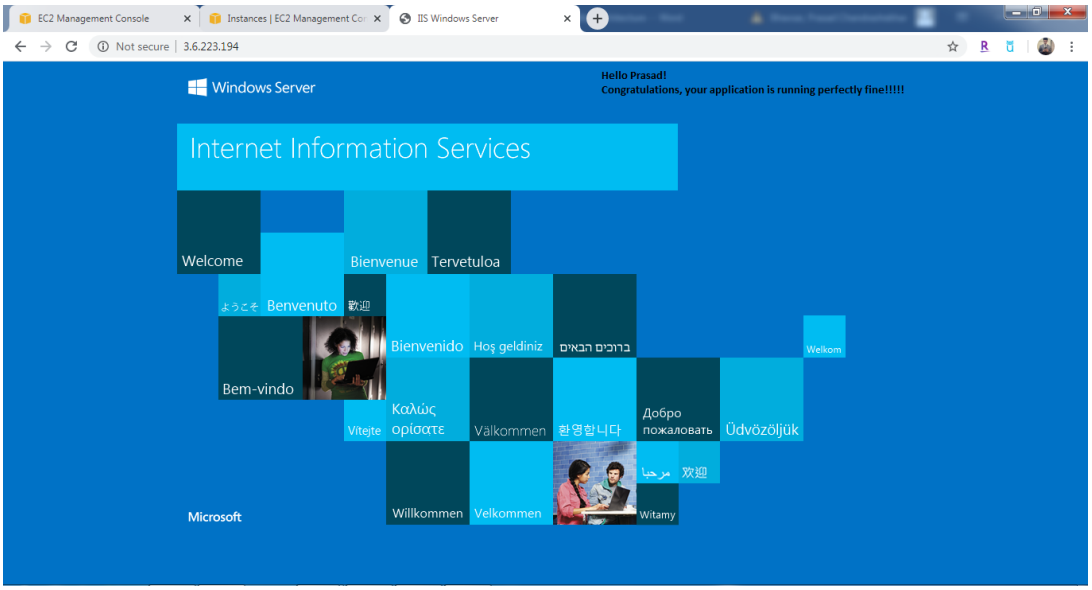


Your Instance is now configured with the Elastic IP Address.



Copy the Elastic IP Address and paste it in your browser.

It should open the Webpage as follows.

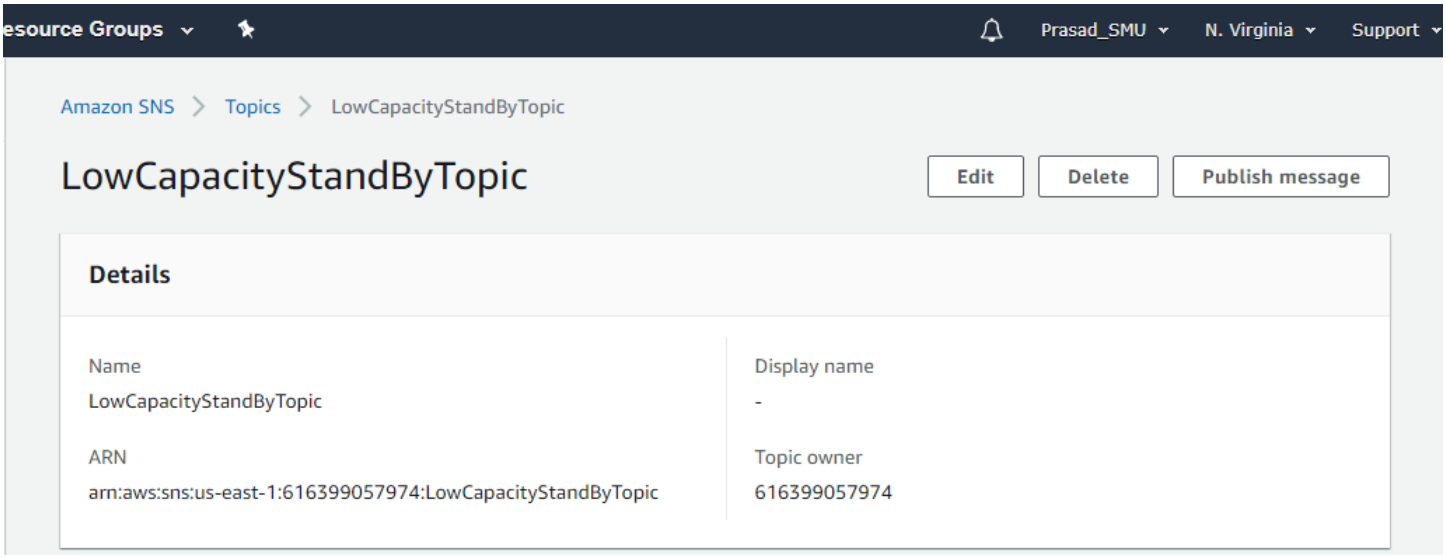


Keep in mind that this regional architecture (Mumbai) is our Primary Architecture.

