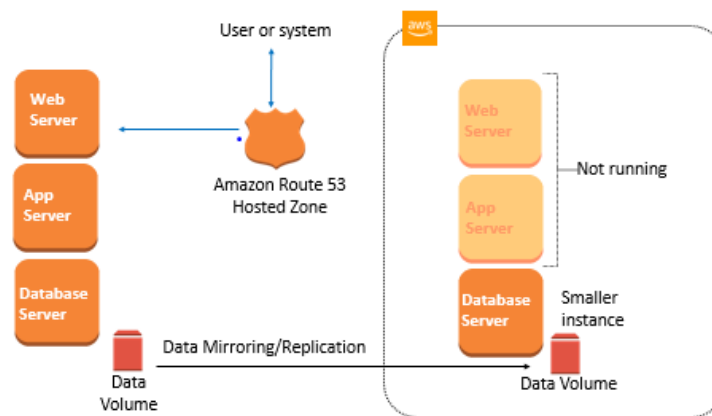


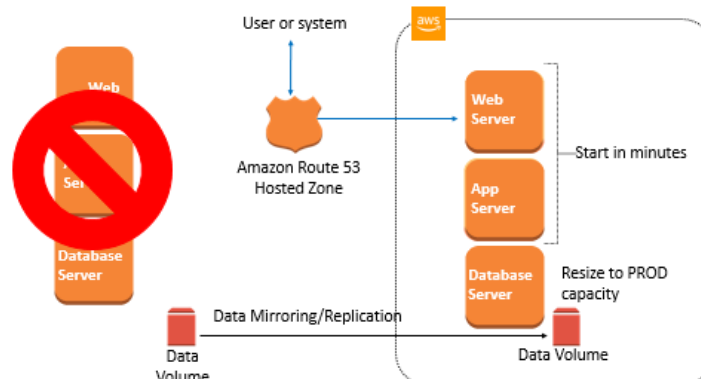
Pilot Light Architecture

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 **Pilot Light** architecture for Disaster Recovery.

With reference to below image, let's consider that you've an on-premise Infrastructure which includes Web Server, App Server and Database Server. Consider that you've replicated the identical environment on AWS but it's not currently Powered ON except for the regular backup of Databases from on-prem Infrastructure to a DB Instance in AWS which is not running at the Production scale. Route 53 is currently pointing towards the on-premise Infrastructure.



Now with reference to below image, let's consider that the entire on-premise Infrastructure shuts down due to Power Supply issues or any catastrophic events. Now the Web Server and App Server Instances which are running on AWS gets powered ON and the Database Instance grows to the Production Capacity and Route 53 is now gets pointed to the AWS environment. Hence now your application is still Up and Running with the latest contents even though entire On-Premise Infrastructure is down.



This is called as Pilot Light Architecture for Disaster Recovery.

Below is the list of Tasks:

- Task 1: Launch EC2 Instances in two different Regions and allocate EIPs
- Task 2: Create a SNS Topic
- Task 3: Route 53 Configurations
- Task 4: Create an IAM Policy and Roles for Lambda Function
- Task 5: Configure a Lambda Function
- Task 6: Setup to test the Pilot Light Scenario
- Task 7: Test the Pilot Light Scenario

Task 1: Launch EC2 Instances in two different Regions and allocate EIPs

In this lab, we are going to consider **US East (N. Virginia) (us-east-1)** as the **Primary Region** and **Asia Pacific (Mumbai) (ap-south-1)** as the **Secondary Region**.

Navigate to AWS Management Console, select region as **US East (N. Virginia) (us-east-1)** and navigate to EC2 Service. Since in the Lab 2, we've created a template for the Windows Server 2016 IIS Server. Using that template launch a new EC2 Instance.


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

Step 1: Choose an Amazon Machine Image (AMI) Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows" ✕

Quick Start
My AMIs
AWS Marketplace
Community AMIs


Windows Server 2016 with IIS Image - ami-0f8809192a92bdc71
Root device type: ebs Virtualization type: hvm Owner: 616399057974 ENA Enabled: Yes

Select
64-bit (x86)

|< < 1 to 1 of 1 AMIs > >|

We are going to launch the EC2 Instance in our Original Architecture which was launched by CloudFormation. Make sure to select the Network as Custom VPC & Subnet as Public Subnet 1.

