Capstone Project: Deploying a Restaurant Application on Cloud

DESCRIPTION

As a Cloud Architect, you are asked to deploy the restaurant application on cloud.

Background of the problem statement:

You have created a new website for the restaurant and used a public cloud for the internet facing website of the restaurant. After deploying the application on cloud, users are complaining about the reloading speed of the page. The website is getting global traffic and static assets like pages are served from a single server. Also, you need to make sure that the traffic coming to the application from different parts of the world is load balanced at DNS level.

You can use either Azure or AWS platforms to design the solution using IaaS OR PaaS.

You have been asked to:

1. Suggest an appropriate solution so that your company can make use of the cloud while keeping the requirements mentioned above for your company in mind
2. Provide an approach to:

   a. Govern all the resources being used for development, testing, and production of the company's website

   b. Keep a separate track of the billing life cycle and cost management of all the services being used for hosting the company's website on Cloud
3. Upload all static content of your web site to cloud
4. Create a CDN endpoint and configure it to serve the static files you have uploaded
5. Use storage service and upload files for your teammates to share
6. Connect a Windows or Linux VM to the Storage service

You must use the following tools:

- AWS: Route 53, S3 Bucket, CloudFront, EC2
- Azure: Azure App Service, CDN, DNS, Azure VM, Azure Traffic Manager

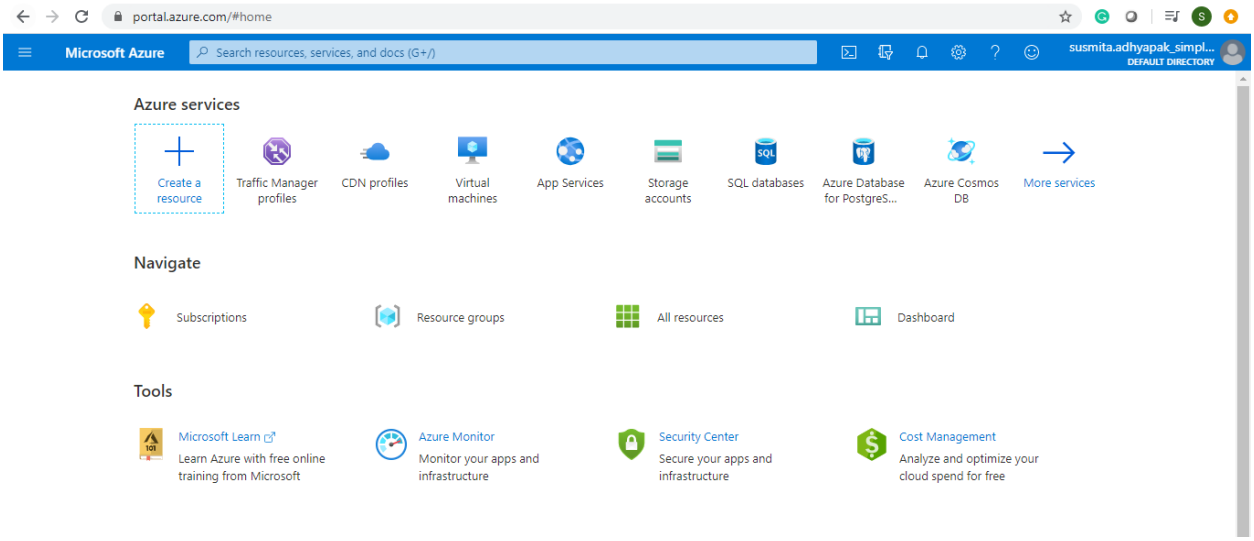
This section will guide you to deploy an application on:

- Azure
- AWS

Azure:

Approach 1:

Step 1: Log into the Azure portal

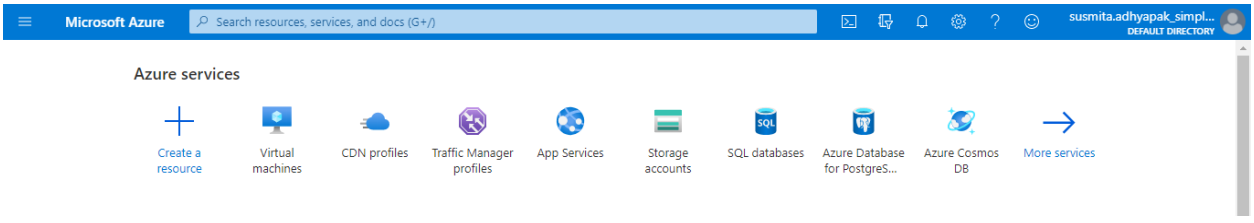


Step 2: Before creating the resources, make sure you apply tags to resources so that you can keep a track of billing later on.

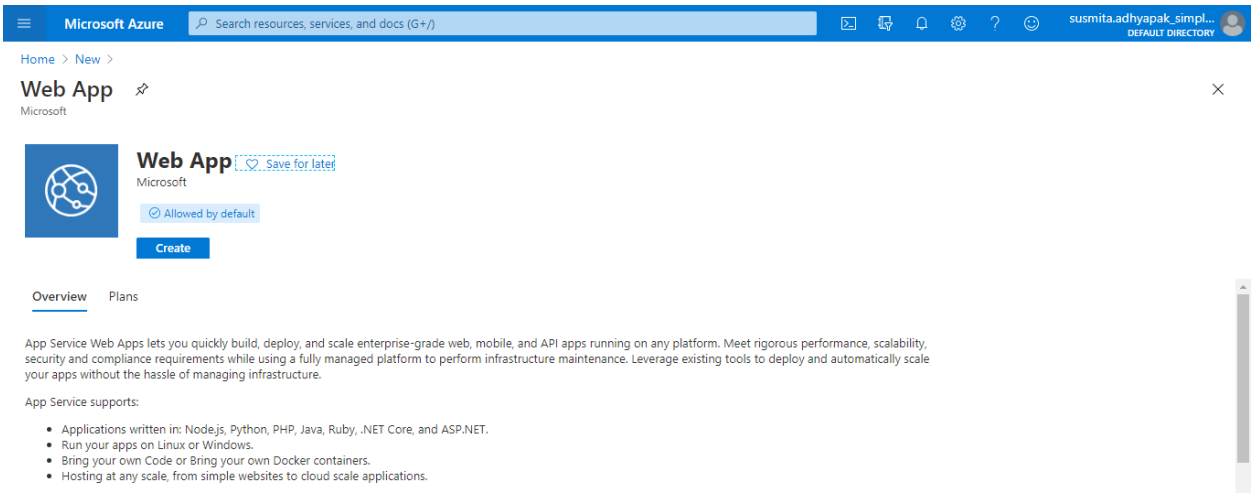
Step 3: To begin, create an Azure App Service Plan in Standard Tier

Step 4: Create an App Service (Web App) using the App Service Plan that you just created

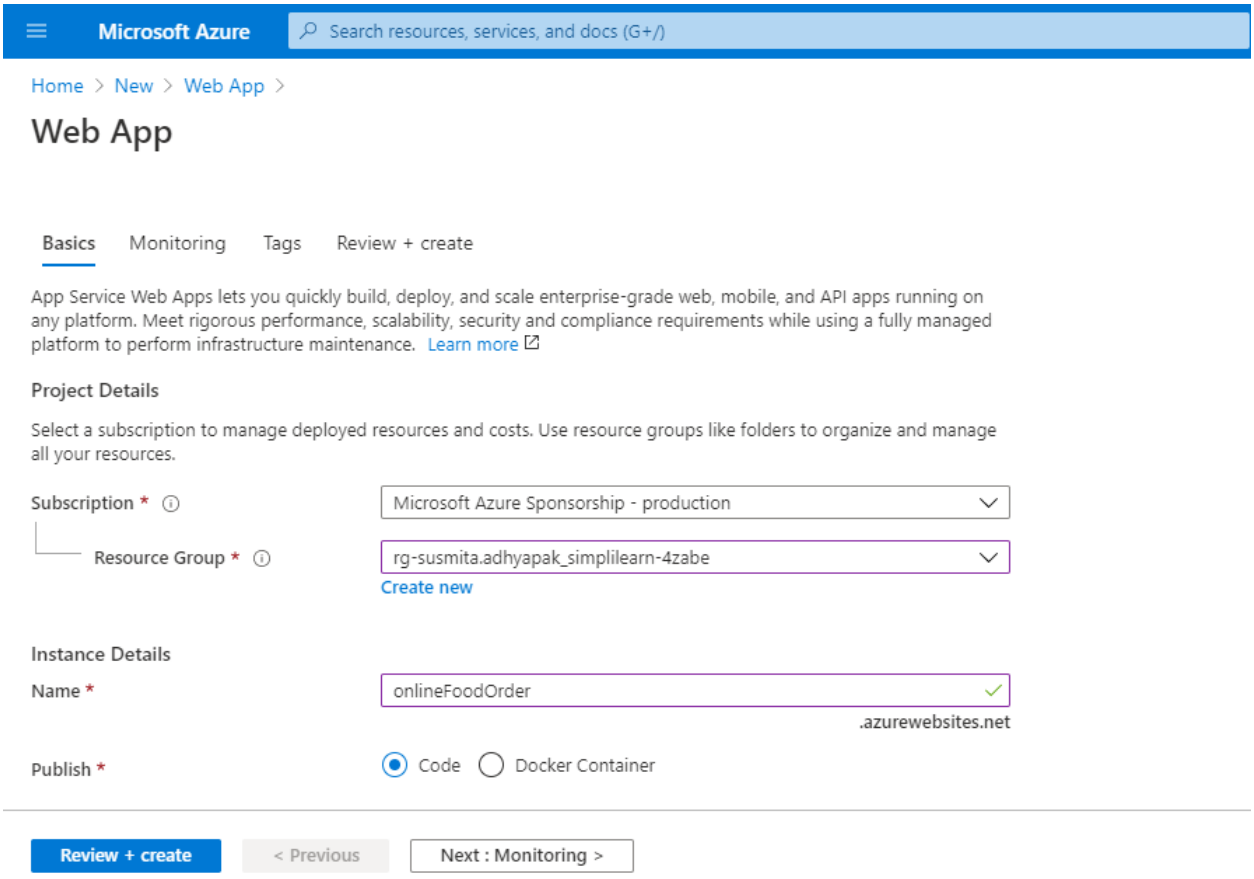
Step 4.1: Click on **Create a resource**

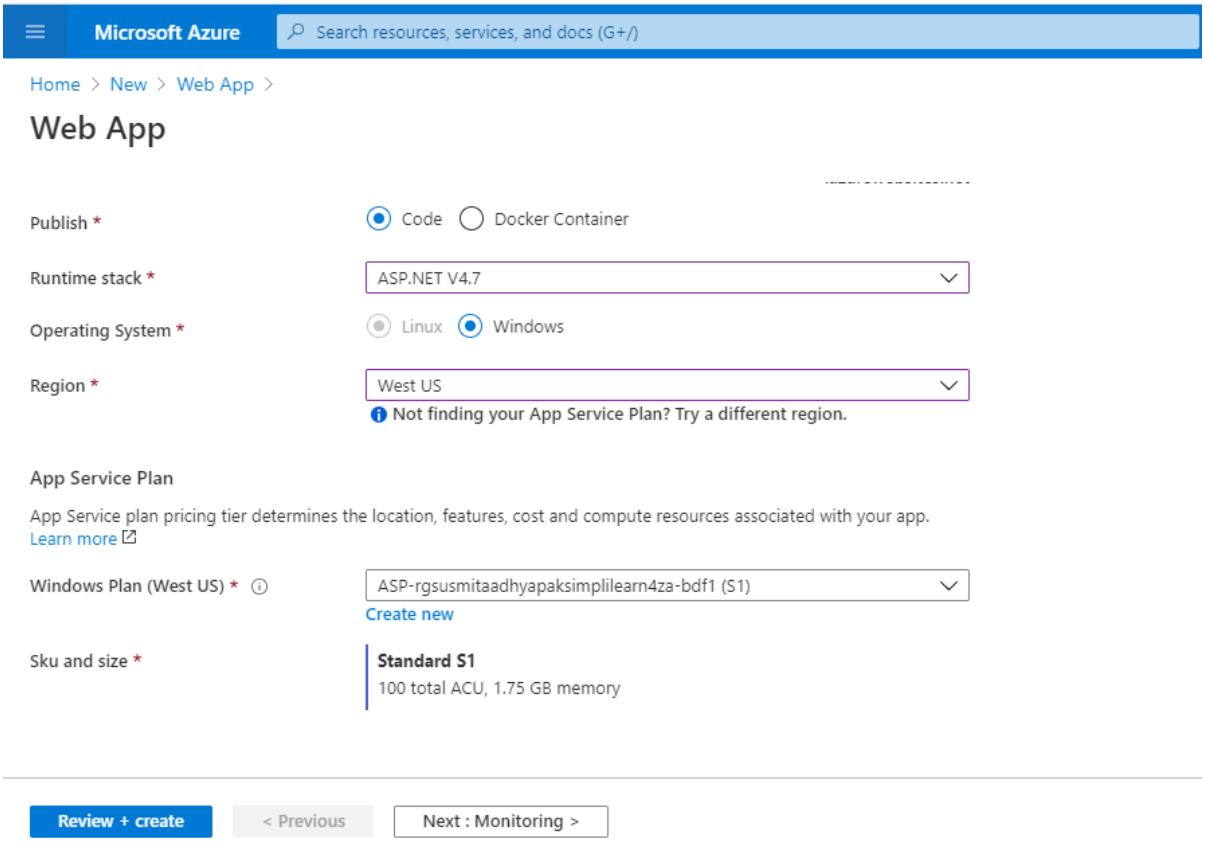


Step 4.2: Search for Web App and click on **Create**



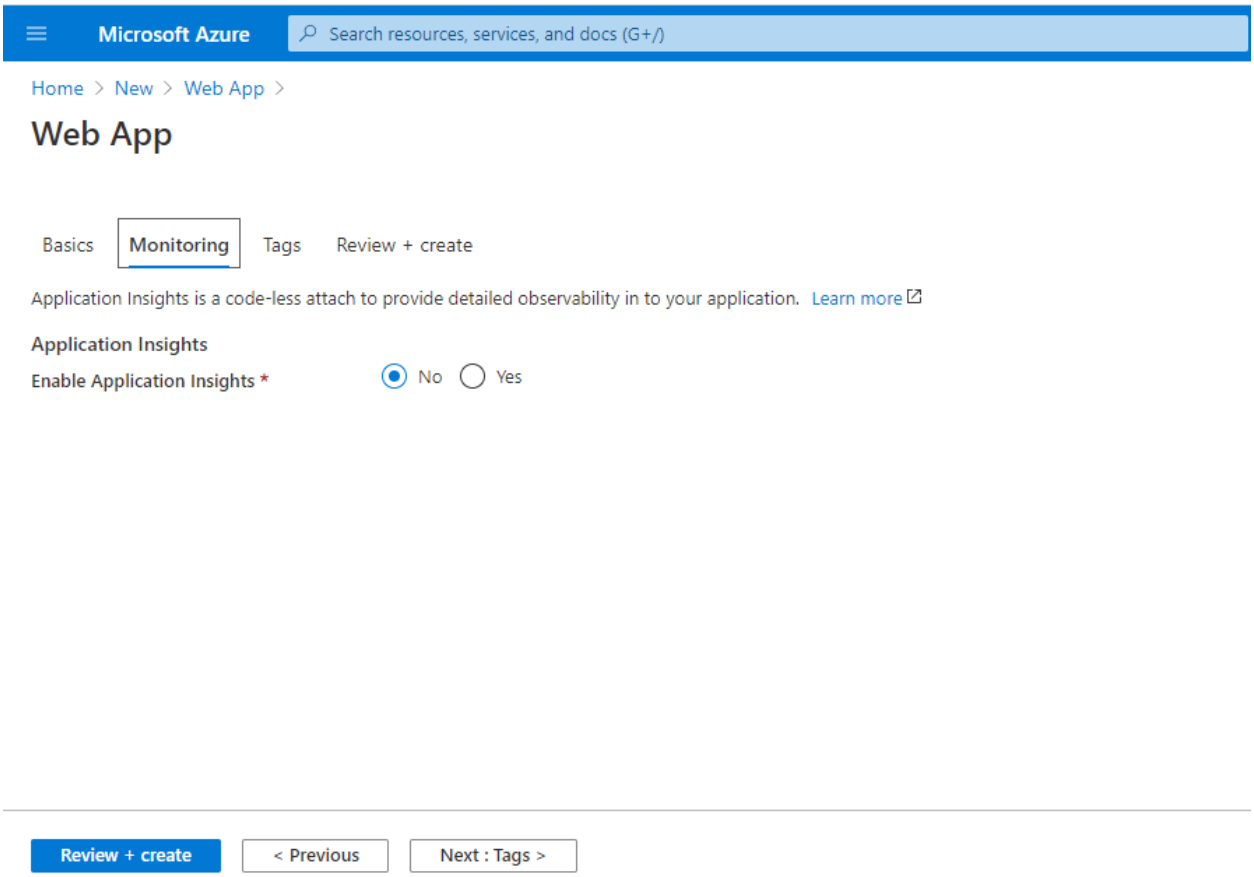
Step 4.3: Provide the basic information for the application





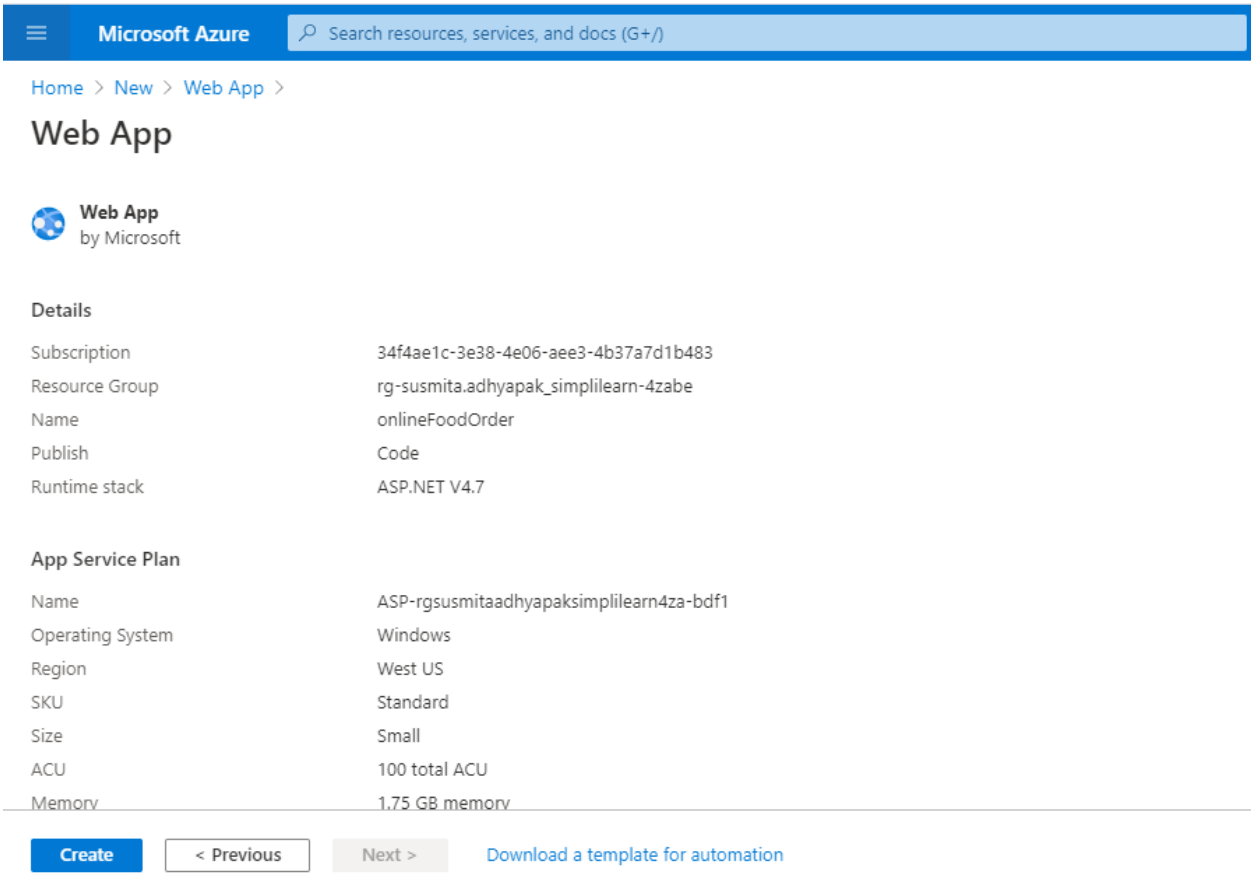
Note: Choose the runtime stack as ASP.NET V4.7 and choose the region as West US or West US 2

Step 4.4: In the Monitoring section, select **No** for **Enable Application Insights**

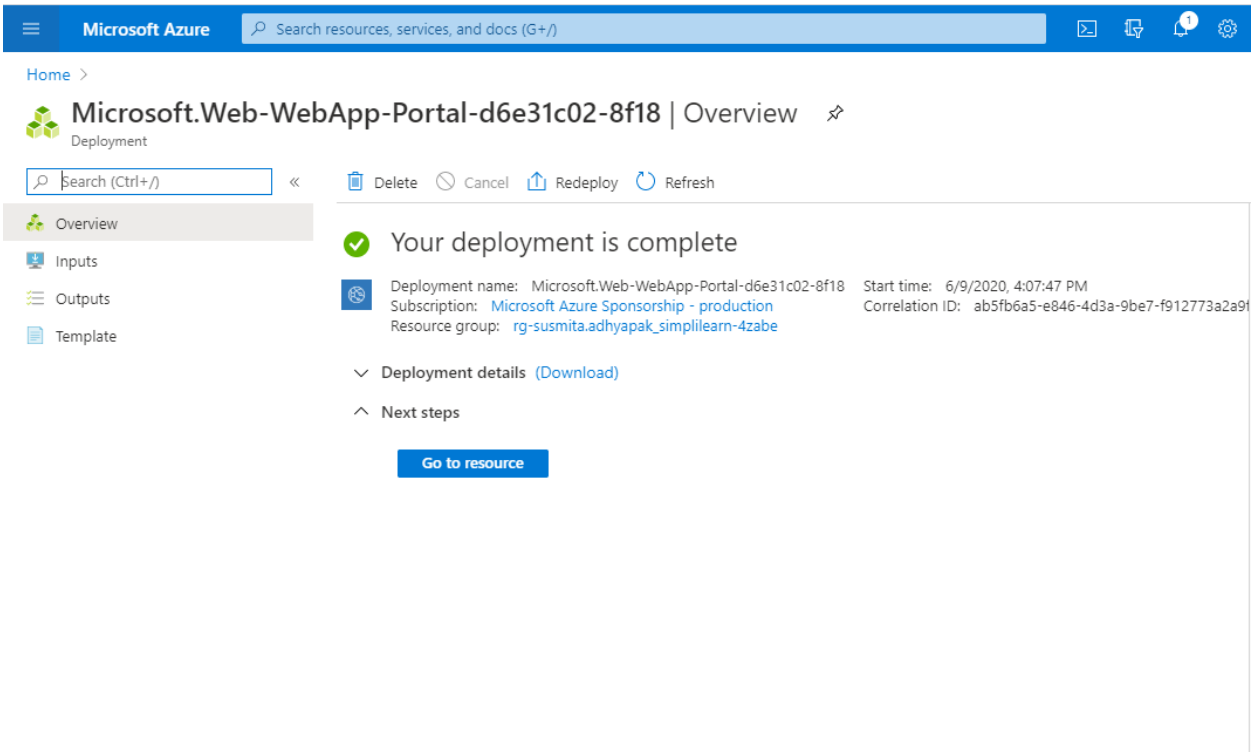


Step 4.5: Click on **Review and Create**

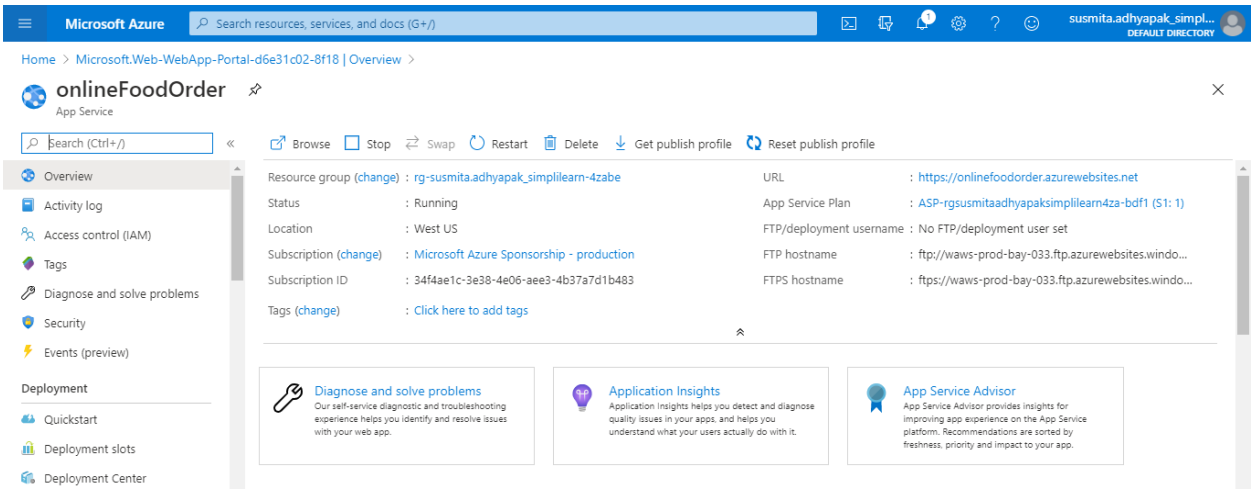
Step 4.6: Click on **Create**



Step 4.7: This will create the Web App on Azure.

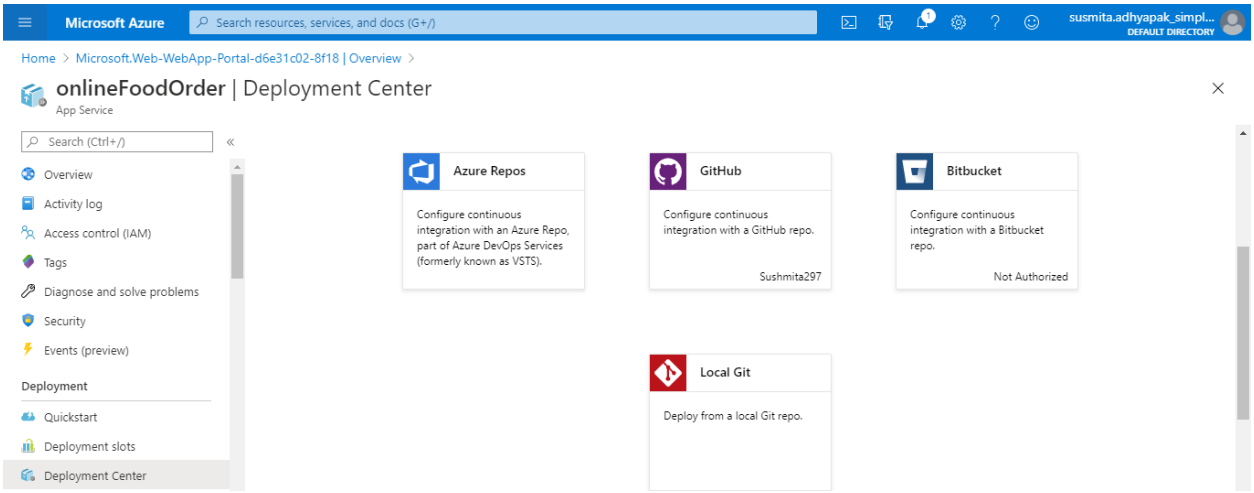


Step 4.8: Click on **Go to resource** to get the overview of the created web app



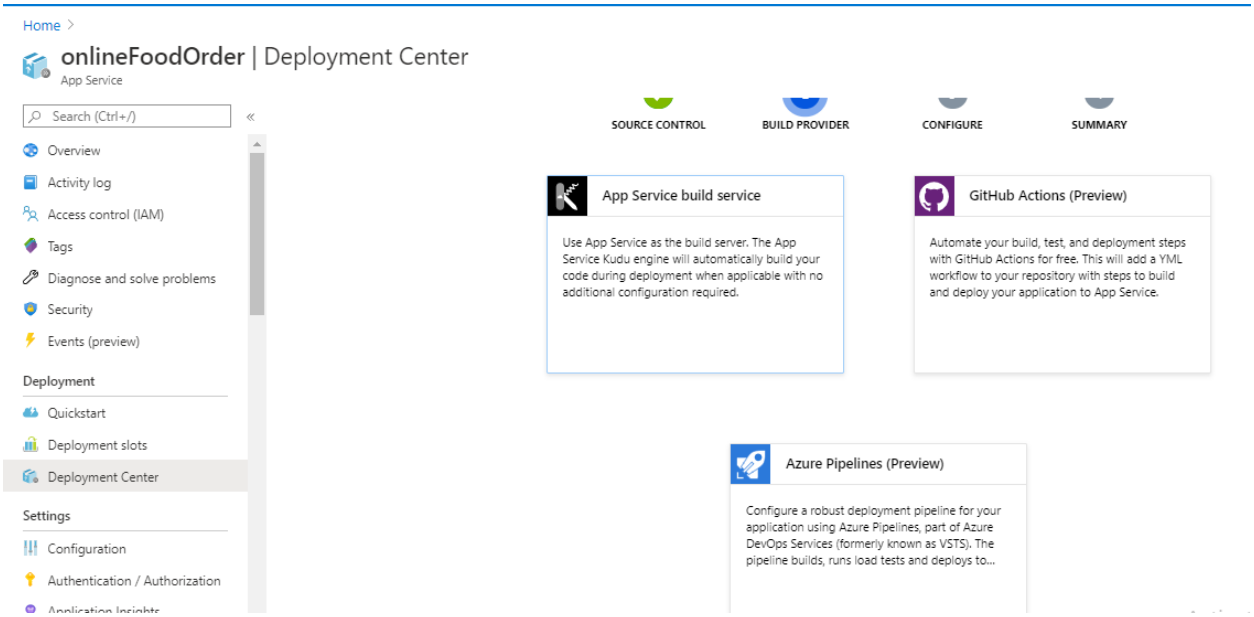
Step 5: Deploy your static web app to Azure App Service (Web App) using a method of your choice such as Visual Studio Code, GitHub, or FTP.