Task 2: Create a SNS Topic

Come back to US East (N. Virginia) (us-east-1)

Navigate to Simple Notification Service and create a SNS Topic with the name of your choice.



Do not add any subscriptions for now.

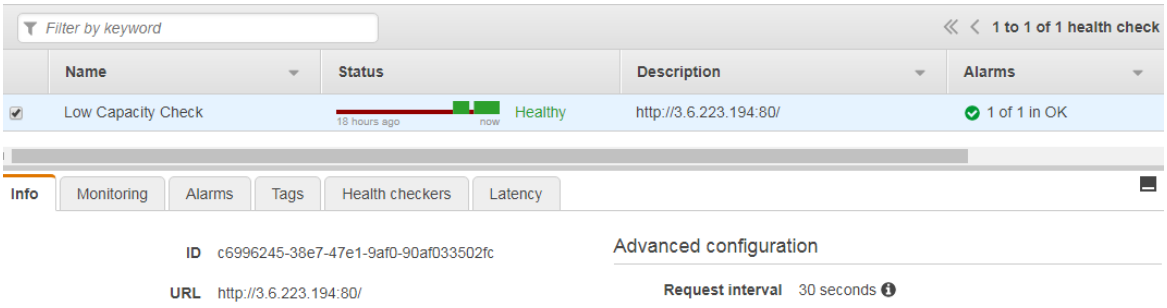
Task 3: Route 53 Configurations

Navigate to the Route 53 Service.

Click on the Hosted Zone that we've created i.e. SMU.EDU

Health Checks:

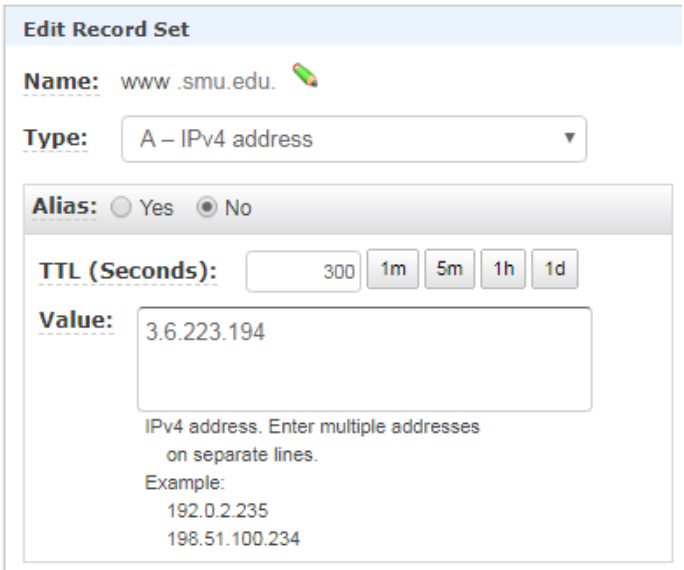
Click on Health Checks and create Health Check for our Primary EC2 Instance in Mumbai region i.e. 3.6.223.194. Make sure that Health Check sends Alarms to the SNS topic.

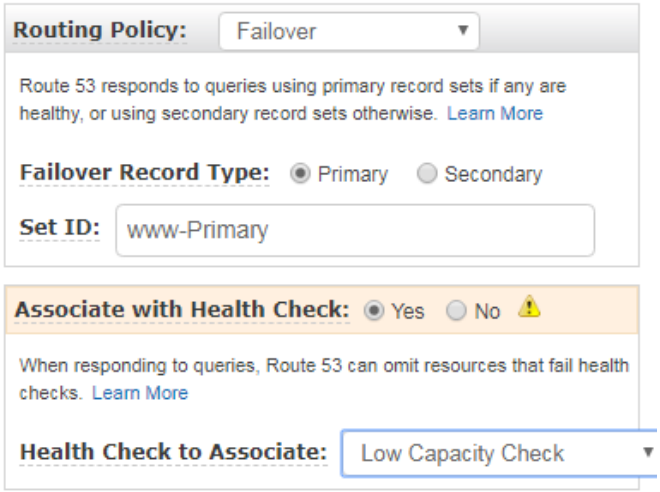


A-Records:

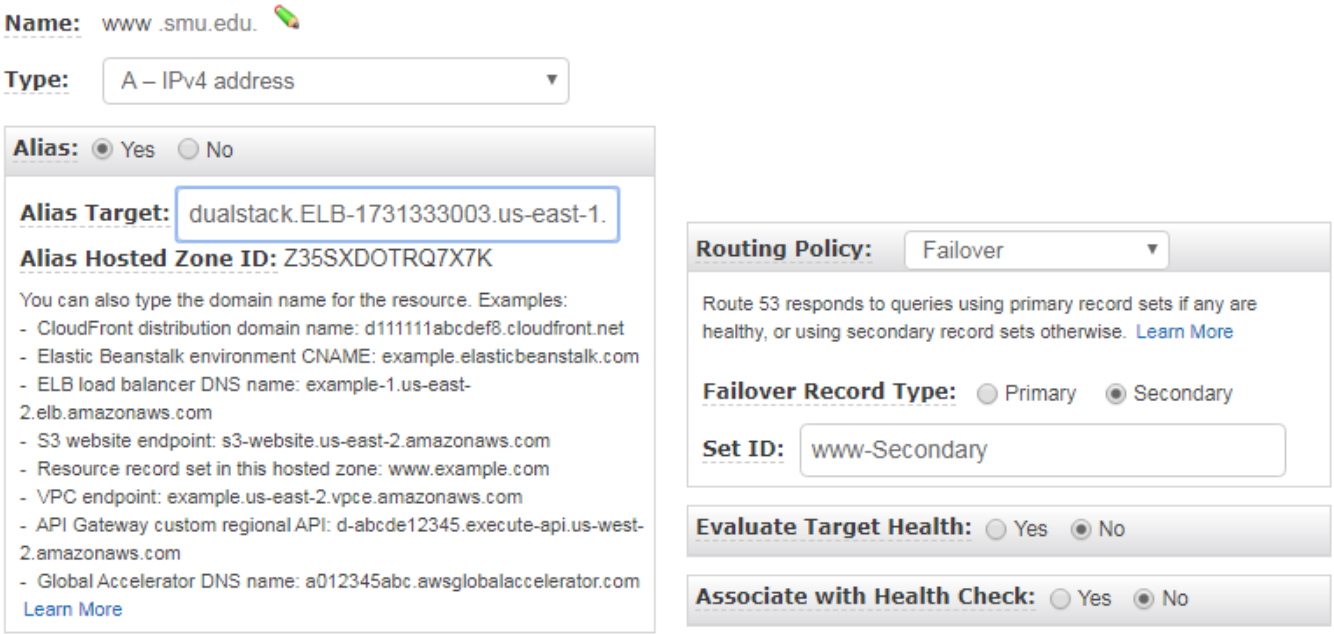
Now configure the Route 53 records with the Failover Policy.

A Record for the Primary Region: Put value as IP Address of the EC2 Instance (3.6.223.194), Select the Routing Policy as Failover-Primary and Select the Health Check which you've configured.





A Record for the Secondary Region: Put the Alias Target as Load Balancer's DNS Name. Select the Routing Policy as Failover-Secondary and no need to select the Health Check here.

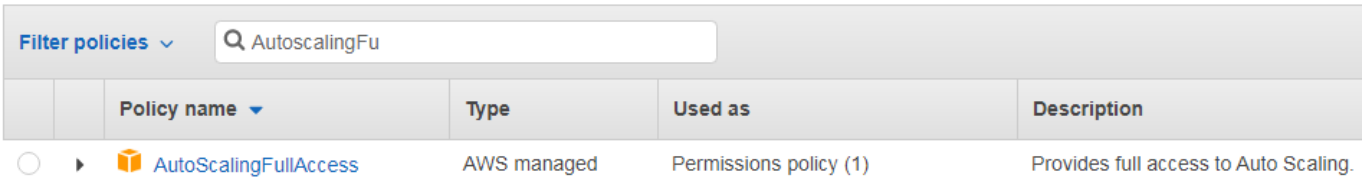


Route 53 Configuration is now completed.

Task 4: IAM Policy and Roles for Lambda Function

IAM Policy:

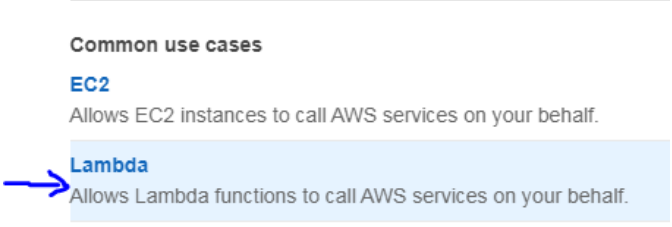
In this lab, we are going to use default IAM Policy which is **AutoScalingFullAccess.**



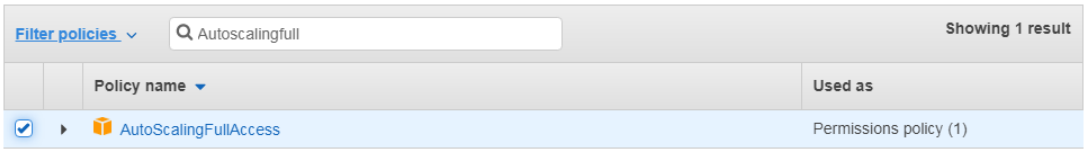
IAM Roles:

On the left-hand side, click on ROLES and click on Create Roles.

