University of Mumbai

PRACTICAL JOURNAL – ELECTIVE II

PSIT3P3c
Cloud Application Development

SUBMITTED
BY

SACHIN DADHIBAL JAISWAR

SEAT NO 30440

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTSFOR
QUALIFYING M.Sc. (I.T.) PART-II (SEMESTER – III) EXAMINATION

2023-2024

DEPARTMENT OF INFORMATION TECHNOLOGY 3RD FLOOR, DR. SHANKAR DAYAL SHARMA BHAVAN, VIDYANAGRI,SANTACRUZ (E), MUMBAI – 400098.

University of Mumbai

Department of Information Technology

Certificate

This is to certify that **Mr. Sachin Dadhibal jaiswar Seat No. 30440** studying in **Master of Science** in **Information Technology Part II Semester III** has satisfactorily completed the Practical of **PSIT3P3c Cloud Application Development** as prescribed by University of Mumbai,during the academic year **2023-24**

Signature Subject-In-Charge	Signature Head of the Department

Signature
External Examiner

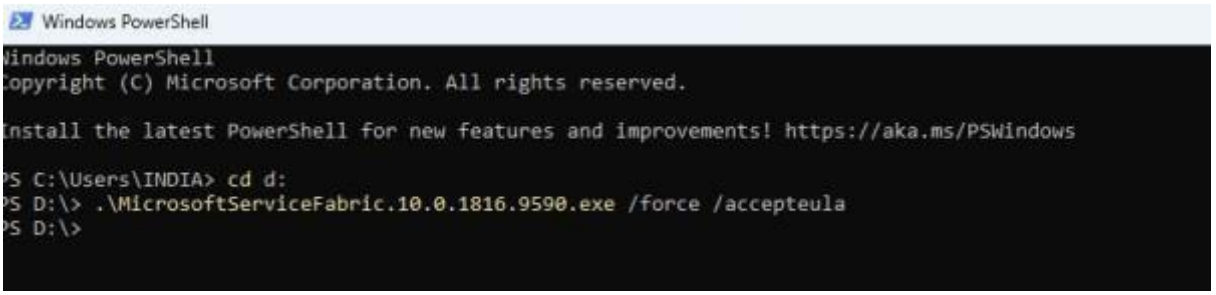
College Seal: _____ **Date:** _____

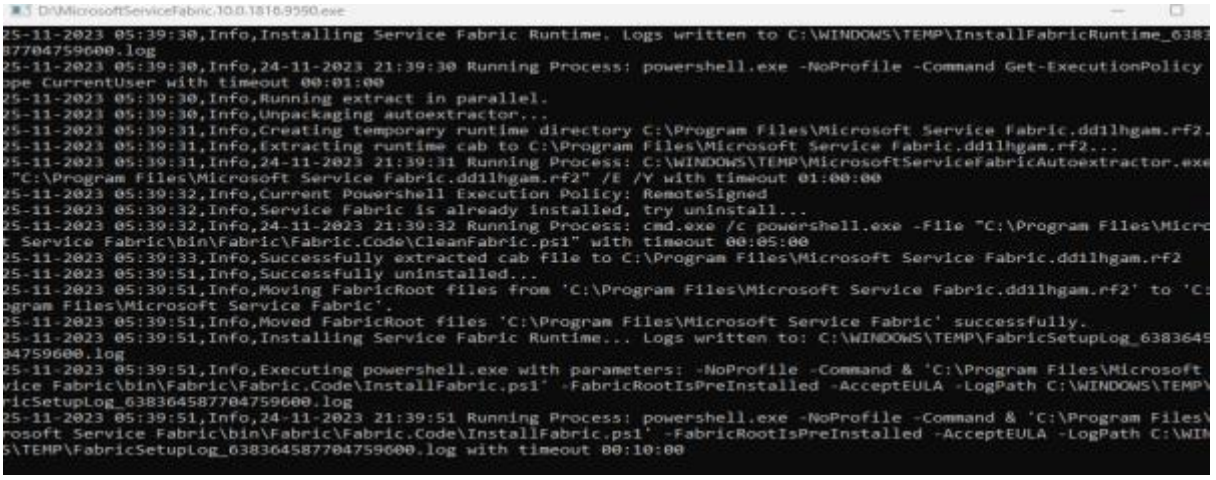
INDEX

Practical No: 1

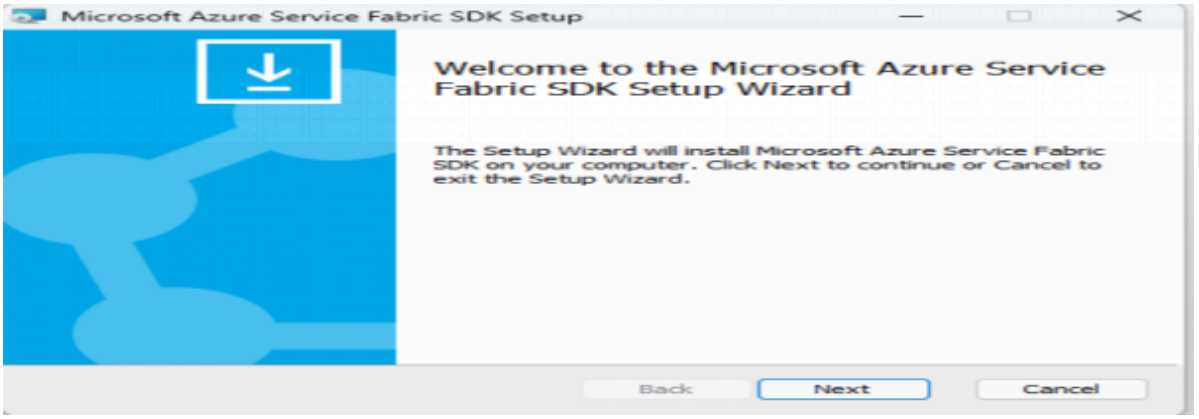
Develop an ASP.NET Core MVC BASED Stateless Web App.

First, we need set up the Service fabric Cluster. For that open Windows PowerShell and fire the following command for setting up a service fabric cluster environment.

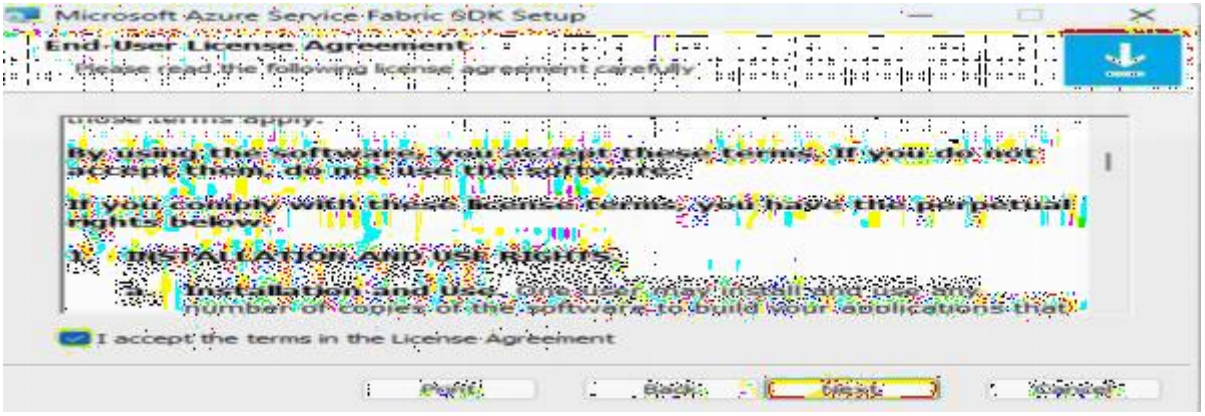




Now install Service Fabric SDK.

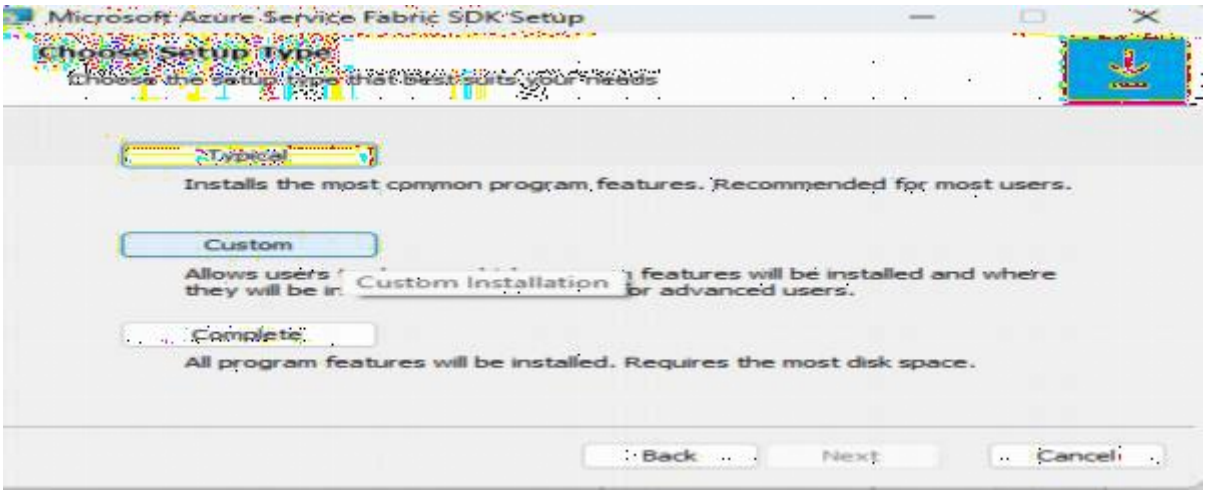


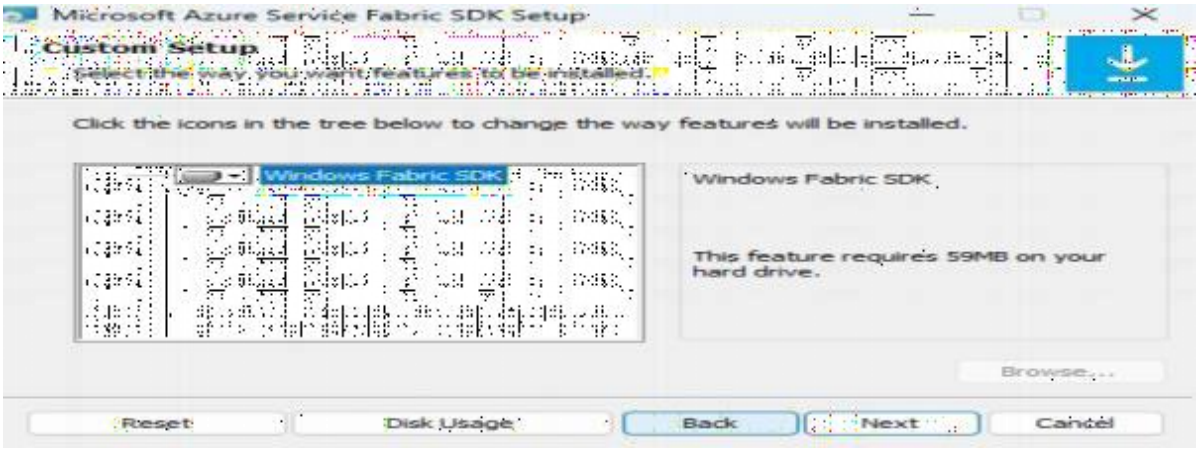
Accept the Liscence agreement.

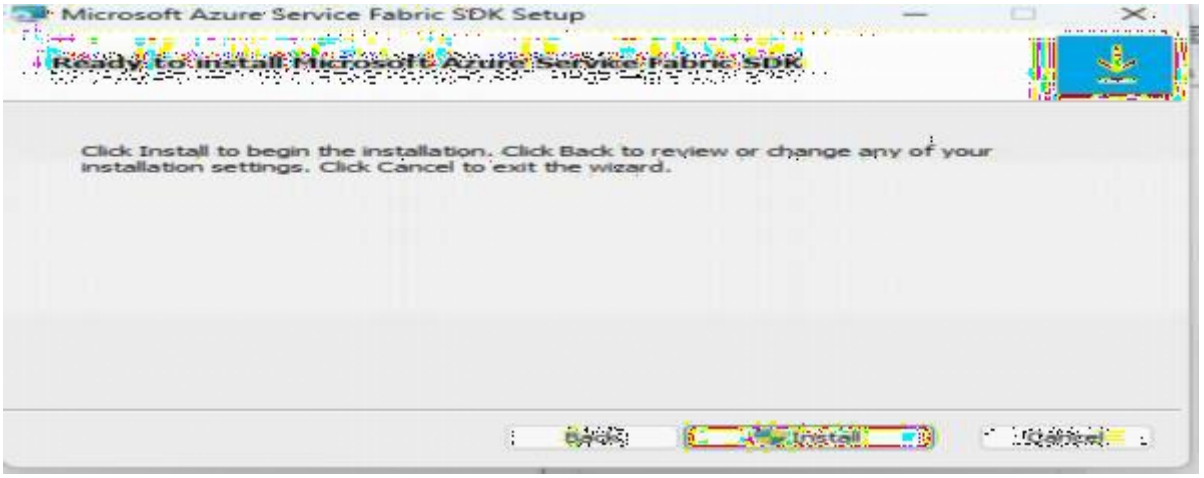


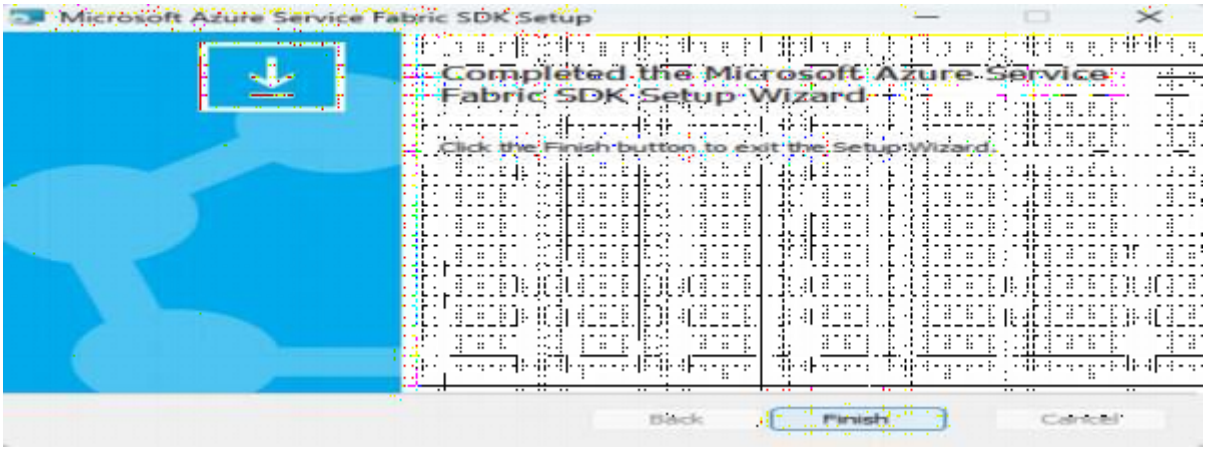
Choose Custom Setup type and click on next.

And install.