Step 5.1: Go to **Deployment Center**

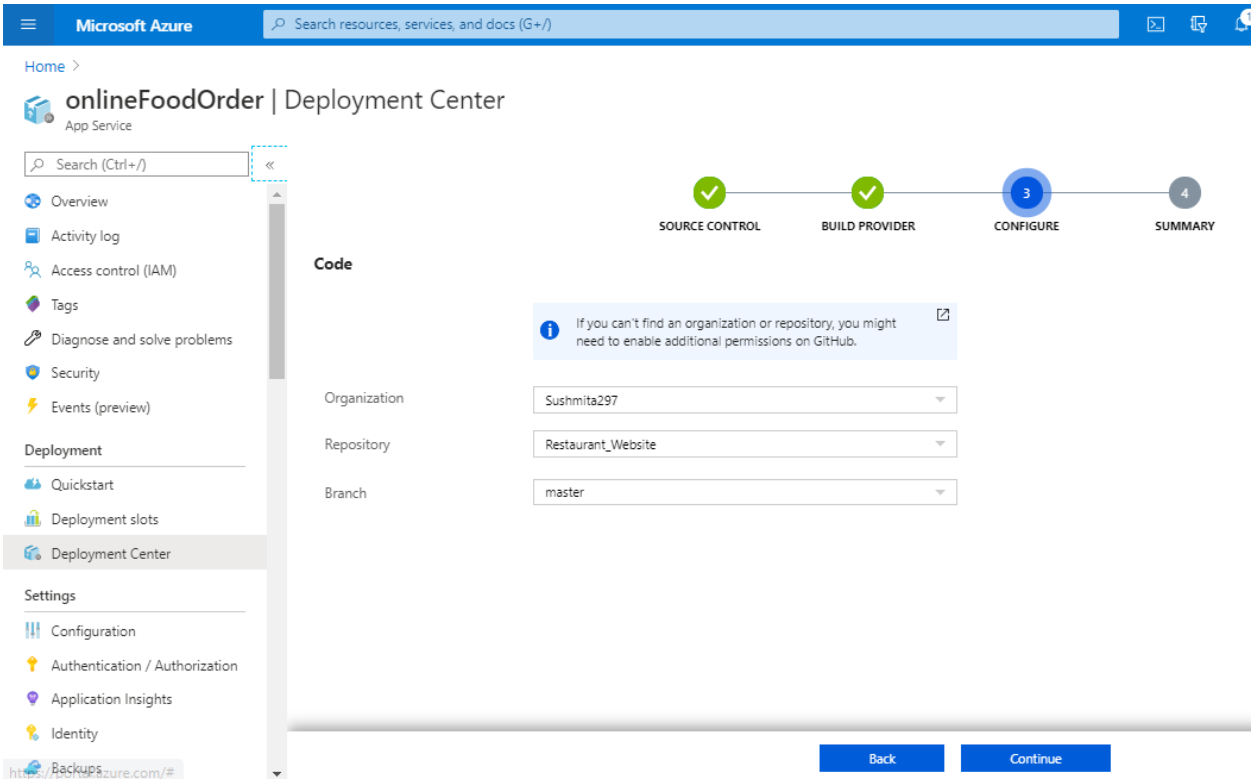


Step 5.2: Select GitHub

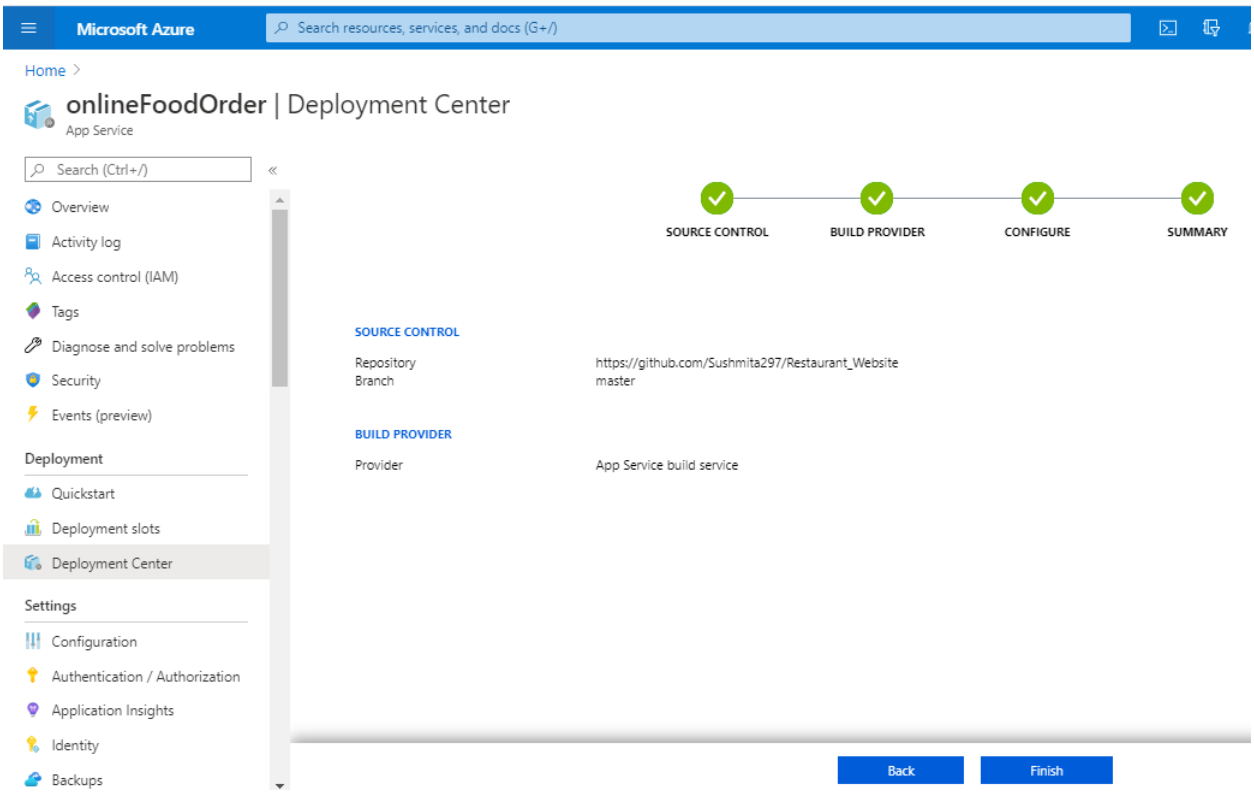
Step 5.3: Authorize your account



Step 5.4: Select the application files uploaded on GitHub and click on **Continue**

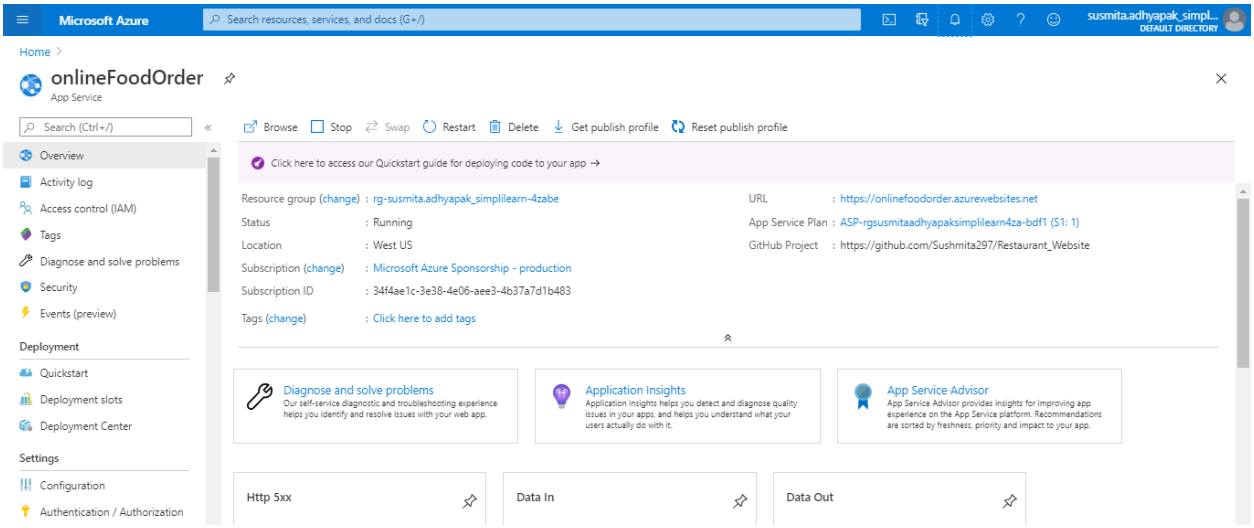


Step 5.5: Click on **Finish**

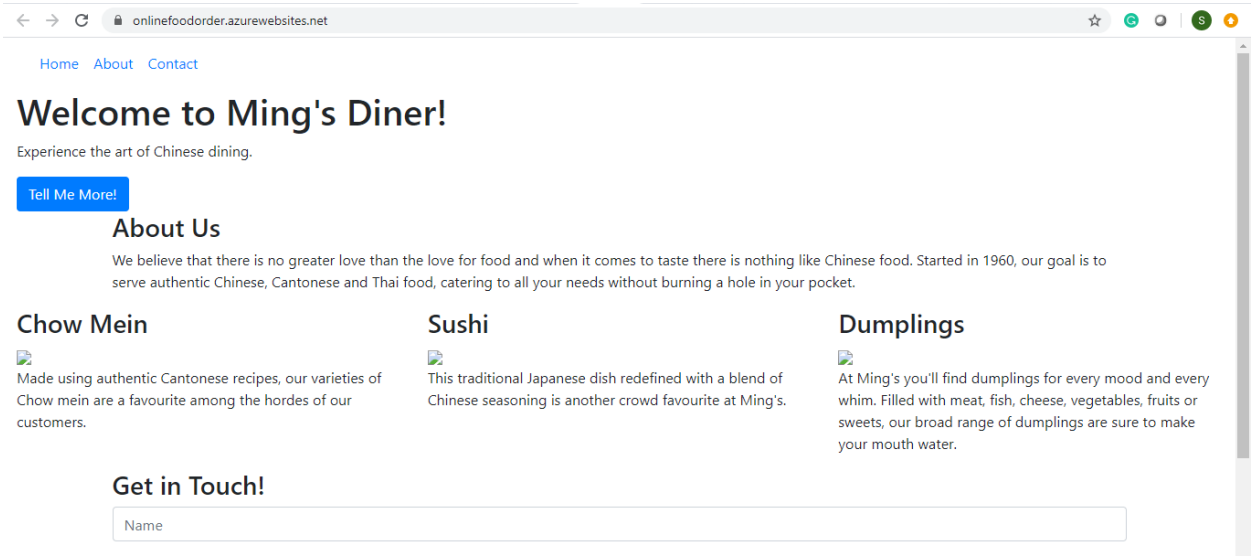


Step 6: Hit the web app endpoint to check if the application is online

Step 6.1: Click on Overview of the web app



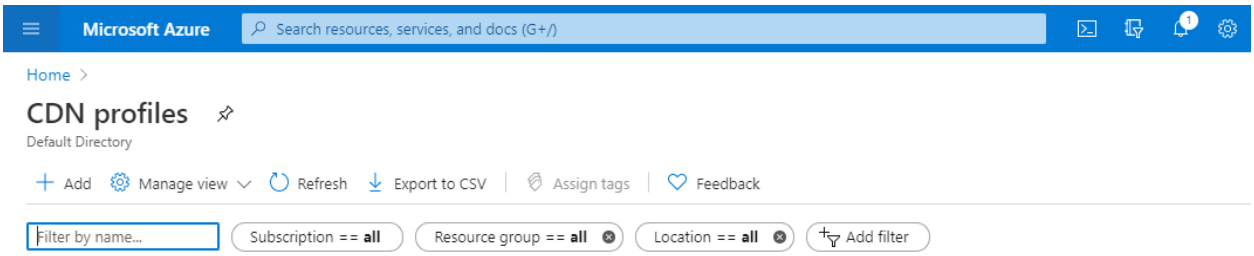
Step 6.2: Click on the URL and you will get the application running



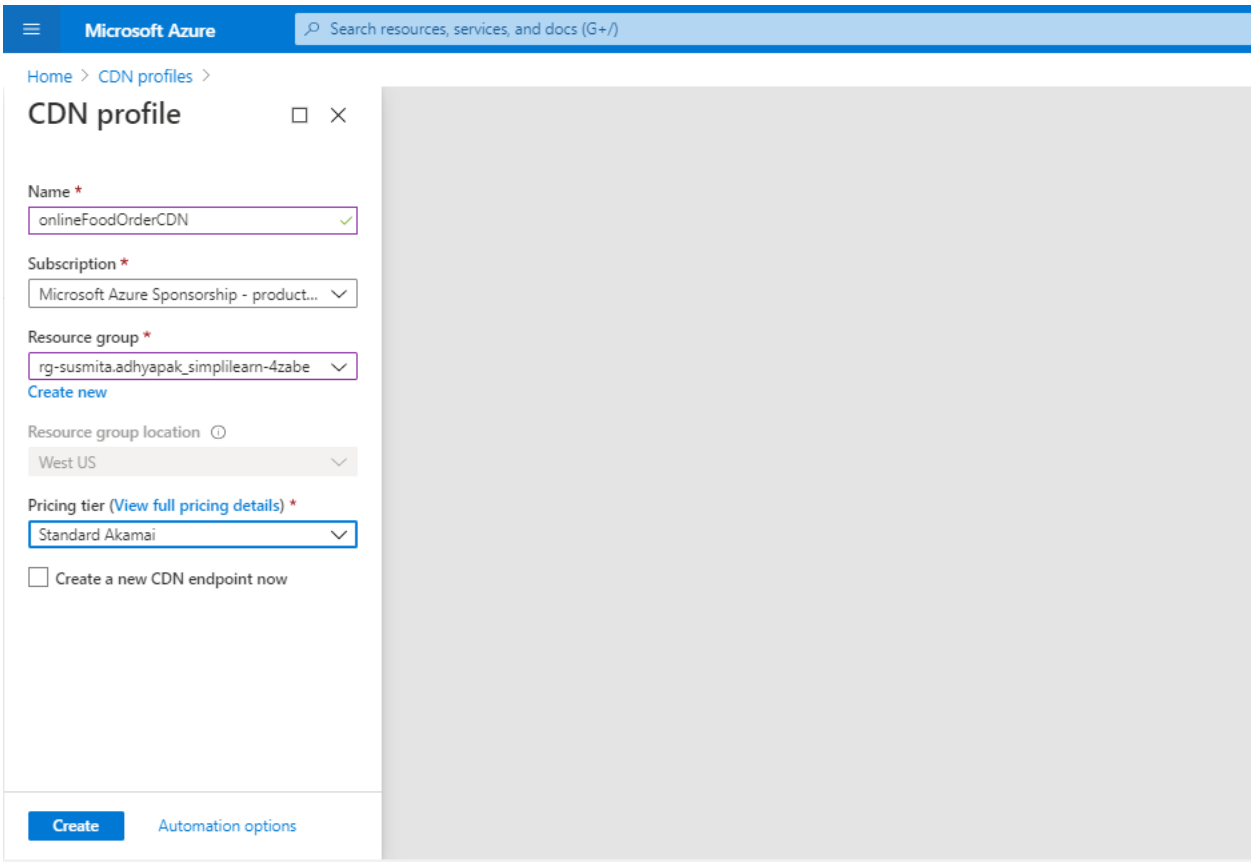
Step 7: Now create a CDN profile

Step 7.1: In the search window, search for CDN profiles

Step 7.2: Click on **Add**

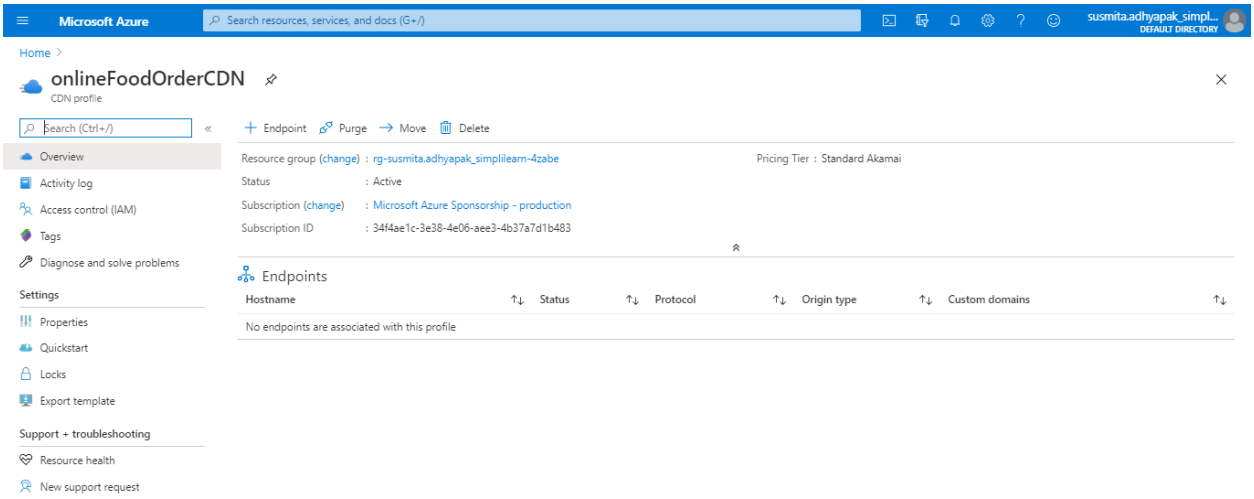


Step 7.3: Provide the information to create the CDN and click on **Create**



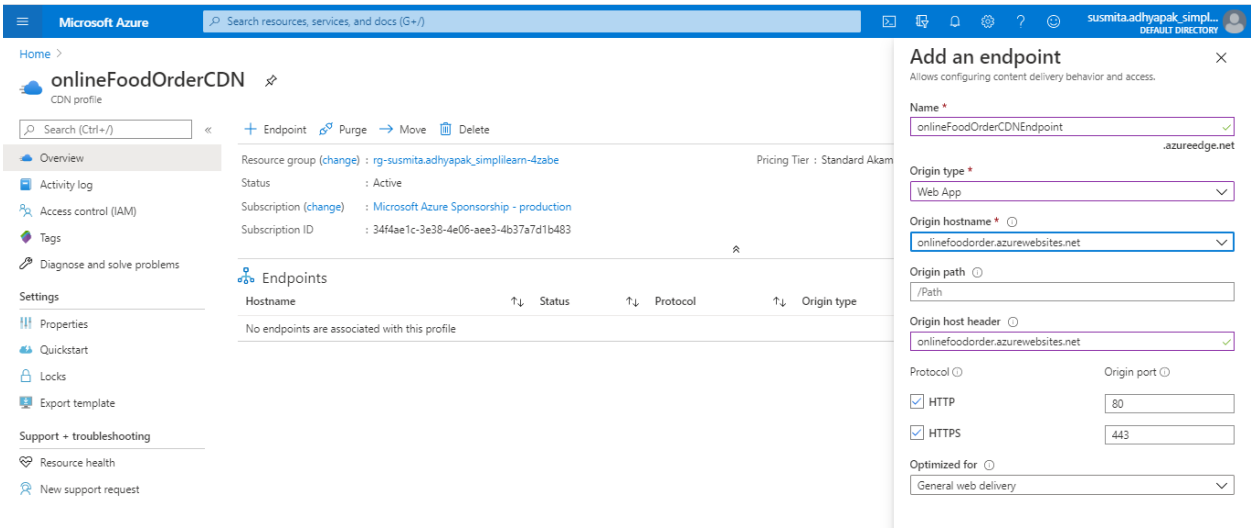
Step 8: Use CDN profile to create an endpoint

Step 8.1: Go to the created CDN

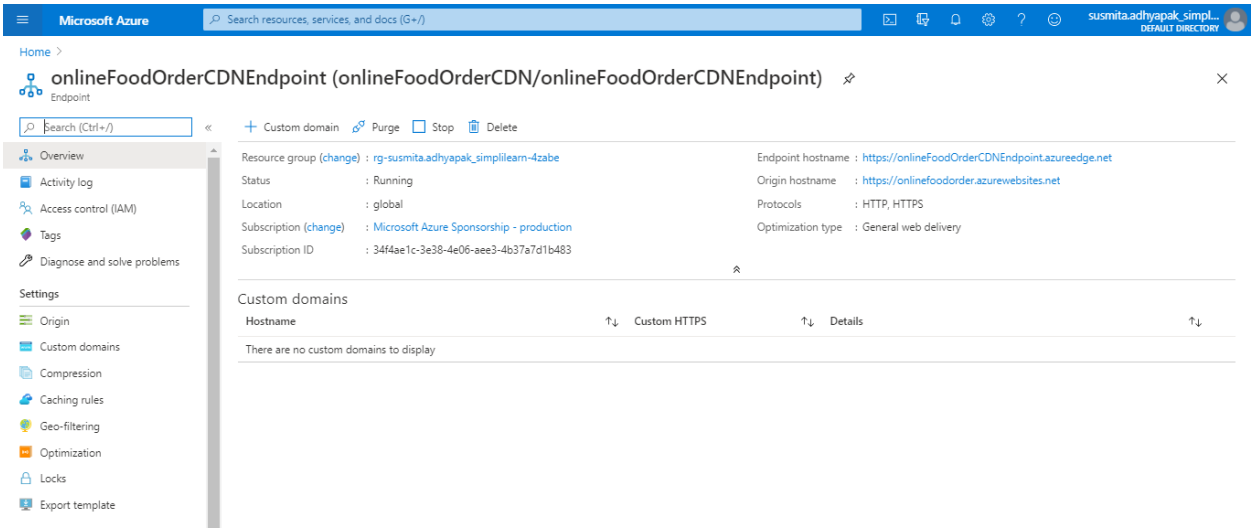


Step 8.2: Click on Endpoint

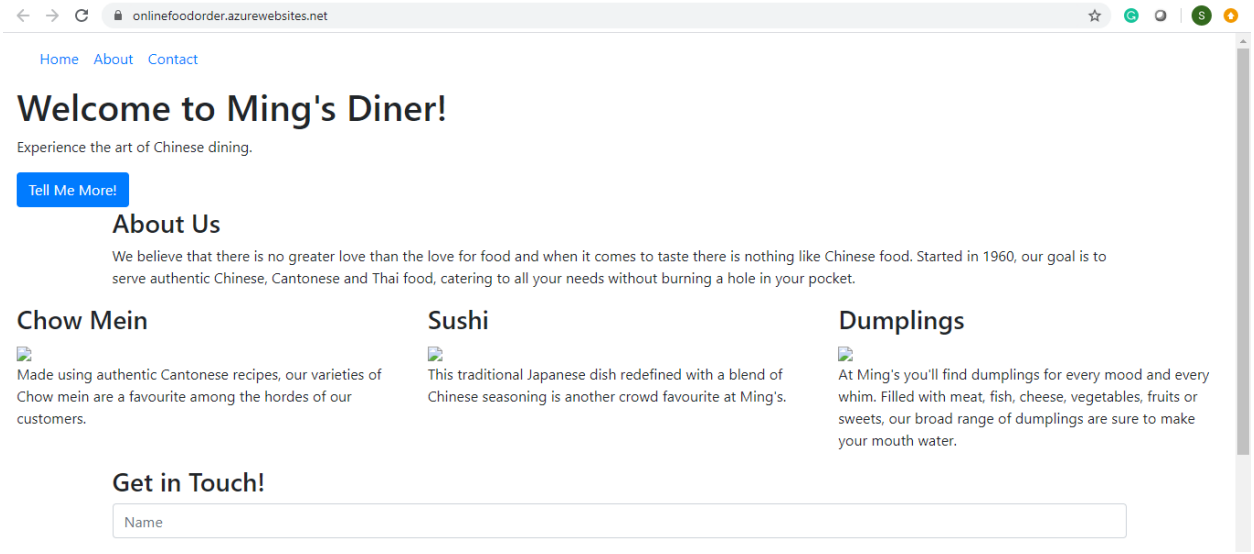
Step 8.3: Provide the basic information about the endpoint and click on **Add**



Step 8.4: Go to the created CDN endpoint and click on **Origin hostname**



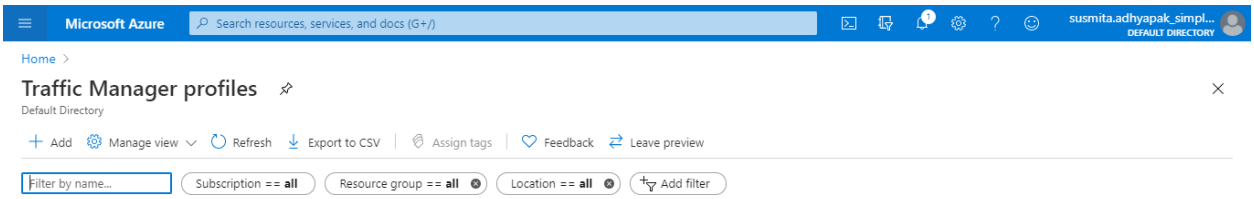
Step 8.5: Your application is running.