Select the Type of Trusted Entity as LAMBDA.

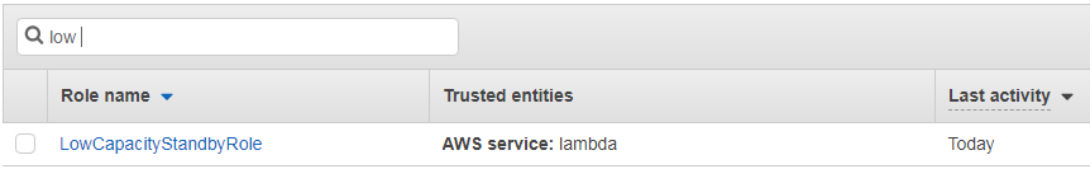


Under Permissions Policies, select the default IAM Policy **AutoScalingFullAccess.**



Give the Role Name as per your choice and click on Create Role.

Role has been created Successfully.

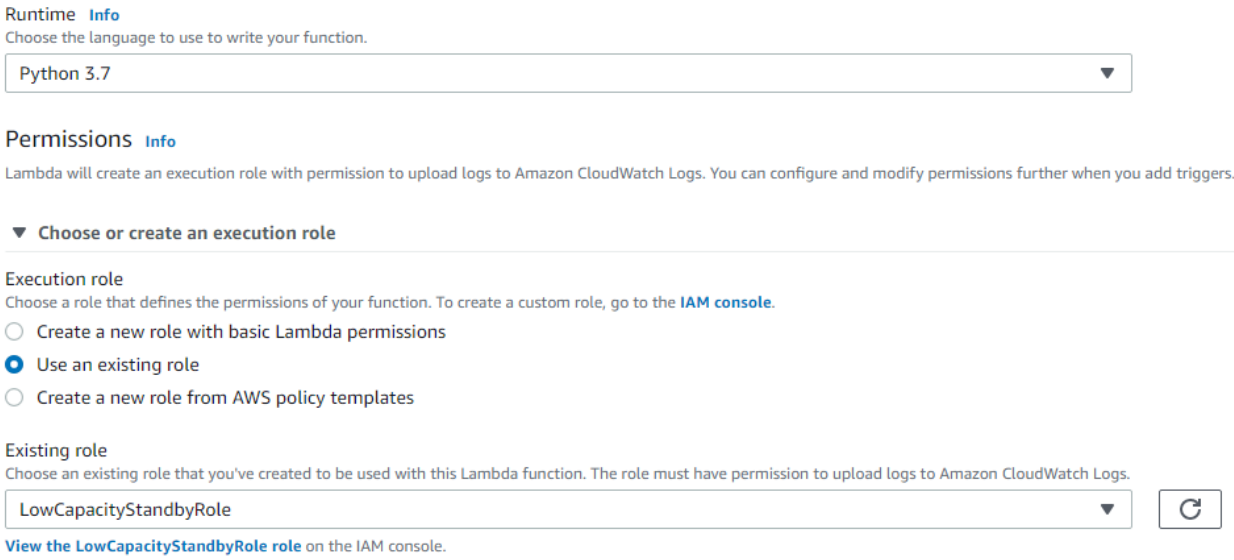


Task 5: Configure a Lambda Function

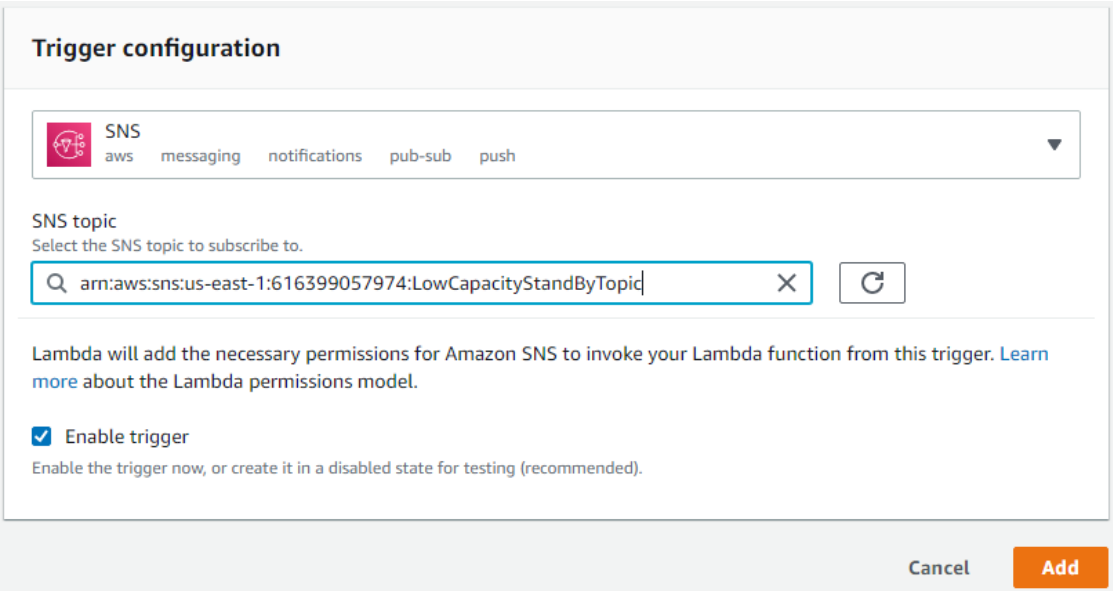
Make sure you're in US East (N. Virginia) (us-east-1) region.