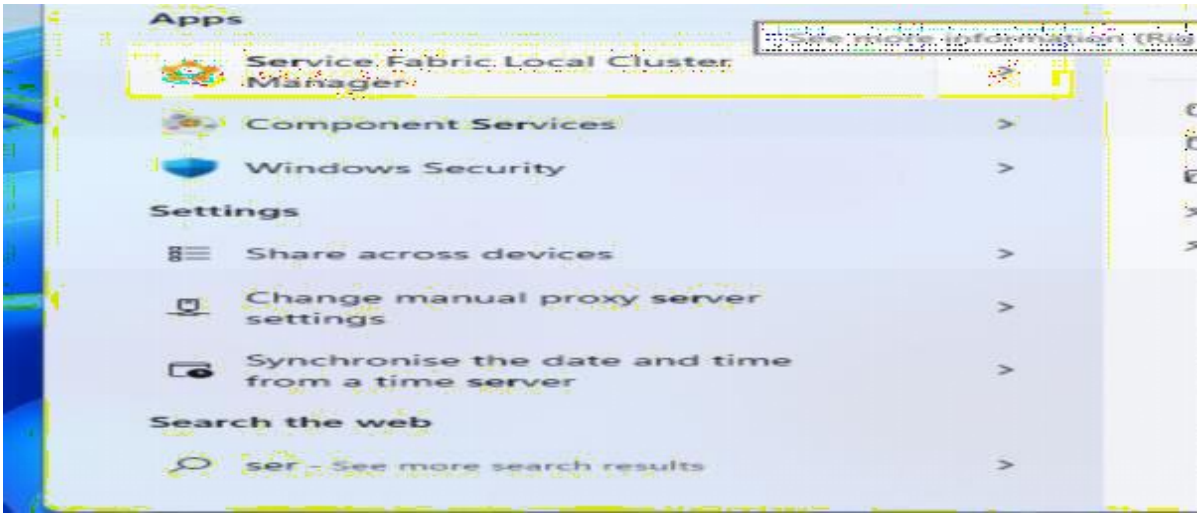






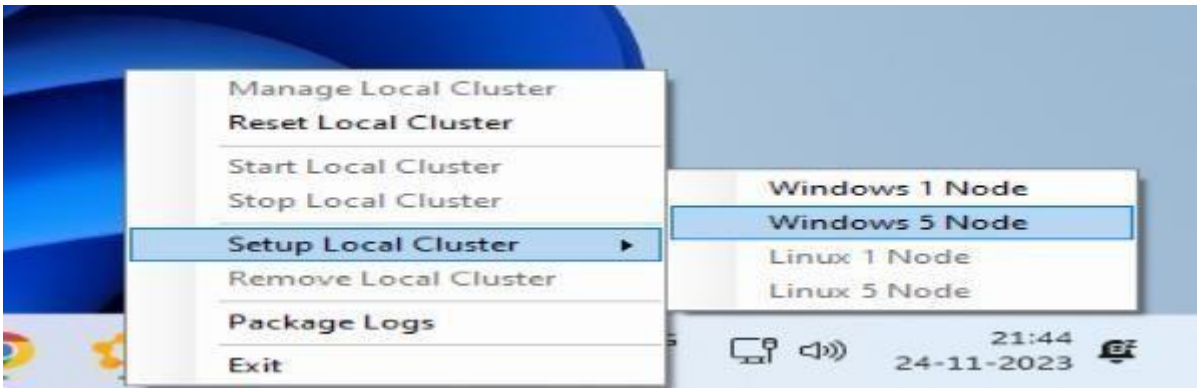


Open search Bar on desktop and search for Service fabric Local Cluster Manager and click on it.

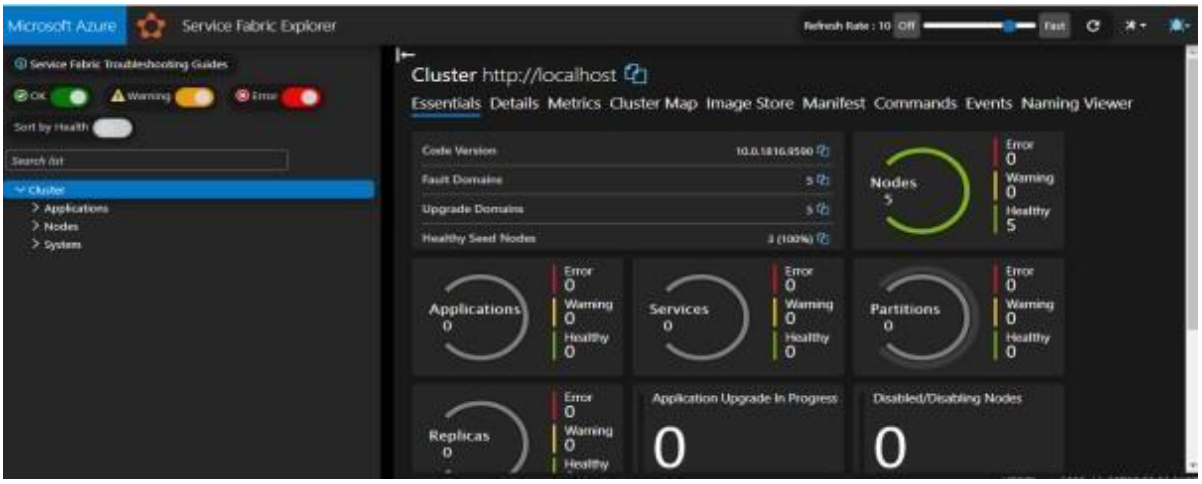


Now Service Fabric Cluster icon is available on the right-hand side icon bar.

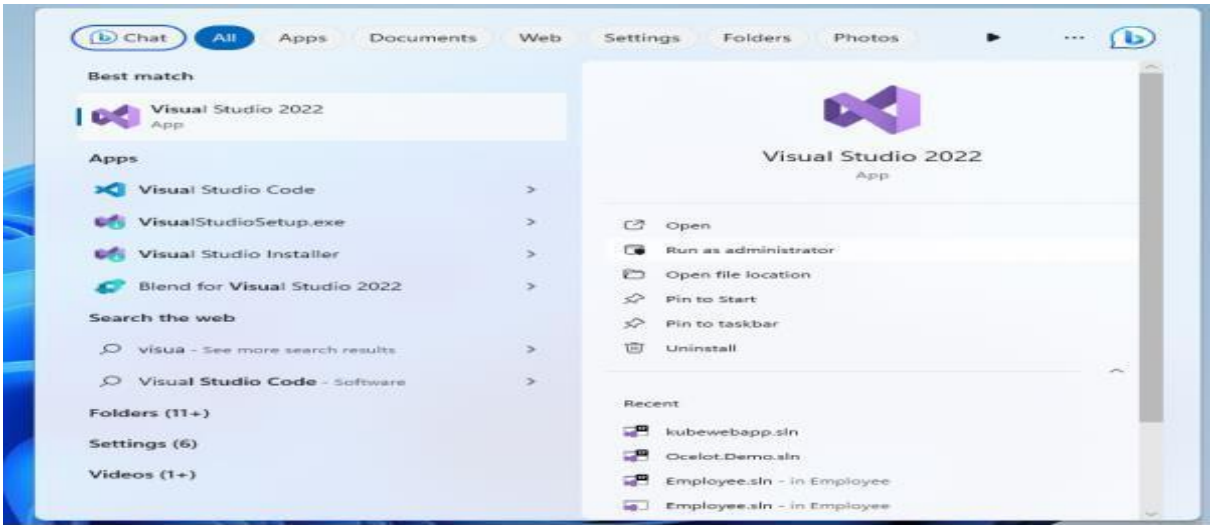
Right click on it and click on Setup local Cluster to windows 5 node. And start local cluster.



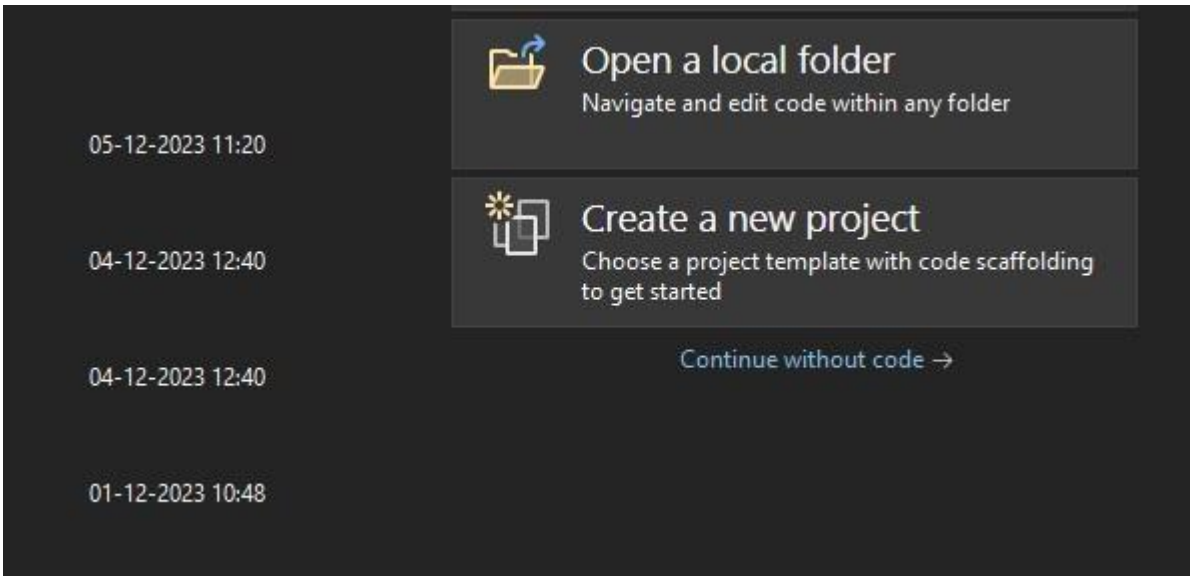
After starting local cluster wait for some time and after that click on Manage Local Cluster. And then following window will open which is Service Fabric Explorer.



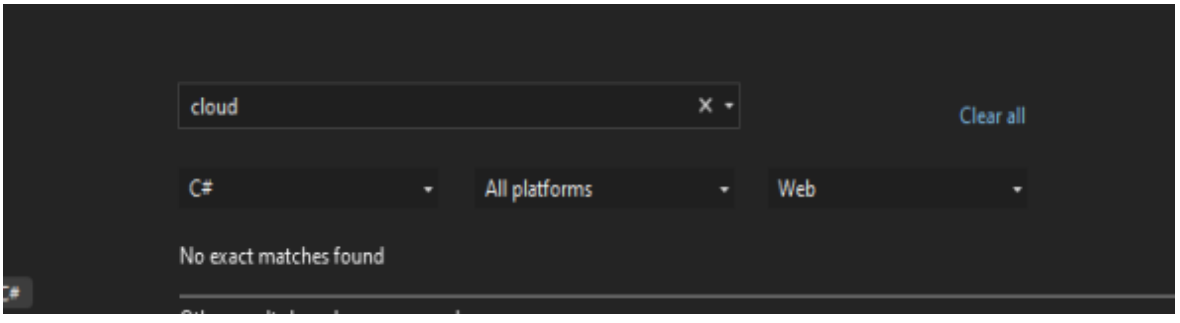
Now, open visual studio 2022 and run as administrator.



Now click on create a new project.

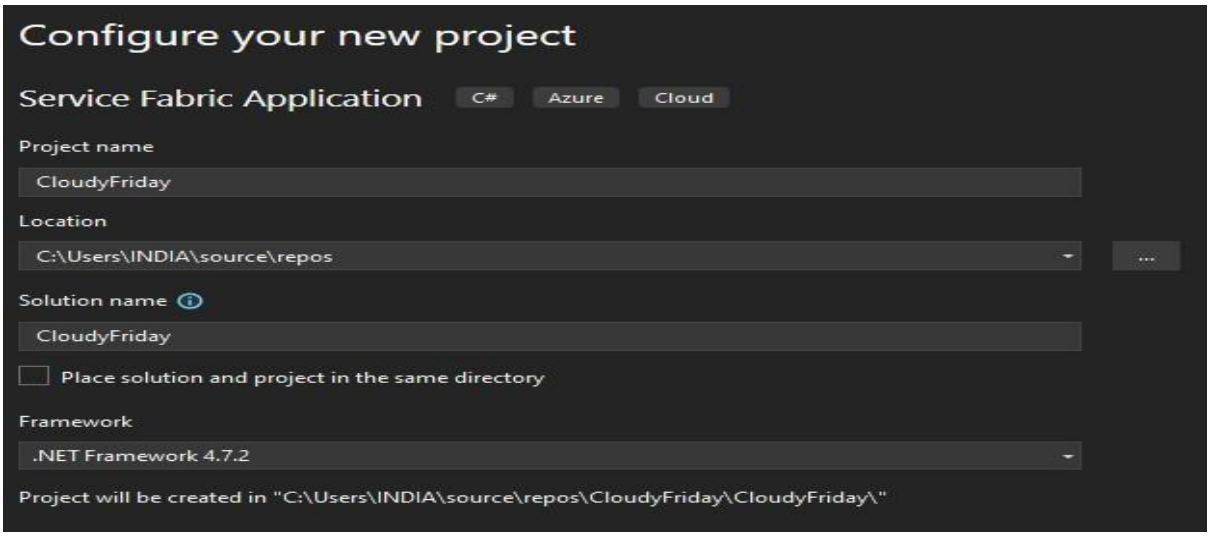


Search for cloud, choose project type c#.

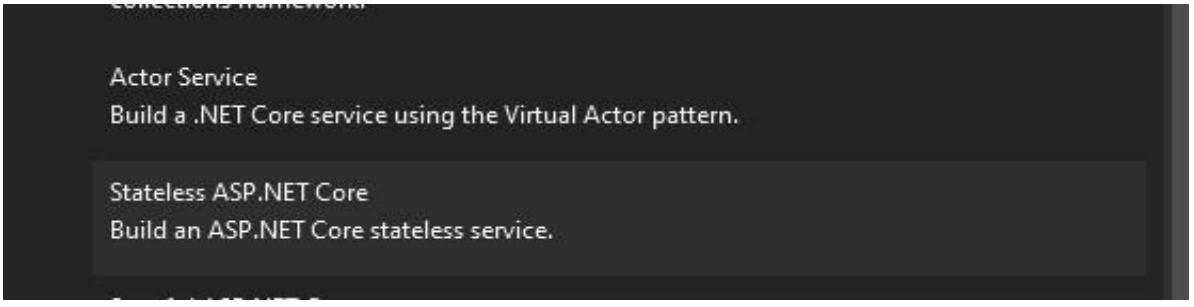


Select Service Fabric Application.

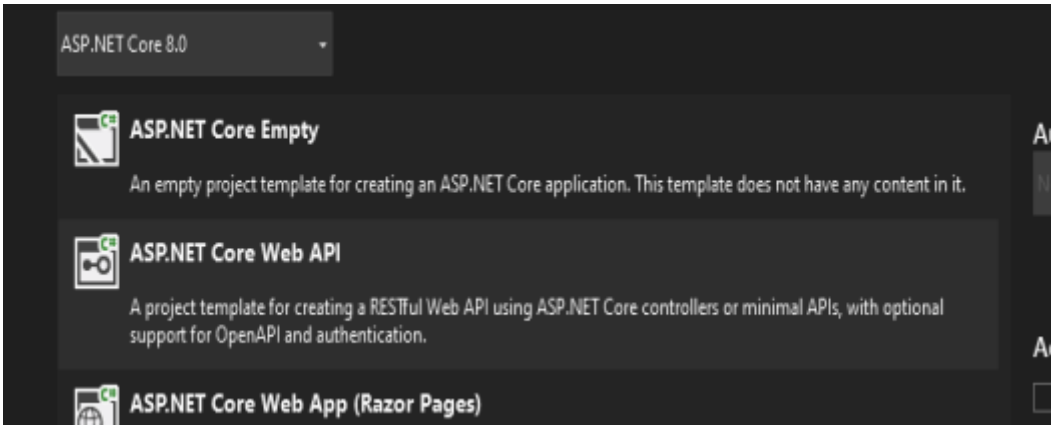
Give project Name and click on next.