Network ⓘ vpc-062814d035612343e | Custom VPC ↕ Create new VPC

Subnet ⓘ subnet-01ee44283bcd09e5c | Public Subnet 1 | us-e ↕ Create new subnet
247 IP Addresses available

Auto-assign Public IP ⓘ Use subnet setting (Enable) ↕

Select the Security Group that we've created for the Windows Server 2016 IIS Web Servers.

sg-04bd568afe8fa65c4 SG-Windows Servers SG-Windows Servers Copy to new

Inbound rules for sg-04bd568afe8fa65c4 (Selected security groups: sg-04bd568afe8fa65c4)

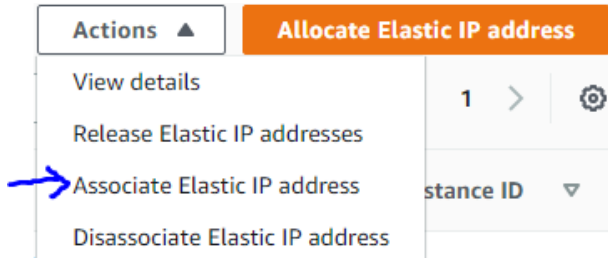
Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	:::/0	

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

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input type="checkbox"/>		i-0f7f4c3af93c18381	t2.micro	us-east-1b	● running	✓ 2/2 checks passed
→ <input checked="" type="checkbox"/>	Pilot Light-N.Virginia	i-0cb4711711c5a18d8	t2.micro	us-east-1a	● running	✓ 2/2 checks passed

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

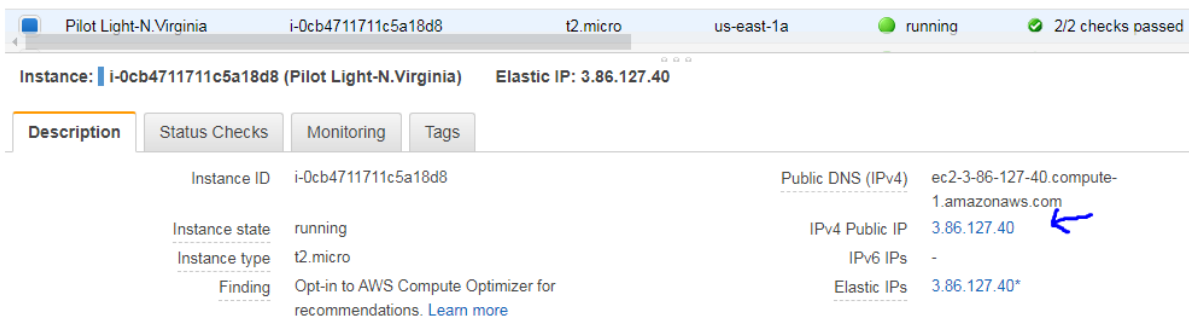
Select the Elastic IP Address and click on Actions and click on Associate Elastic IP Address.



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

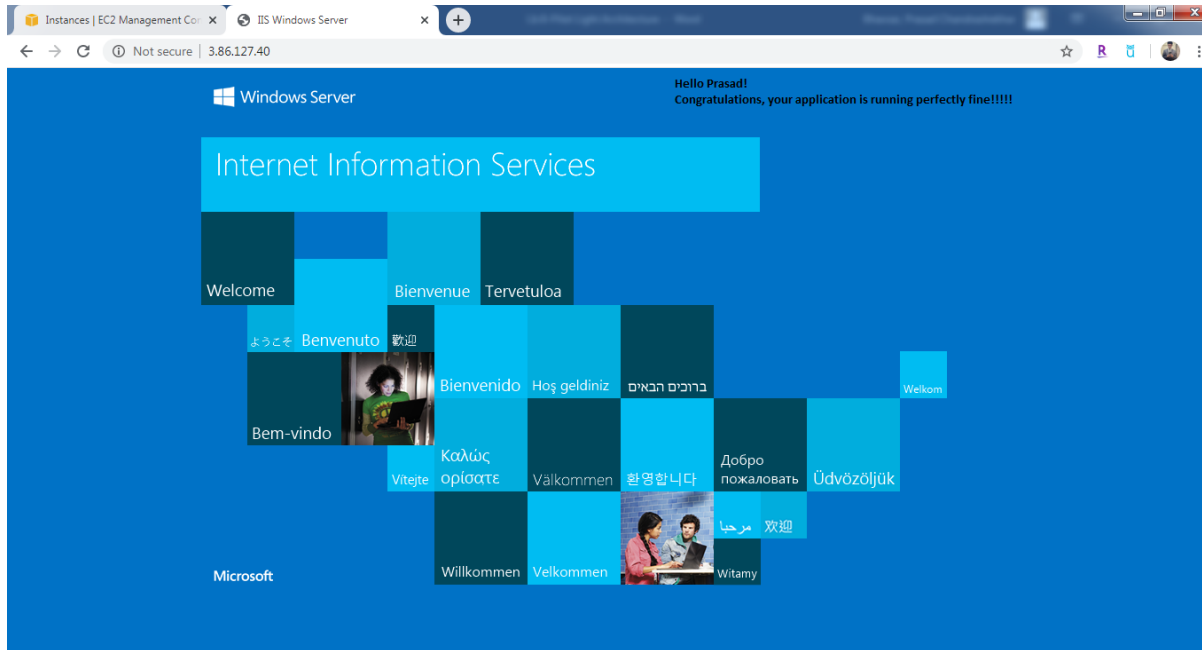
The screenshot shows the 'Associate Elastic IP address' dialog box. The 'Instance' field is populated with 'i-0cb4711711c5a18d8'. The 'Private IP address' field is empty. The 'Reassociation' checkbox is unchecked. The 'Associate' button is highlighted.