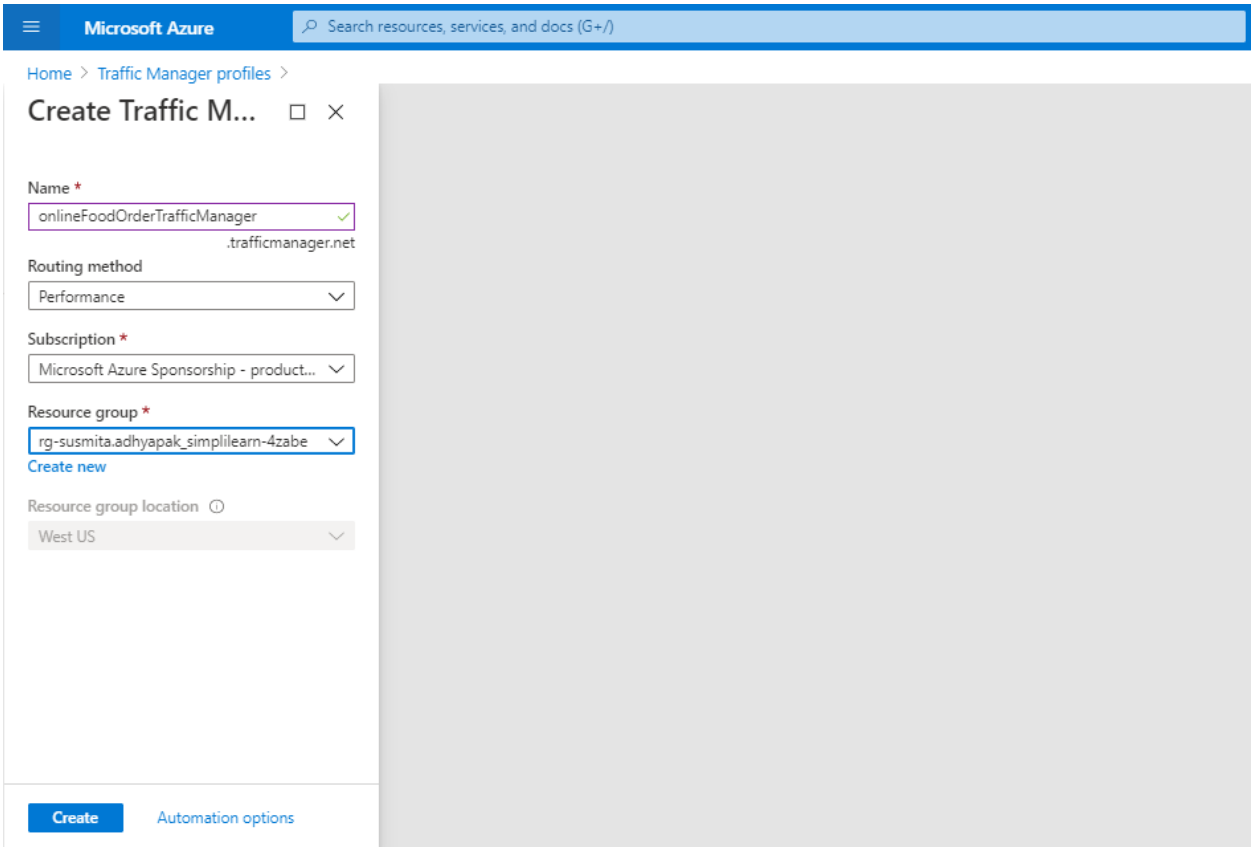
Step 9: Repeat steps 2 to 7 to create multiple deployments of your application in different regions so that you can meet the global traffic demand

Step 10: To make sure that traffic coming from different parts of the world is load balanced at DNS level, create a Traffic Manager Profile

Step 10.1: In the search window, search for Traffic Manager Profile. Click on **Add** to create a new traffic manager profile

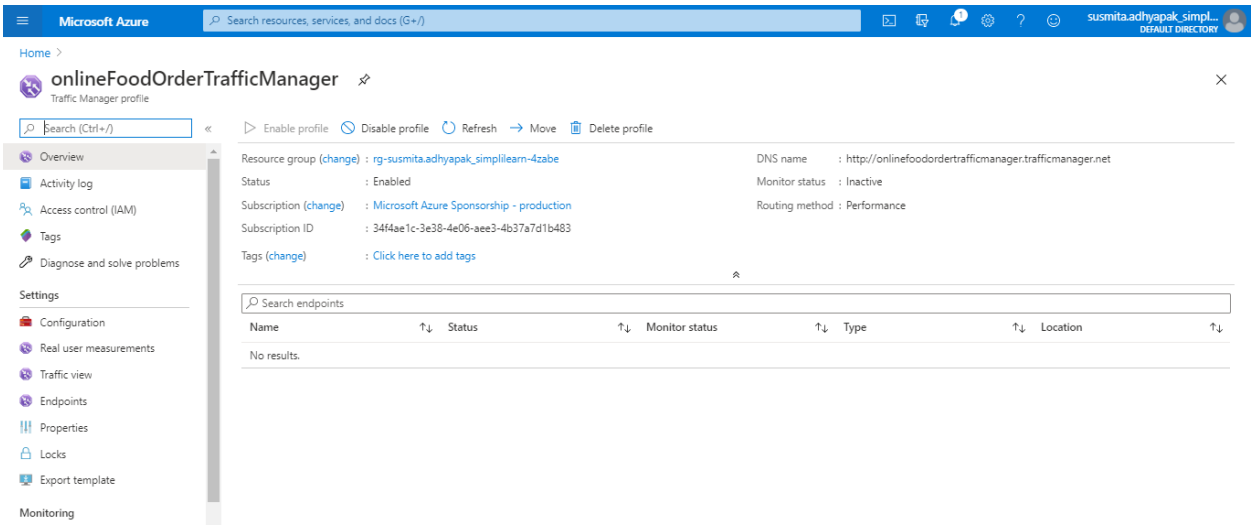


Step 10.2: Provide the required information and click on **Create** to create the Traffic Manager Profile

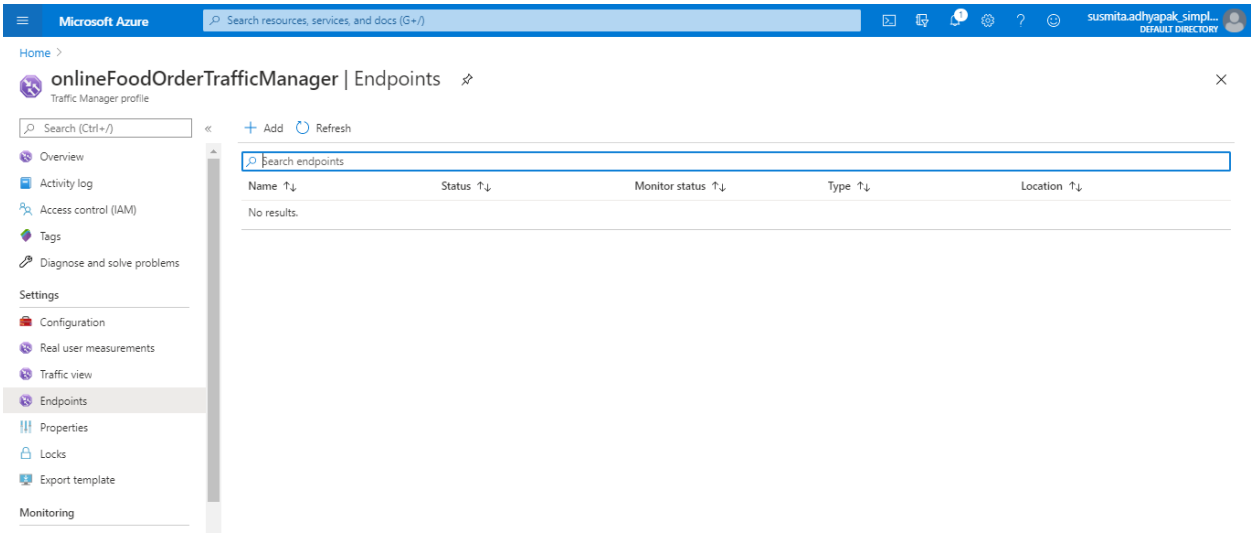


Step 11: Create endpoints in the traffic manager corresponding to each CDN endpoints that you have created

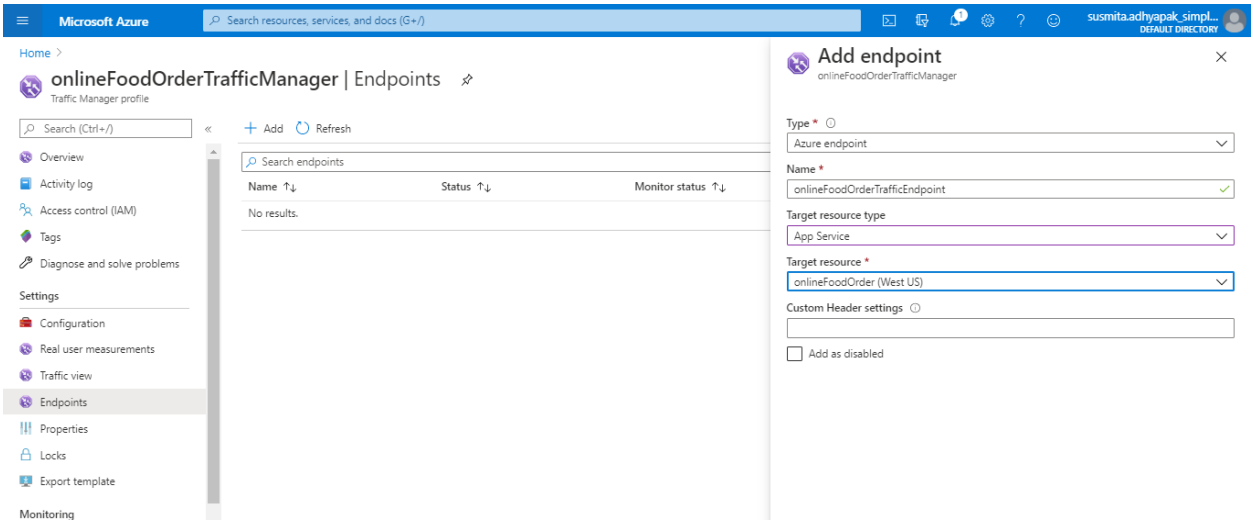
Step 11.1: Go to the created Traffic Manager Profile



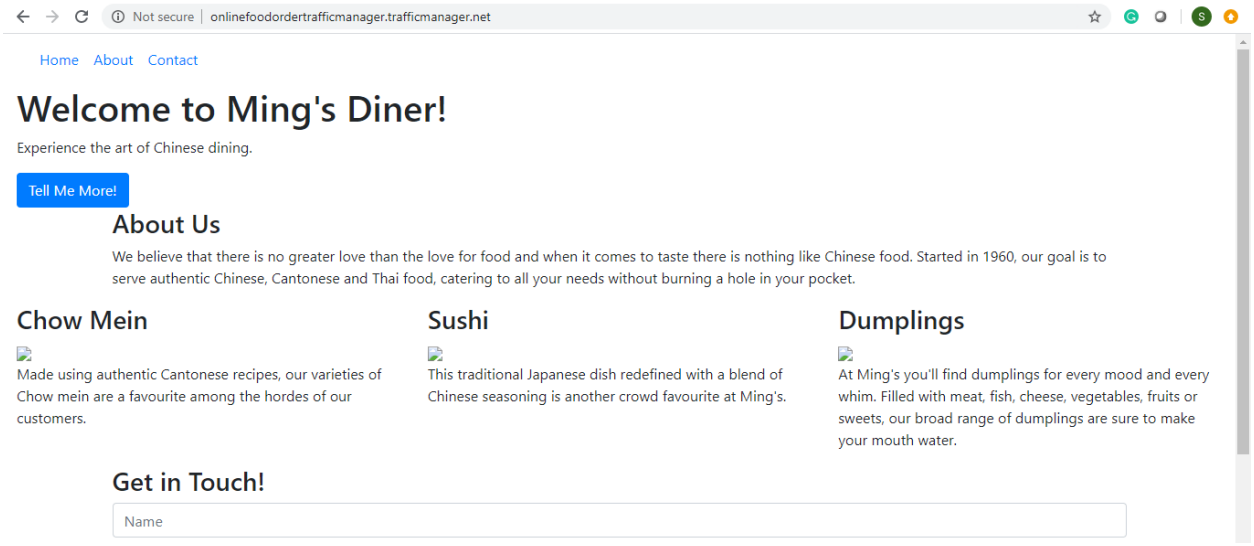
Step 11.2: Click on **Endpoints**. Click on **Add** to add new endpoints



Step 11.3: Provide the required information and click on **Add**



Step 11.4: Once the monitor status is online, copy the link of the DNS name and check whether the application is online



Step 12: Optionally, if you want to add the application in your own domain, you can configure the traffic manager to point to a custom domain.

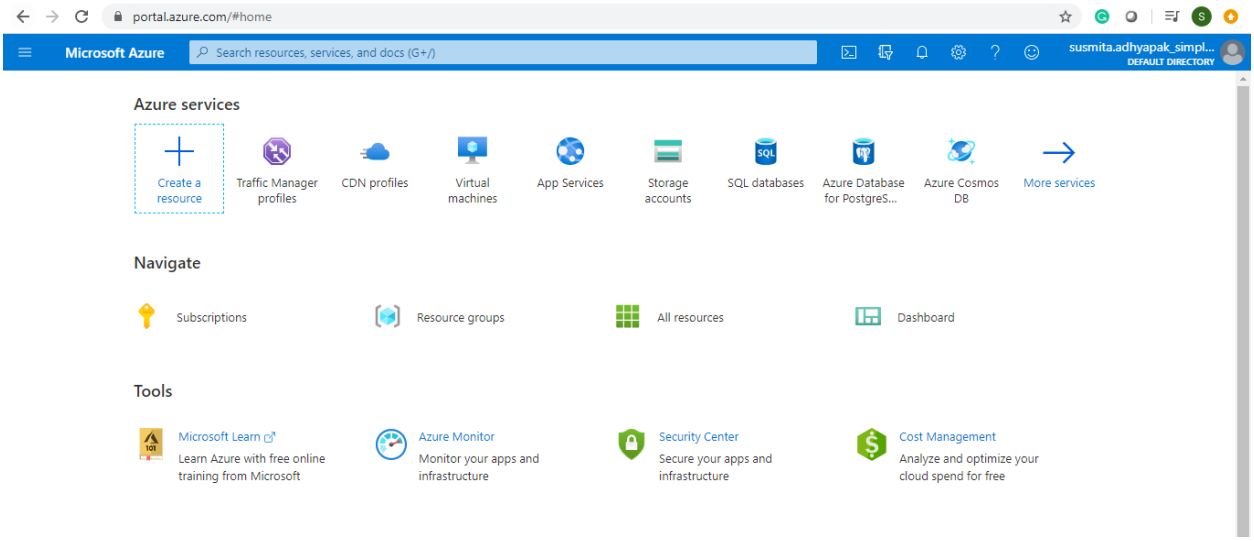
Step 13: As good practice, follow the principle of least privilege so that you give

access to the services that need to be accessed within the azure portal

Azure:

Approach 2:

Step 1: Log into the Azure portal

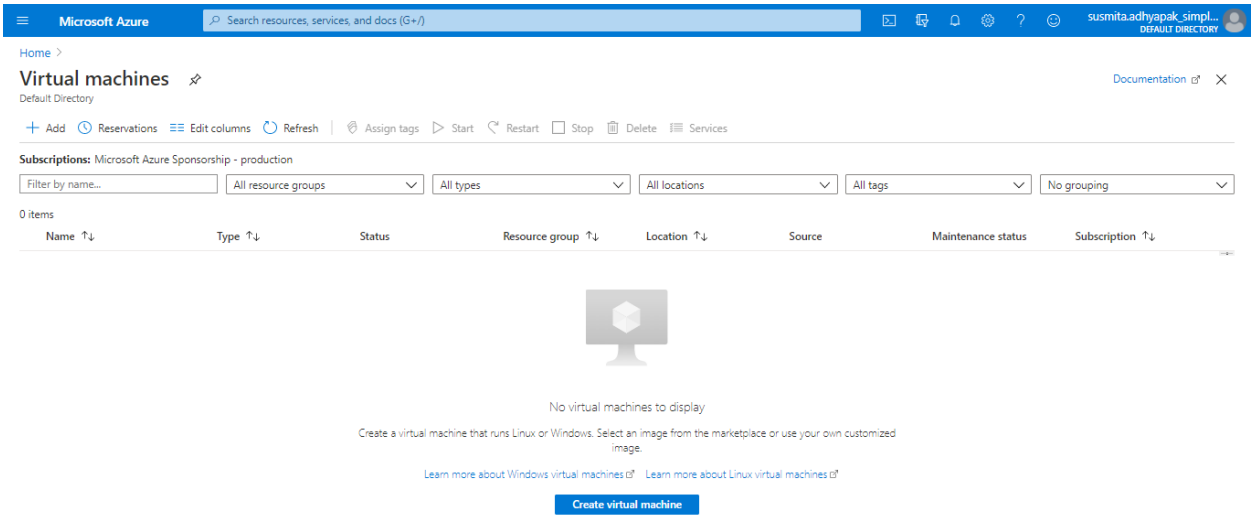


Step 2: Before creating the resources, make sure you apply tags to resources so that

you can keep a track of billing later on.

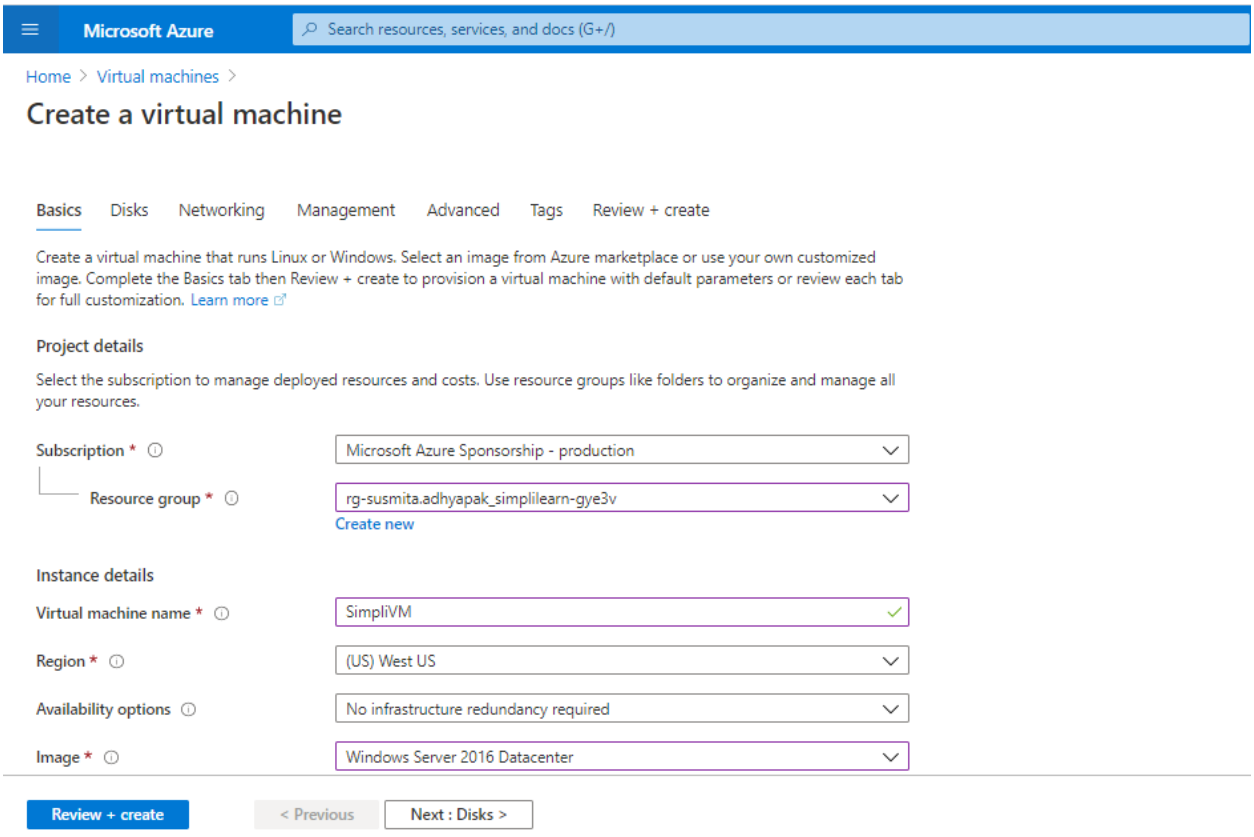
Step 3: To begin , create an Azure VM

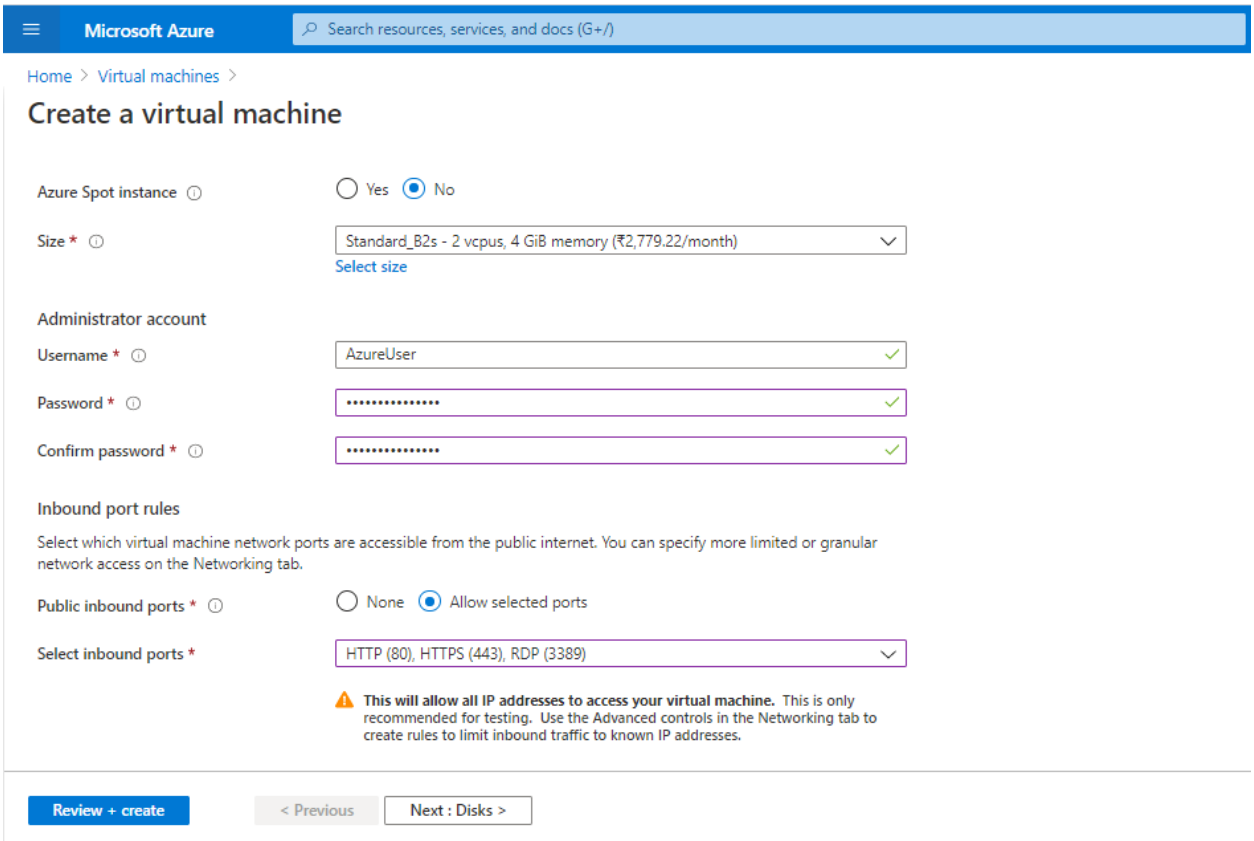
Step 3.1: Search for Virtual Machines and click on **Add**



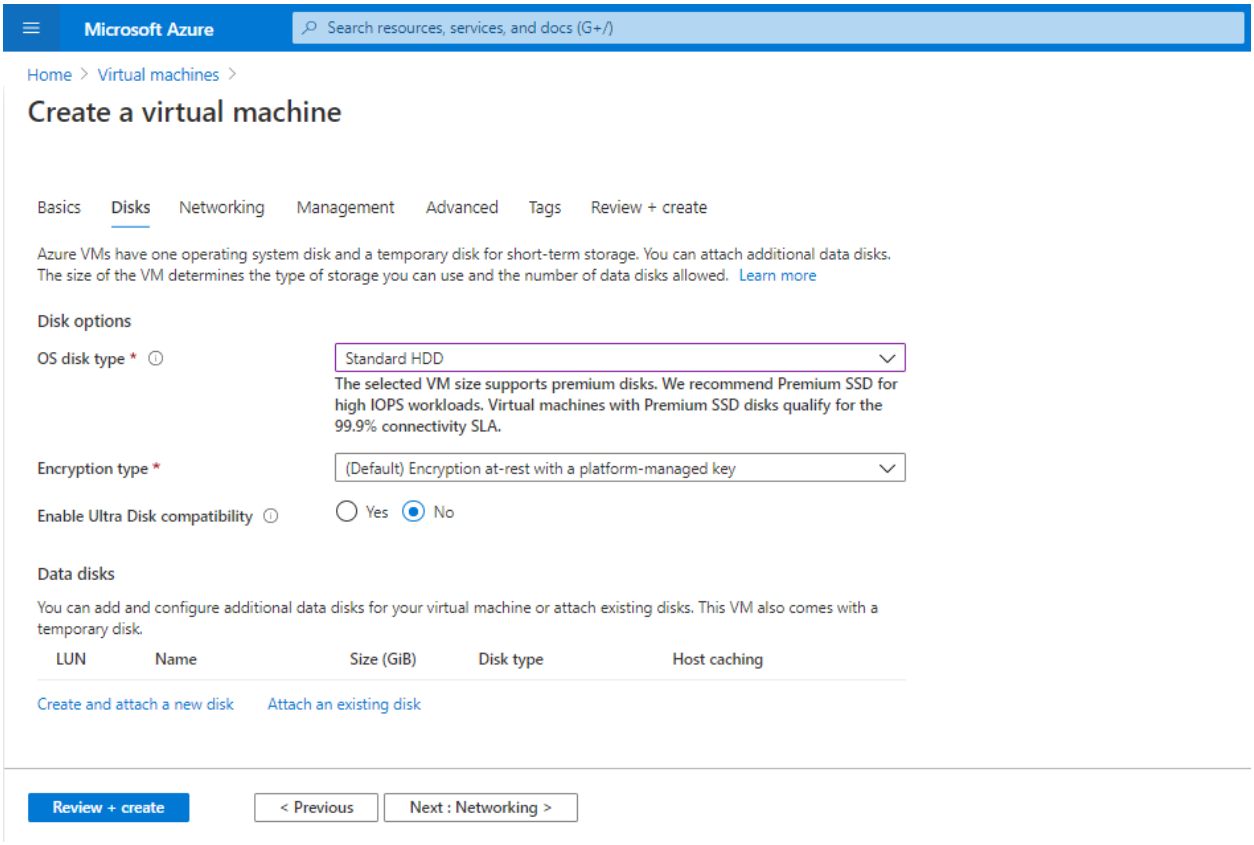
Step 3.2: Provide basic information about the VM

simplilearn

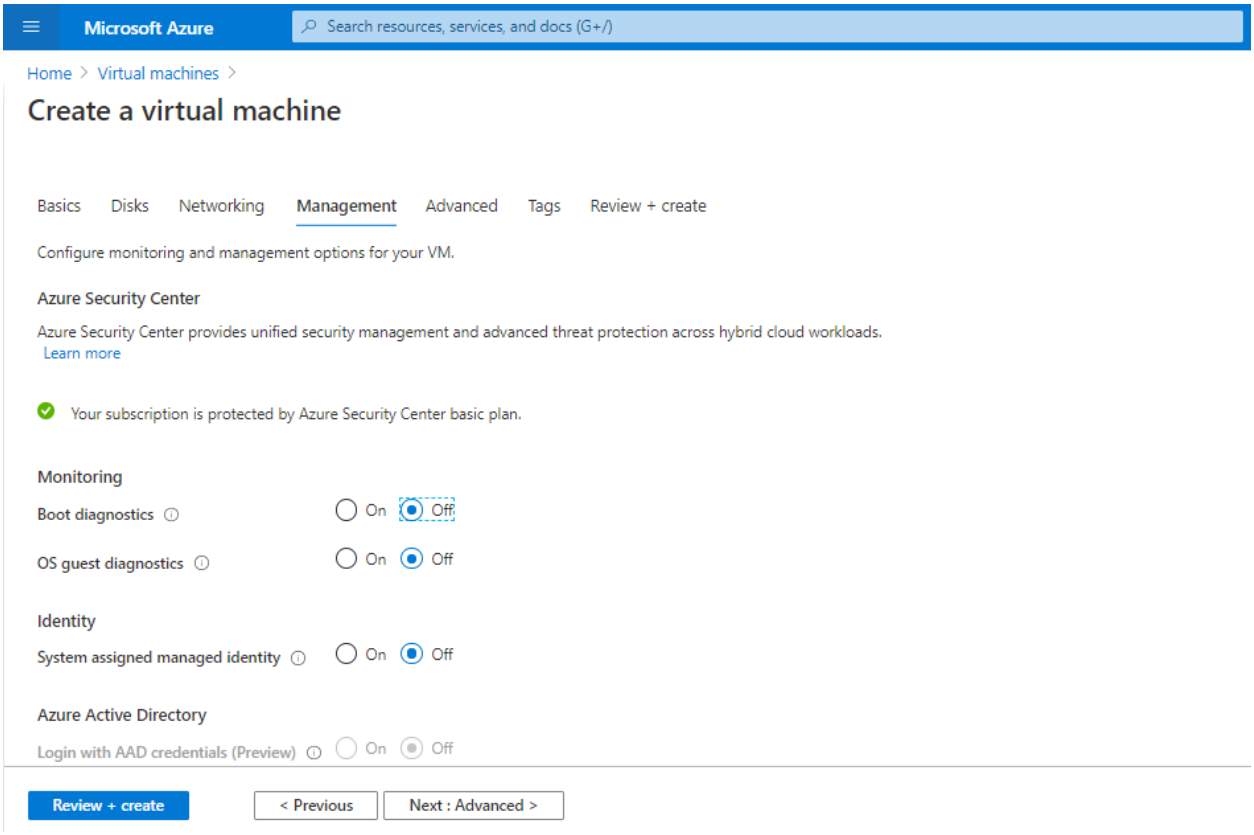




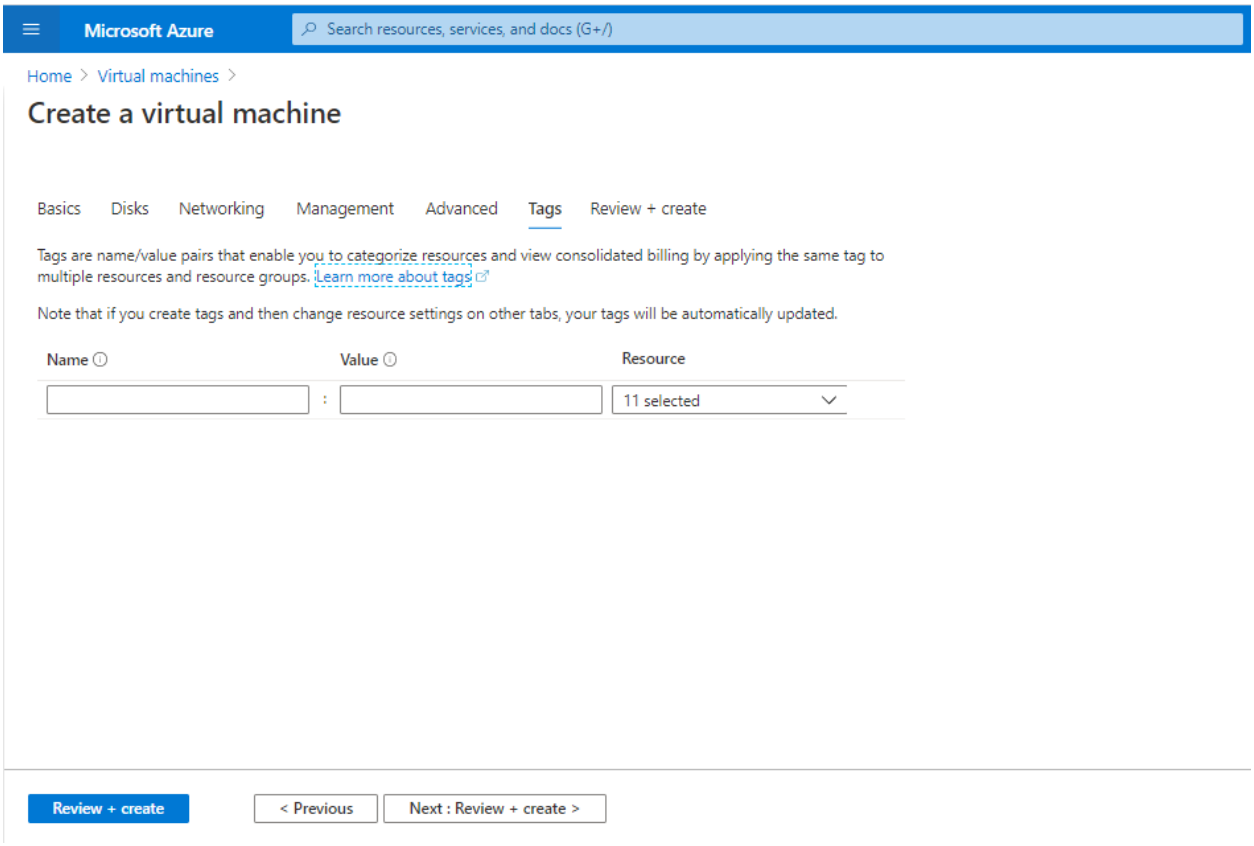
Step 3.3: In the Disks section, provide the required information