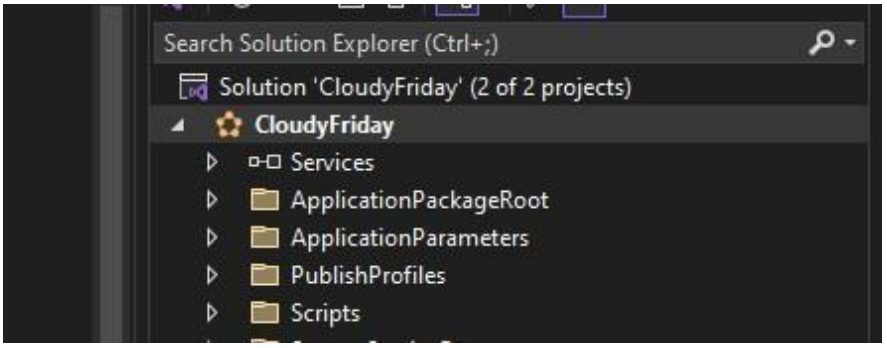
Choose stateless ASP.NET Core and give service name GeographyService.



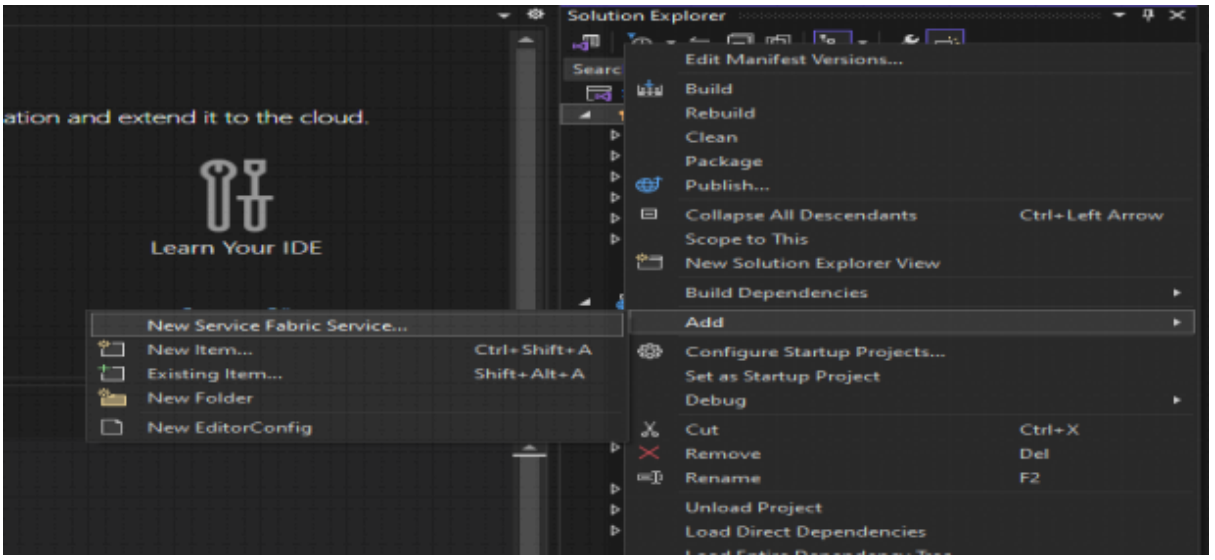
Choose ASP.NET Core Web API and create project.



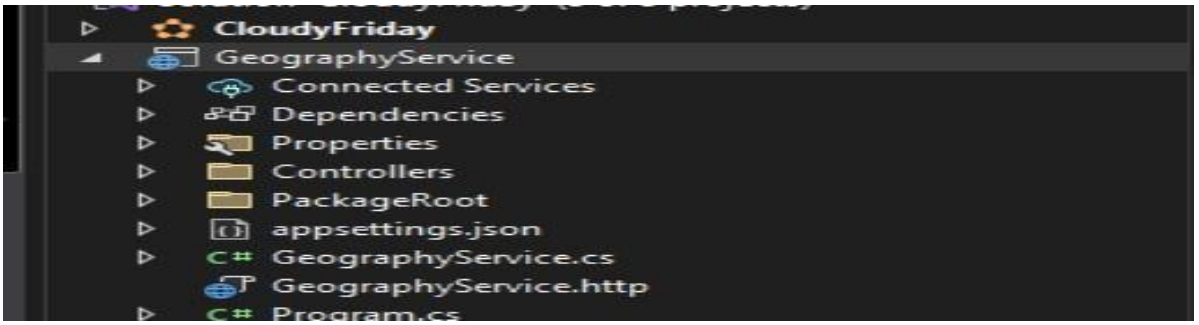
Right click on CloudyFriday.

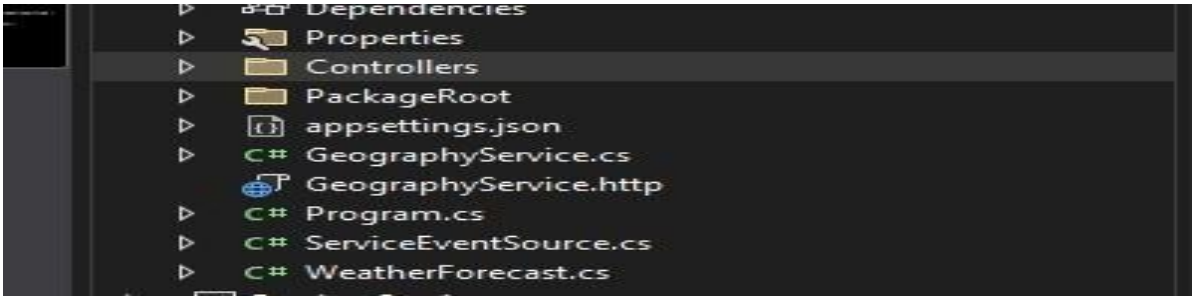


Add a new Service Fabric Service.

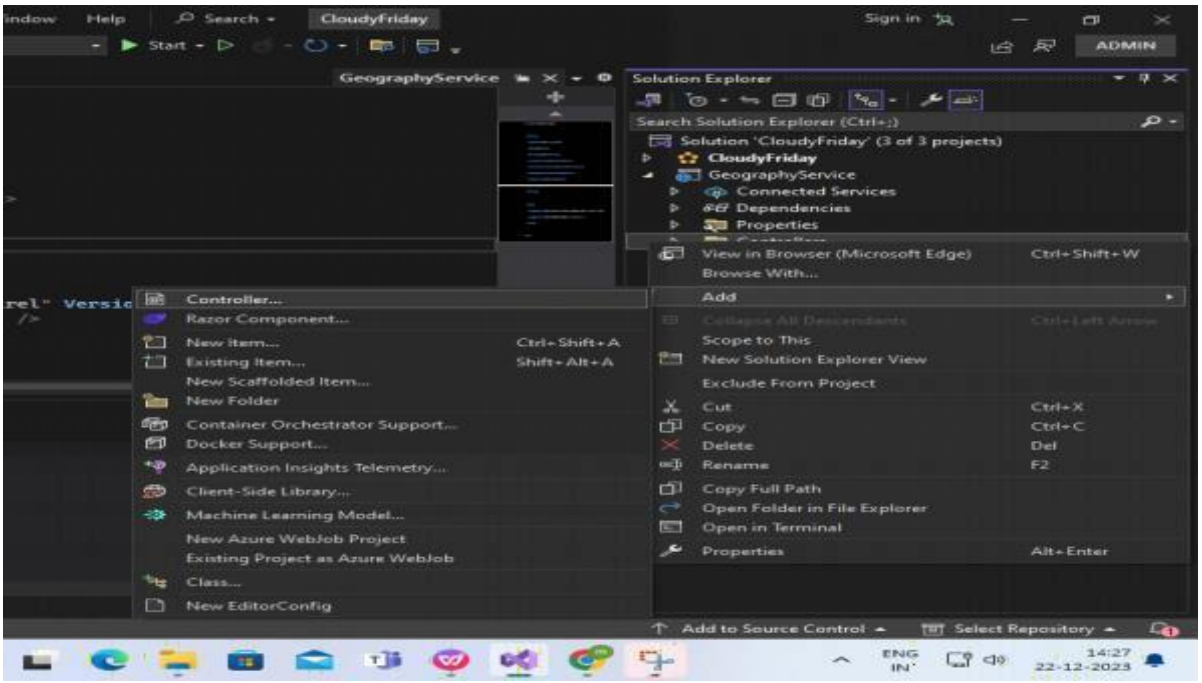


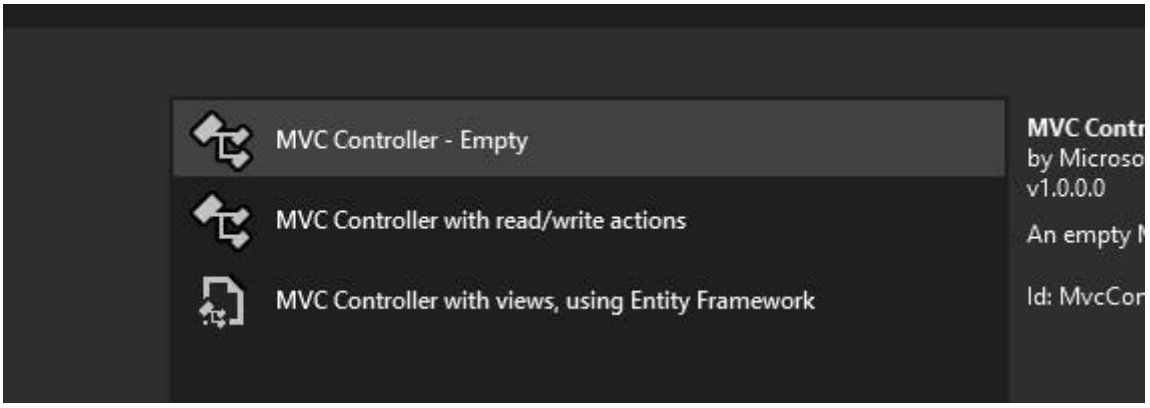
In CloudyFriday project, right click on controllers in GeographyService.

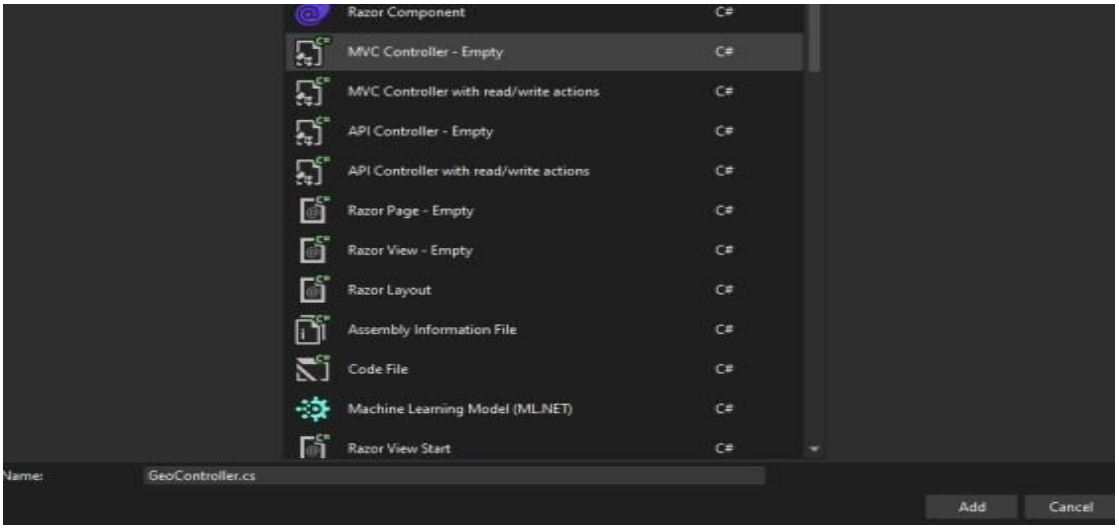




Add controller and select MVC Controller – Empty and give name GeoController.cs. And add.



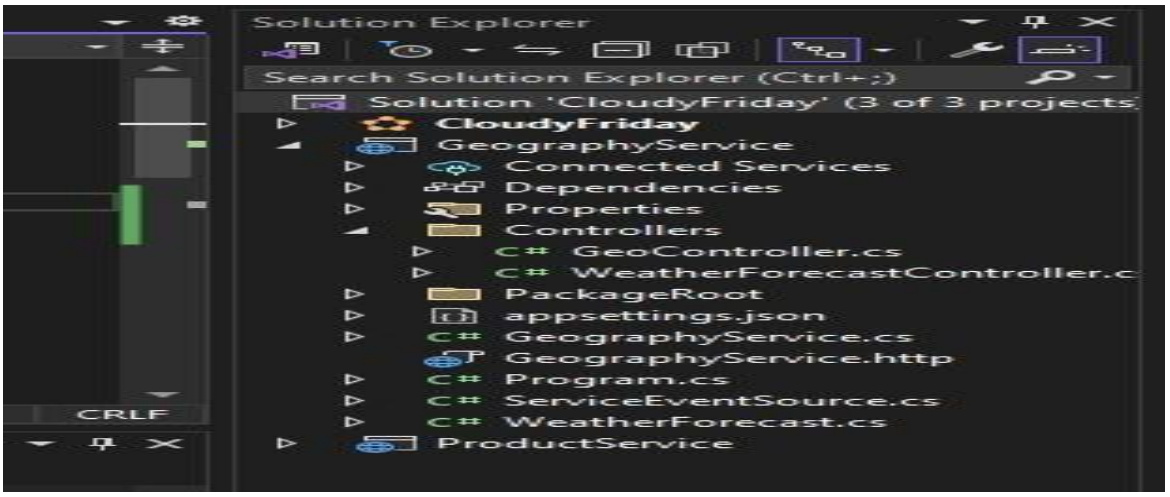


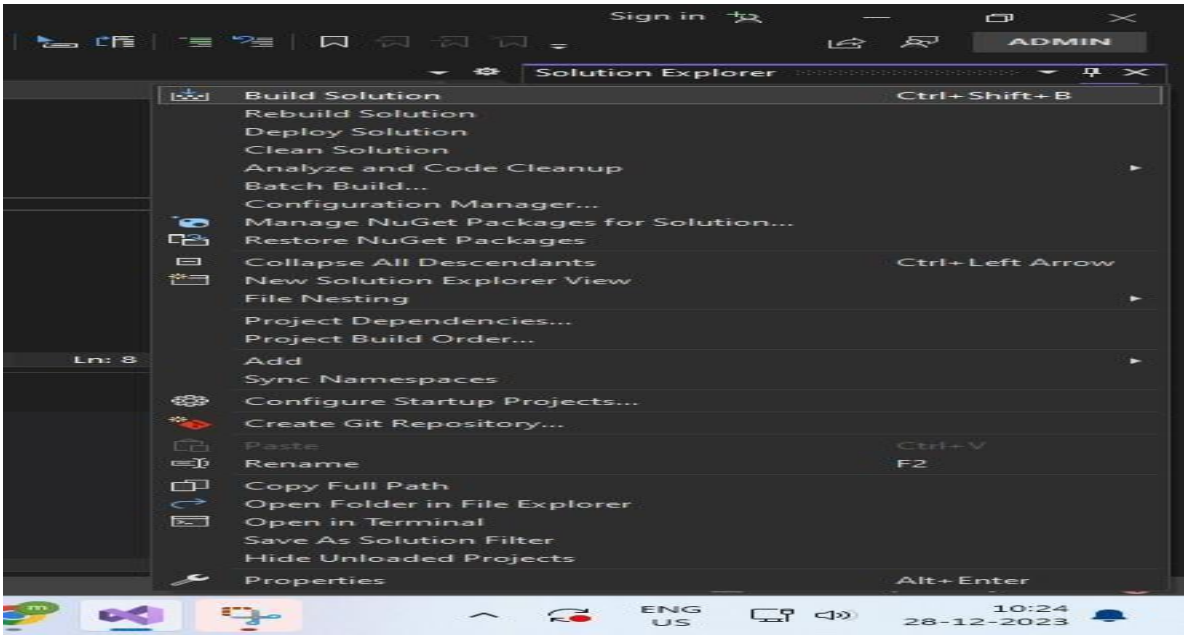


And add the following code in GeoController.cs:

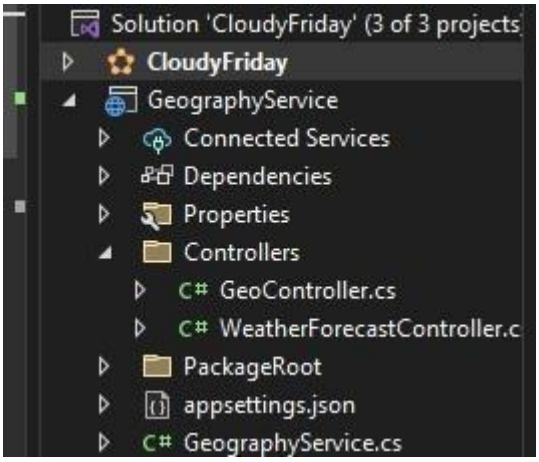
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;

namespace GeographyService.Controllers
{
class Geography
{
public String Name { get; set;  }

public int ID { get; set; }
}

[Route("api/[controller]")]
public class GeoController : Controller
{

List<Geography> geographies = new List<Geography>();
public IActionResult Index()
{
geographies.Add(new Geography() { ID = 1, Name = "GEO1" });
geographies.Add(new Geography() { ID = 2, Name = "GEO2" });
return new JsonResult(geographies);
}
}
}
```

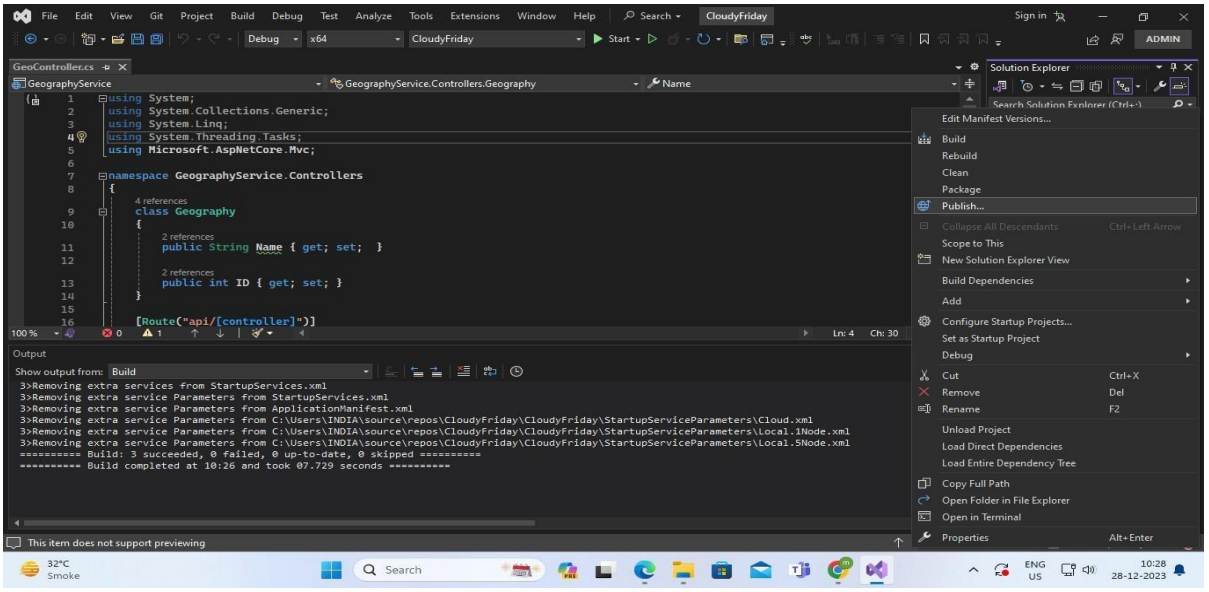
Right click on solution and click on build solution.



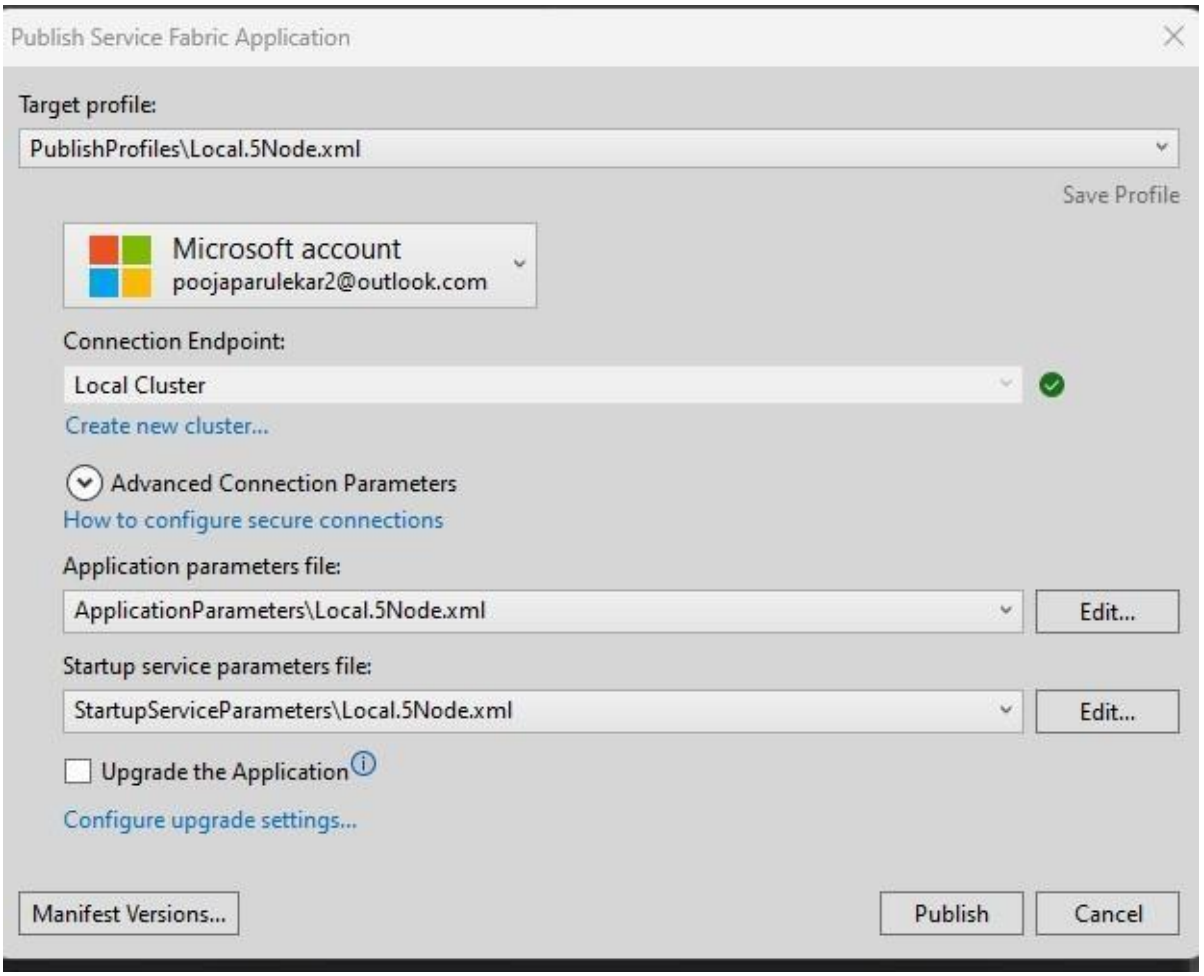


Right click on Cloudy Friday and click on publish.

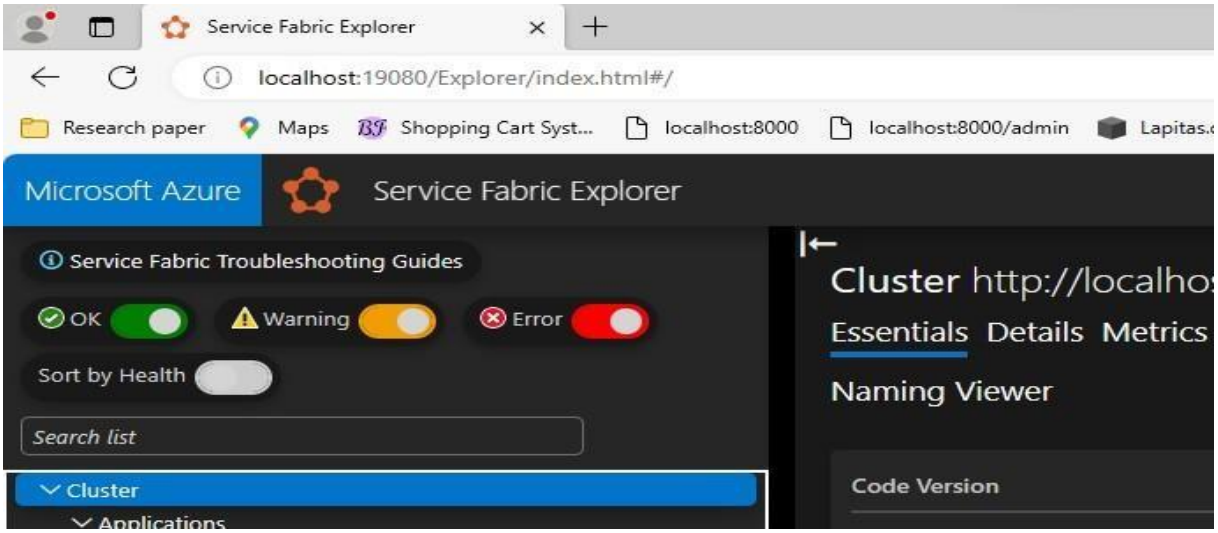


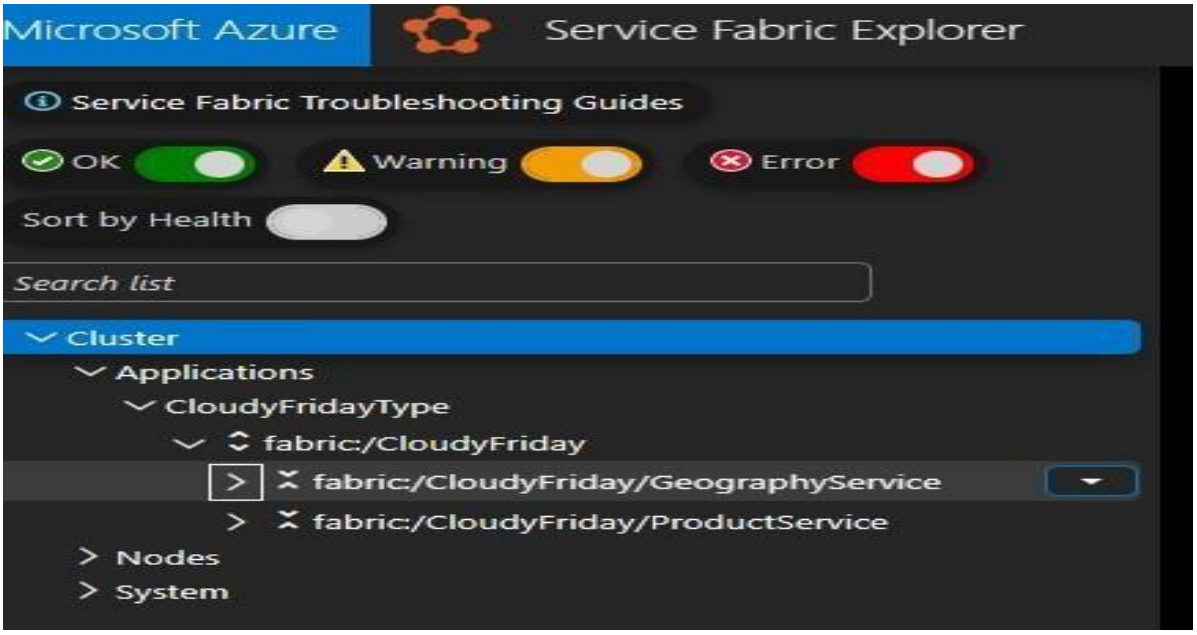


Choose Local.5Node.xml because we are using 5 node cluster. Choose Microsoft account and if not then sign in to it. And publish.

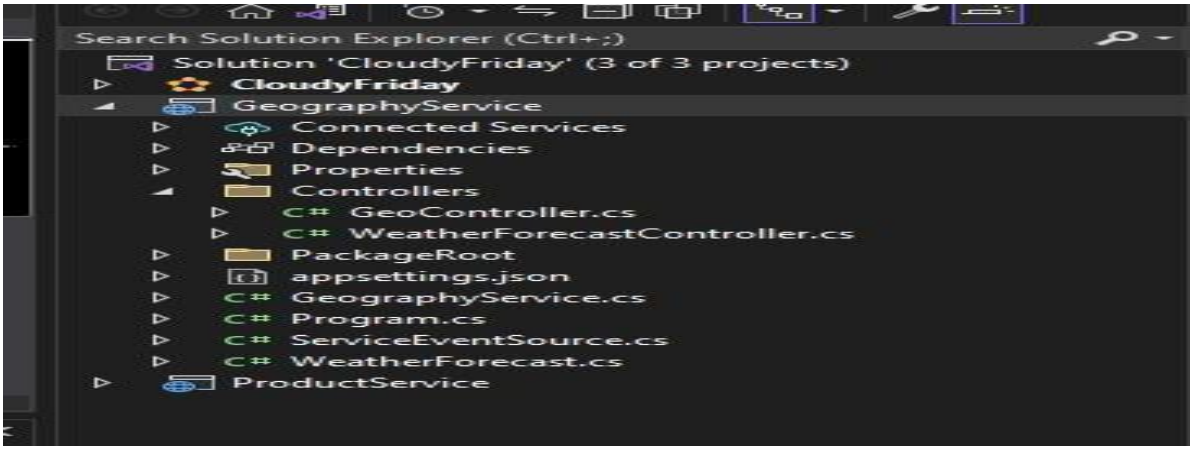


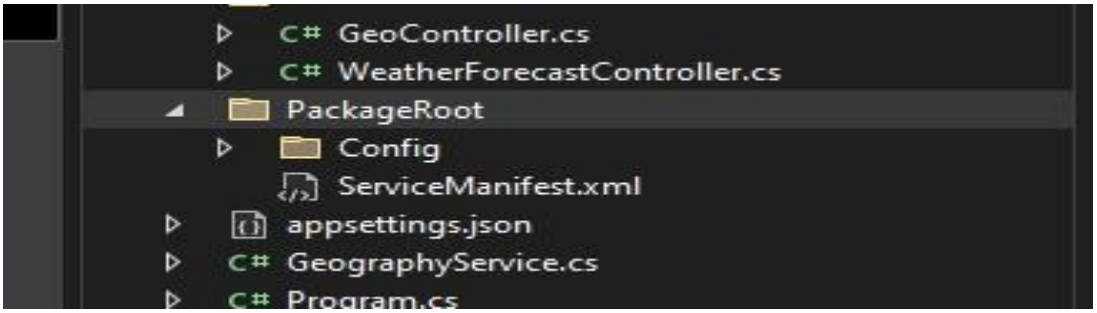
Now our project is deployed on Service Fabric .

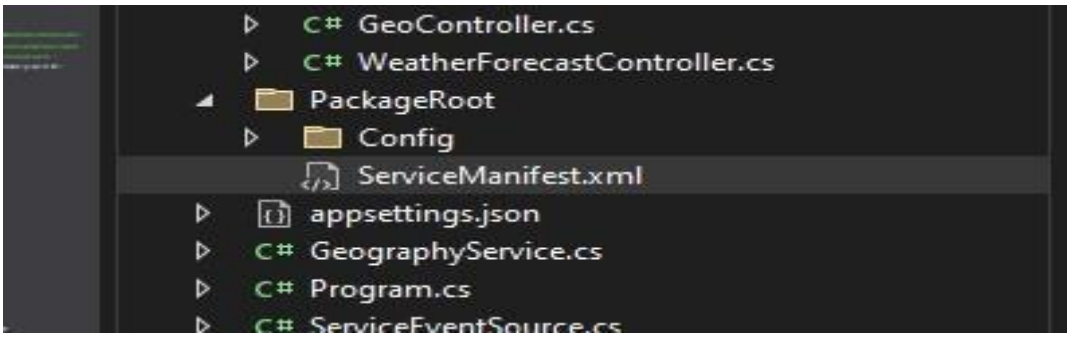




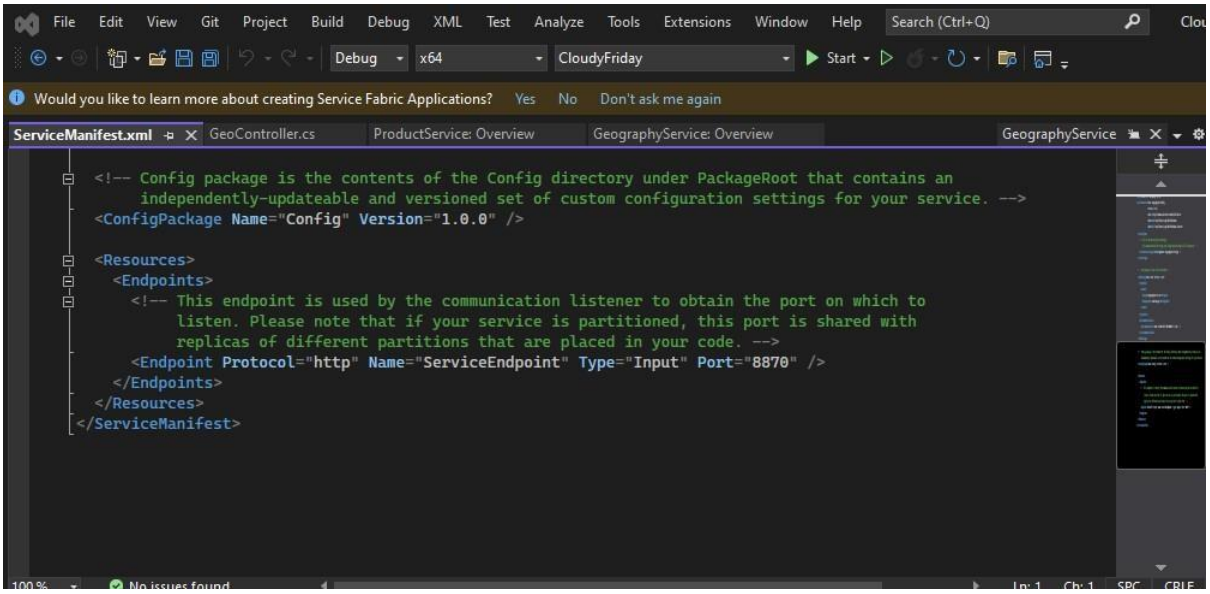
Open ServiceManifest.xml file in config in package root in GeographyService.


Search Solution Explorer (Ctrl+;)
Solution 'CloudyFriday' (3 of 3 projects)
CloudyFriday
GeographyService