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.



Now switch it to the **Asia Pacific (Mumbai) (ap-south-1) Region**, and launch a new EC2 Instance.

Launch this Instance using the same AMI Image which we've copied earlier in Lab 3. Make sure to select the VPC as Custom VPC and Subnet as Public Subnet. Allocate an Elastic IP Address and assign it to the EC2 Instance.

You've now successfully launched an EC2 Instance in **Asia Pacific (Mumbai) (ap-south-1) Region**.

→

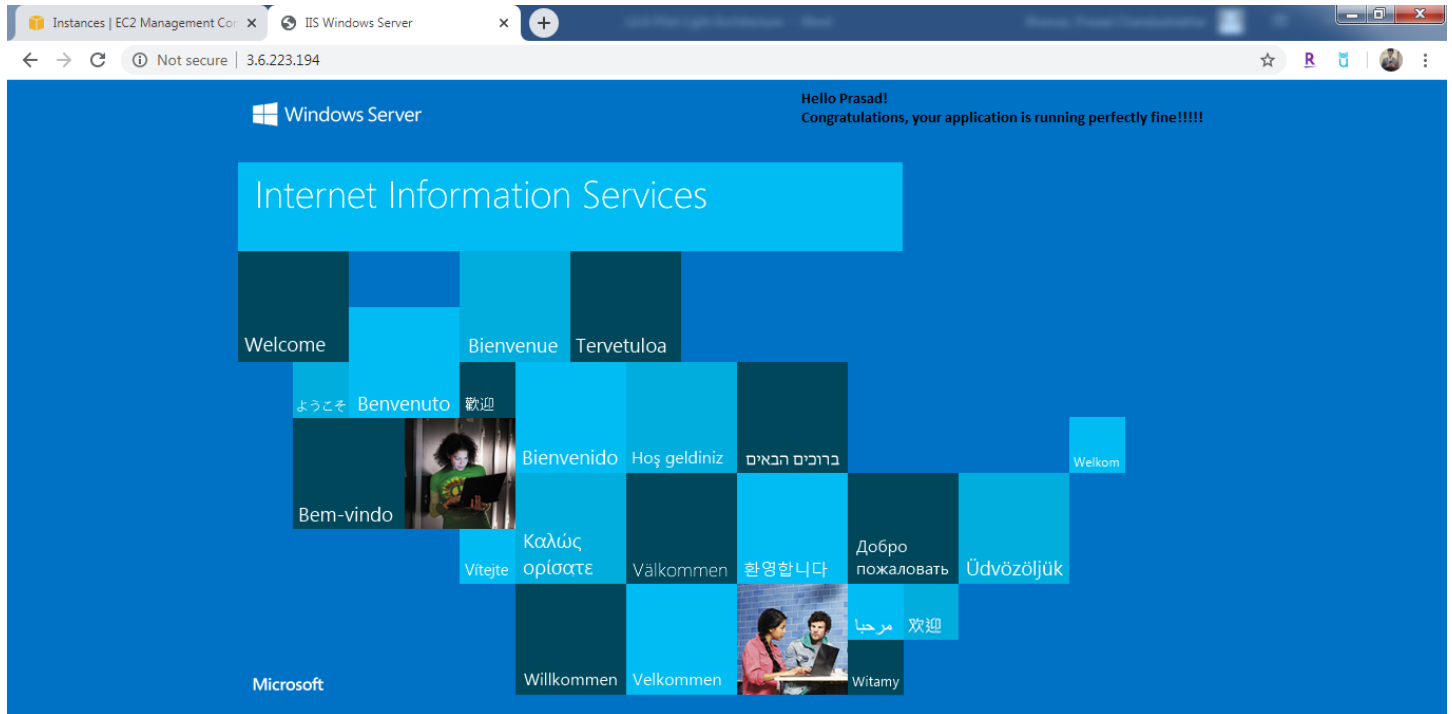
Instance Name	Instance ID	Instance Type	Region	Status	Checks	Tags
Pilot Light-Mumbai	i-066243cb3b8932c1f	t2.micro	ap-south-1a	running	2/2 checks ...	None

Instance: **i-066243cb3b8932c1f (Pilot Light-Mumbai)** Elastic IP: **3.6.223.194**

Description		Status Checks		Monitoring		Tags	
Instance ID	i-066243cb3b8932c1f	Public DNS (IPv4)	ec2-3-6-223-194.ap-south-1.compute.amazonaws.com	Instance state	running	IPv4 Public IP	3.6.223.194
Instance type	t2.micro	IPv6 IPs	-	Instance type	t2.micro	Elastic IPs	3.6.223.194*
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more						

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

It should open the Webpage as follows.