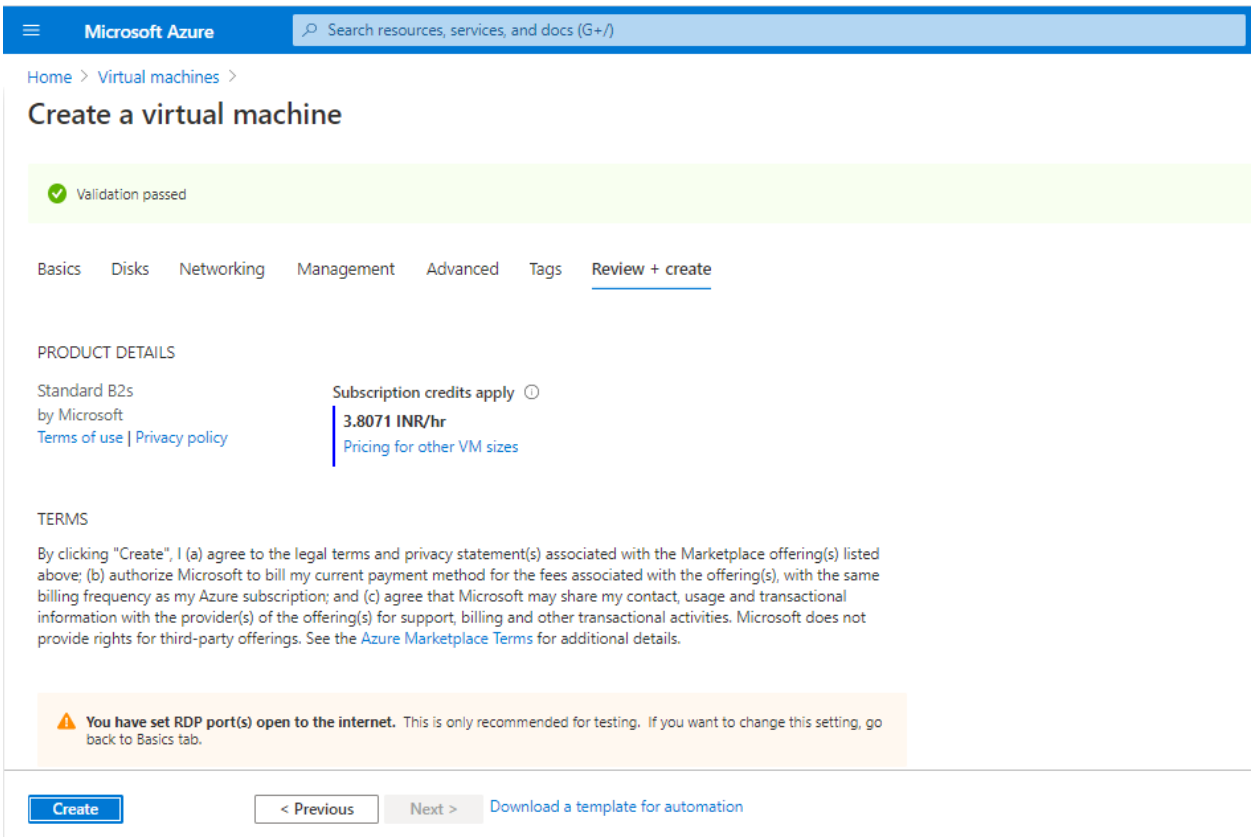
Step 3.4: In the Management section, turn off the Boot diagnostics



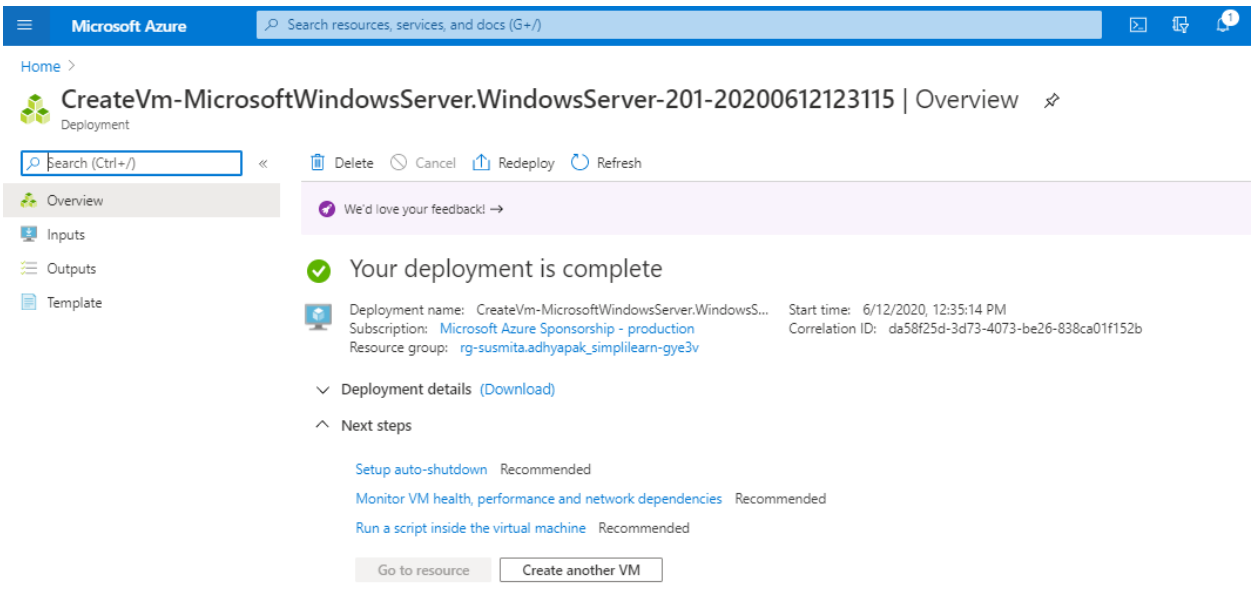
Step 3.4: Click on **Review and Create**



Step 3.5: Click on **Create**



Step 3.6: Your VM will get deployed.

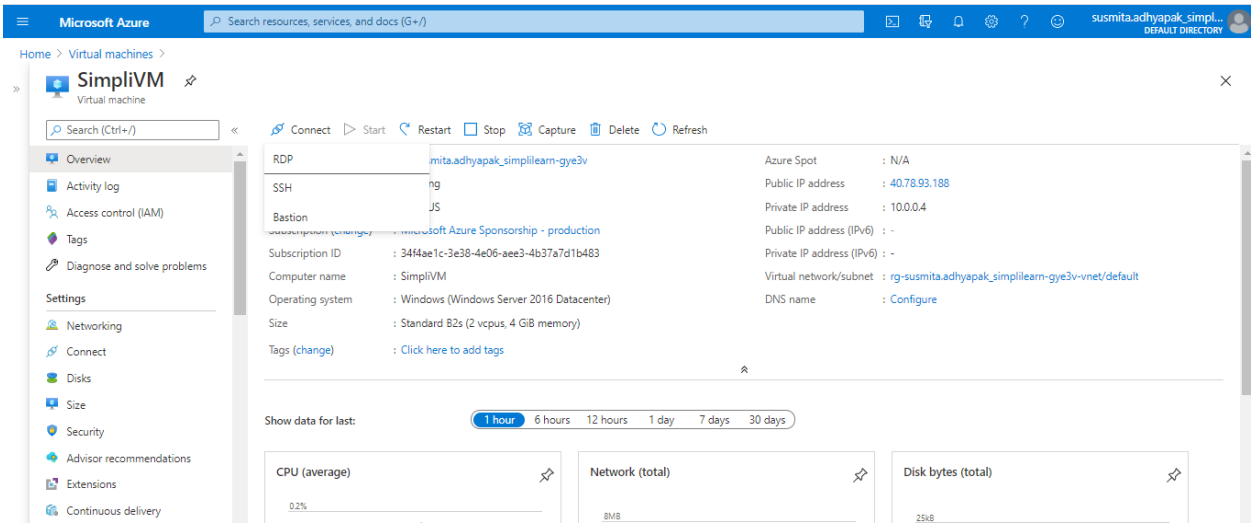


Step 4: Make sure you have inbound traffic on port 80 and port 443 open.

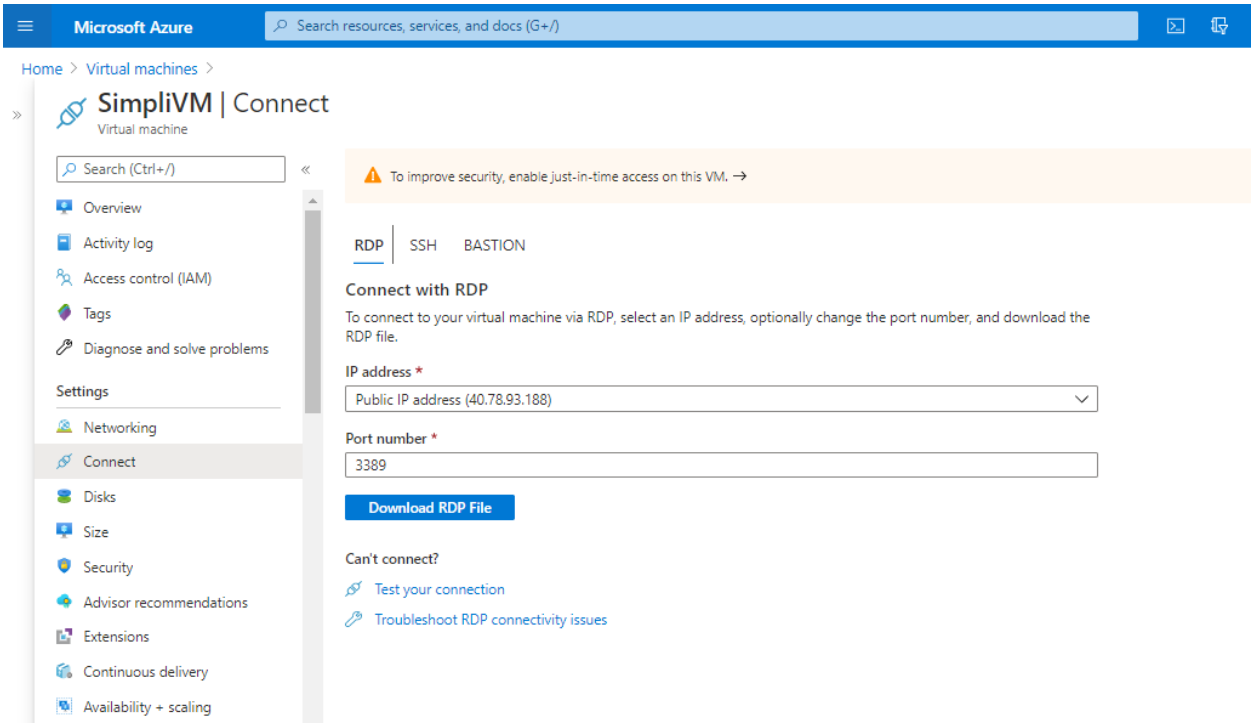
Step 5: Make sure port 445 is open on your VMs so that teammates can use common file share to access and share files if needed.

Step 6: Log into Azure VM and spin up a web server of your choice on port 80

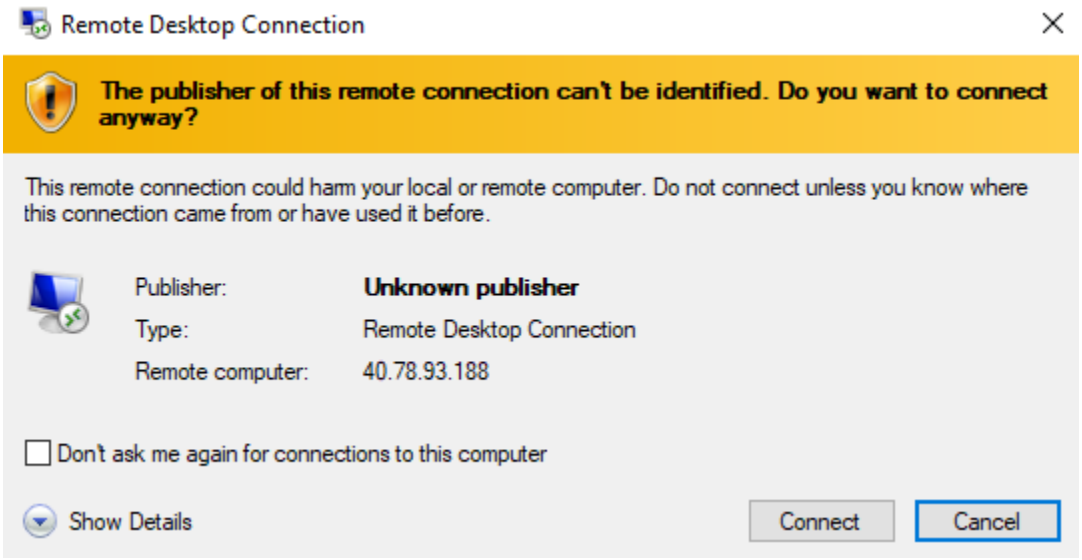
Step 6.1: Click on **Connect** and select **RDP**



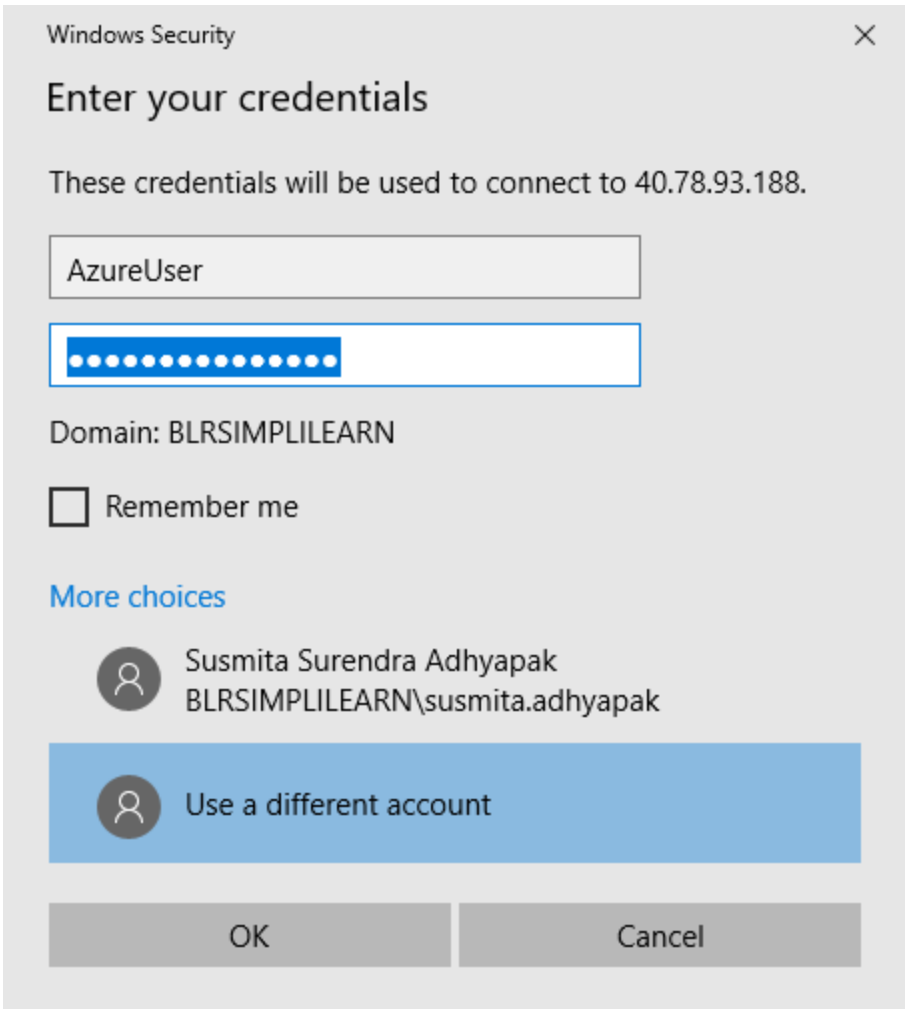
Step 6.2: Click on **Download RDP file.** It will download the created VM in your system.



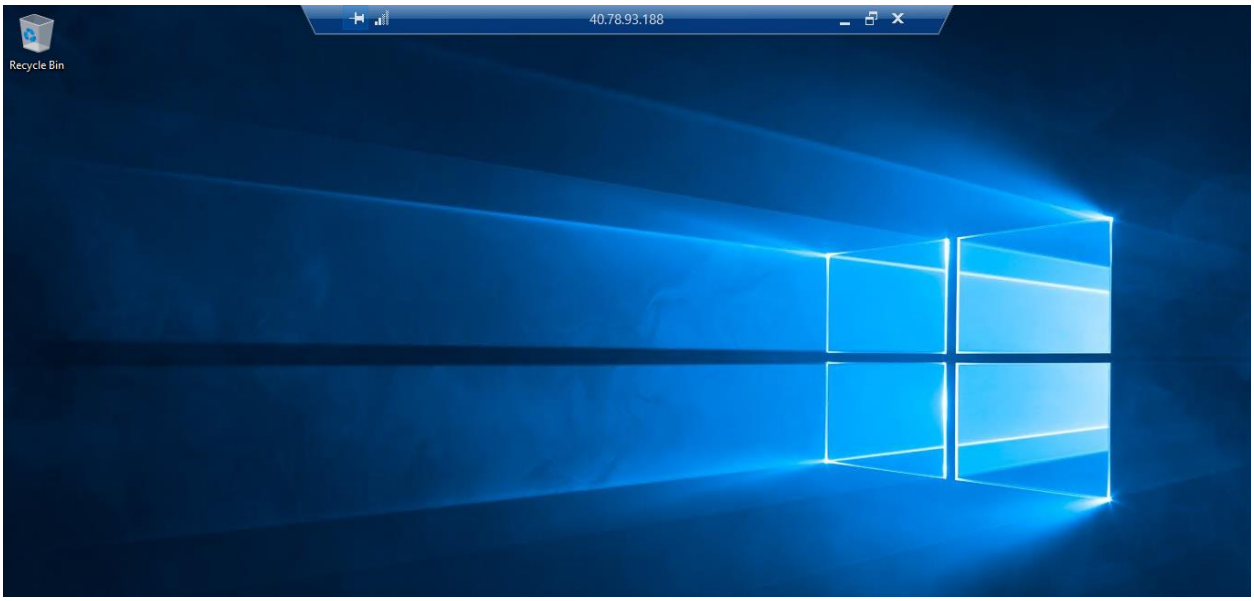
Step 6.3: Open the downloaded VM. Click on **Connect**



Step 6.4: Enter the credentials to login into VM and click on **OK**

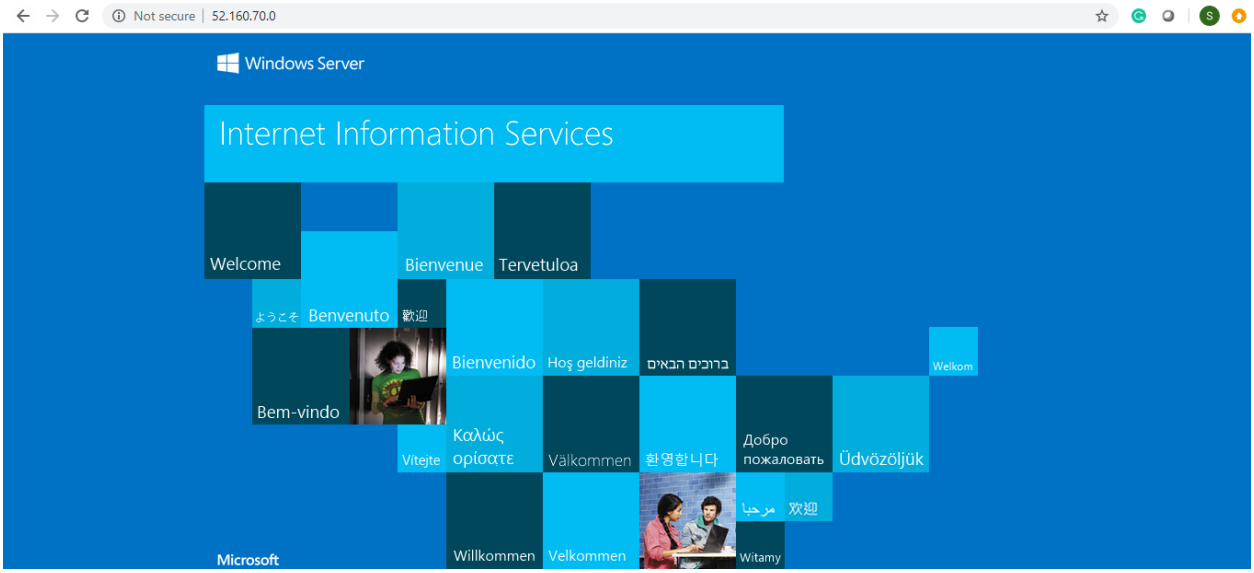


Step 6.5: This will open the VM in your system.



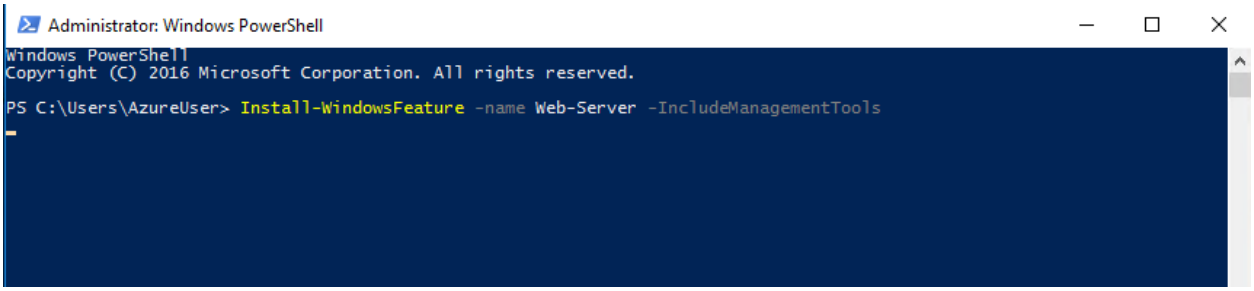
Step 7: Deploy your application on the web server that you have created within the virtual machine.

Step 7.1: In the portal, select the VM and in the overview of the VM, use the Click to copy button to the right of the IP address to copy it and paste it into a browser tab. The default IIS welcome page will open, and should look like this:

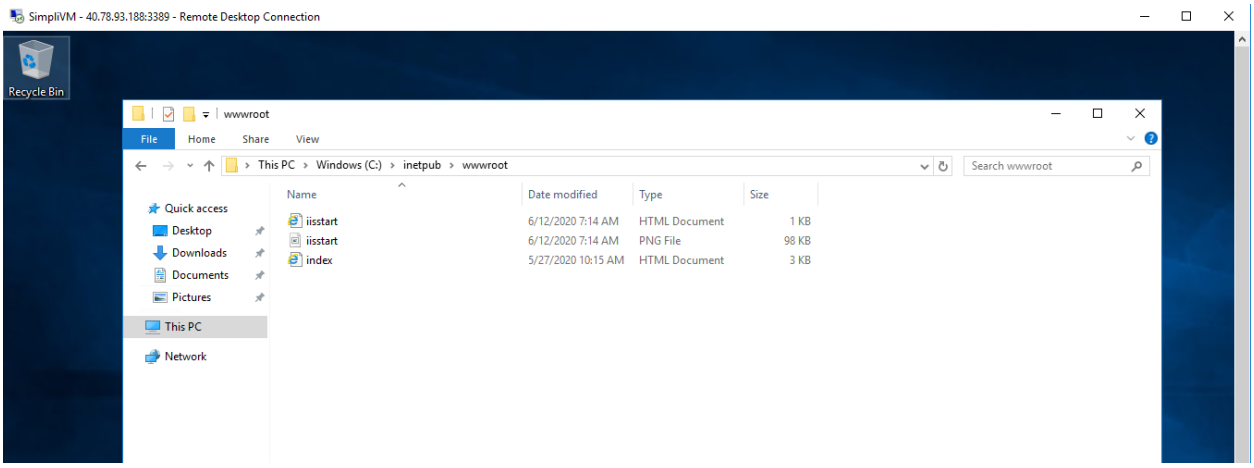


Step 7.2: Open the powershell and type the following command:

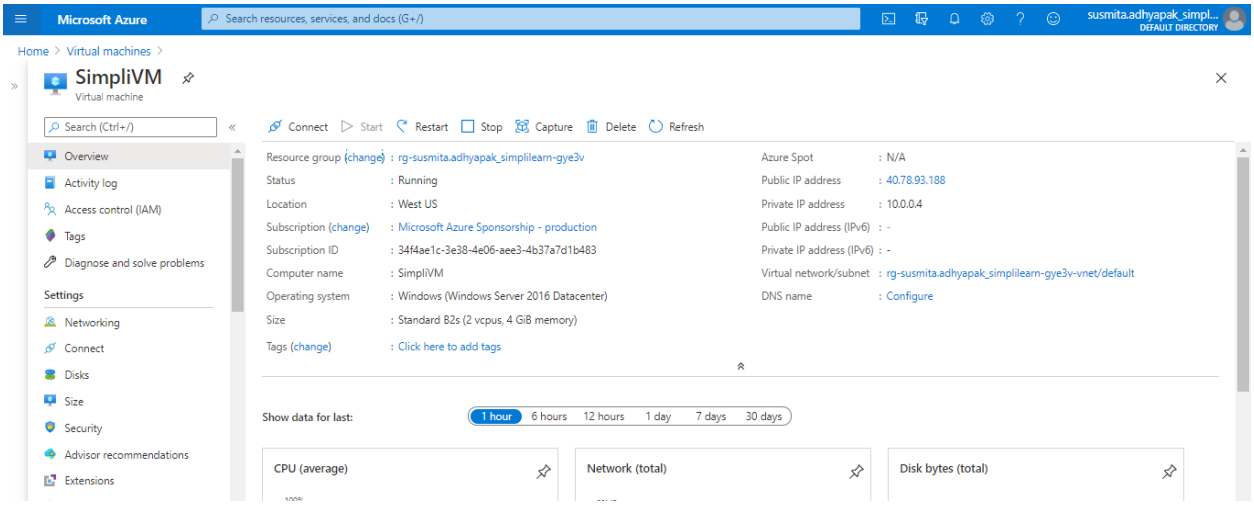
Install-WindowsFeature -name Web-Server -IncludeManagementTools

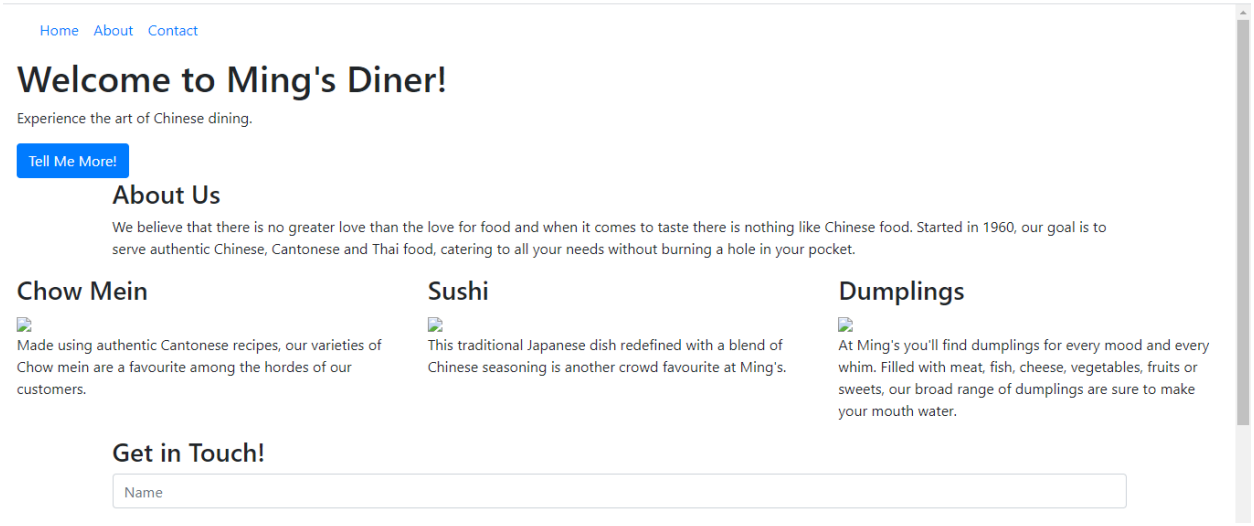


Step 7.3: The above command will create an inetpub folder in your C drive. Go to C:\inetpub\wwwroot and copy-paste the index.html file of your application which you want to deploy here:



Step 7.4: Copy and paste the public IP address of the created VM in the browser and you will get the output