Connected Services
Dependencies
Properties
Controllers
GeoController.cs
WeatherForecastController.cs
PackageRoot
appsettings.json
GeographyService.cs
Program.cs
ServiceEventSource.cs
WeatherForecast.cs
ProductService


GeoController.cs
WeatherForecastController.cs
PackageRoot
Config
ServiceManifest.xml
appsettings.json
GeographyService.cs
Program.cs

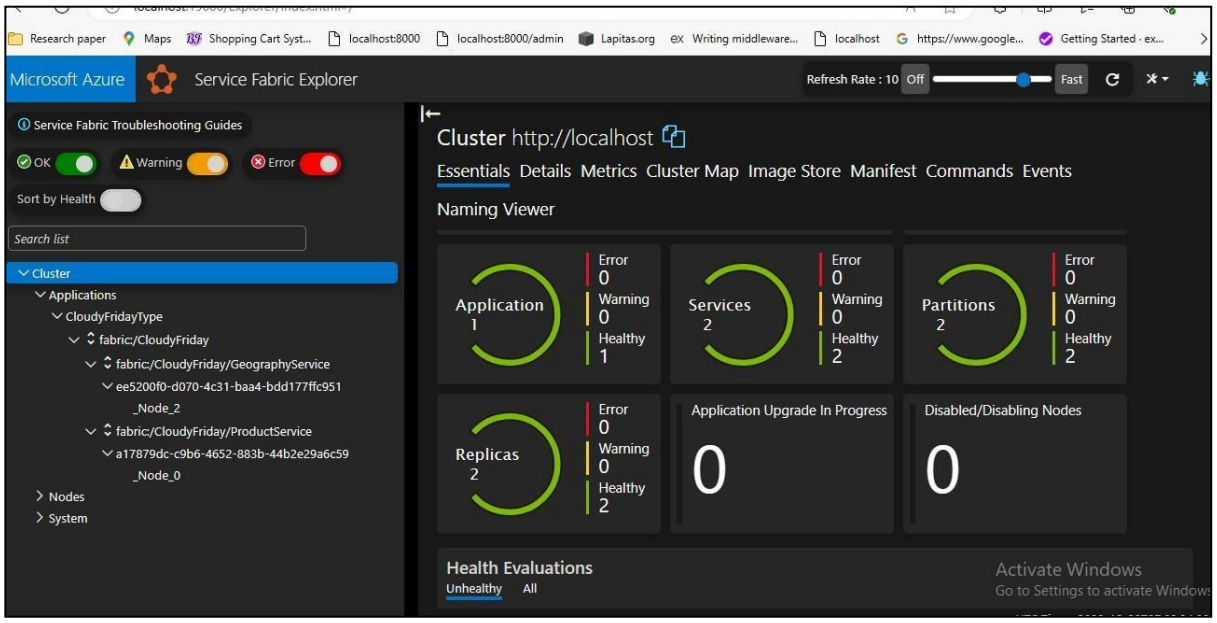

GeoController.cs
WeatherForecastController.cs
PackageRoot
Config
ServiceManifest.xml
appsettings.json
GeographyService.cs
Program.cs

At the End of File check the port number . and go to browser and type localhost:8870/ api/Geo and enter. 8870 is a port number and Geo is a controller name.



And here's the Output.

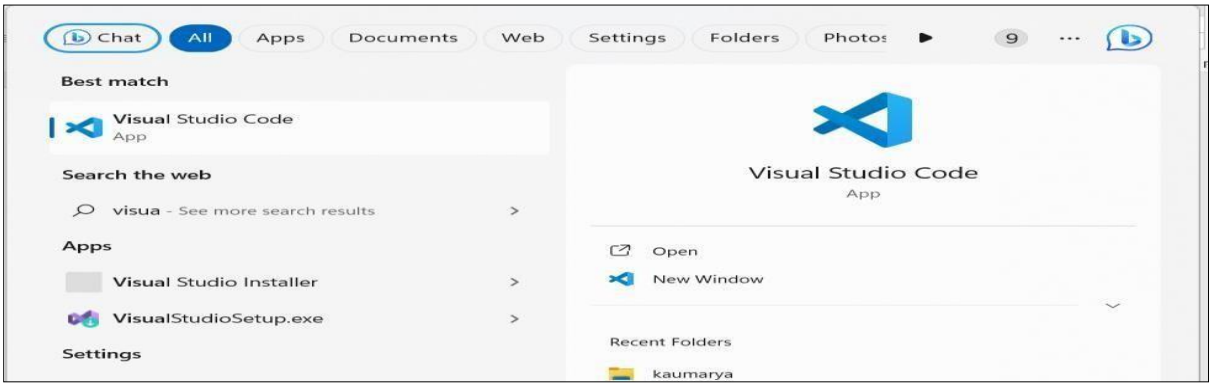
Service Fabric shows all nodes



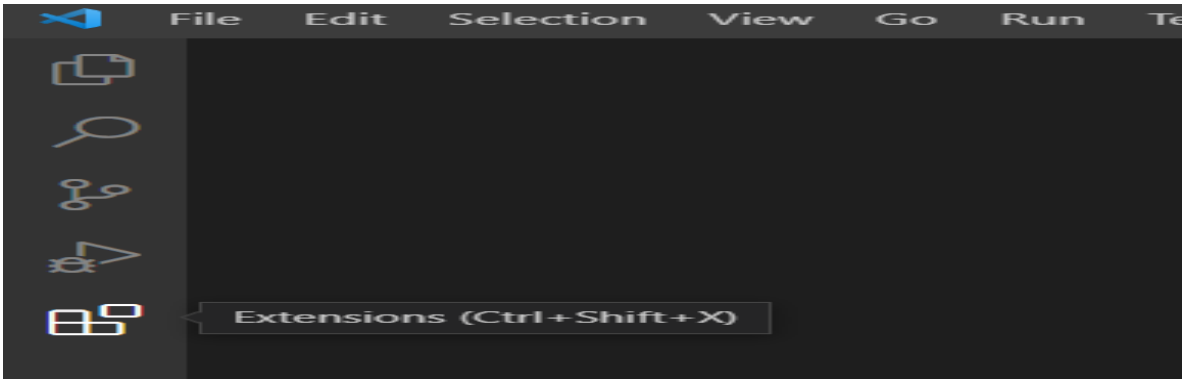
Practical No: 2

Develop a Spring Boot API.

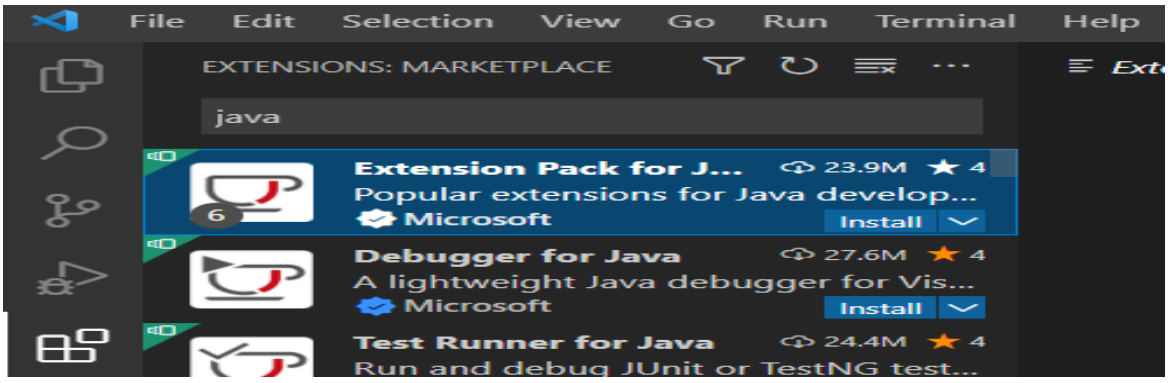
First open a visual studio code.



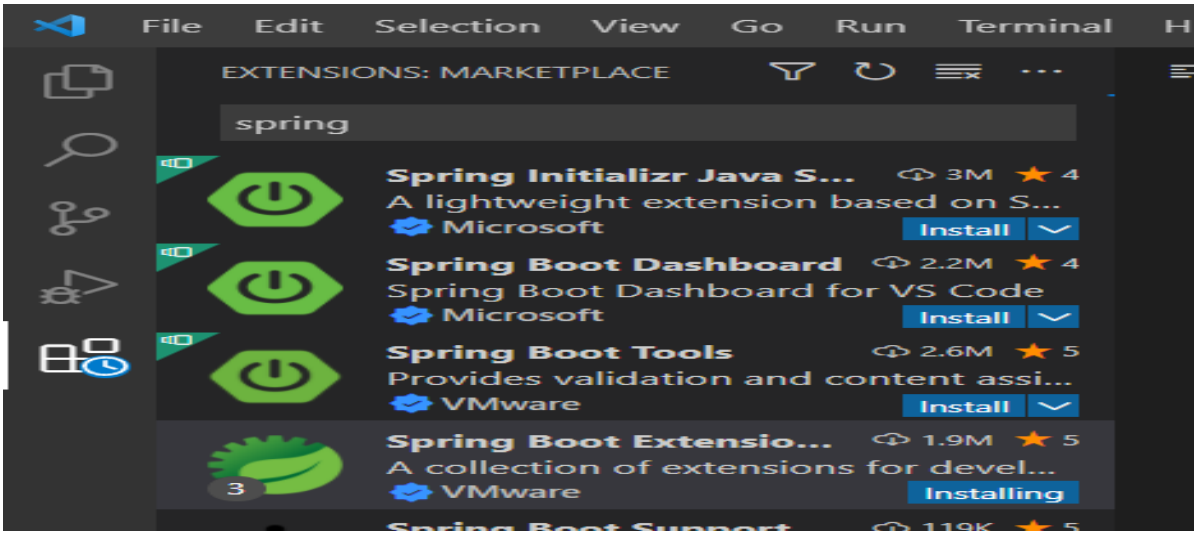
In visual studio code go to extensions.



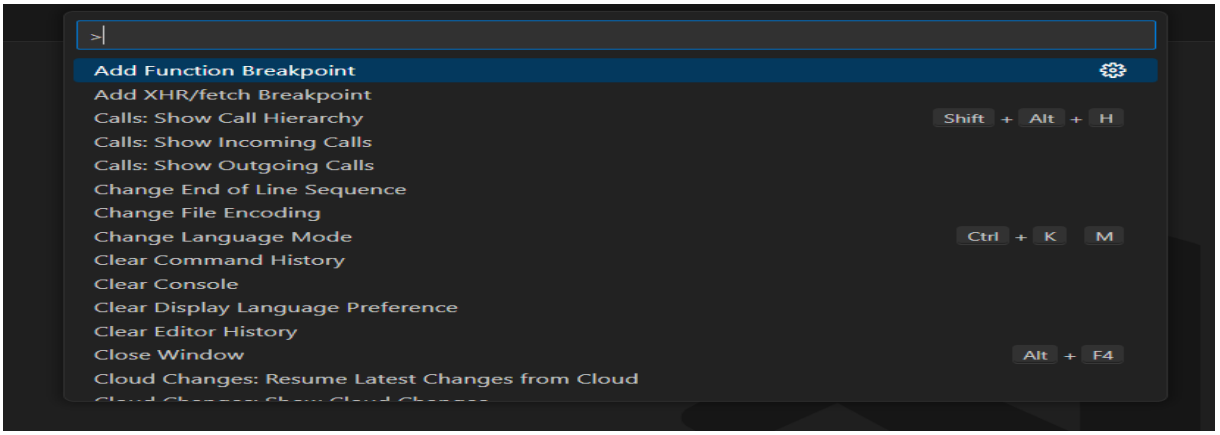
And search for java extension pack and install it.



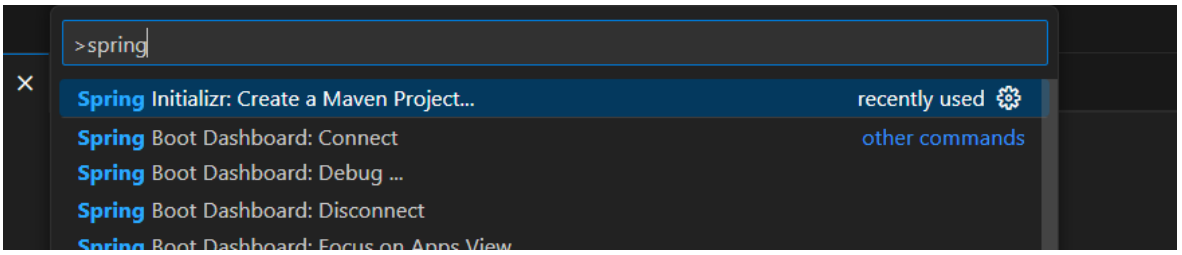
After installing java extension pack now install spring boot extension pack .



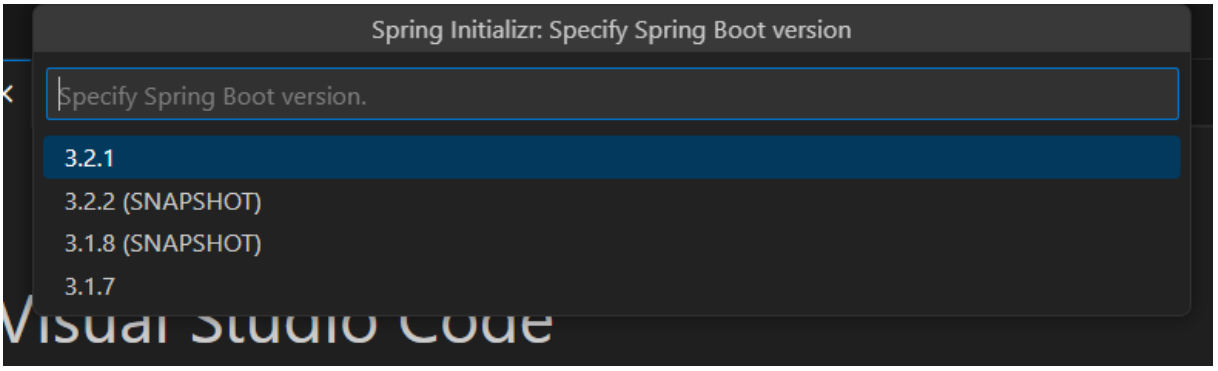