Navigate to Lambda Service and click on Create Function.

Give the Function Name of your Choice, select Runtime as Python 3.7 and select the existing Role which you've created and click on Create Function.

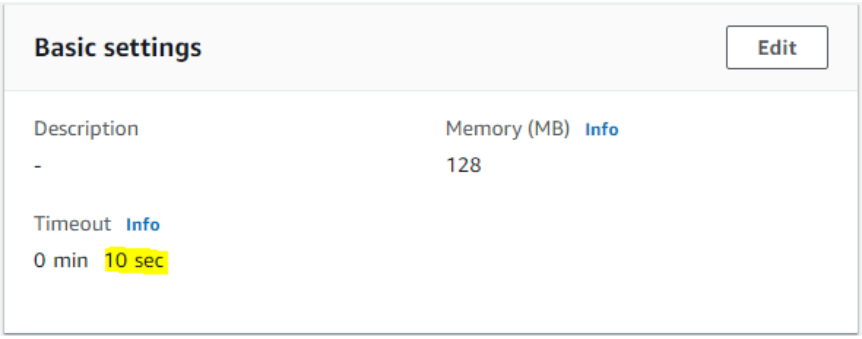


Once the Lambda Function is Created, op top left corner, click on Add Trigger. Select the SNS Service and select the Topic which we've created in Task 2 and click on Add.

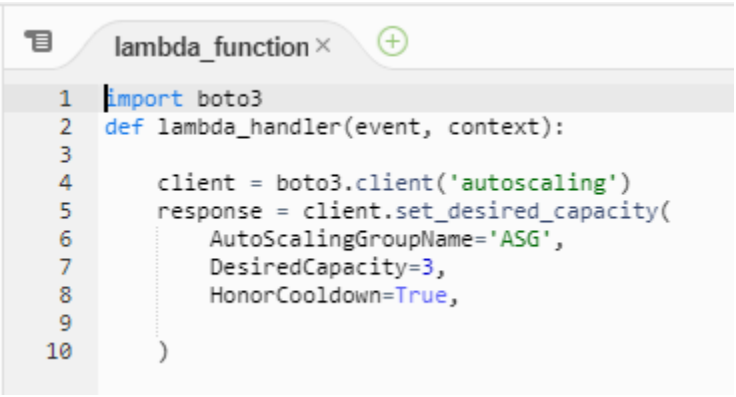


This adds Lambda as a Subscriber to the SNS Topic.

Now under basic settings, click on Edit and set the Timeout Intervals as 10 Seconds.



Now write down the Lambda Function Code as follows:



In line number 6, make sure to replace the **AutoScalingGroupName**.

This Lambda Function sets the Autoscaling Group's Desired capacity to 3.

Finally, click on SAVE.

Lambda function is now configured properly.

Function name	Description	Runtime	Code size	Last modified
LowCapacityStandBy		Python 3.7	297 bytes	18 hours ago

Task 6: Setup to test the Fully Working Low-Capacity Standby Scenario

Select US East (N. Virginia) (US-EAST-1) region.

Navigate to EC2 Service and click on Autoscaling Group.

Make a note of Desired Capacity which is currently 2.