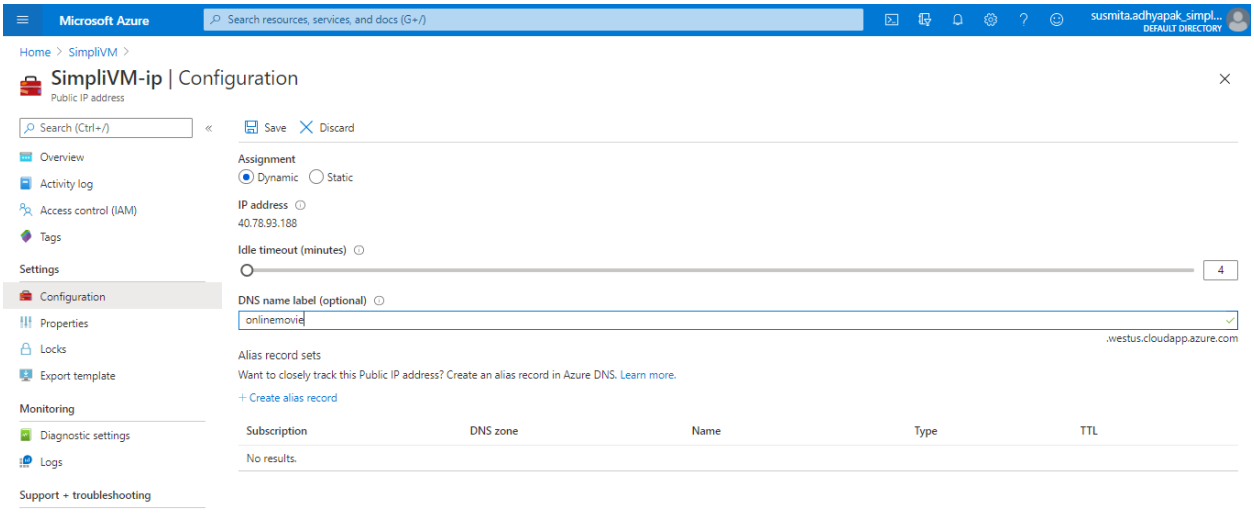


Step 8: Repeat steps 2 to 6 to create multiple deployments of your application in different regions so that you can meet the global traffic demand

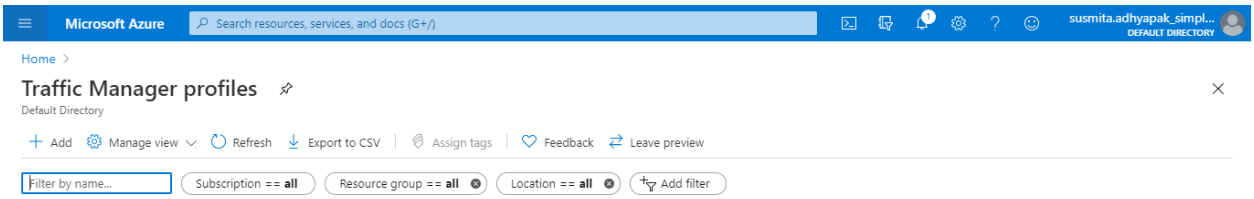
Step 9: To make sure that traffic coming from different parts of the world is load balanced at DNS level, create a Traffic Manager Profile

Step 9.1: Click on the Public IP address of the created VM

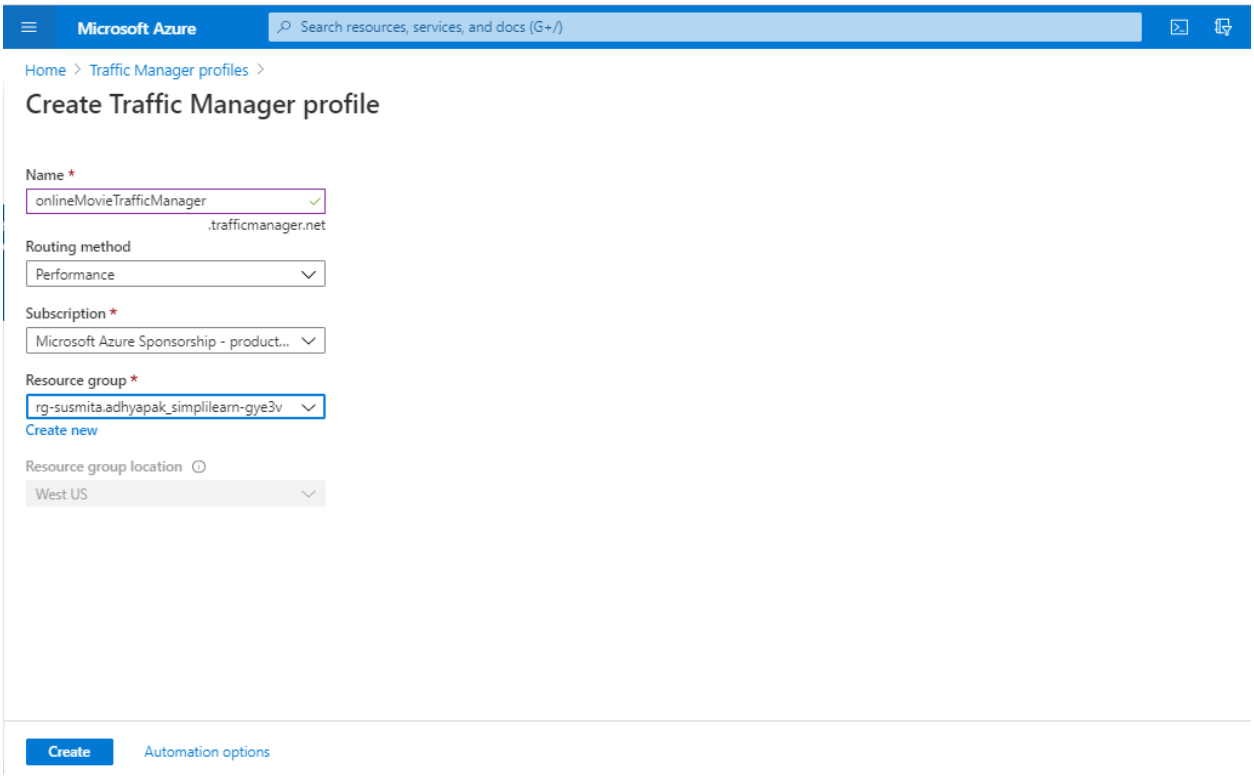
Step 9.2: Enter the DNS name and click on **Save**



Step 9.3: In the search window, search for Traffic Manager Profile. Click on **Add** to create a new traffic manager profile

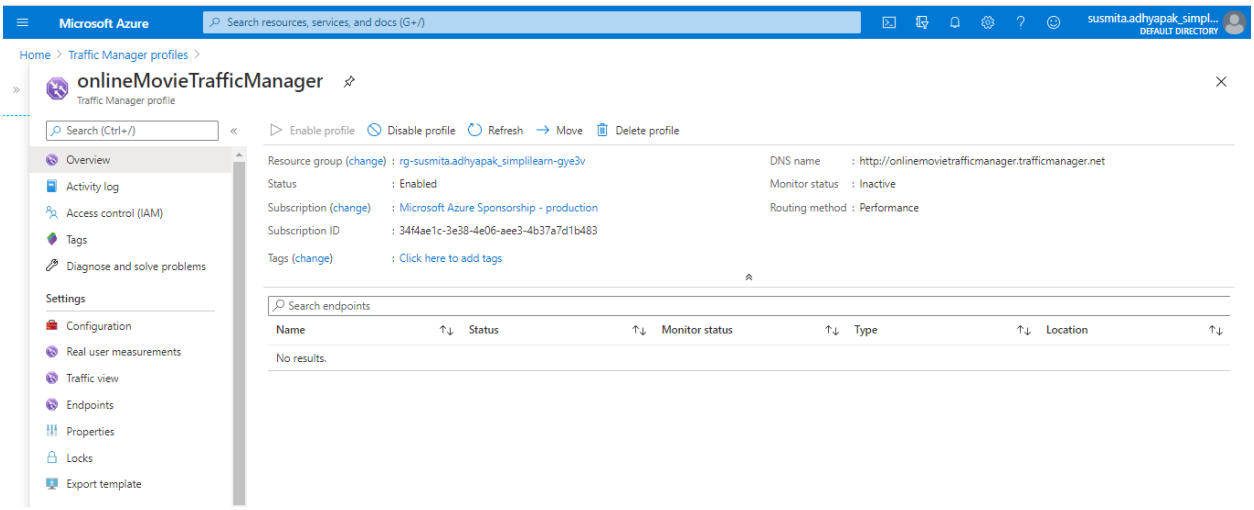


Step 9.4: Provide the required information and click on **Create** to create the Traffic Manager Profile

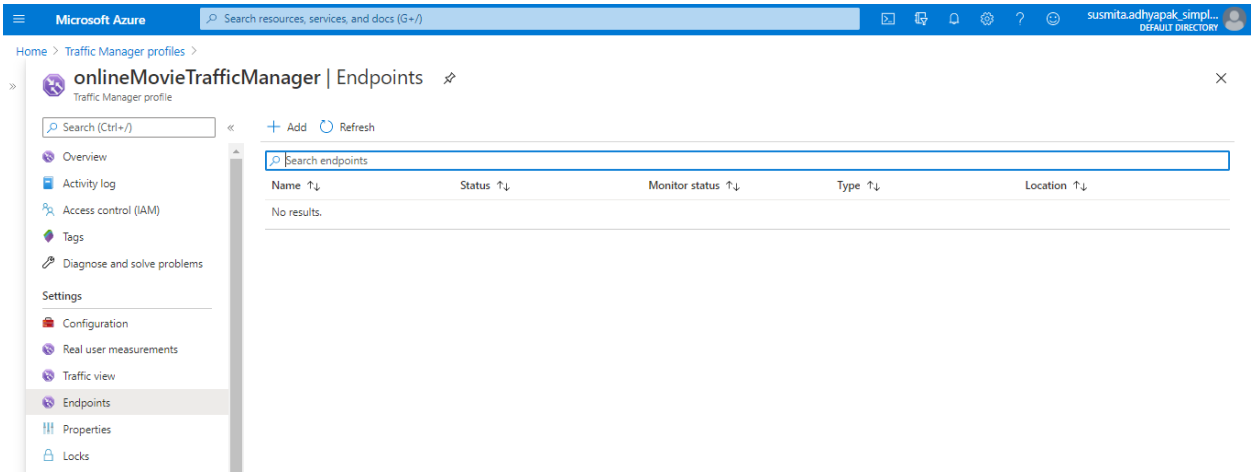


Step 10: Create endpoints in the traffic manager corresponding to the public IP of each virtual machine that you have created.

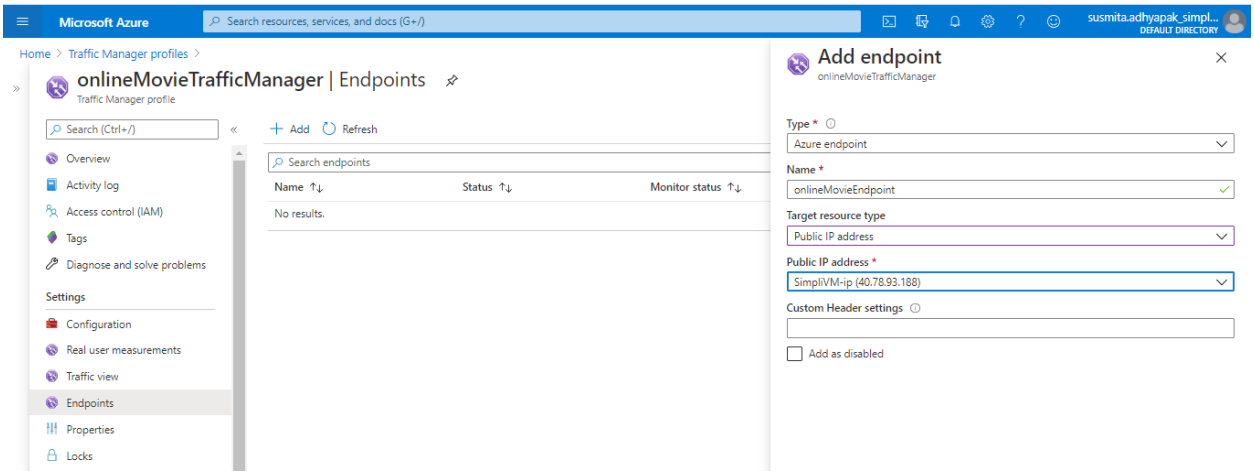
Step 10.1: Go to the created Traffic Manager Profile



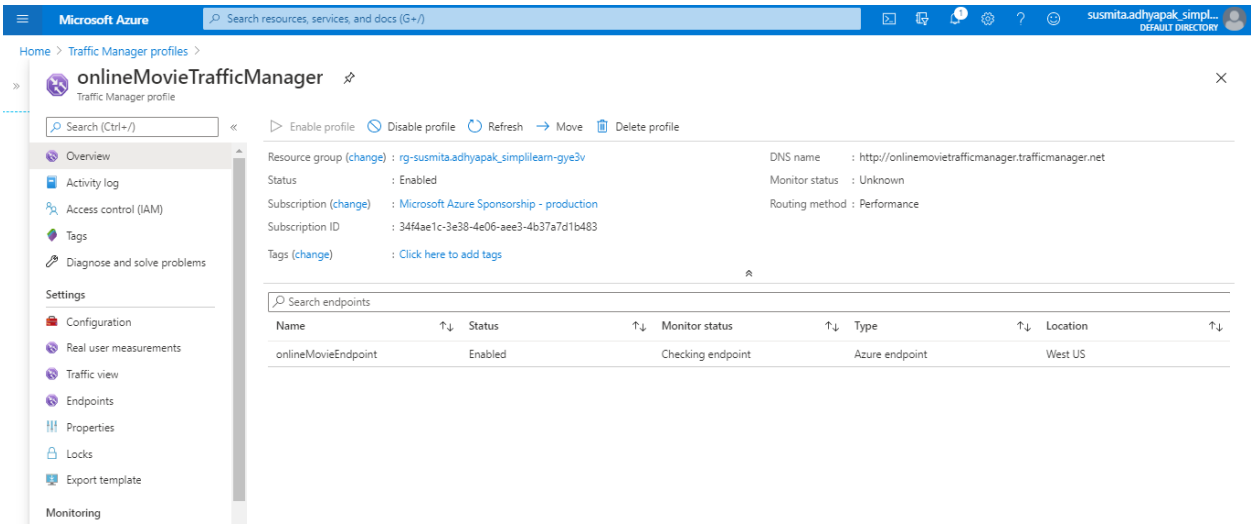
Step 10.2: Click on **Endpoints** and click on **Add** to add new endpoints

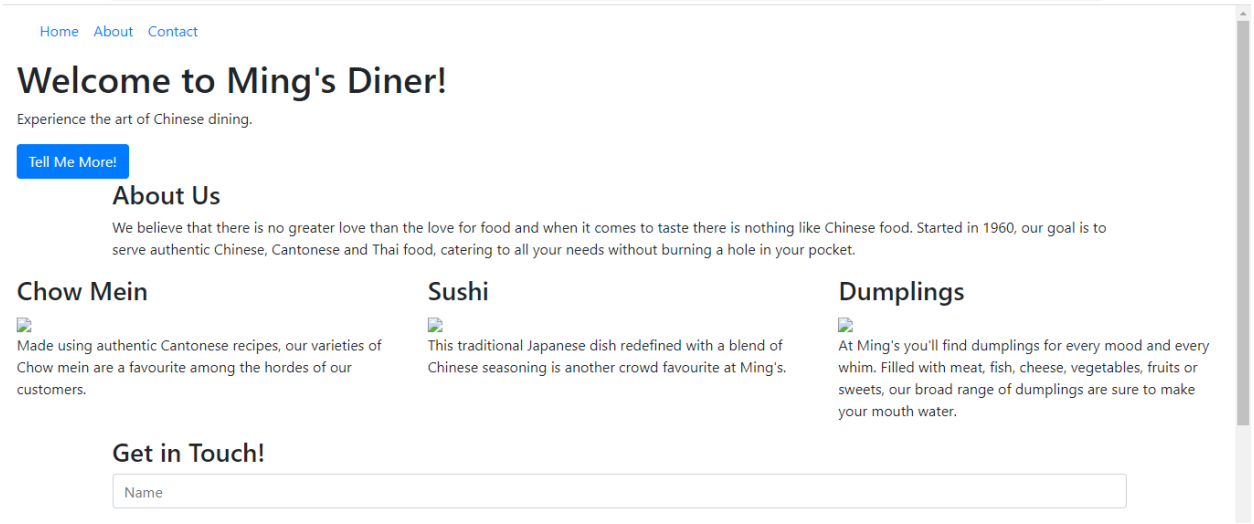


Step 10.3: Provide the required information and click on **Add**



Step 10.4: Copy and paste the DNS name link in the web browser and you'll get the output





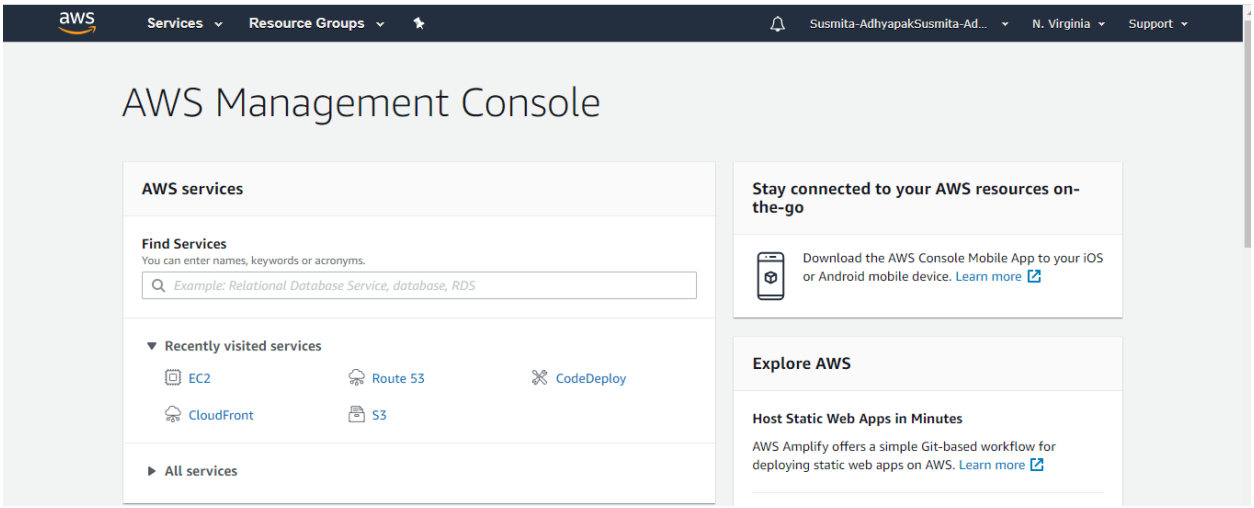
Step 11: Optionally, if you want to add the application in your own domain, you can configure the traffic manager to point to a custom domain.

Step 12: As good practice, follow the principle of least privilege so that you give access to the services that need to be accessed within the Azure portal

AWS:

Approach 1:

Step 1: Log into the AWS console

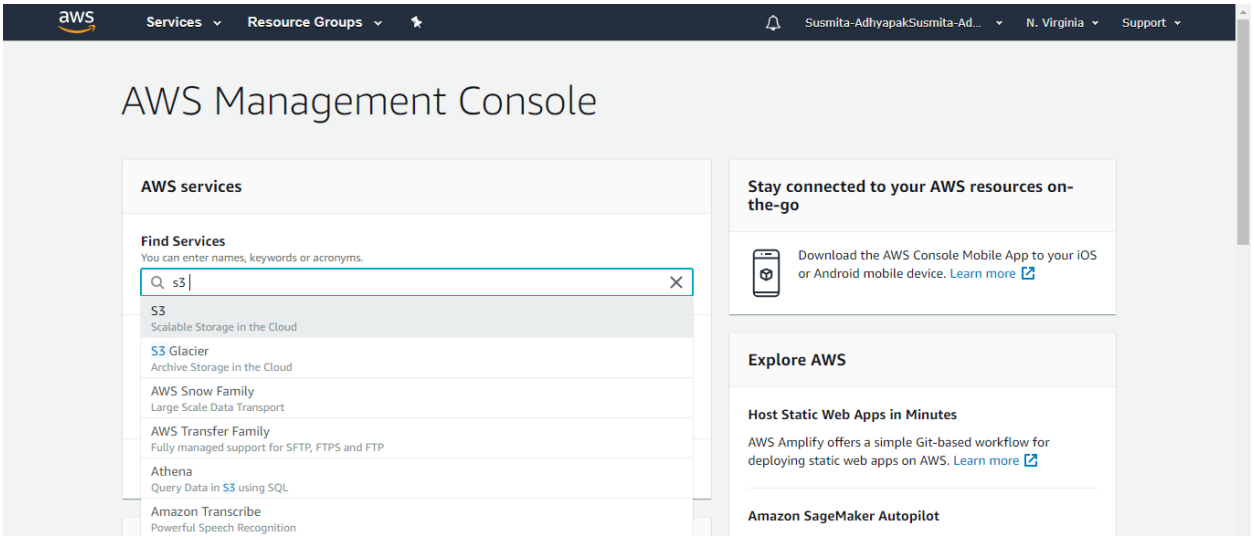


Step 2: Before creating the resources, make sure you apply cost allocation tags to resources so that you can keep a track of billing later on.

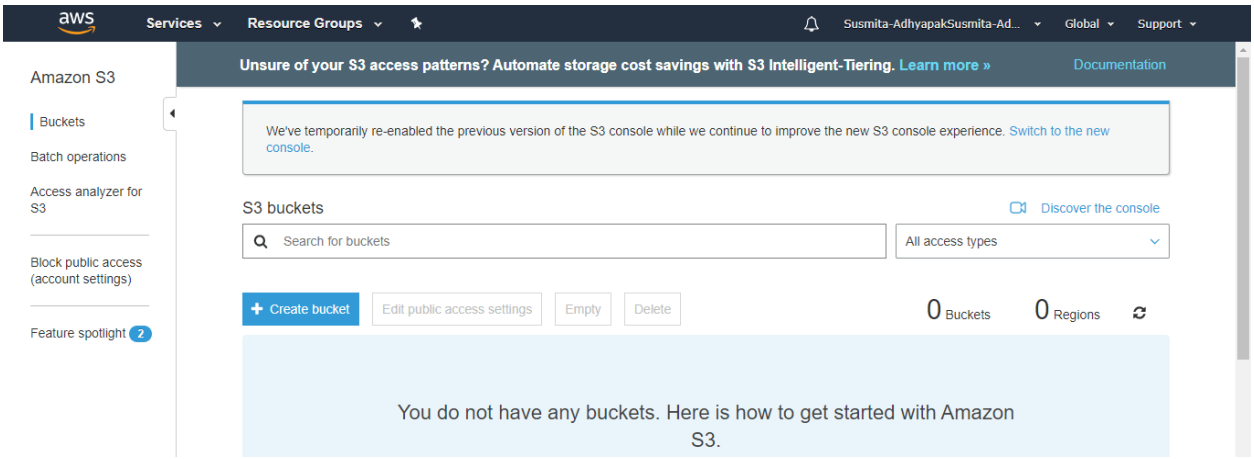
Step 3: To begin with, create Route 53 and add a hosted zone if you have your own domain. This is an optional step to configure a custom domain for your web app.

Step 4: Create an S3 bucket.

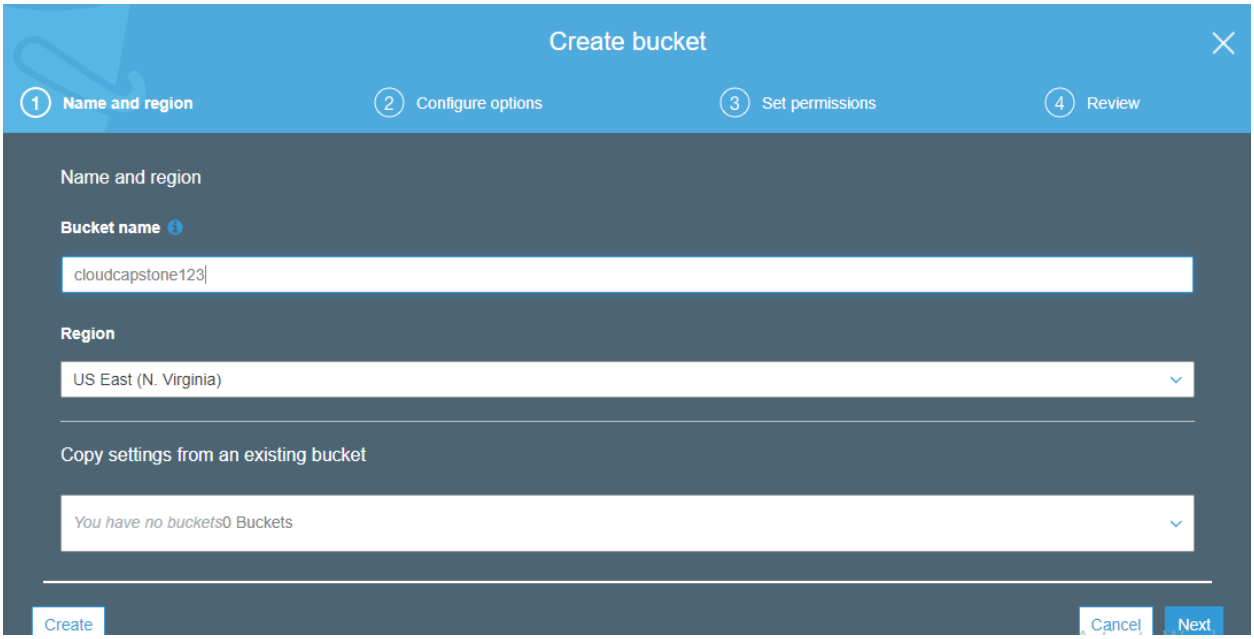
Step 4.1: In the search window, search for S3 service



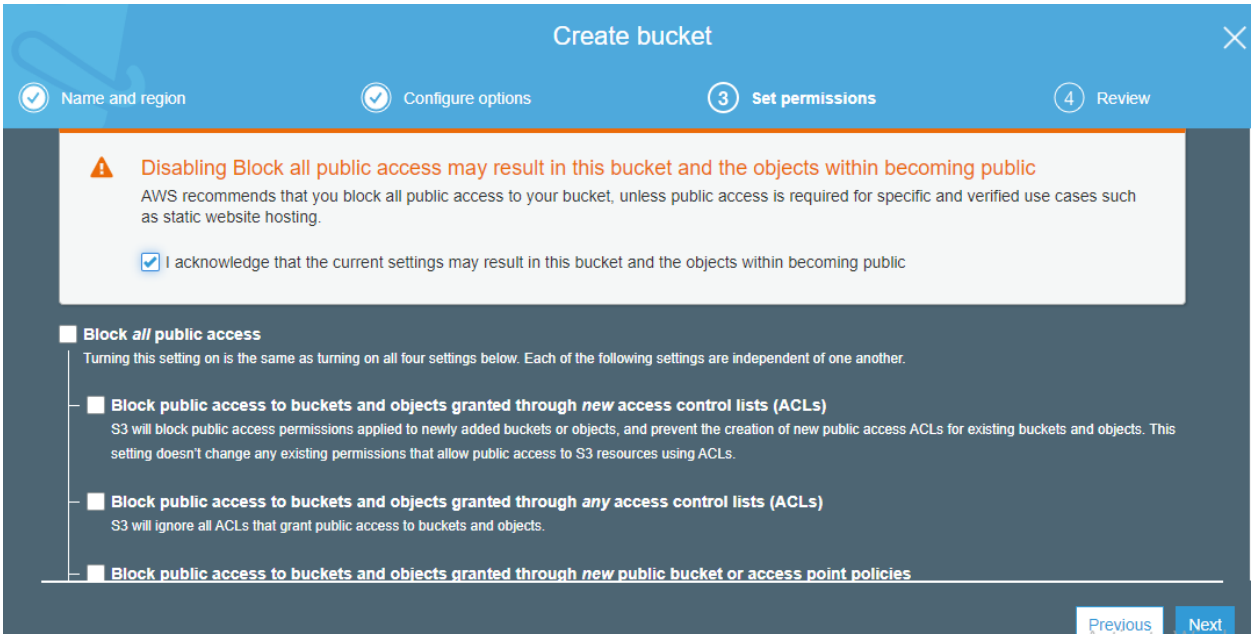
Step 4.2: Click on **Create bucket**



Step 4.3: Provide the bucket name, select the region, and click on **Next**

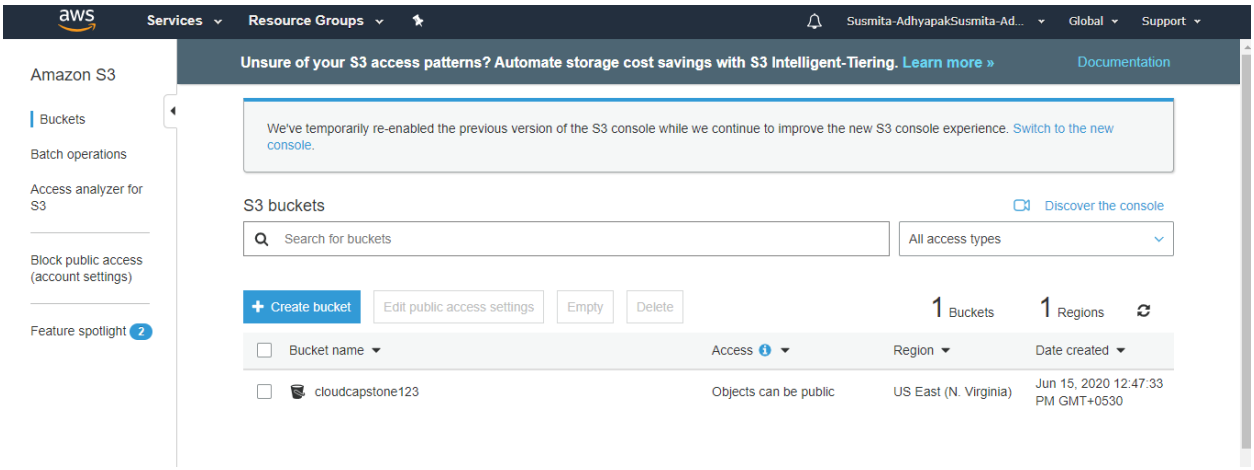


Step 4.4: In the Set permissions section, uncheck the box of Block all public access and acknowledge the terms and click on **Next**