After installing extensions, we are going to implement our spring boot application . for that first of all open command palette using ctrl + shift + p .



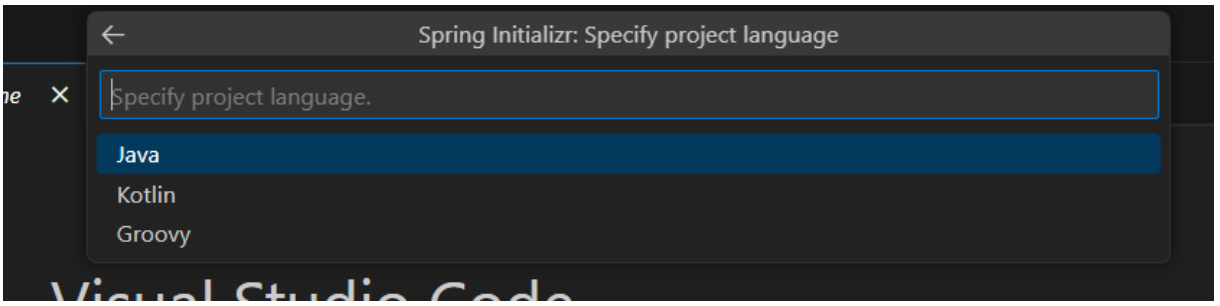
Search for spring and select Spring initializr : create a maven project.



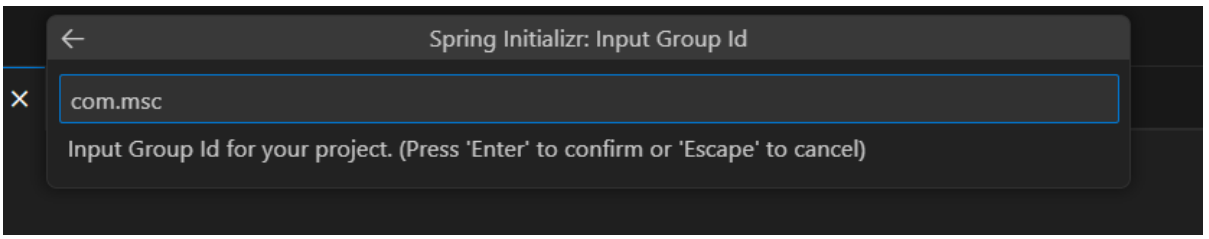
Now , select a spring boot version 3.2.1 .



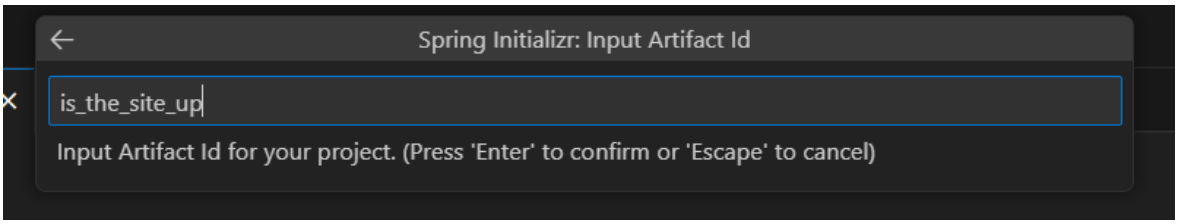
Select project language java.



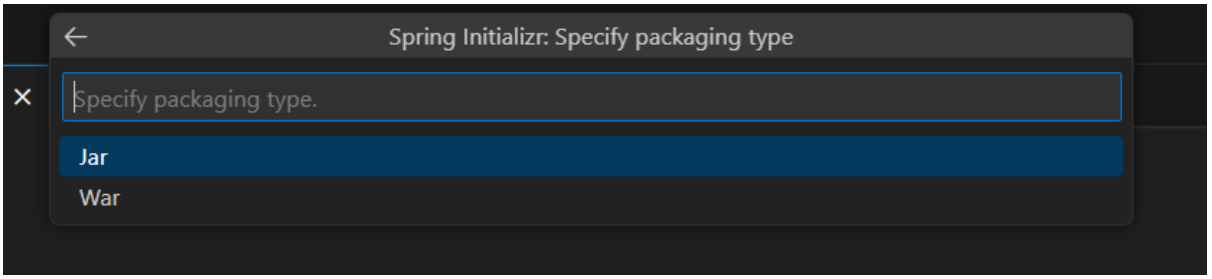
Give Group id com.msc and press enter.



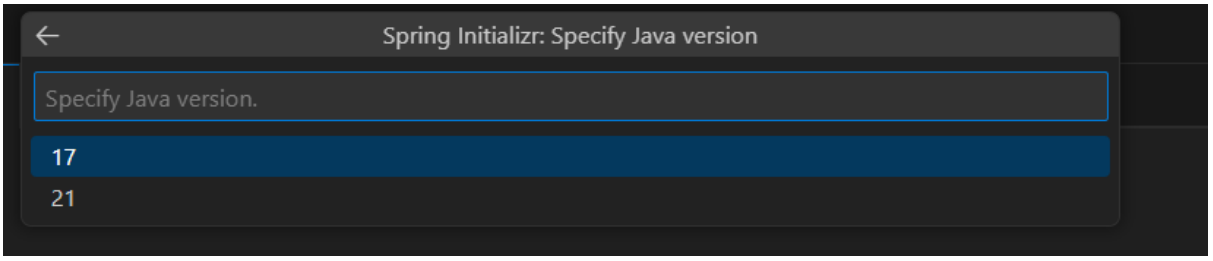
Give artifact id is_the_site_up and press enter.



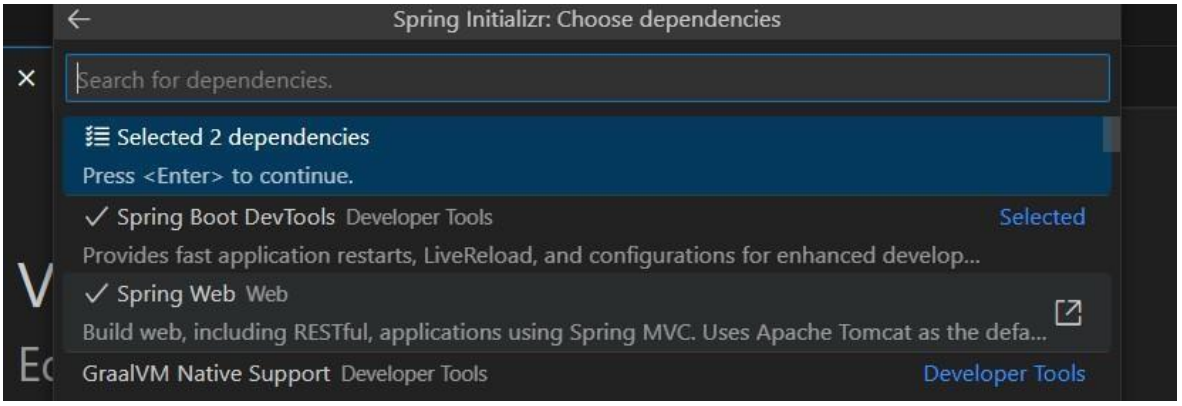
Select packaging type jar.



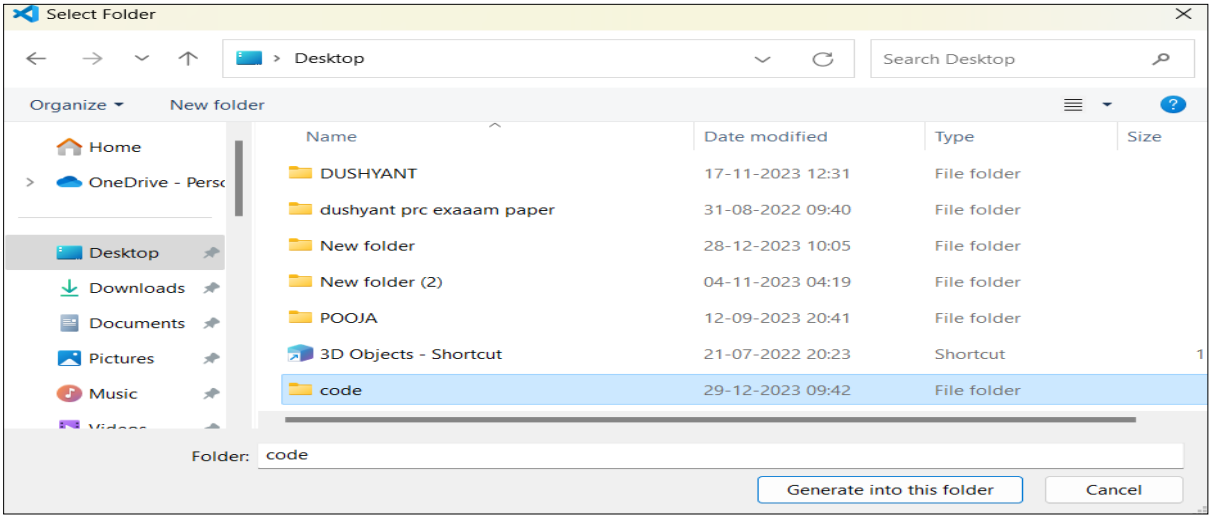
Select java version 17.



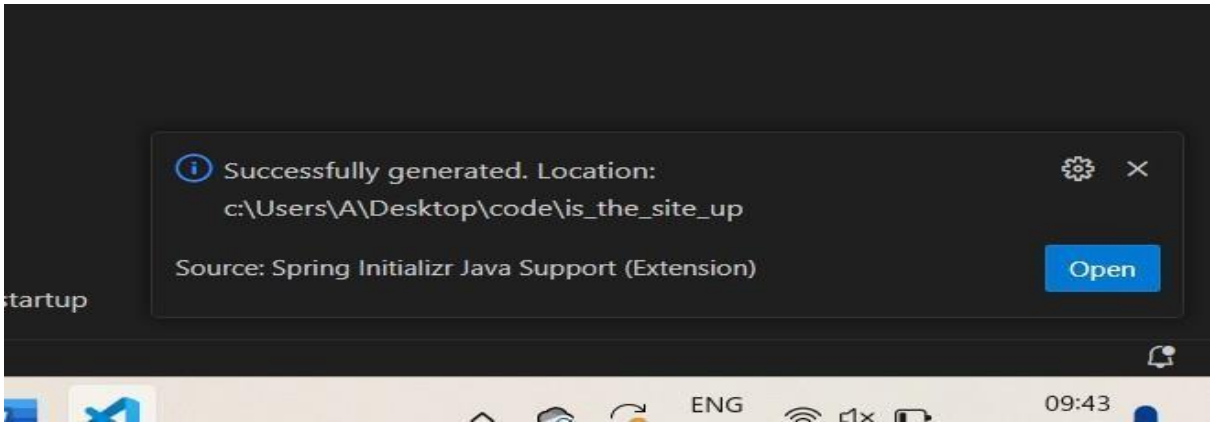
Select the dependencies spring boot devtools and spring web.



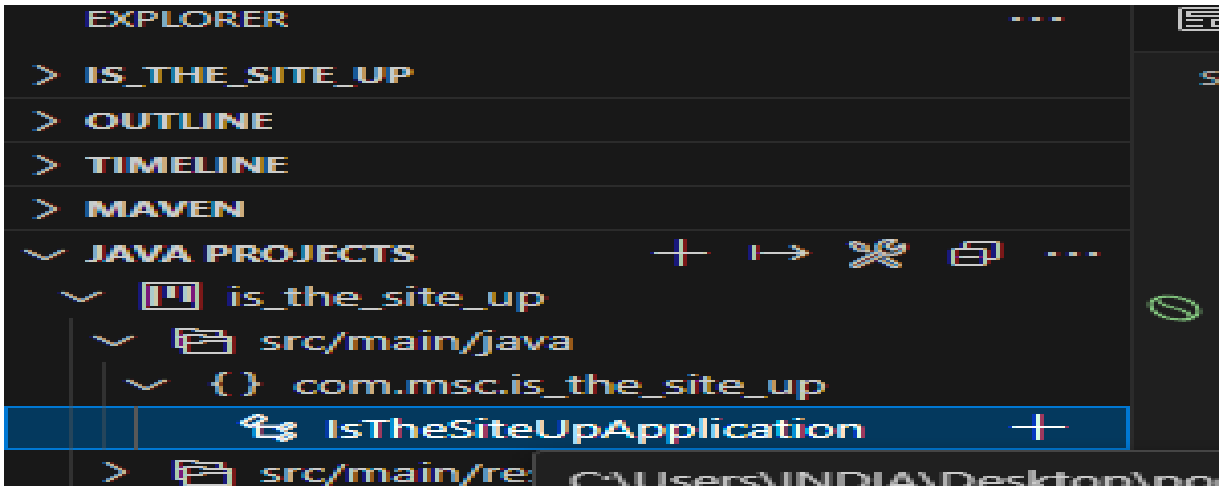
Select the where you want to generate your spring boot application and click on generate into this folder.



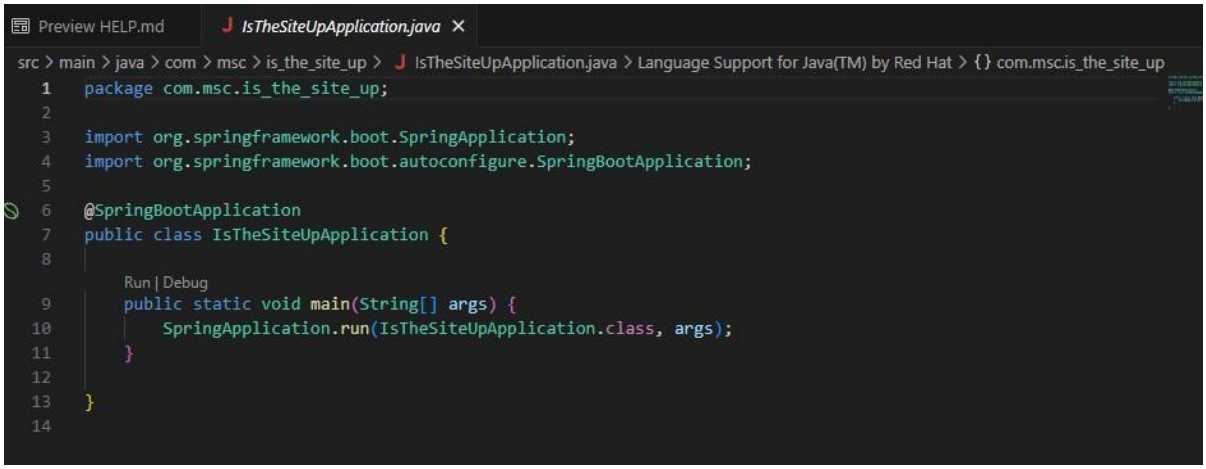
On the right-hand side notification will pop up for open a project .



In application got to java projects - src/main/java – com.msc.is_the_site_up and open the application.

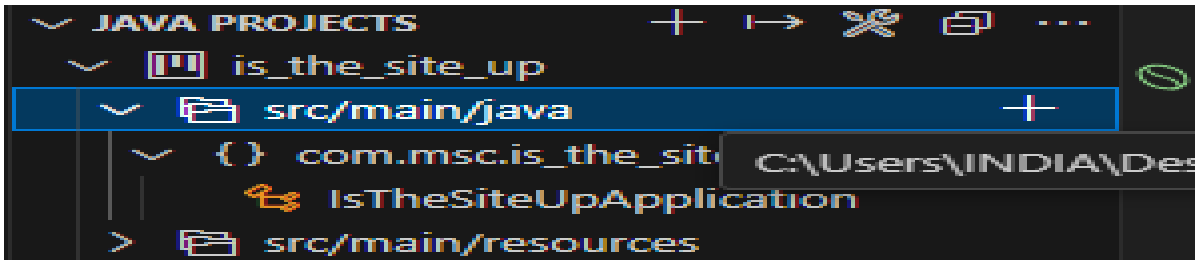


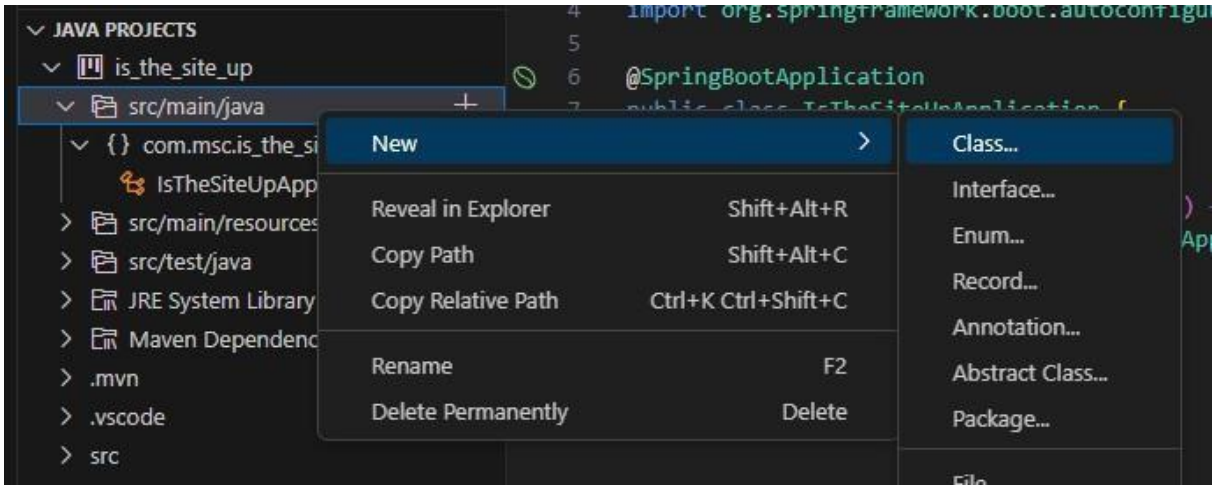
Run the IsTheSiteApplication.java and check the port where application is running.

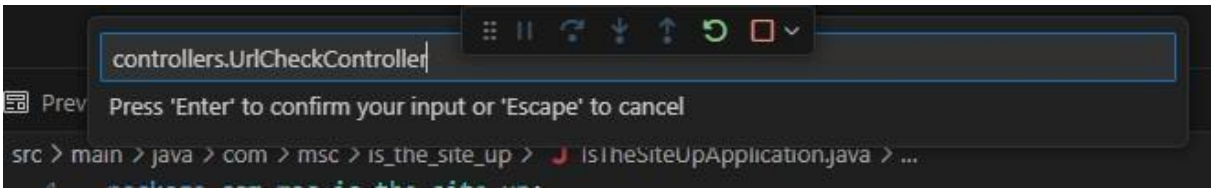