Filter: Filter Auto Scaling groups... 1 to 1 of 1 Auto Scaling Groups

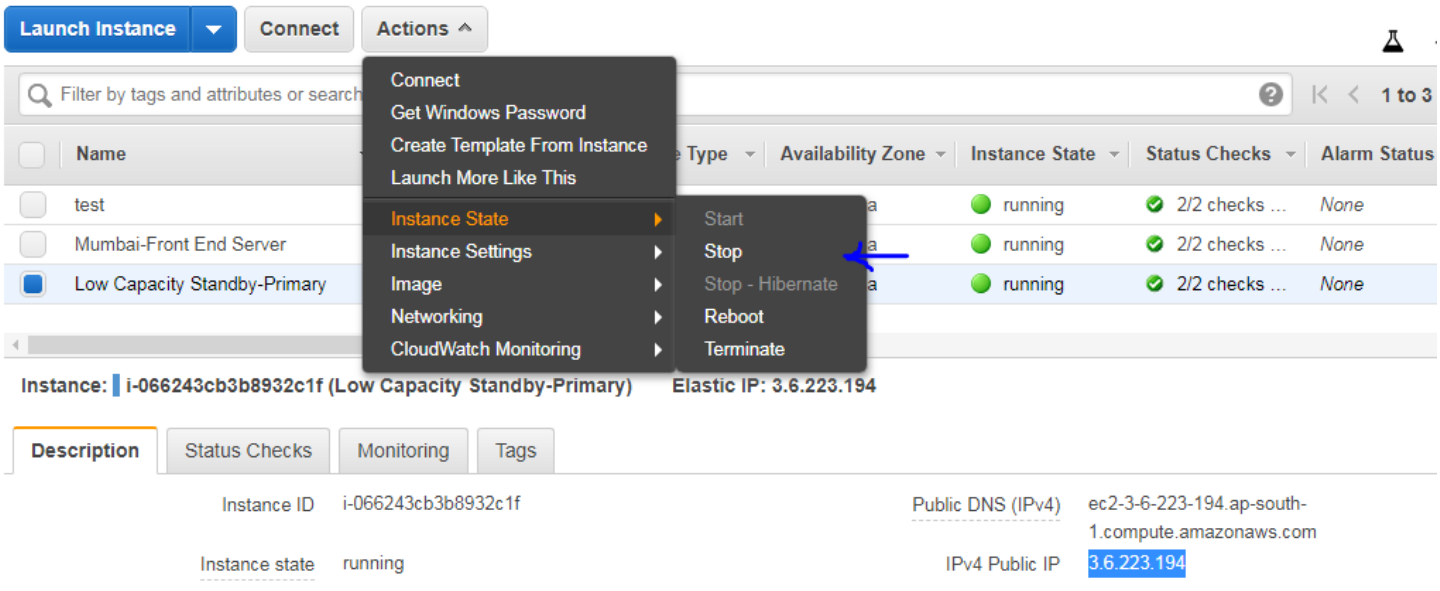
Name	Launch Configuration /	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Check
ASG	LC	2	2	2	3	us-east-1a, us-east-1b	300	300

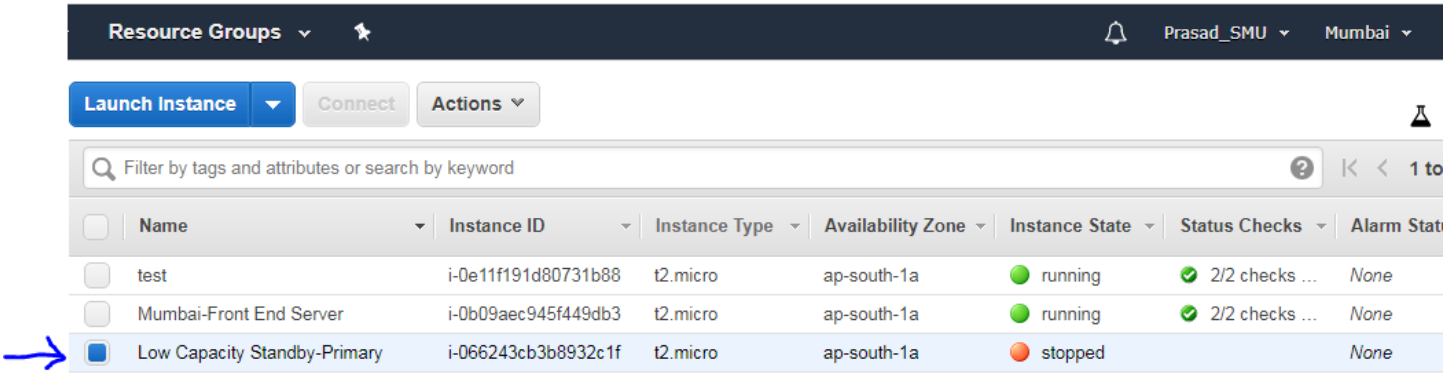
Task 7: Test the Fully Working Low-Capacity Standby Architecture

Select the Asia Pacific (Mumbai) (ap-south-1) region.

Navigate to the EC2 Service, click on our Primary EC2 Instance.