Step 4.5: Click on **Create bucket**

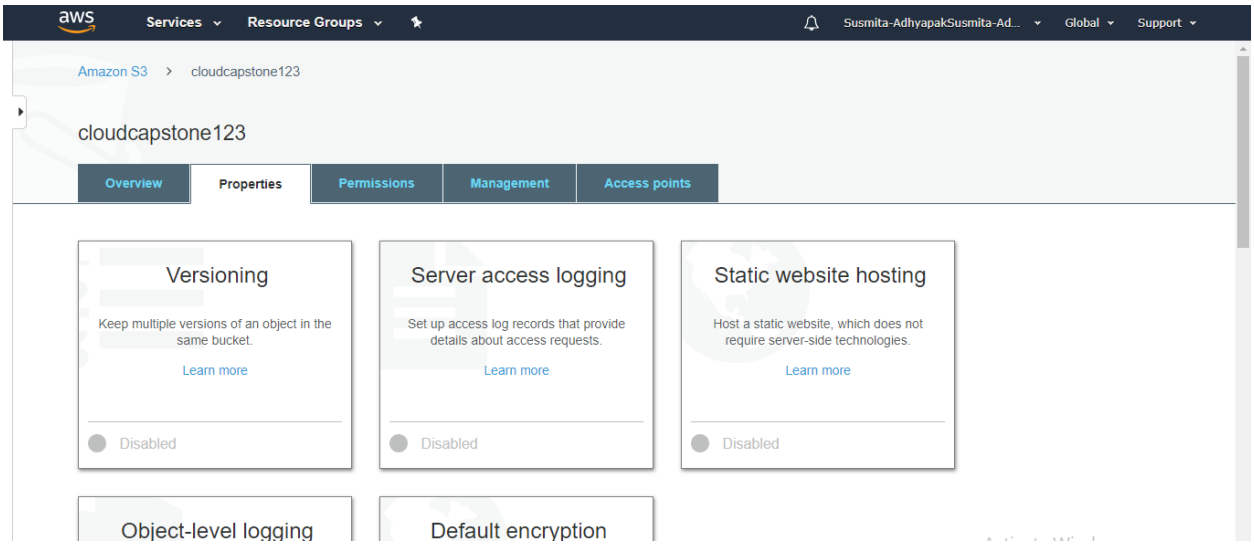
Step 4.6: The created bucket will be visible in the portal



Step 5: In the properties of S3 bucket, configure the S3 bucket to enable Static website hosting

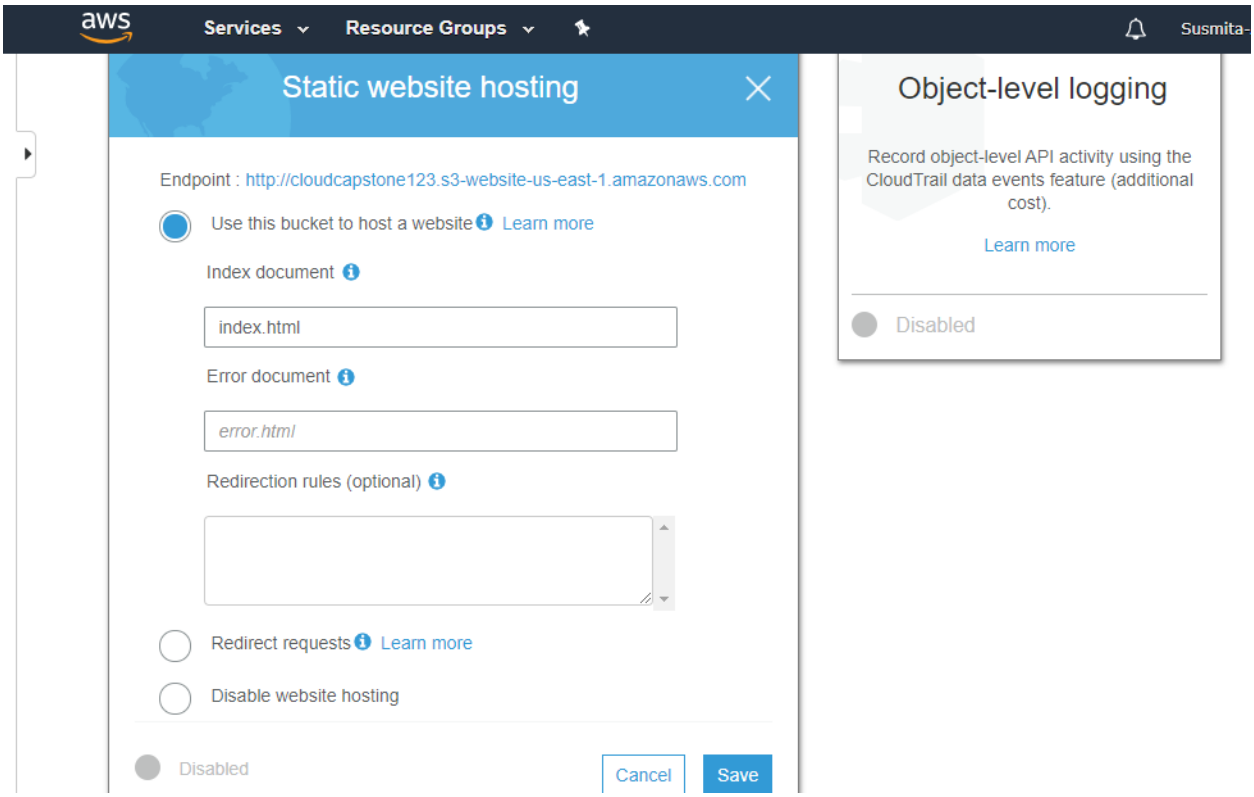
Step 5.1: Click on the created bucket

Step 5.2: Go to **Properties**



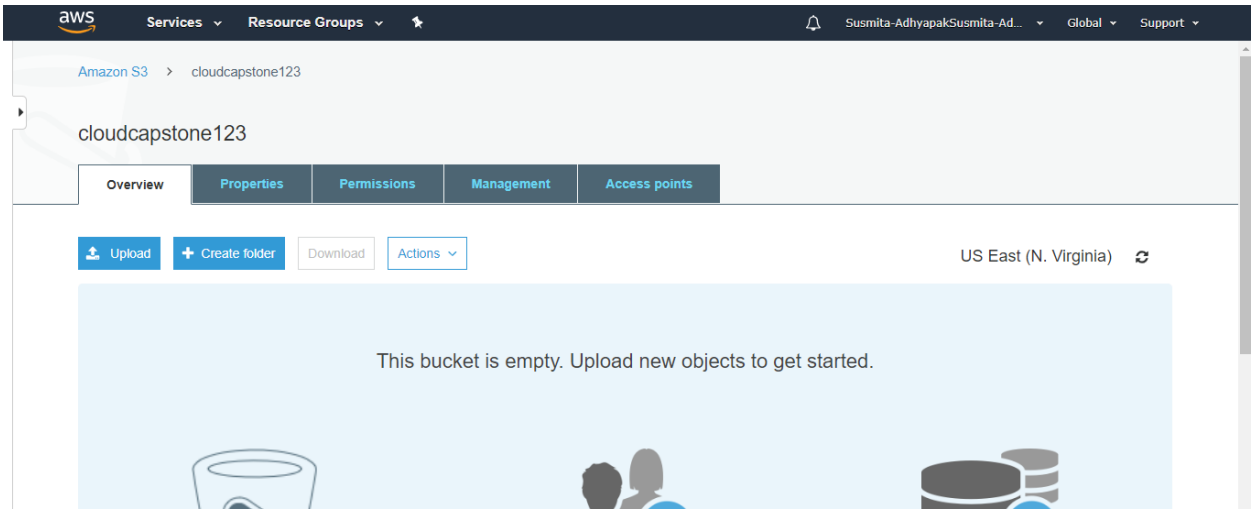
Step 5.3: Select Static web hosting

Step 5.4: Select **Use this bucket to host a website**, provide the required information and click on **Save**

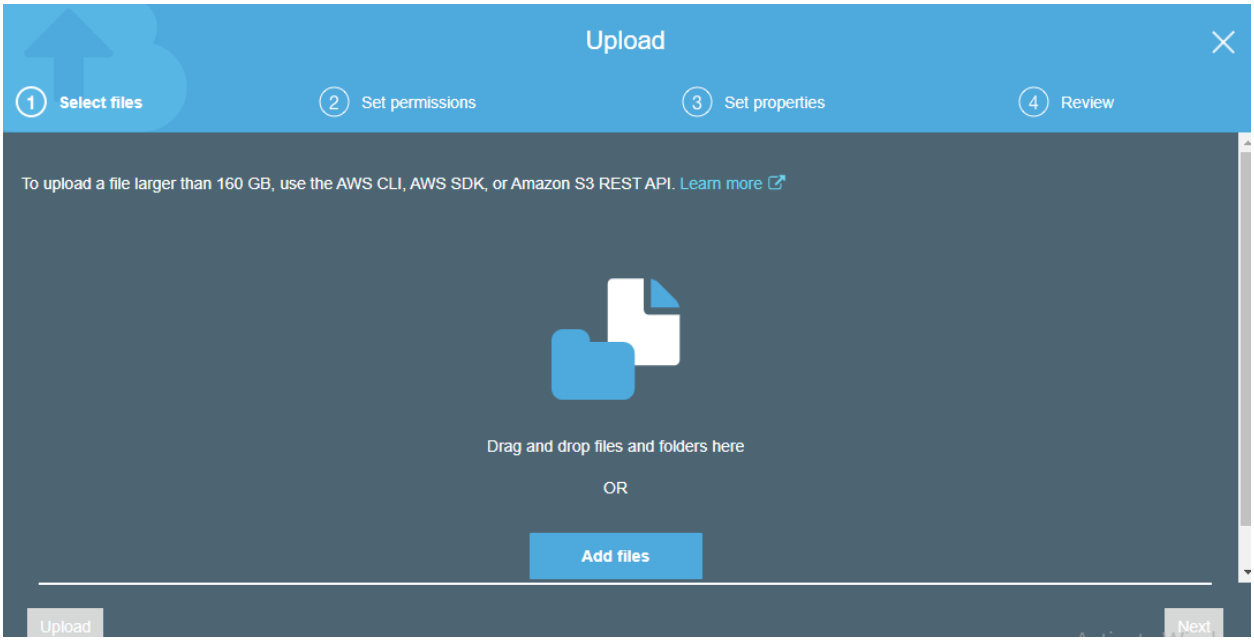


Step 6: Upload your static content (web app files) to the S3 bucket

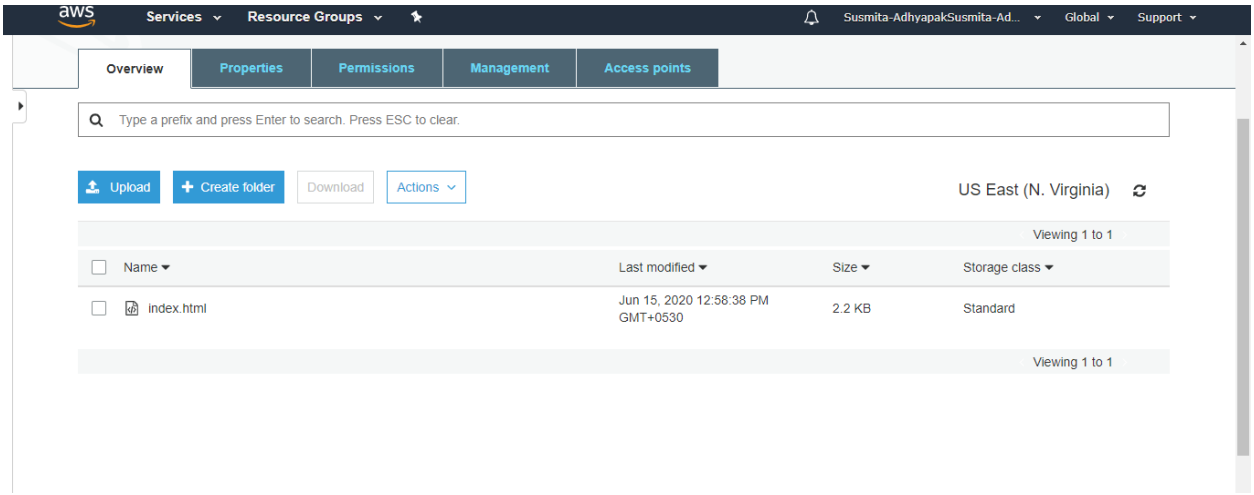
Step 6.1: Go to Overview tab of the created bucket and click on **Upload**



Step 6.2: Select the files of your application which you want to deploy and click on **Next** and click on **Upload**



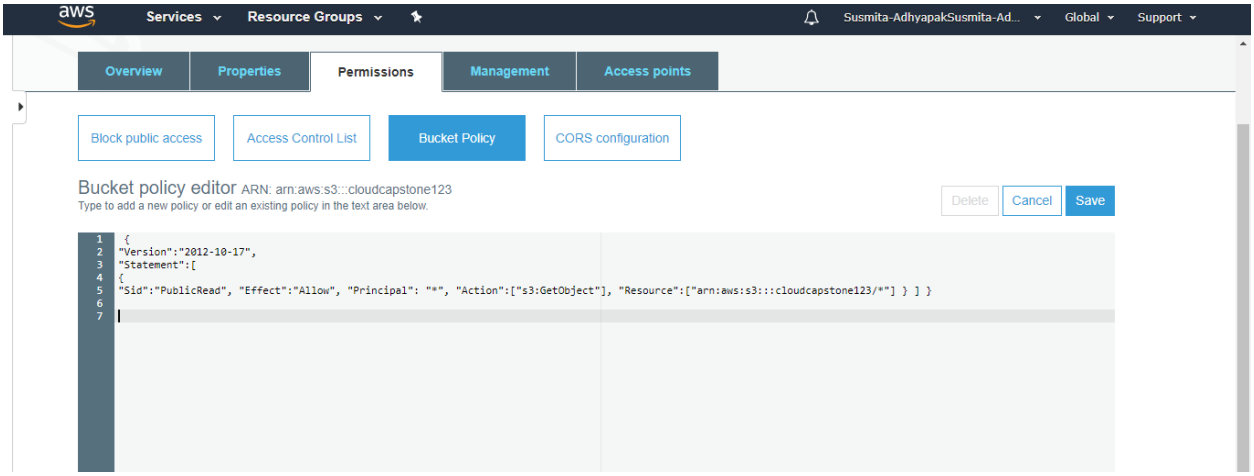
Step 6.3: This will add the files of your application in the S3 bucket



Step 7: Configure permissions in S3 and add the below bucket policy to give read only access to the static web app endpoint

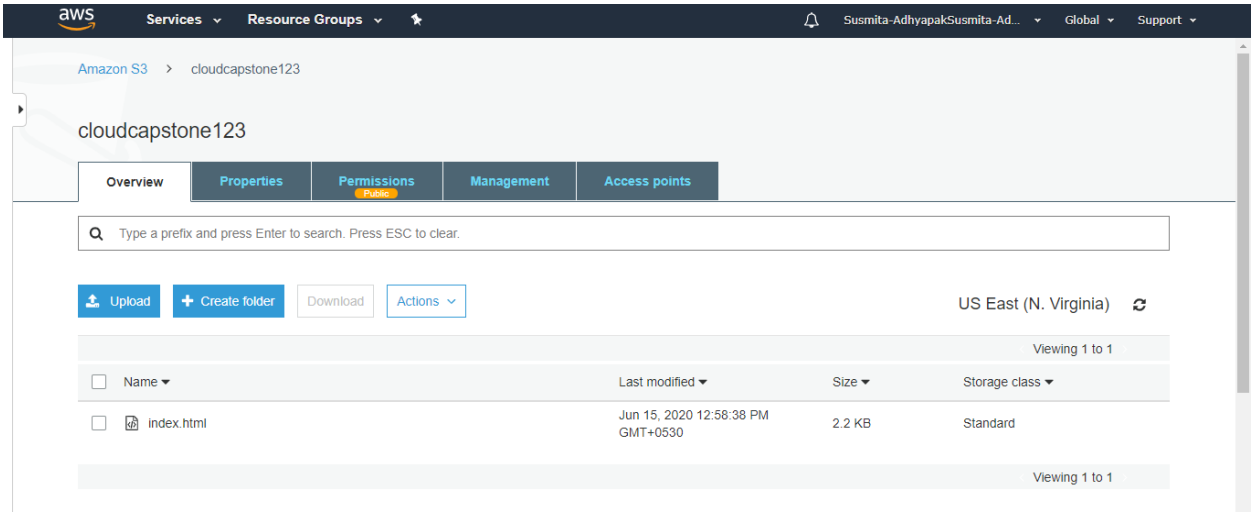
Step 7.1: Go to **Permissions**, click on **Bucket policy**, add the following code there and click on **Save**

```
{

    "Version":"2012-10-17",
       "Statement":[
          {
"Sid":"PublicRead", "Effect":"Allow", "Principal": "*",
"Action":["s3:GetObject"],
"Resource":["arn:aws:s3:::cloudcapstone123/*"] } ] }
```

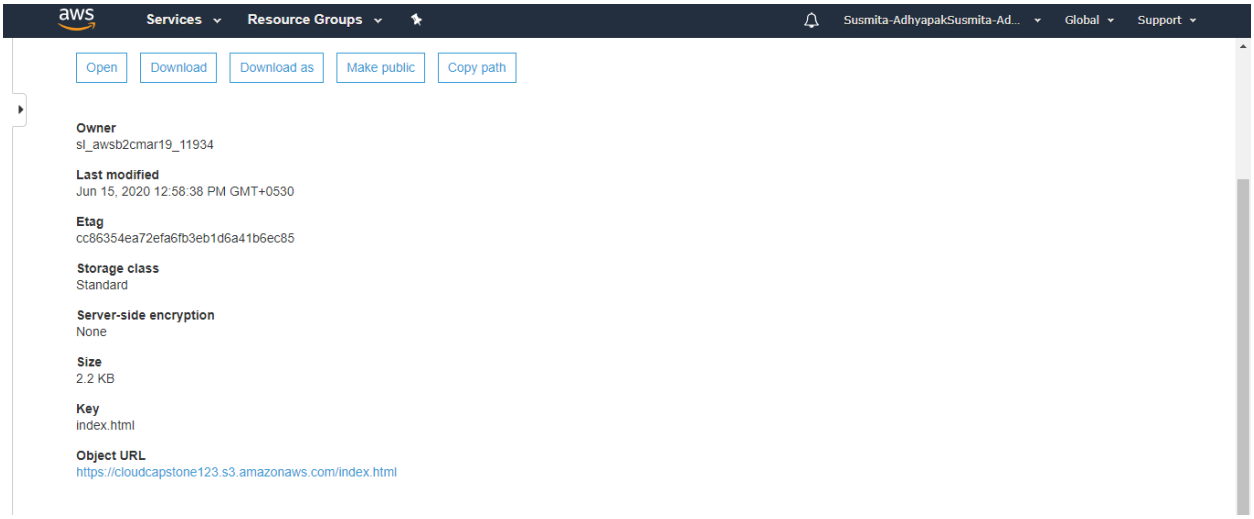


Step 8: Hit the web app endpoint to check if the application is online

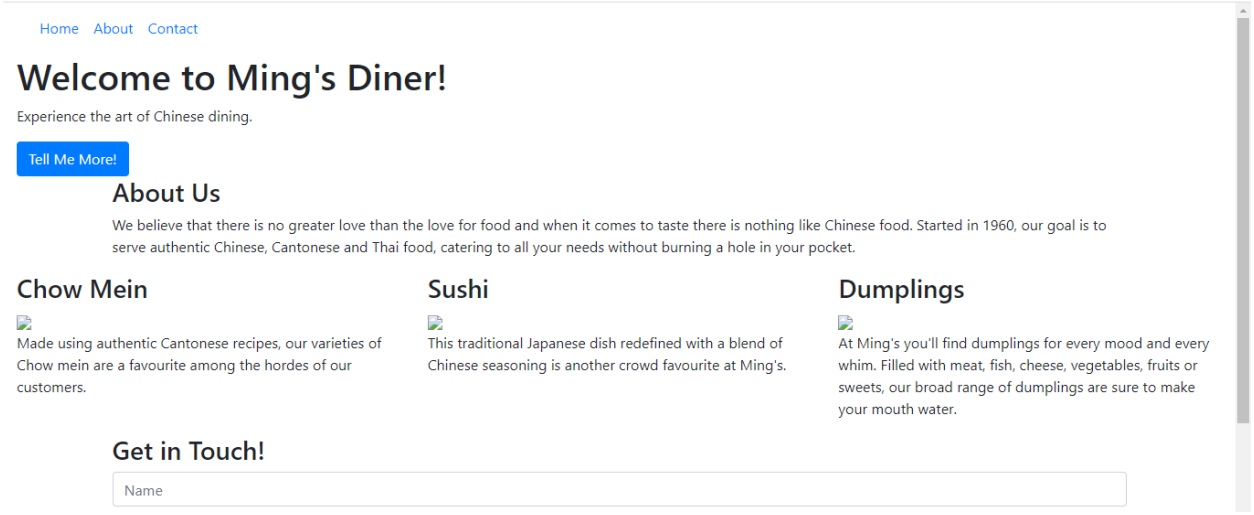
Step 8.1: Click on the Overview tab of the created bucket and click on the uploaded file



Step 8.2: Click on the **Object URL**

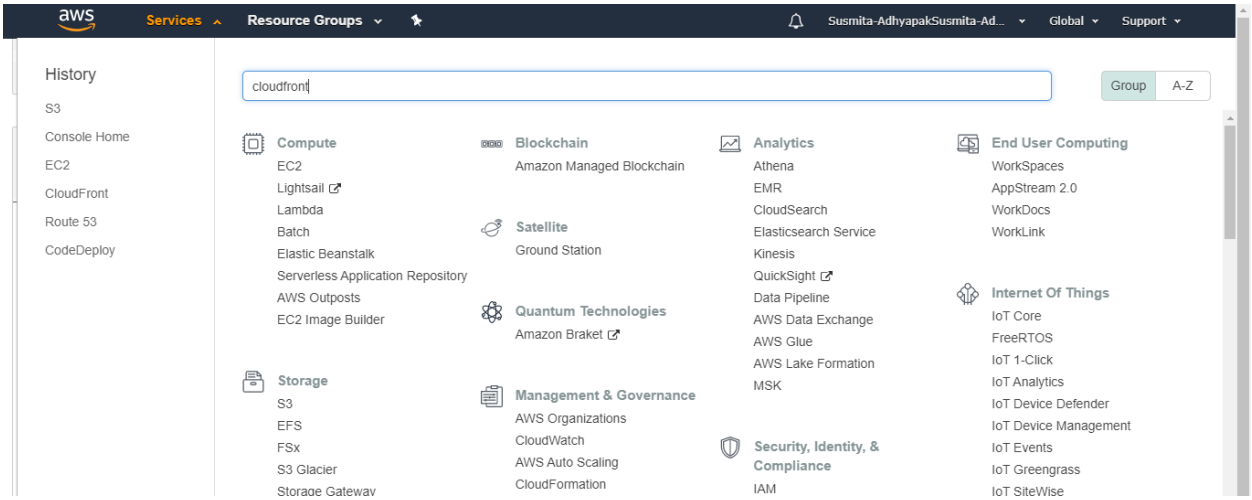


Step 8.3: Your application will start running.

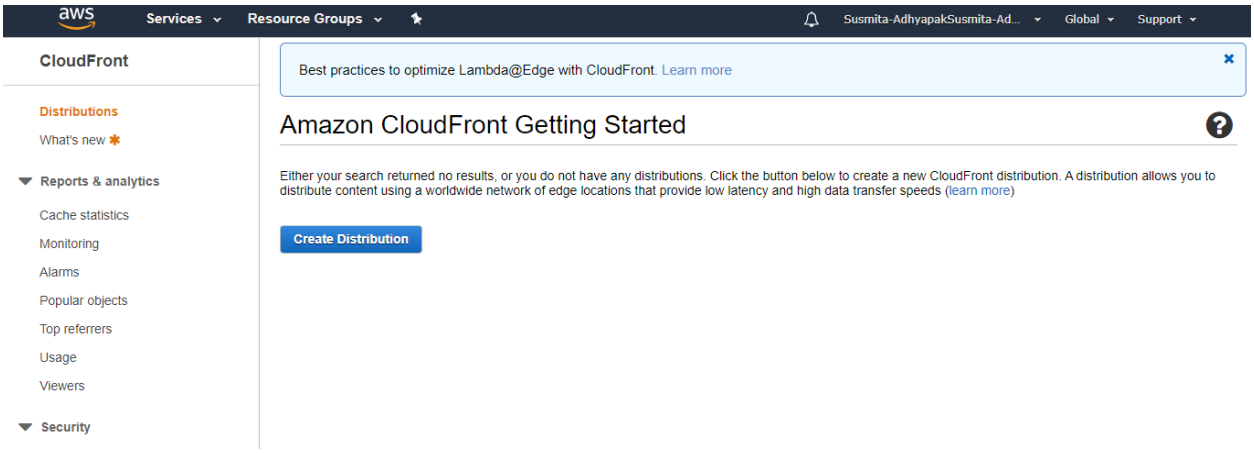


Step 9: Now create a CloudFront distribution corresponding to the static web app endpoint

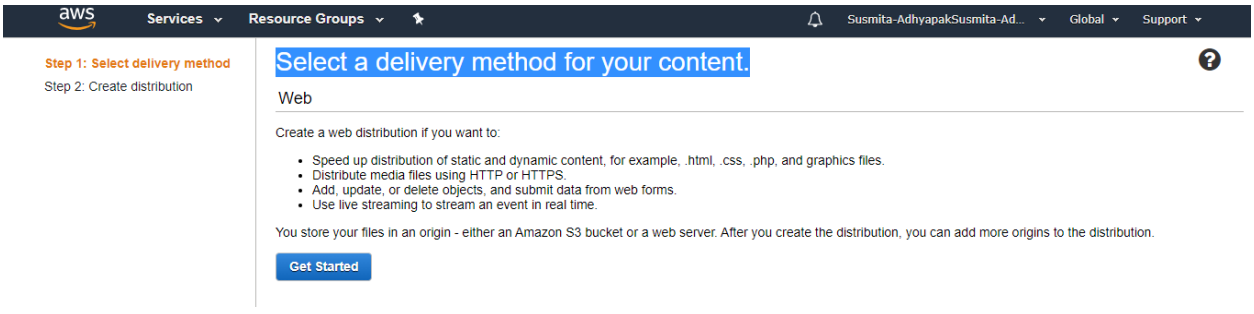
Step 9.1: Go to services and search for CloudFront



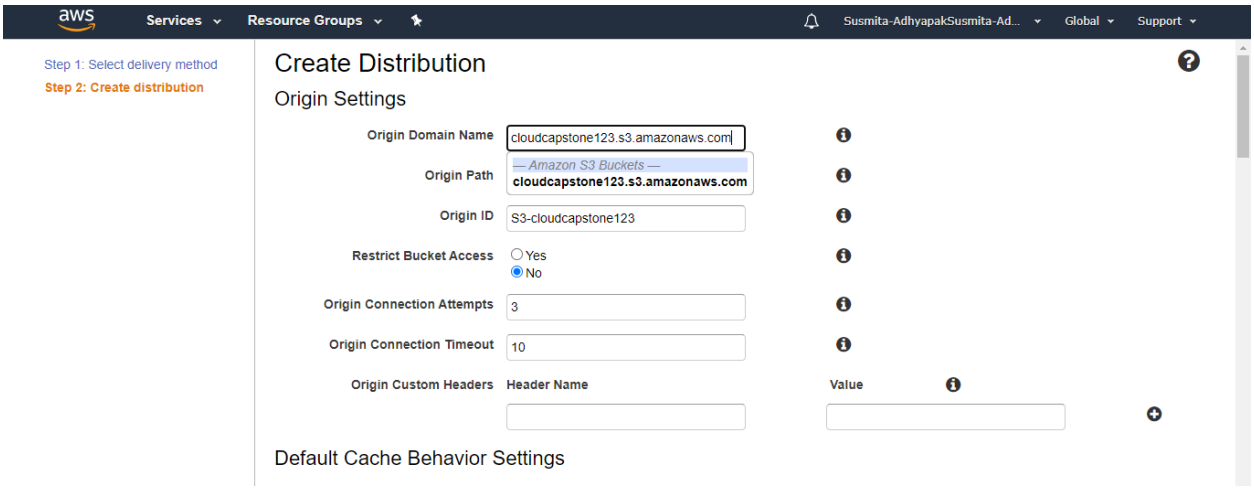
Step 9.2: Click on **Create Distribution**