```
2023-12-30T09:44:51.324+05:30  INFO 10148 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initial
ization completed in 819 ms
2023-12-30T09:44:51.626+05:30  INFO 10148 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on por
t 35729
2023-12-30T09:44:51.673+05:30  INFO 10148 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http)
with context path ''
```

Right click on src/main/java and add new class and name its controllers.UrlCheckController.







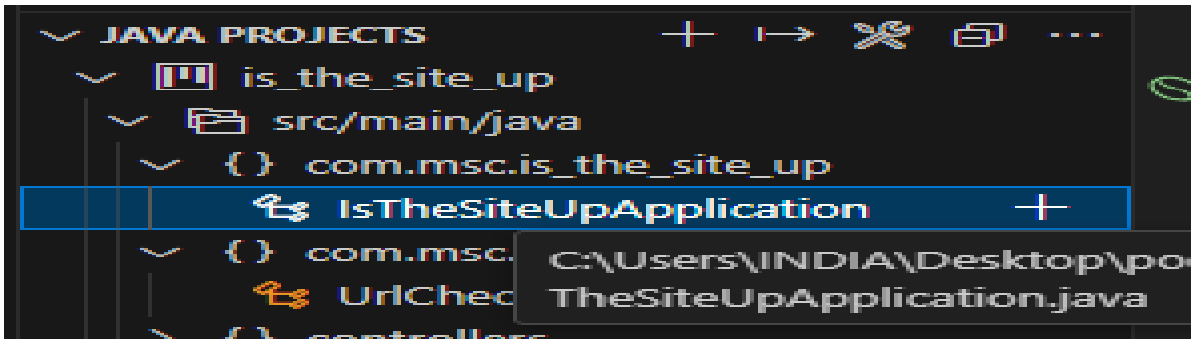
Write a following code in UrlCheckController.

Code :