Finally, make a note of IP Addresses of EC2 Instances running in two different Regions.

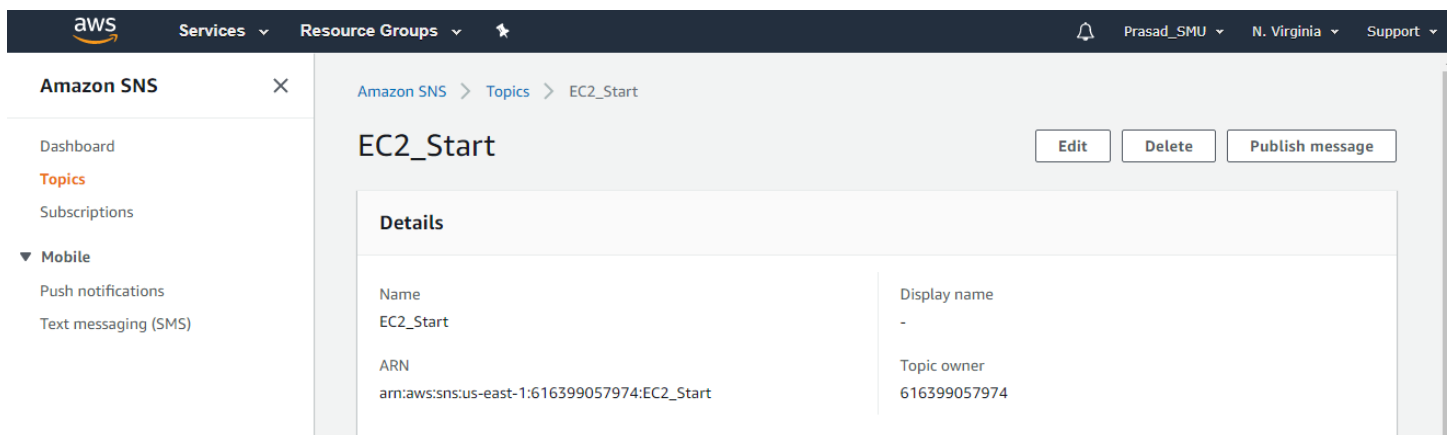
N. Virginia: 3.86.127.40

Mumbai: 3.6.223.194

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 N. Virginia region i.e. 3.86.127.40

The screenshot shows the AWS Route 53 Health Checks console. At the top, there's a search bar and a breadcrumb trail: << < 1 to 2 of 2 health checks. Below this is a table with columns: Name, Status, Description, and Alarms. The table contains one entry: 'check 1' with a status of 'Healthy' (indicated by a green bar and '15 hours ago now') and a description 'http://3.86.127.40:80/'. The 'Alarms' column shows '1 of 1 in OK'. Below the table are tabs for 'Info', 'Monitoring', 'Alarms', 'Tags', 'Health checkers', and 'Latency'. The 'Info' tab is selected, showing the 'ID' (0970483a-f82f-4fcd-b480-1215777d1144) and 'URL' (http://3.86.127.40:80/). A blue arrow points to the URL. Below the URL, it says 'Specify endpoint by IP address'. To the right, under 'Advanced configuration', there are settings for 'Request interval' (30 seconds) and 'Failure threshold' (3).

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 (3.86.127.40), Select the Routing Policy as Failover and Select the Health Check which you've configured.

Name: www.smu.edu

Type: A - IPv4 address

Alias: ☐ Yes ☒ No

TTL (Seconds): 300 1m 5m 1h 1d

Value: 3.86.127.40

IPv4 address. Enter multiple addresses on separate lines.
Example:
192.0.2.235
198.51.100.234

Routing Policy: Failover

Route 53 responds to queries using primary record sets if any are healthy, or using secondary record sets otherwise. [Learn More](#)

Failover Record Type: ☒ Primary ☐ Secondary

Set ID: www-Primary

Routing Policy: Failover

Route 53 responds to queries using primary record sets if any are healthy, or using secondary record sets otherwise. [Learn More](#)

Failover Record Type: ☒ Primary ☐ Secondary

Set ID: www-Primary

Associate with Health Check: ☒ Yes ☐ No

When responding to queries, Route 53 can omit resources that fail health checks. [Learn More](#)

Health Check to Associate: check 1

A Record for the Secondary Region: Put value as IP Address of the EC2 Instance (3.6.223.194), Select the Routing Policy as Failover and no need to select the Health Check here.