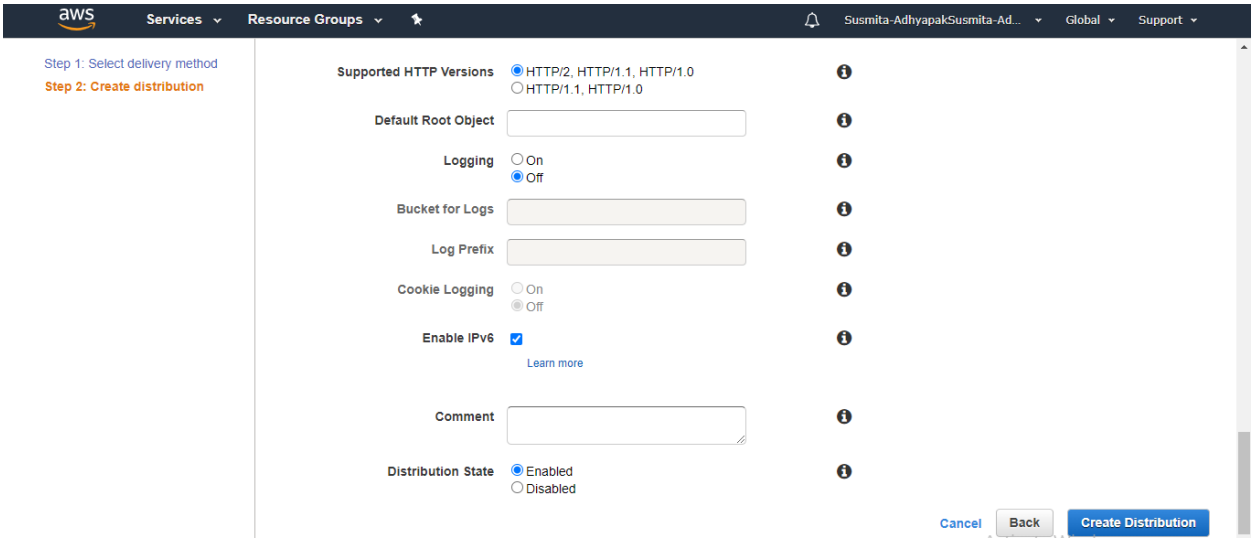
Step 9.3: Select a delivery method for your content as Web and click on Get Started



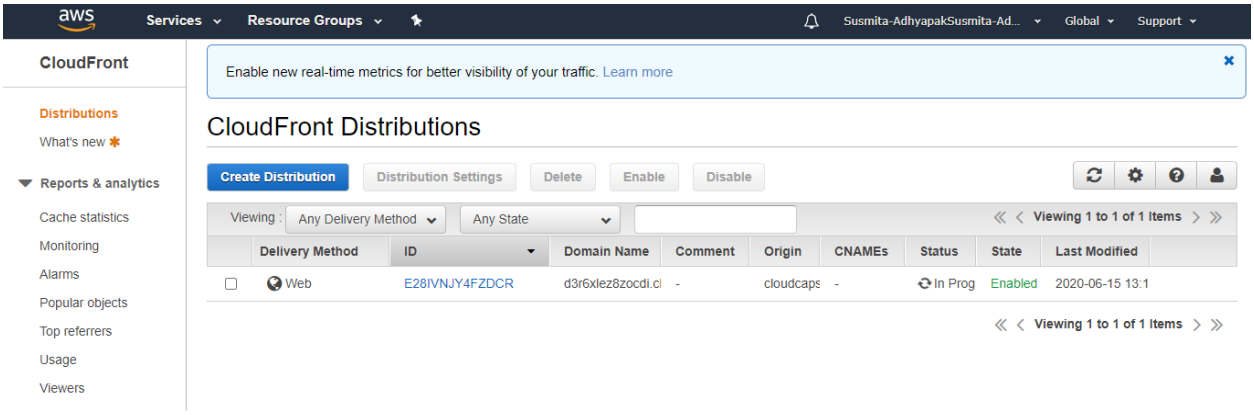
Step 9.4: Provide the Origin Domain Name and Origin ID



Step 9.5: Keep all the values as default and click on **Create Distribution**



Step 9.6: These steps will create your CloudFront distribution

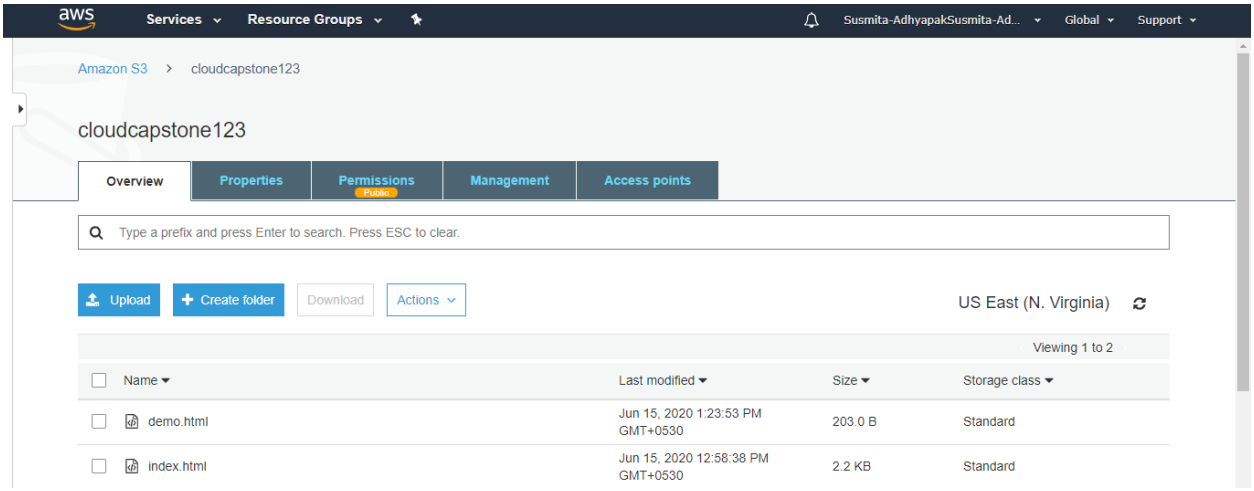


Step 10: Configure the CloudFront distribution to point to your domain by editing the configuration and adding the domain name in Alternate Domain Name field.

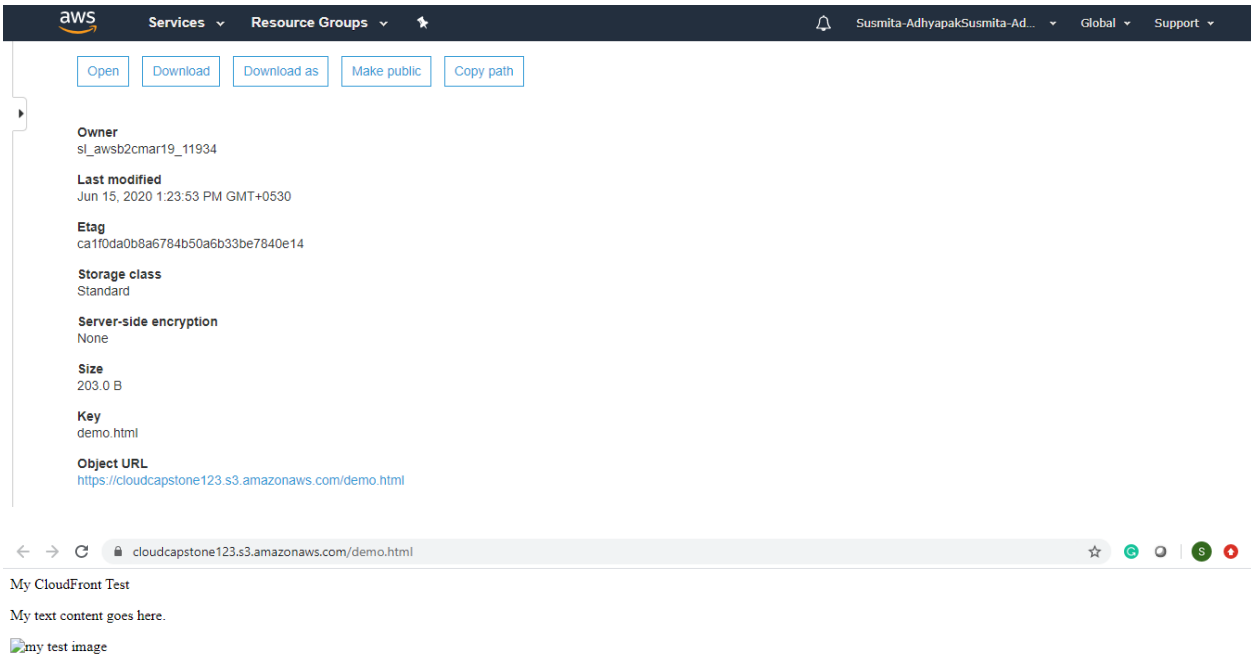
Step 10.1: Once the CloudFront service gets deployed, create a new file of .html extension in your system and copy the following content in it

```
<html>

    <head>My CloudFront Test</head>

    <body>

    <p>My text content goes here.</p>

    <p><img src=https://d111111abcdef8.cloudfront.net/.html
alt="my test image"/>

    </body>

    </html>
```

Step 10.2: Upload the same file in the created S3 bucket



Step 10.3: Click on the newly uploaded file, and go to the Object URL



Step 11: Repeat Steps 5 to 11 to create multiple deployments of your application in different regions so that you can meet the global traffic demand.

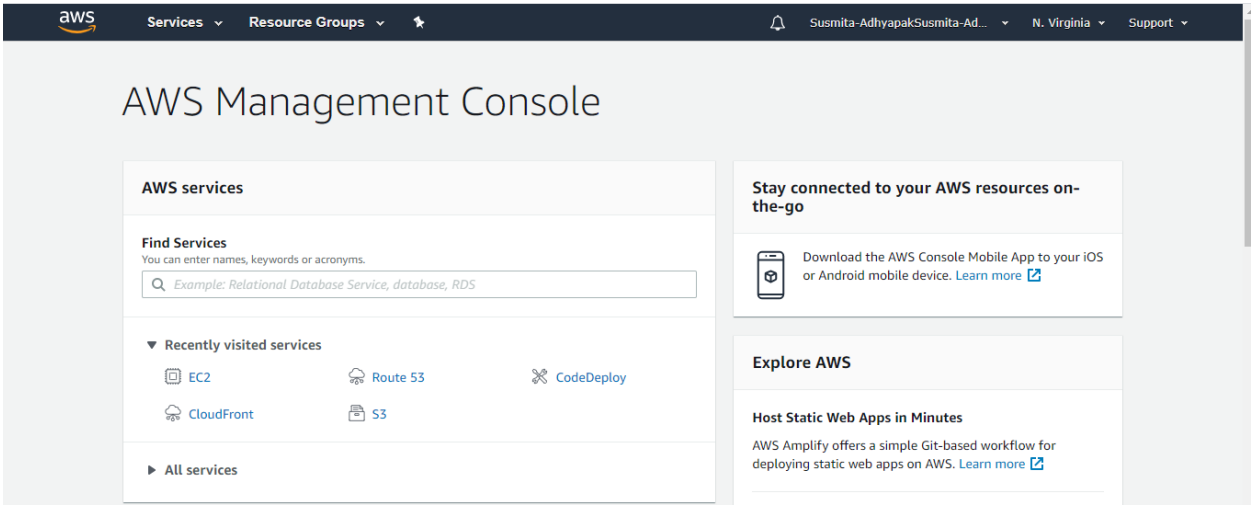
Step 12: Use the traffic flow editor to create traffic policy to route traffic to different endpoints across the globe

Step 13: As good practice, follow the principle of least privilege so that you give access to the services that need to be accessed within the AWS console.

AWS:

Approach 2:

Step 1: Log into the AWS console

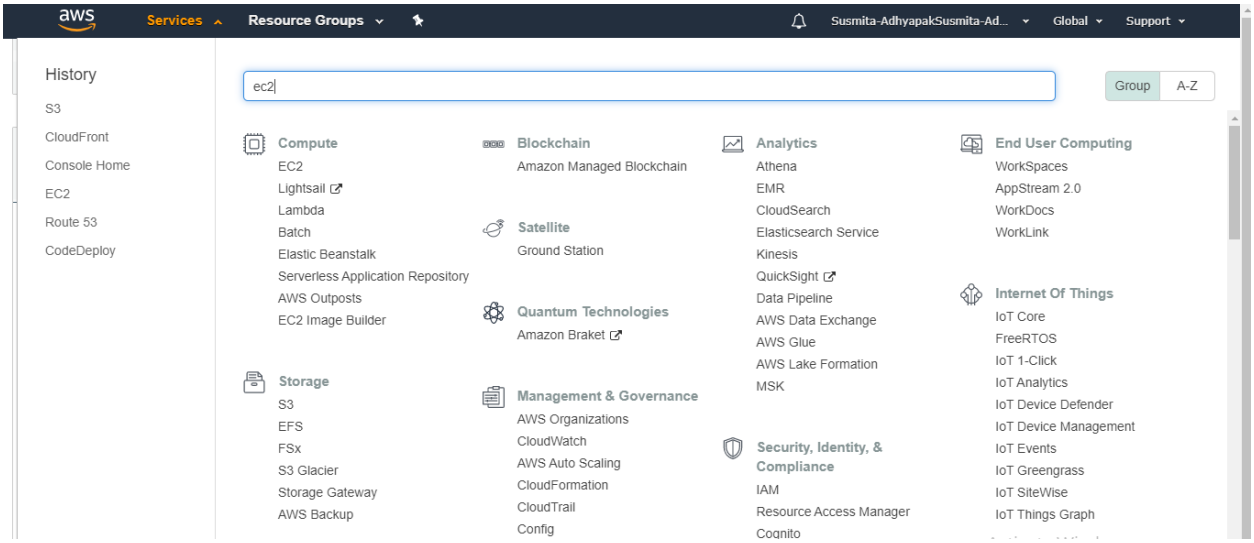


Step 2: Before creating the resources, make sure you apply cost allocation tags to resources so that you can keep a track of billing later on

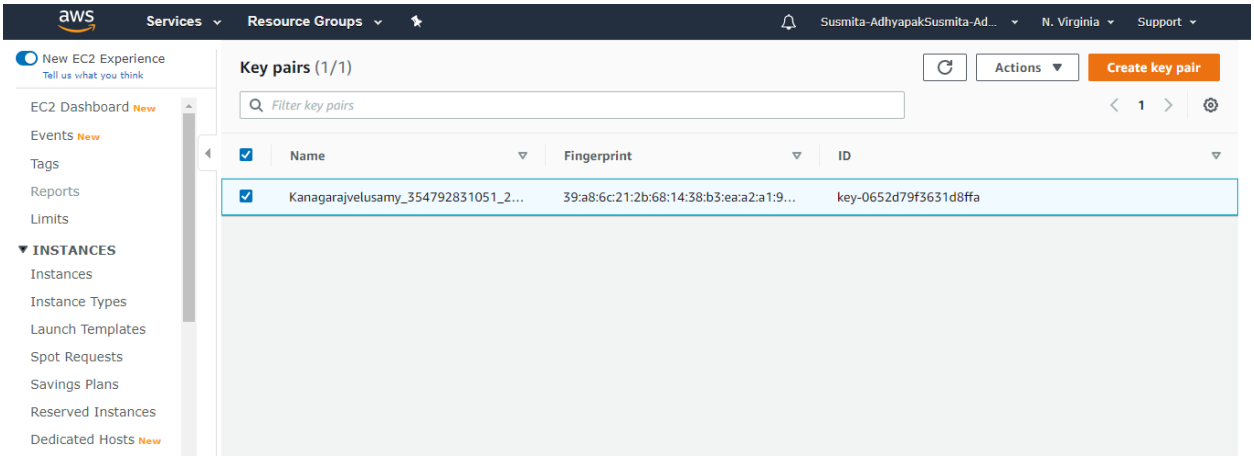
Step 3: To begin, create Route 53 and add a hosted zone if you have your own domain, this is an optional step to configure a custom domain for your web app

Step 4: Create an EC2 instance

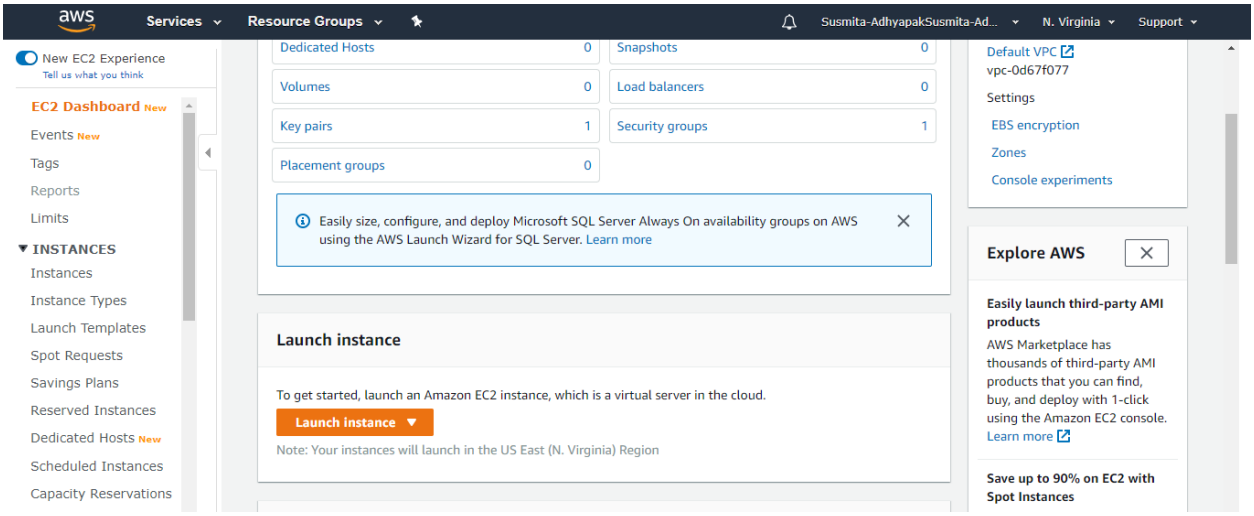
Step 4.1: In the search window, search for EC2



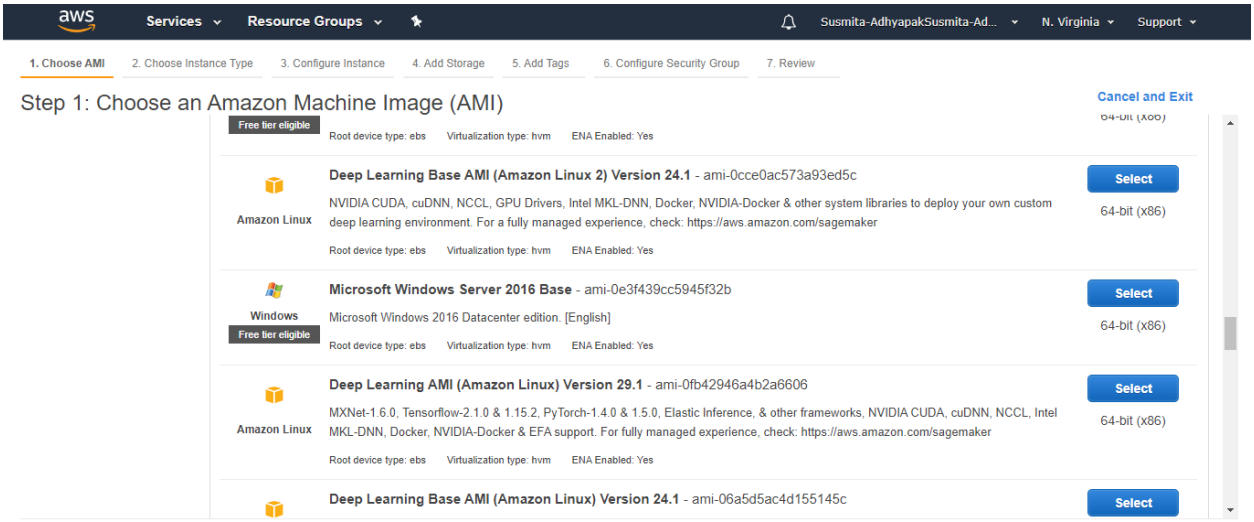
Step 4.2: In the EC2 instance, check whether a key-value pair is created or not. If not then create one



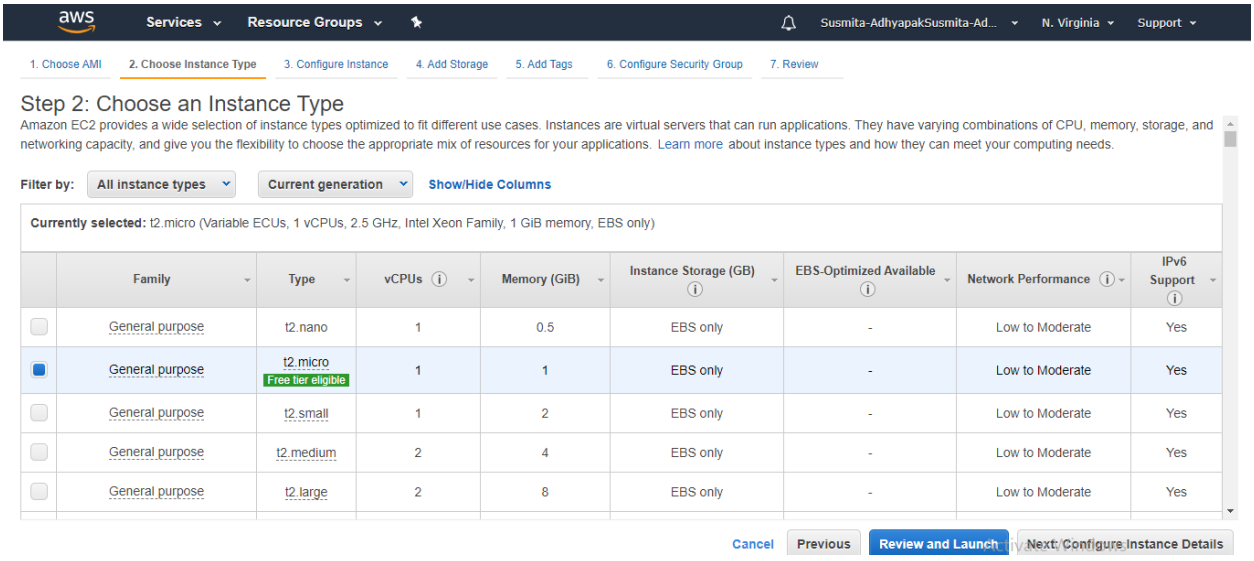
Step 4.3: Click on Launch instance



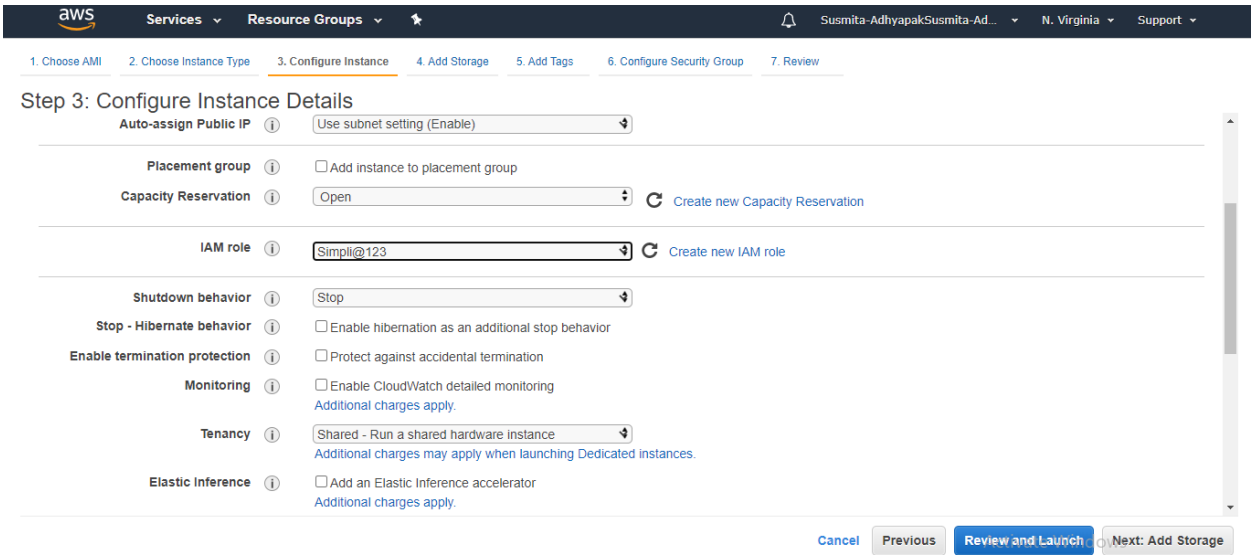
Step 4.4: Choose an Amazon Machine Image (AMI) (Free tier only) and click on **Select**



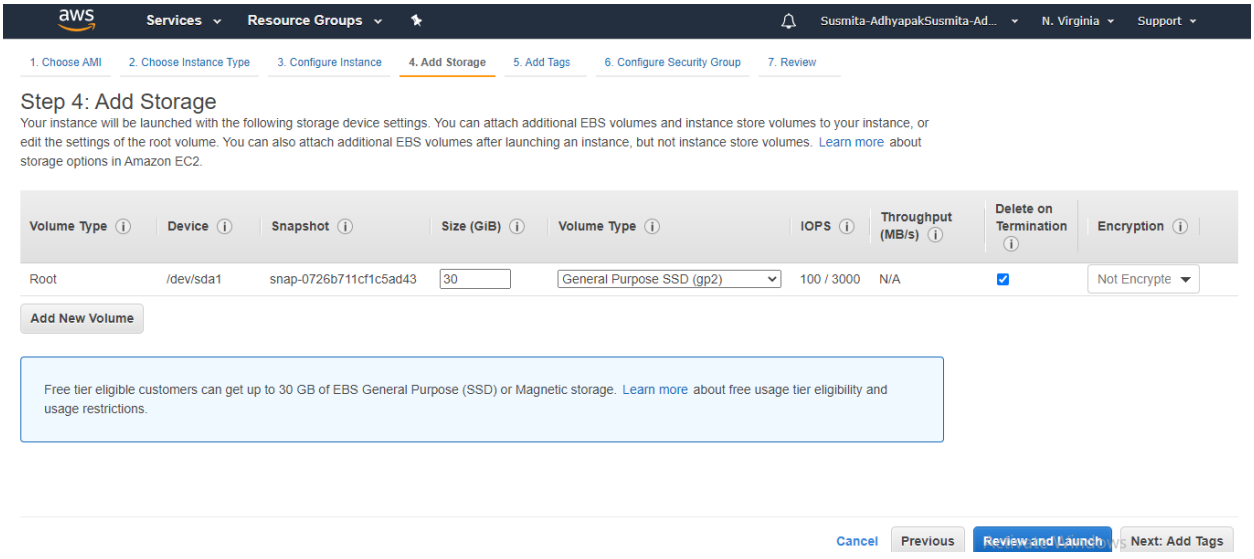
Step 4.5: Select a proper instance type (Select t2 micro) and anc click on **Next: Configure Instance Details**



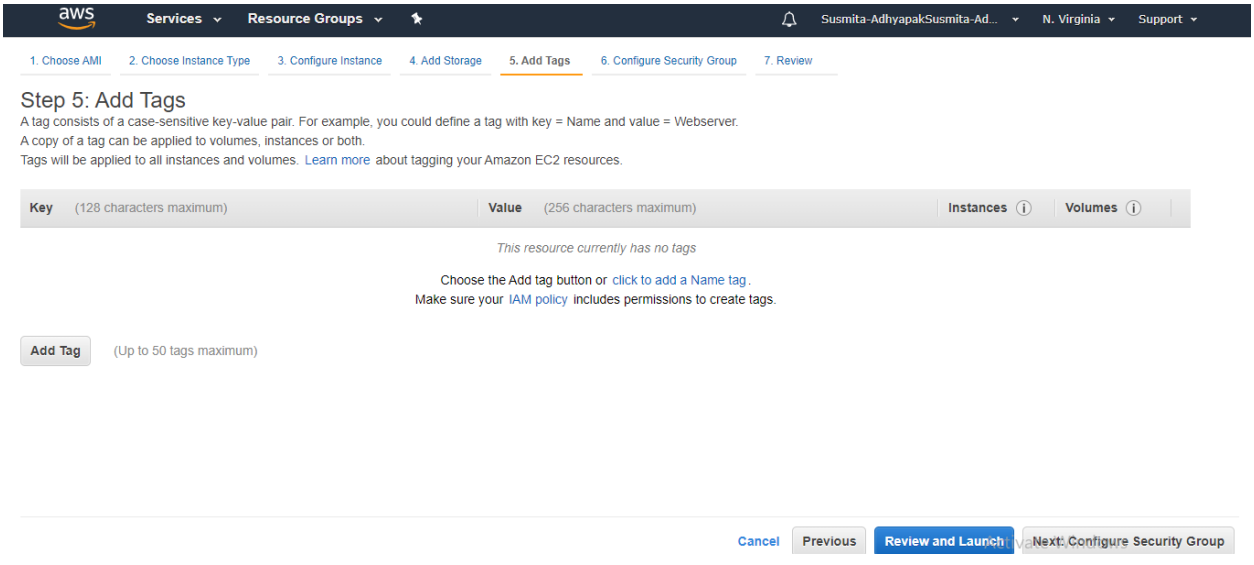
Step 4.6: In the EC2 dashboard, click on **Next: Add Storage**