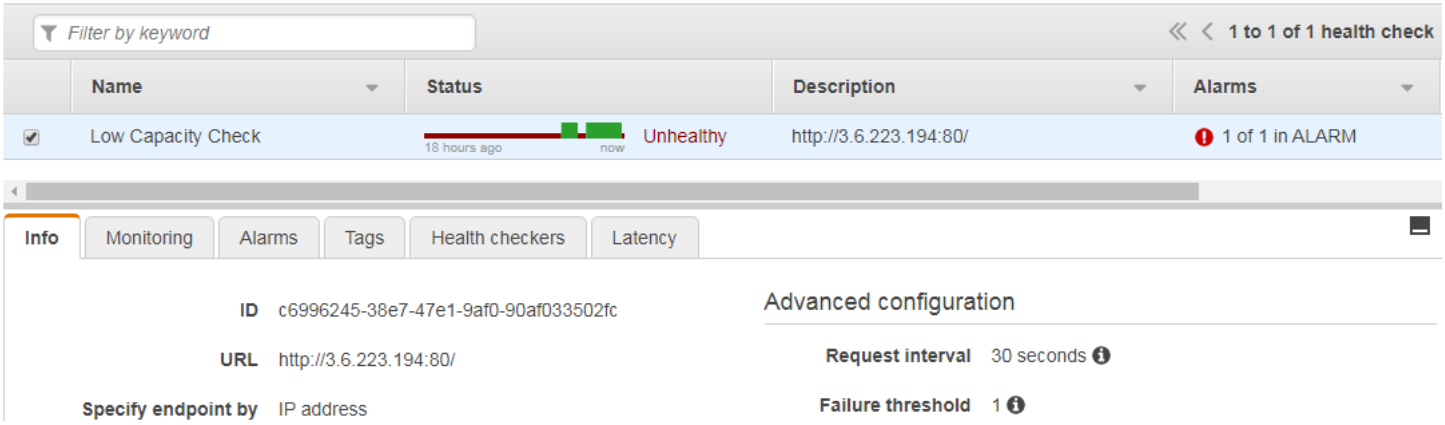
Click on Actions > Instance State > STOP



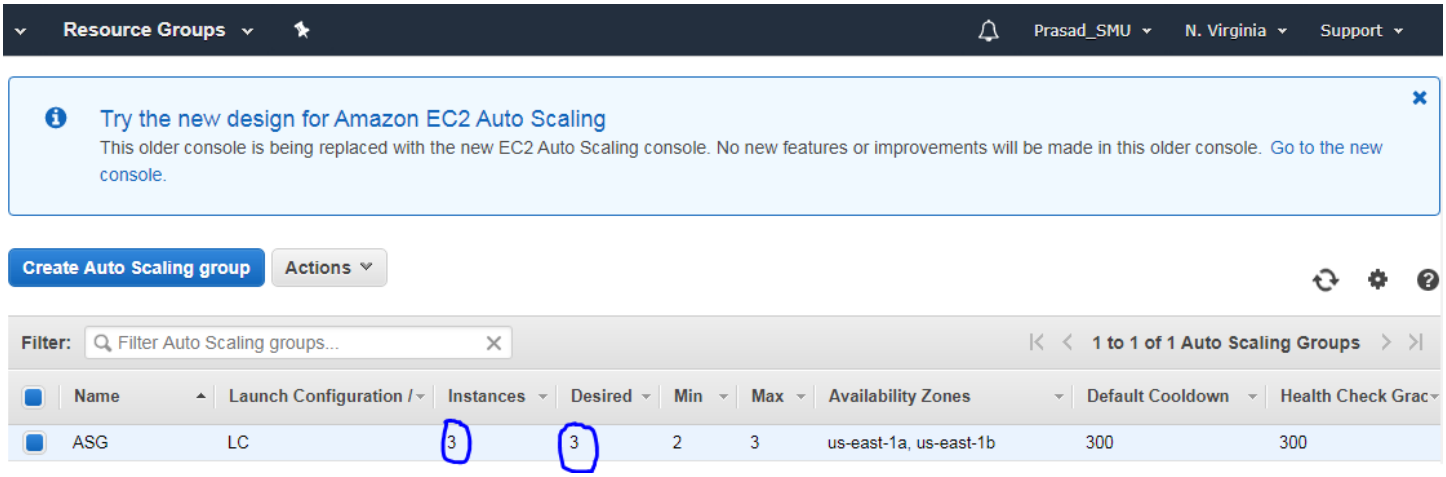


Let's look at the Route53 Health Check now.

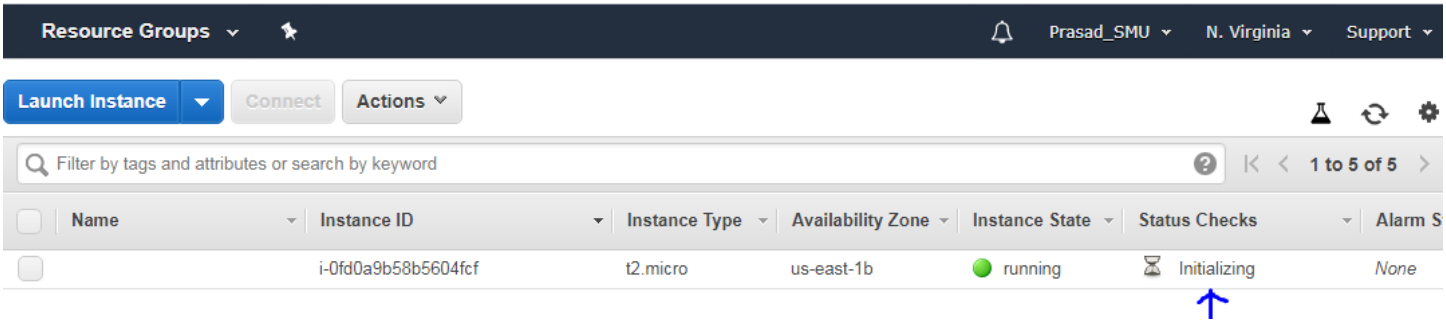
Health Check shows as Unhealthy, this would have triggered our SNS Topic.