Edit Record Set

Name:

Type:

Alias: ☐ Yes ☒ No

TTL (Seconds):

Value:

IPv4 address. Enter multiple addresses on separate lines.
Example:
192.0.2.235
198.51.100.234

Routing Policy:

Route 53 responds to queries using primary record sets if any are healthy, or using secondary record sets otherwise. [Learn More](#)

Failover Record Type: ☐ Primary ☒ Secondary

Set ID:

Associate with Health Check: ☐ Yes ☒ No

Route 53 Configuration is now completed.

Task 4: Create an IAM Policy and Roles for Lambda Function

IAM Policy:

Navigate to the IAM Service., Click on Policies on the Left-Hand side and click on Create Policy.
Now click on JSON format and put the below code in the JSON Editor.

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "logs:CreateLogGroup",
8         "logs:CreateLogStream",
9         "logs:PutLogEvents"
10      ],
11      "Resource": "arn:aws:logs:*:*:*"
12    },
13    {
14      "Effect": "Allow",
15      "Action": [
16        "ec2:Start*"
17      ],
18      "Resource": "*"
19    }
20  ]
21 }
```

I'll give you this code in a .txt file.

Click on review the Policy, Give the Policy Name of your Choice.

At the bottom, look the resources on which this policy is going to give an access.

Filter			
Service	Access level	Resource	Request condition
Allow (2 of 228 services) Show remaining 226			
CloudWatch Logs	Limited: Write	arn:aws:logs:*:*	None
EC2	Limited: Write	All resources	None

Finally, Click on Create Policy.

IAM Policy has been created successfully.

Filter policies Start				
	Policy name	Type	Used as	Description
<input type="radio"/>	StartEC2Instance	Customer managed	Permissions policy (1)	This policy starts an Instance

IAM Roles:

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

Select the Type of Trusted Entity as LAMBDA.

Common use cases	
EC2	Allows EC2 instances to call AWS services on your behalf.
Lambda	Allows Lambda functions to call AWS services on your behalf.

Under Permissions Policies, select the Policy that you've created.

Filter policies Start	
	Policy name
<input checked="" type="checkbox"/>	StartEC2Instance

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

Role has been created Successfully.

Q EC2		
Role name ▾	Trusted entities	Last activity ▾
<input type="checkbox"/> StartEC2InstanceRole	AWS service: lambda	Today

Task 5: Configure a Lambda Function

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.

Runtime [Info](#)
Choose the language to use to write your function.

Python 3.7 ▾

Permissions [Info](#)
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add trigger:

▼ Choose or create an execution role


Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates


Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

StartEC2InstanceRole ▾ 


[View the StartEC2InstanceRole role](#) on the IAM console.

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.

Trigger configuration

 SNS
aws messaging notifications pub-sub push ▾

SNS topic
Select the SNS topic to subscribe to.

Q arn:aws:sns:us-east-1:616399057974:EC2_Start X 

Lambda will add the necessary permissions for Amazon SNS to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

☒ Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).

Cancel 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.

Basic settings Edit

Description	Memory (MB)	Info
-	128	
Timeout Info		
0 min	10 sec	

Now write down the Lambda Function Code as follows:

```
1 import boto3
2 region = 'ap-south-1'
3 instances = ['i-066243cb3b8932c1f']
4 ec2 = boto3.client('ec2', region_name=region)
5
6 def lambda_handler(event, context):
7     ec2.start_instances(InstanceIds=instances)
8     print('started your instances: ' + str(instances))
9
10
11
12
13
14
15
16
17
```

You just have to change the Instance ID in the Line Number 3. Copy the Instance ID of your EC2 Instance running in Mumbai (ap-south-1) region and Paste it in Line 3.

Finally, click on SAVE.

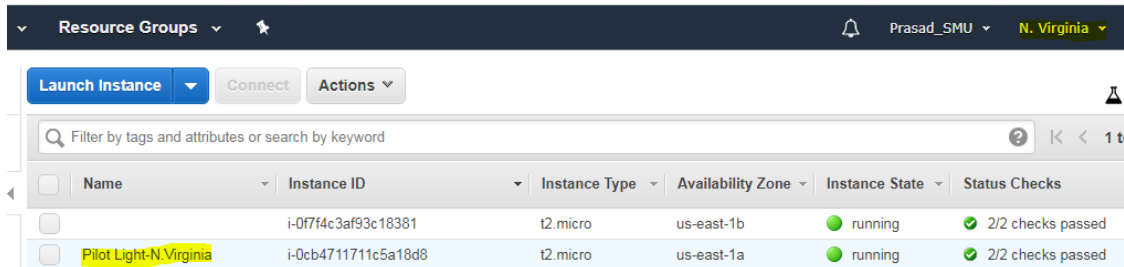
Lambda function is now configured properly.

☐ **StartEC2Function** Python 3.7 318 bytes

Task 6: Setup to test the Pilot Light Scenario

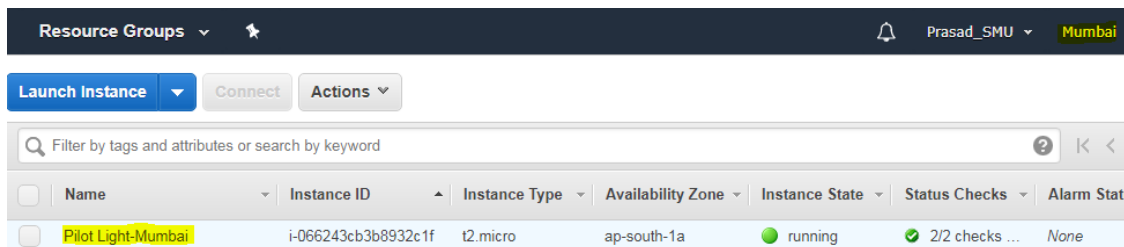
Follow the below steps to make setup to test the Pilot Light Scenario.

1. Verify the both the running EC2 Instances.



The screenshot shows the AWS Management Console for the 'N. Virginia' region. The 'Resource Groups' header is visible. Below the navigation bar, there are buttons for 'Launch Instance', 'Connect', and 'Actions'. A search bar is present with the text 'Filter by tags and attributes or search by keyword'. Below the search bar, a table lists EC2 instances. The table has columns: Name, Instance ID, Instance Type, Availability Zone, Instance State, and Status Checks. Two instances are listed: one with ID 'i-077f4c3af93c18381' and another named 'Pilot Light-N.Virginia' with ID 'i-0cb4711711c5a18d8'. Both are 't2.micro' instances in the 'us-east-1a' availability zone and are in the 'running' state with '2/2 checks passed'.

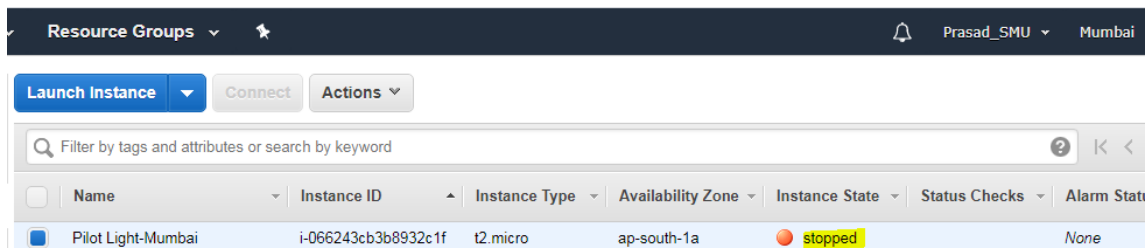