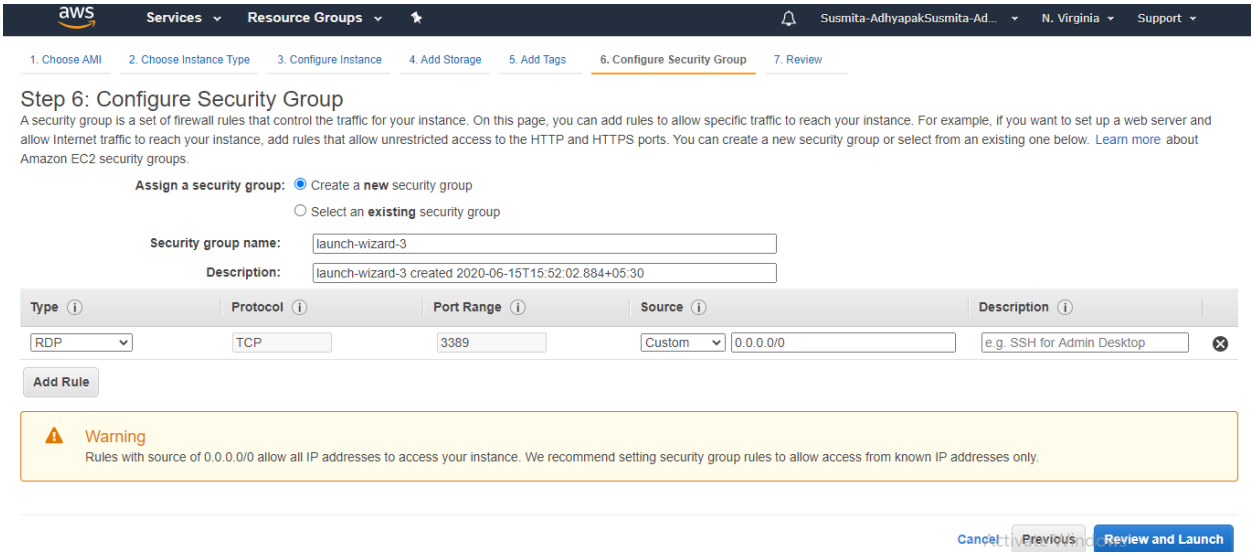
Step 4.7: Click on **Next: Add Tags**



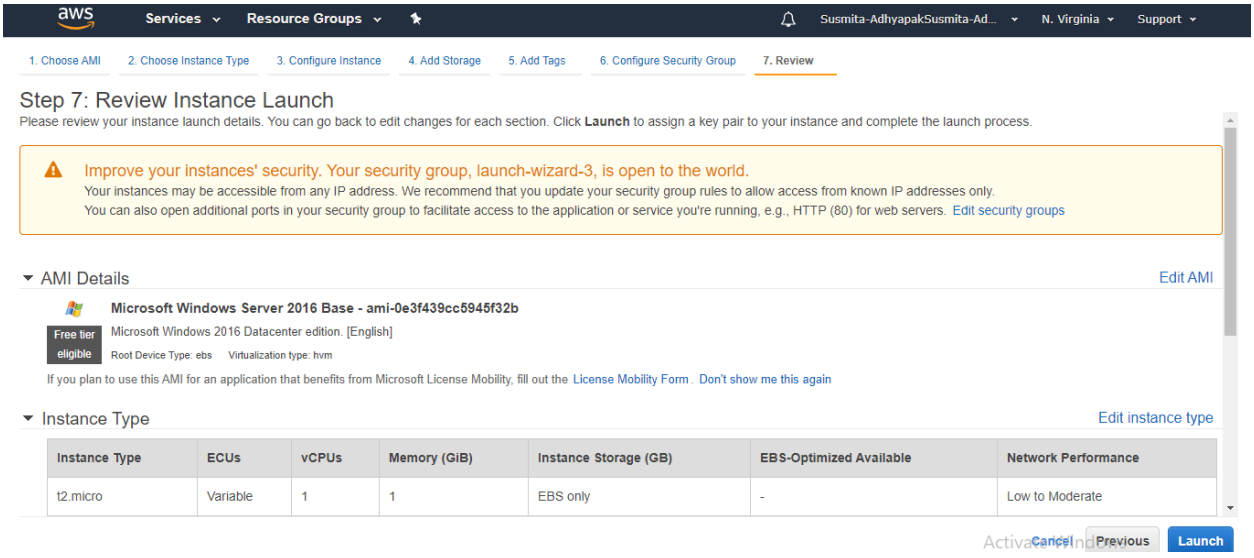
Step 4.8: Click on **Next: Configure Security Groups**



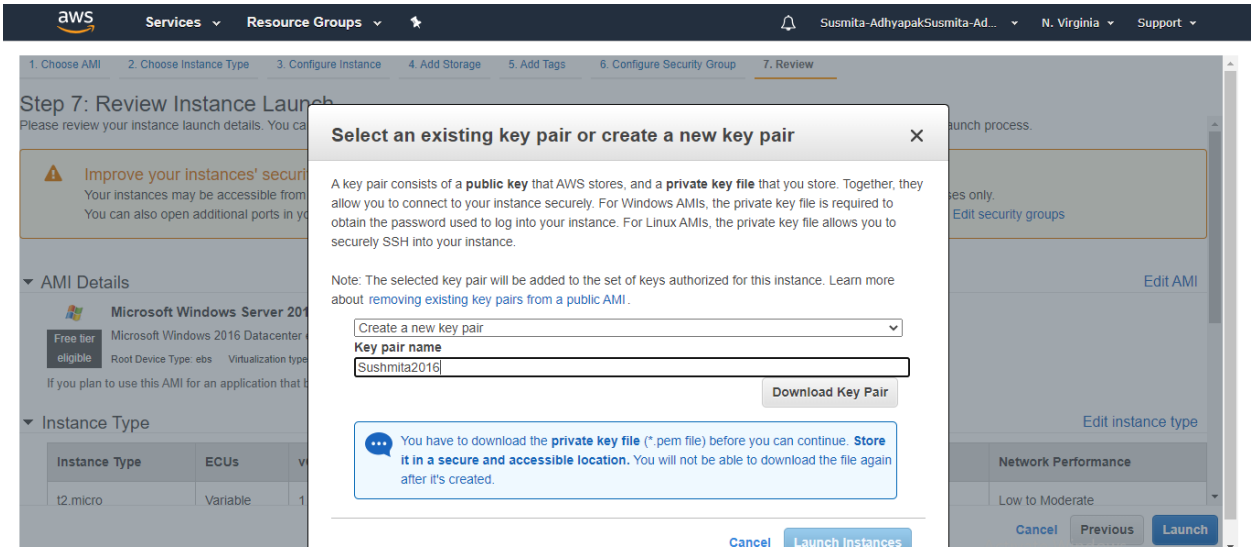
Step 4.9: Click on **Review and Launch**



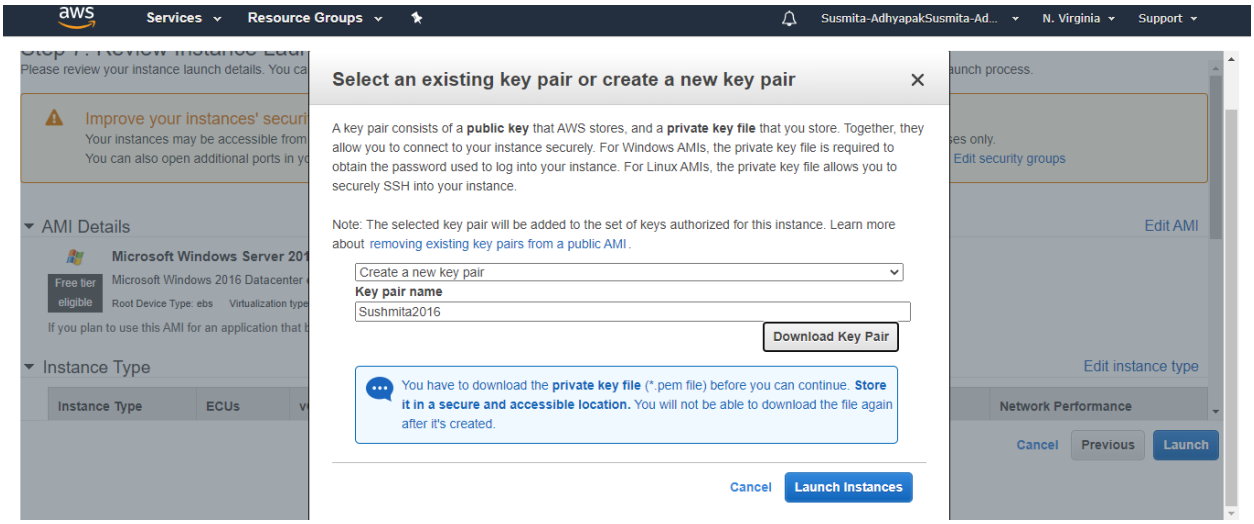
Step 4.10: Click on **Launch**



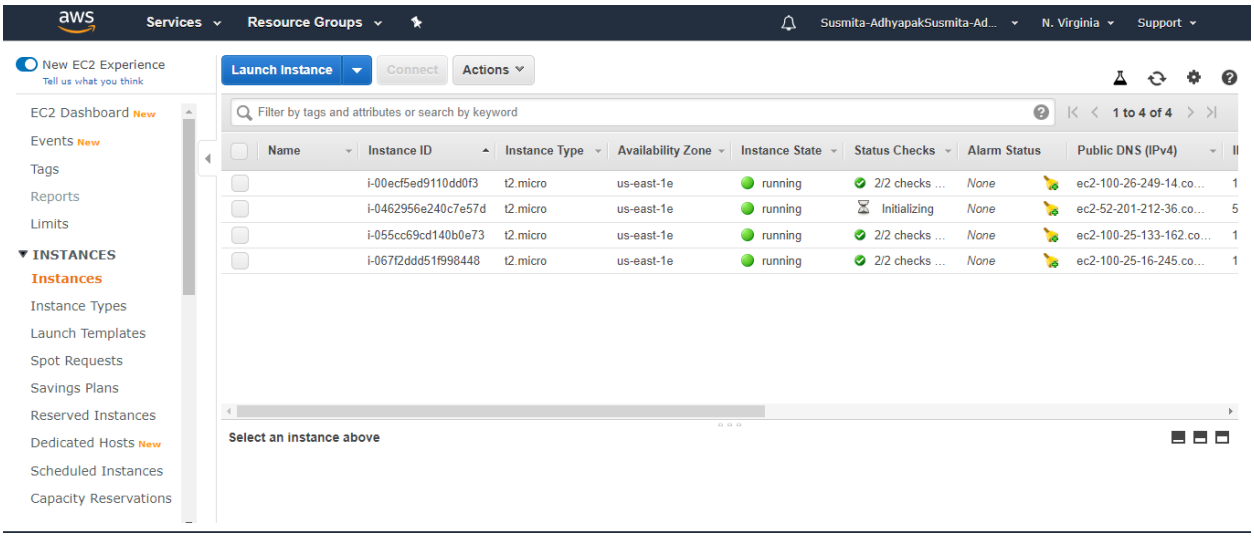
Step 4.11: Create a new key-pair, provide the name of the file and click on **Download Key Pair**



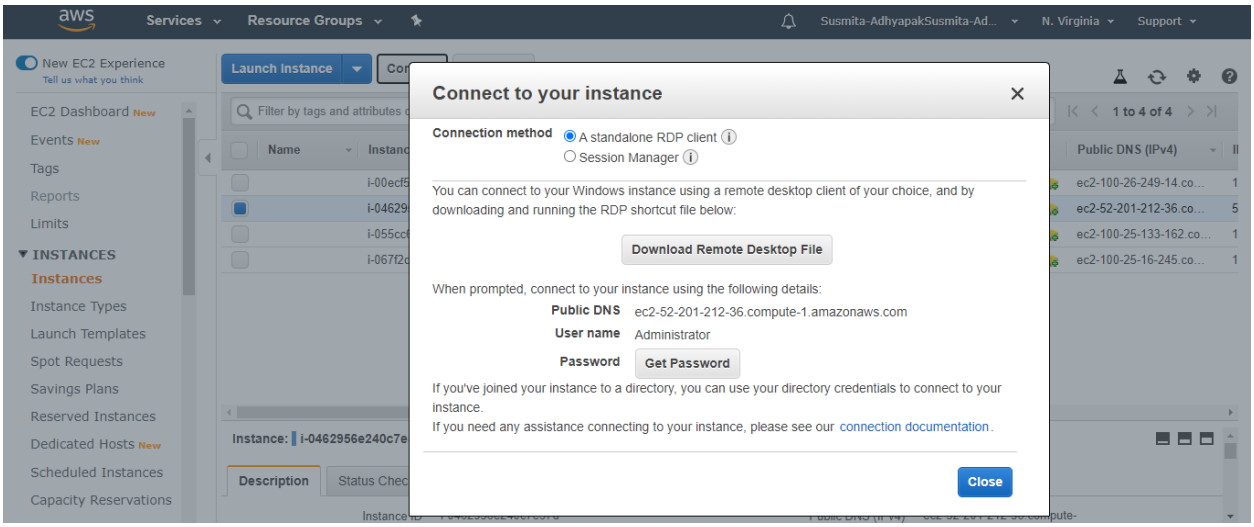
Step 4.12: Click on **Launch Instances**



Step 4.13: Go to the EC2 dashboard, select the created EC2 instance and click on **Connect**

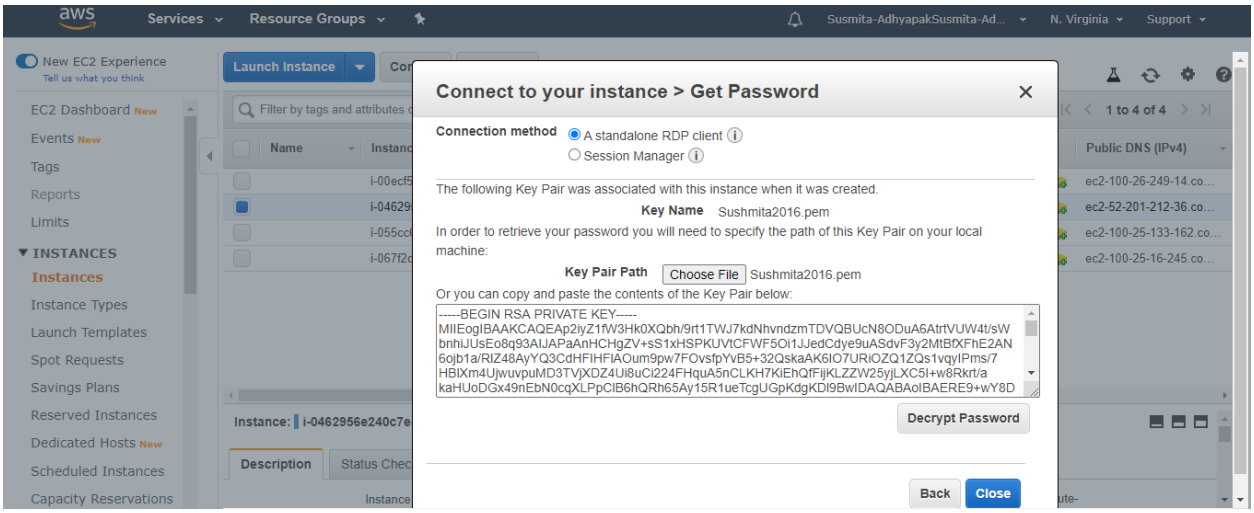


Step 4.14: Click on **Download Remote Desktop File**

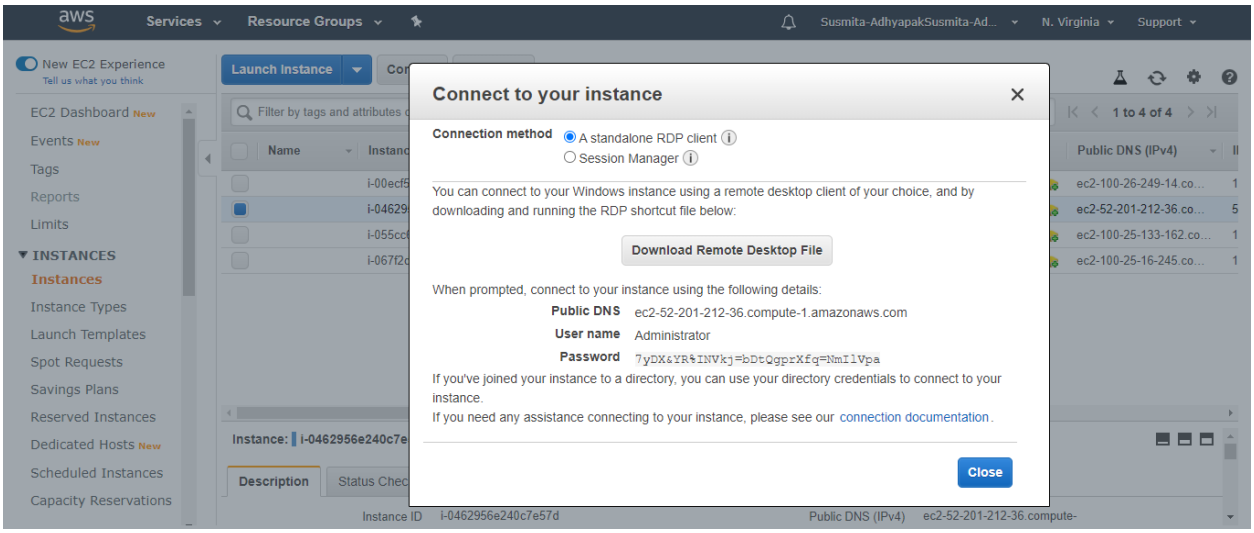


Step 4.15: Click on **Get Password**

Step 4.16: Browse to the Key Pair File you have downloaded using **Choose File** option



Step 4.17: Click on **Decrypt Password**



Step 4.18: Copy the decrypted password

Step 5: Make sure you have inbound traffic on port 80 and port 443 open

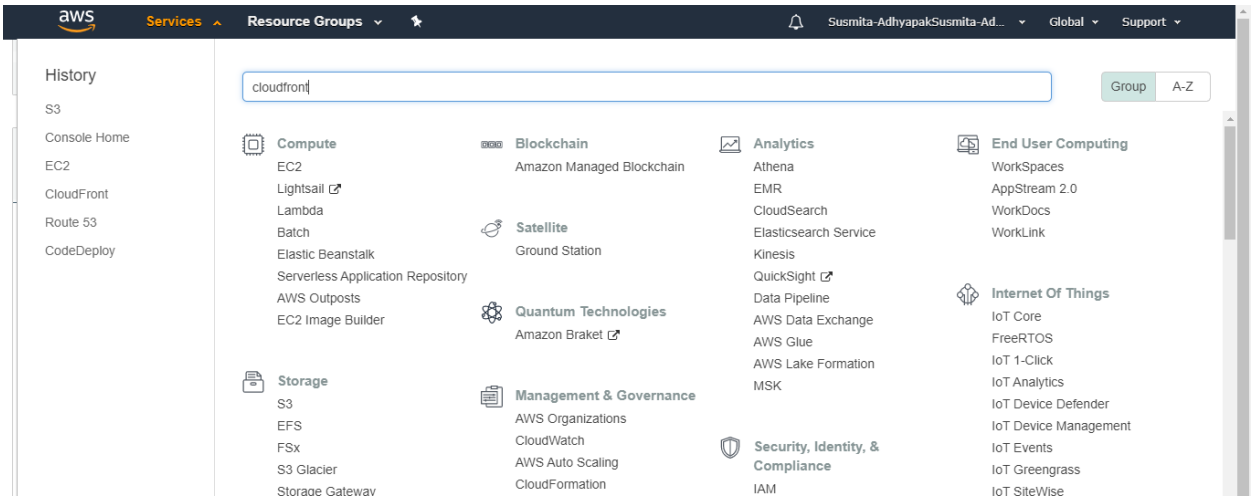
Step 6: Log into EC2 instance and spin up a web server of your choice on port 80

Step 7: Deploy your application on the web server that you have created within the virtual machine

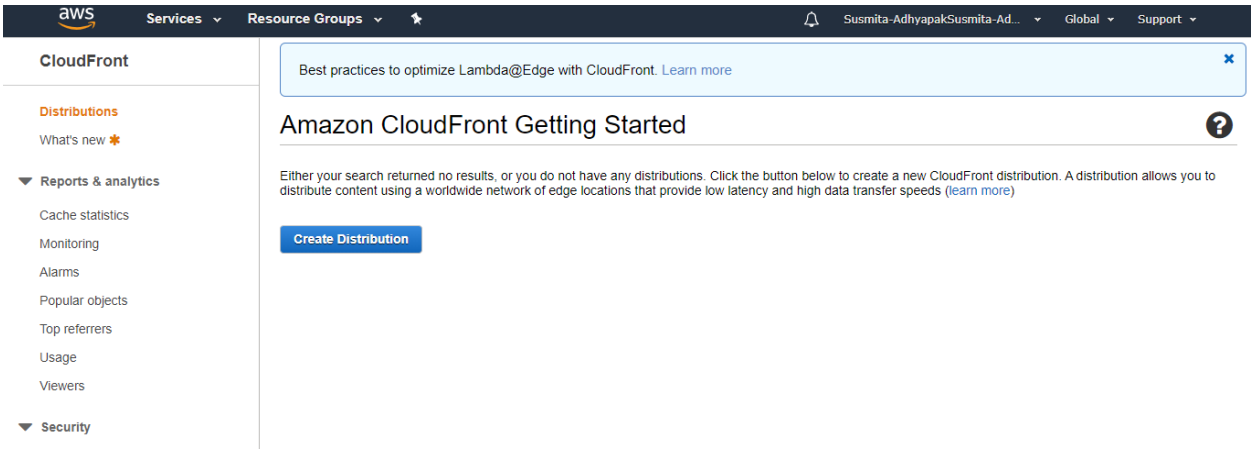
Step 8: Hit the Public IP of EC2 instance (web app endpoint) to check if the application is online

Step 9: Now create a CloudFront distribution corresponding to the static web app endpoint.

Step 9.1: Go to services and search for CloudFront

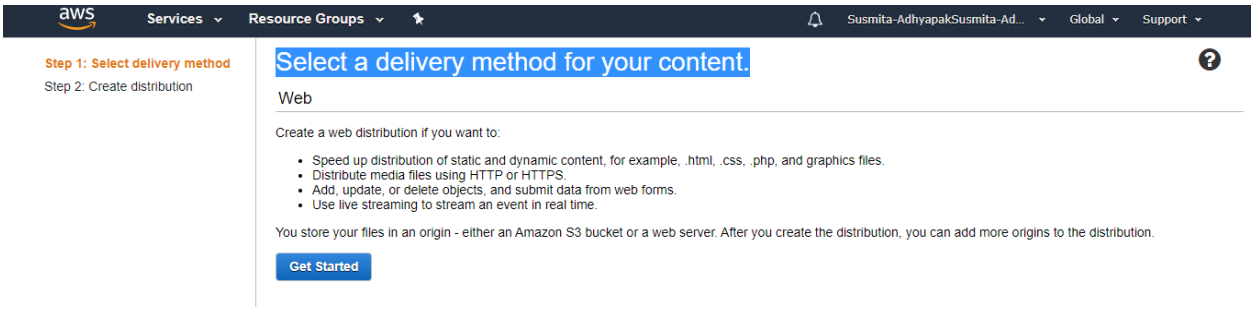


Step 9.2: Click on **Create Distribution**

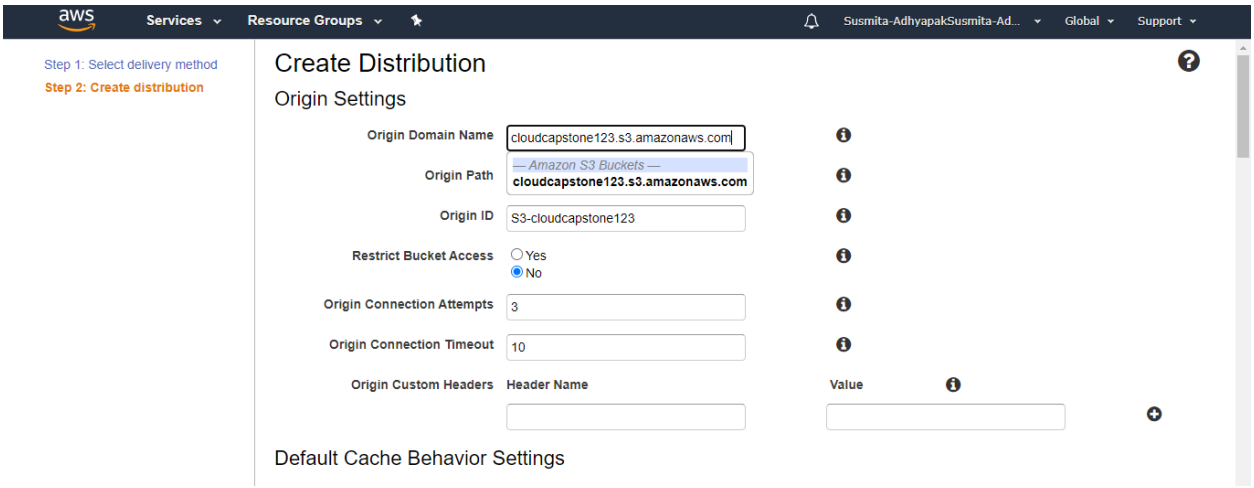


Step 9.3: Select a delivery method for your content as Web and click on Get

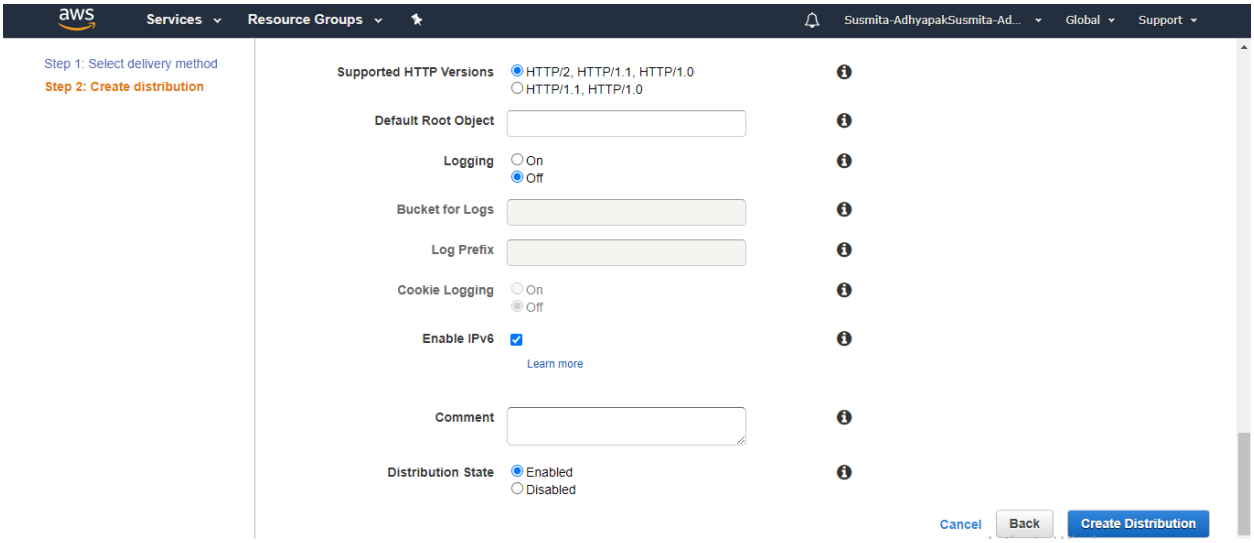
Started



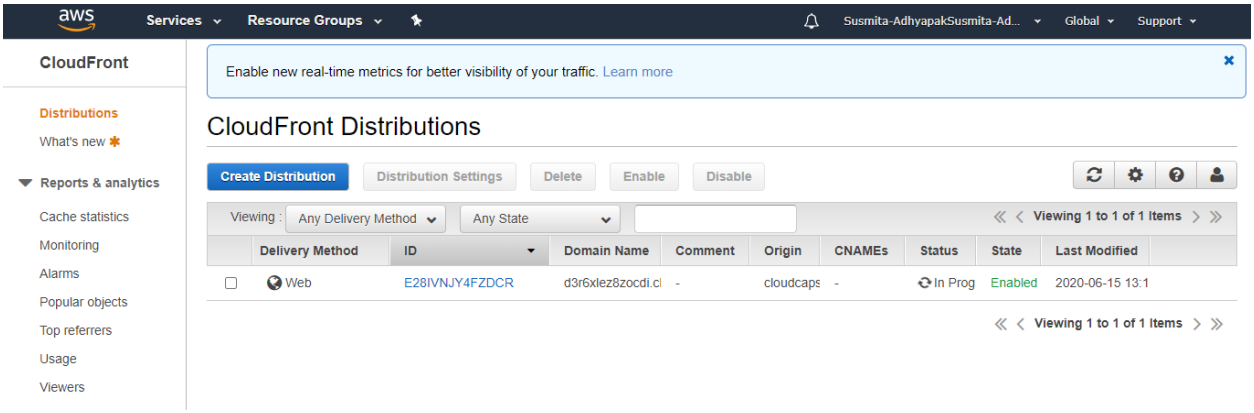
Step 9.4: Provide the Origin Domain Name (DNS name of the3 created EC2 instance) and Origin ID



Step 9.5: Keep all the values as default and click on **Create Distribution**



Step 9.6: These steps will create your CloudFront distribution.



Step 10: Configure the CloudFront distribution to point to your domain by editing the configuration and adding the domain name in Alternate Domain Name field

Step 11: Repeat steps 4 to 10 to create multiple deployments of your application in different regions so that you can meet the global traffic demand

Step 12: Use the traffic flow editor to create traffic policy to route traffic to different

endpoints across the globe

Step 13: As good practice, follow the principle of least privilege so that you give access to the services that need to be accessed within the AWS console