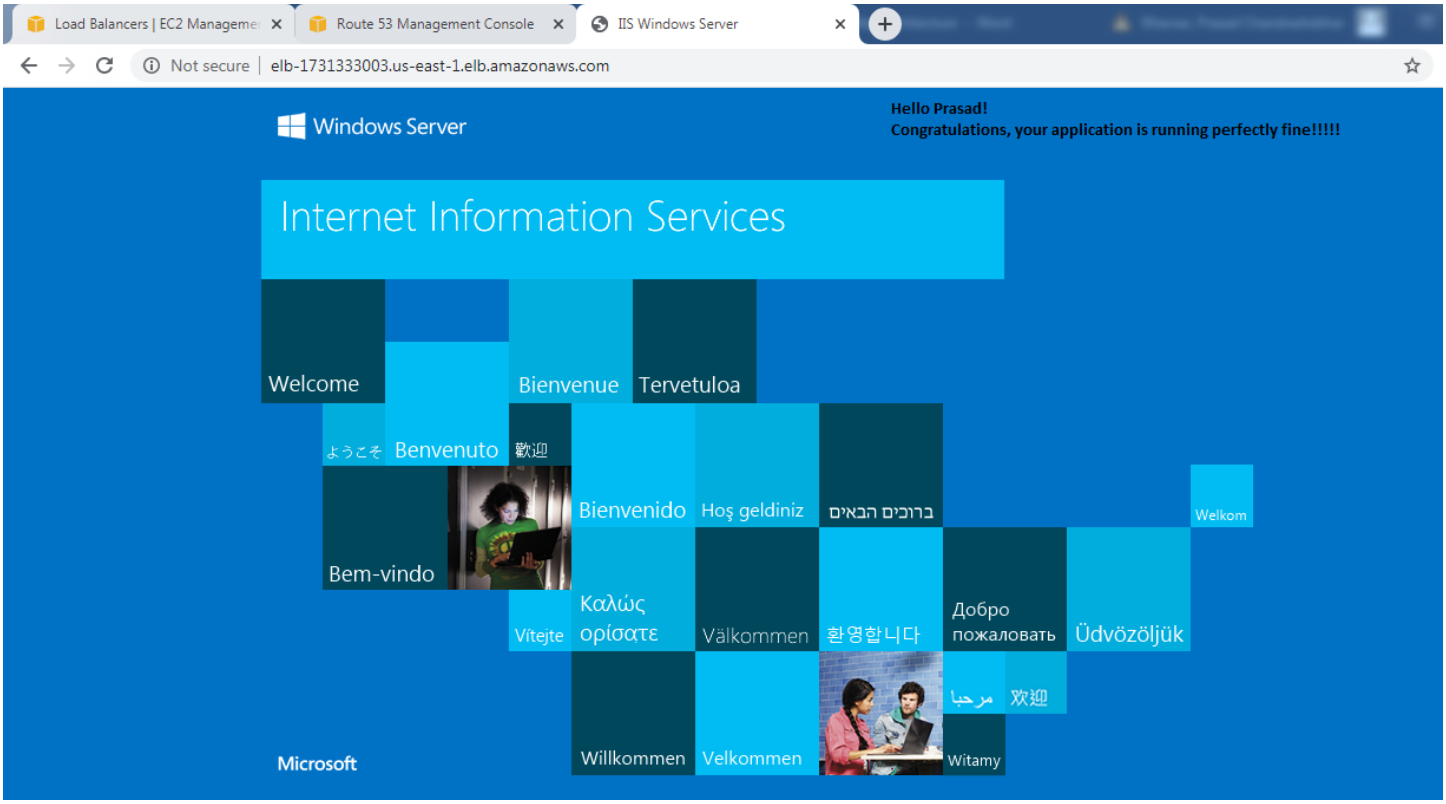
SNS Topic then would have triggered the Lambda Function and Lambda Function would have changed the Desired Capacity of Autoscaling Group to 3 in N. Virginia region.



You can also notice that a new Instance got launched in the N. Virginia region to main the desired capacity of 3.



Simultaneously, the Route 53 Failover Policy has now detected the Failover and routed the traffic towards the Load Balancer in N. Virginia Region.



This proves the Fully Working Low-Capacity Standby Architecture.

For questions, contact me on pbhavsar@smu.edu .