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
	i-077f4c3af93c18381	t2.micro	us-east-1b	running	2/2 checks passed
Pilot Light-N.Virginia	i-0cb4711711c5a18d8	t2.micro	us-east-1a	running	2/2 checks passed



The screenshot shows the AWS Management Console for the 'Mumbai' region. The 'Resource Groups' header is visible. Below the navigation bar, there are buttons for 'Launch Instance', 'Connect', and 'Actions'. A search bar is present with the text 'Filter by tags and attributes or search by keyword'. Below the search bar, a table lists EC2 instances. The table has columns: Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, and Alarm Status. One instance is listed: 'Pilot Light-Mumbai' with ID 'i-066243cb3b8932c1f'. It is a 't2.micro' instance in the 'ap-south-1a' availability zone and is in the 'running' state with '2/2 checks ...' and 'None' alarm status.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
Pilot Light-Mumbai	i-066243cb3b8932c1f	t2.micro	ap-south-1a	running	2/2 checks ...	None

2. Let's Power OFF the EC2 Instance running in the Mumbai Region.



The screenshot shows the AWS Management Console for the 'Mumbai' region. The 'Resource Groups' header is visible. Below the navigation bar, there are buttons for 'Launch Instance', 'Connect', and 'Actions'. A search bar is present with the text 'Filter by tags and attributes or search by keyword'. Below the search bar, a table lists EC2 instances. The table has columns: Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, and Alarm Status. One instance is listed: 'Pilot Light-Mumbai' with ID 'i-066243cb3b8932c1f'. It is a 't2.micro' instance in the 'ap-south-1a' availability zone and is now in the 'stopped' state with 'None' alarm status.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
Pilot Light-Mumbai	i-066243cb3b8932c1f	t2.micro	ap-south-1a	stopped		None

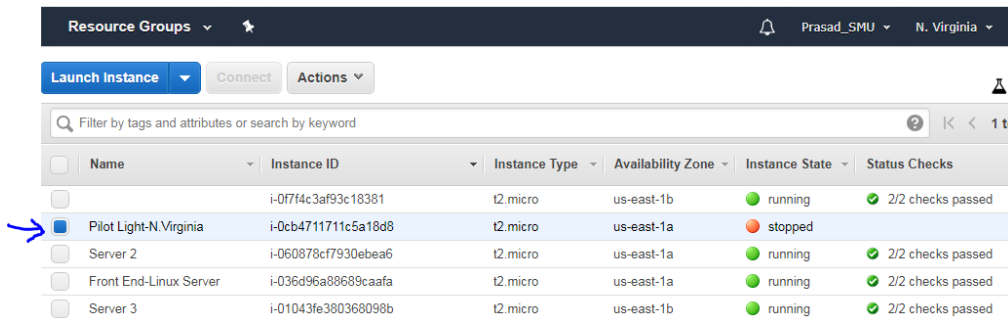
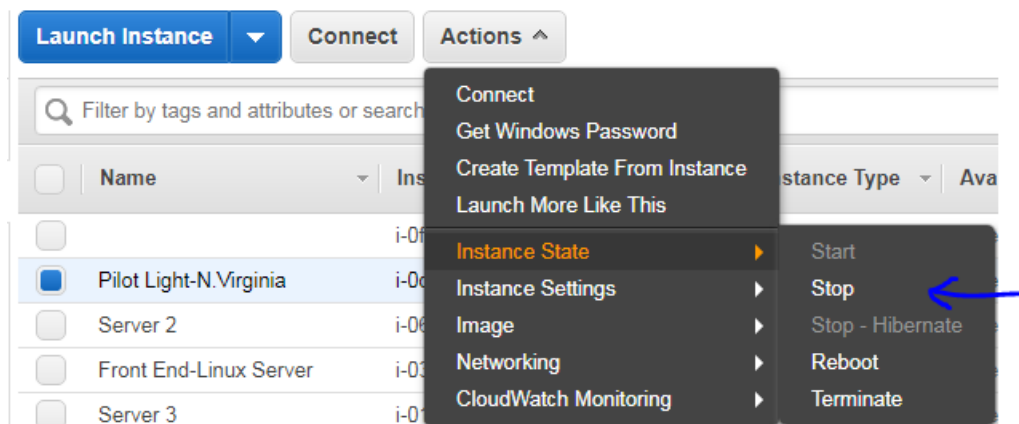
Task 7: Test the Pilot Light Scenario

What's going to happen now?

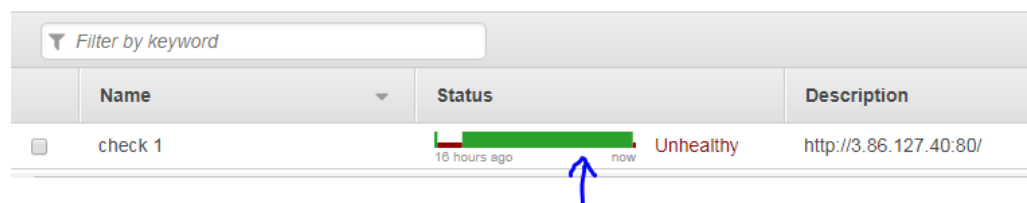
If we Power OFF our Primary EC2 Instance which is running in the N. Virginia region, then our Secondary EC2 Instance which is currently Powered OFF in the Mumbai Region is going to get Powered ON and RUNNING.

Let's verify it.

STOP the EC2 Instance which is running in the N. Virginia Region.



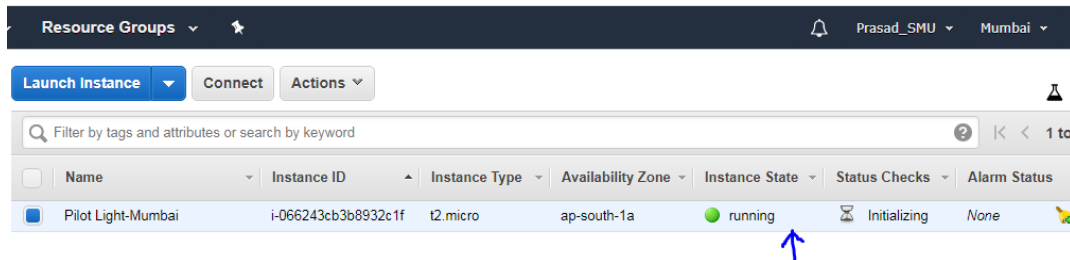
Let's look at the Route 53 Health Check.



You can see that the Health Status is now UNHEALTHY.

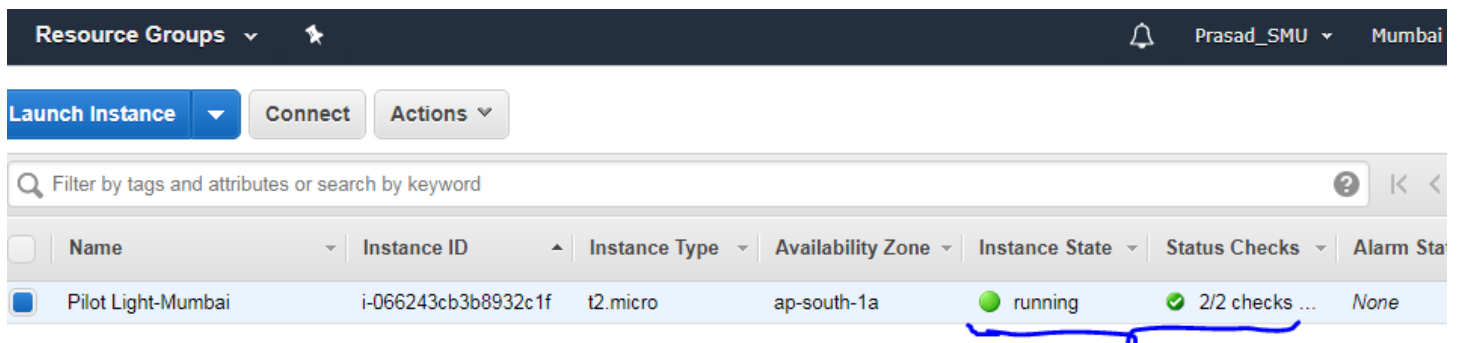
This has now triggered the SNS Topic and since the SNS Topic has Lambda Subscription, it has triggered the Lambda Function.

Lambda Function will now Power ON the EC2 Instance which is currently STOPPED in the Mumbai Region.

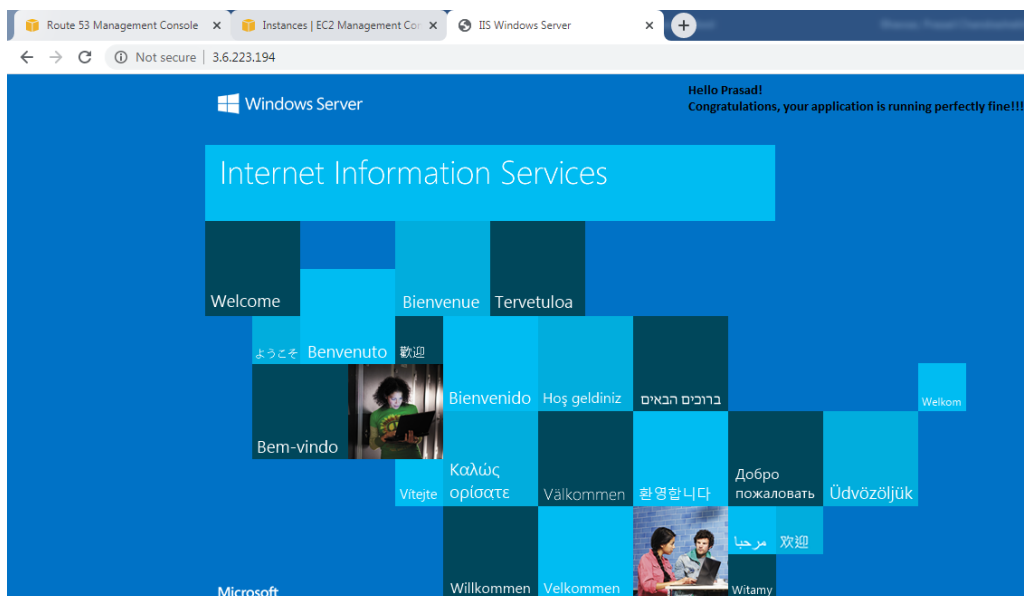


You can see that the EC2 Instance in Mumbai Region is now Initializing.

Now the EC2 Instance which is in Mumbai region is fully operational automatically.



Also, the Route 53 Failover Policy has now detected the Failover and routed the traffic towards the EC2 Instance in Mumbai Region.



This proves the Pilot Light Architecture.

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