```
package com.msc.is_the_site_up.controllers;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
@RestController
public class UrlCheckController {
private final String SITE_IS_UP = "Site is Up!";
private final String SITE_IS_DOWN = "Site is Down!";
private final String URL_INCORRECT = "Provided URL is Incorrect,Please Check URL!";
@GetMapping("/check")
public String GetUrlStatusMsg(@RequestParam String url) {
String returnMsg = "";
try {
URL urlObj = new URL(url);
HttpURLConnection conn = (HttpURLConnection)
urlObj.openConnection();
conn.setRequestMethod("GET");
conn.connect();
int responseCodeCategory = conn.getResponseCode() / 100;
/*
* if (responseCodeCategory != 2 || responseCodeCategory !=
3) {
* returnMsg = SITE_IS_DOWN;
* } else {
* returnMsg = SITE_IS_UP;
* }
```

```
*/
if (responseCodeCategory == 2 || responseCodeCategory == 3)
{
returnMsg = SITE_IS_UP;
} else {
returnMsg = SITE_IS_DOWN;
}
} catch (MalformedURLException e) {
returnMsg = URL_INCORRECT;
} catch (IOException e) {
returnMsg = SITE_IS_DOWN;
}
return returnMsg;
}
}
```

Before running the application kill the terminal and again run the application.



Output:


Cloud Application Developmen
cloud_app_dev_2_pract - Goog
WhatsApp
localhost:8080/check?url=https
localhost:8080/check?url=https://www.google.com
Site is Up!


Cloud Application Developmen
cloud_app_dev_2_pract - Goog
WhatsApp
localhost:8080/check?url=https
localhost:8080/check?url=https://www.googyul.com
Site is Down!

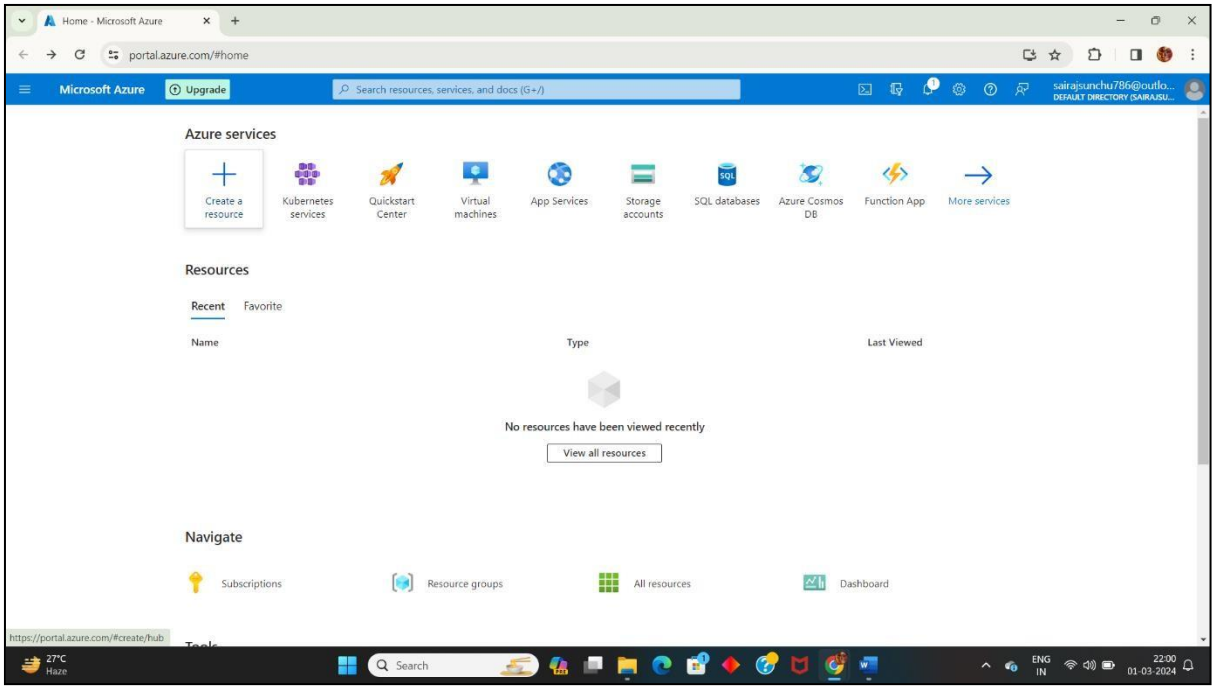
}

Practical No: 3

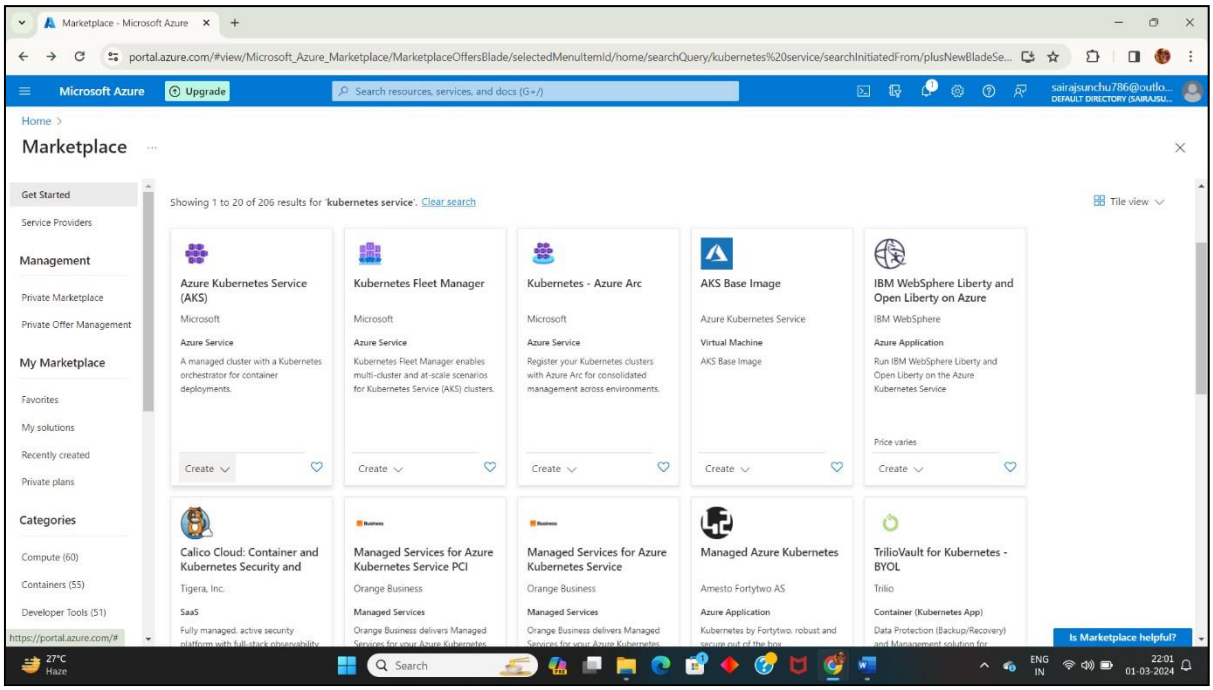
3A. Create an Azure Kubernetes Service Cluster

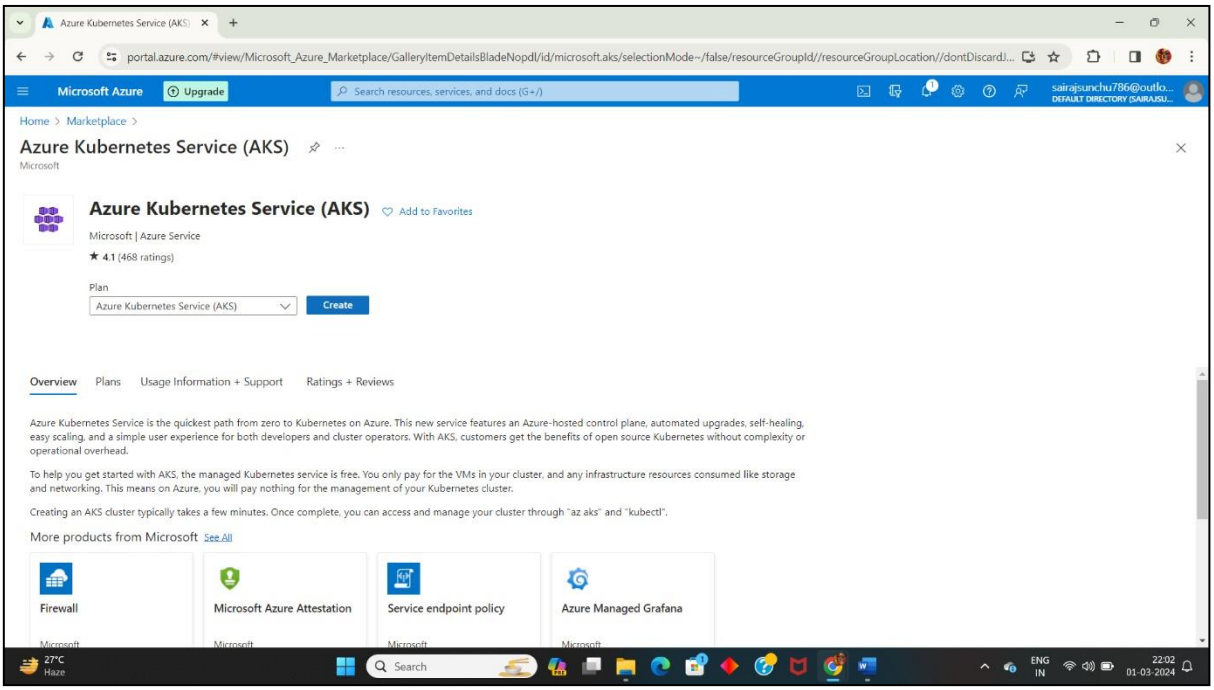
Step 1: Sign in Azure Portal (https://portal.azure.com/)

Step 2: Create Resource by clicking on Create Resource option.

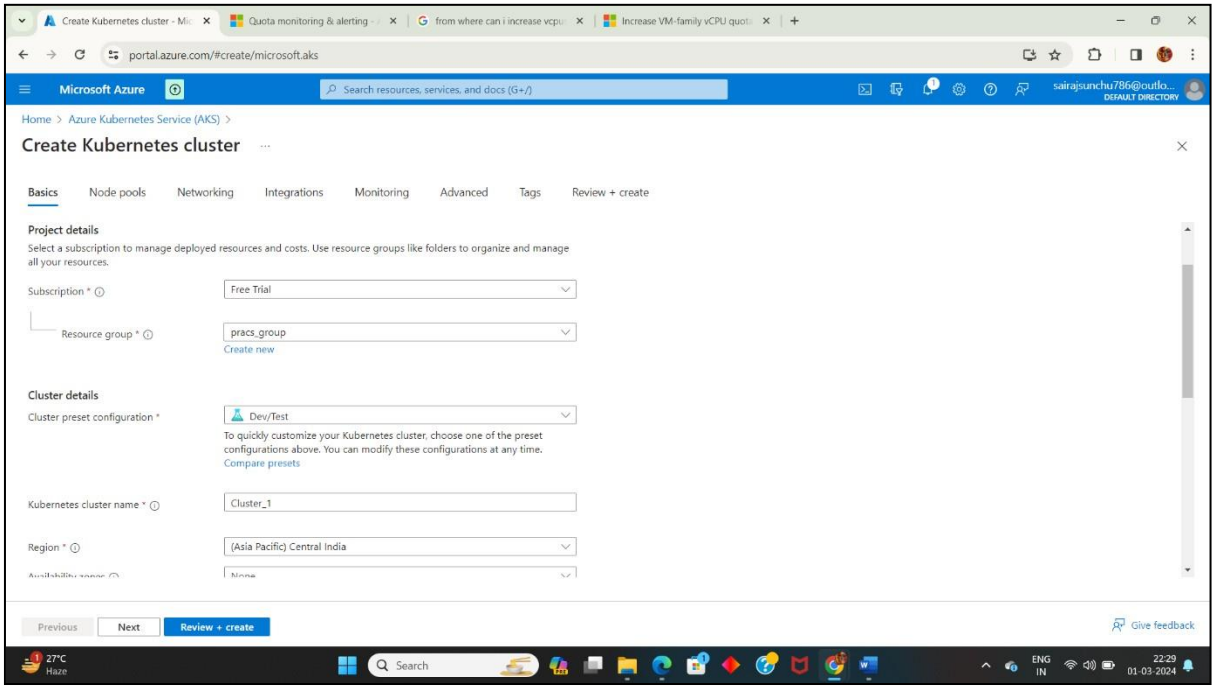


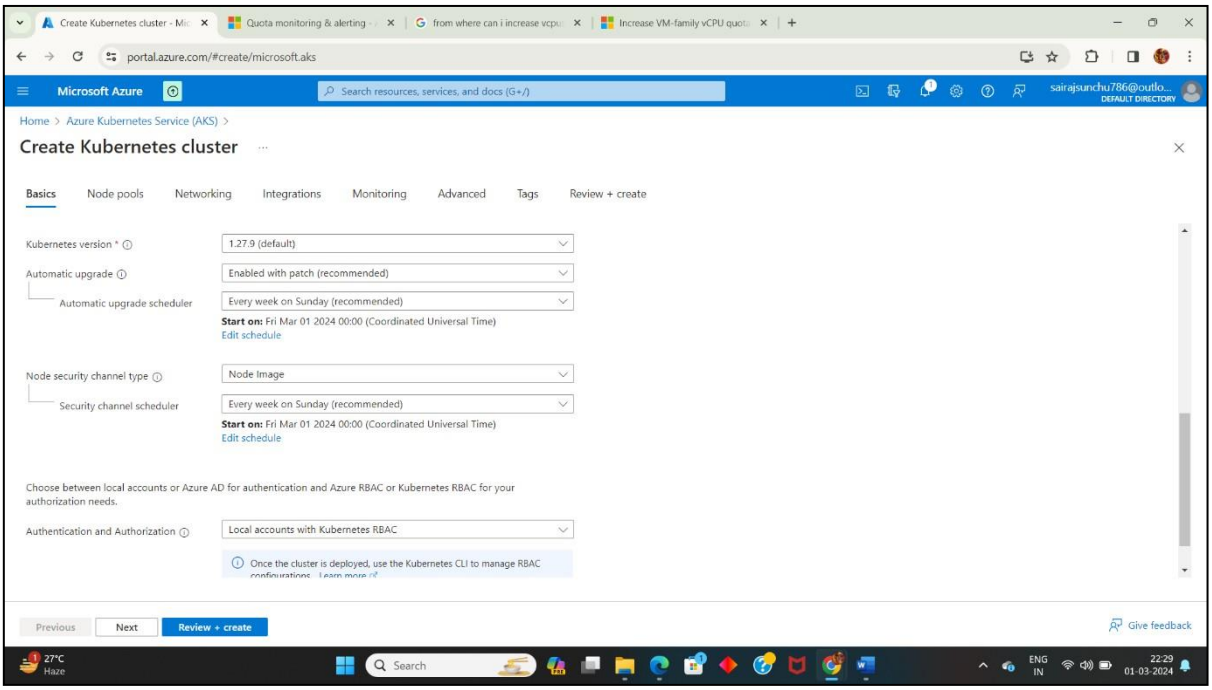
Step 3: Select Kubernetes Services



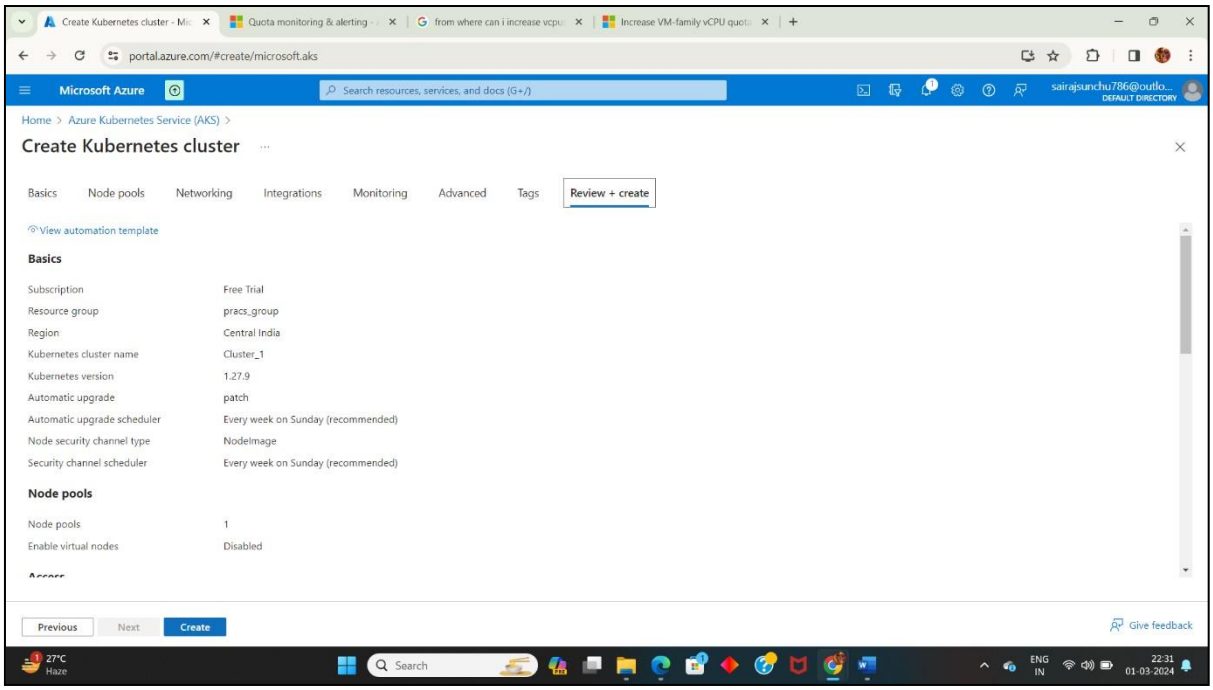


Step 4: Enter the subscription, resource group, Kubernetes Cluster name, Region, Kubernetes version, & DNS Name prefix

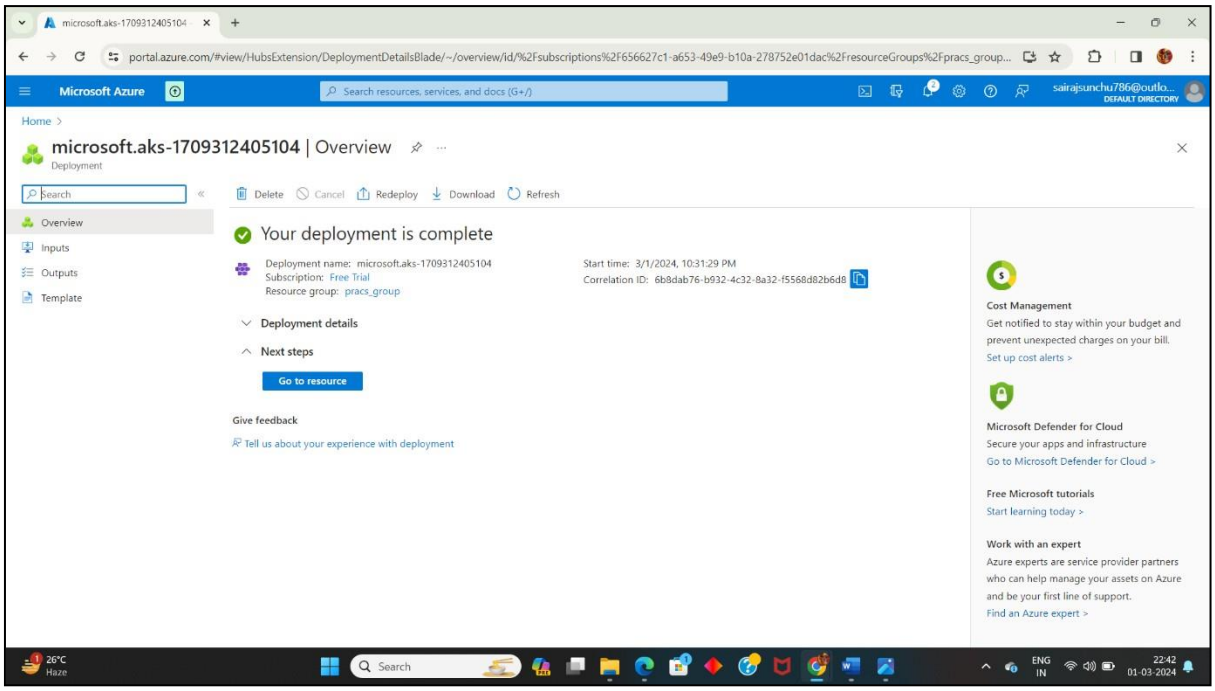




Step 5: Click on Review + Create Button



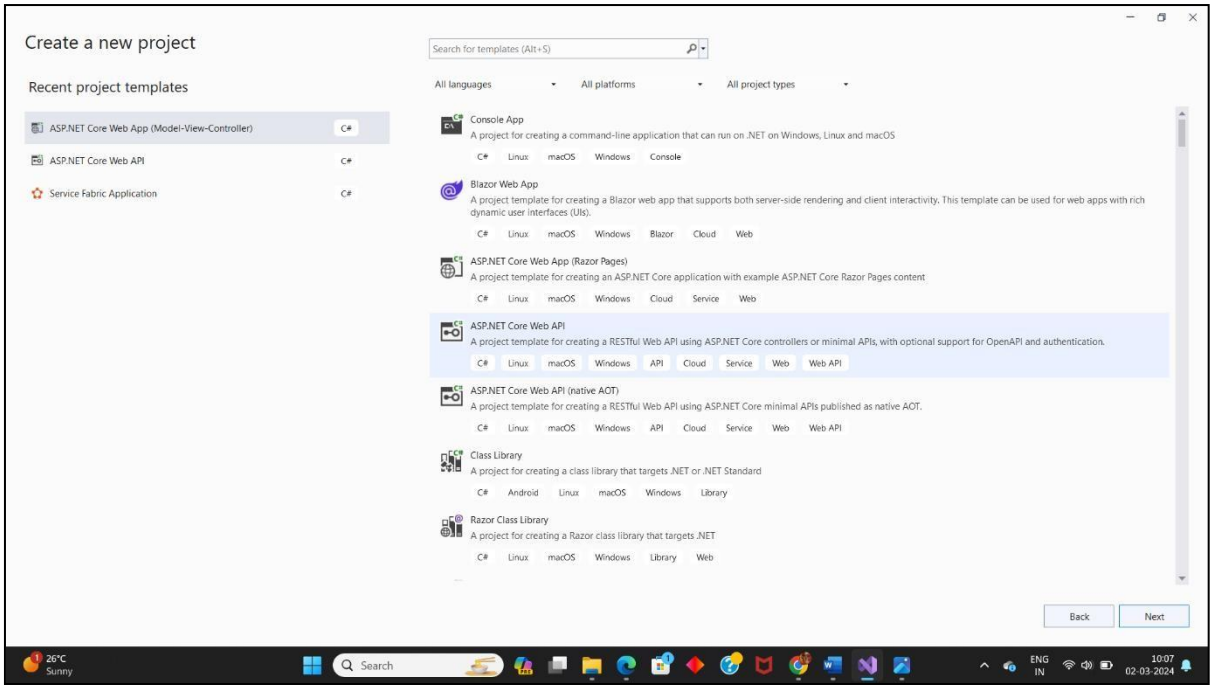
Step 6: Click on Create Button

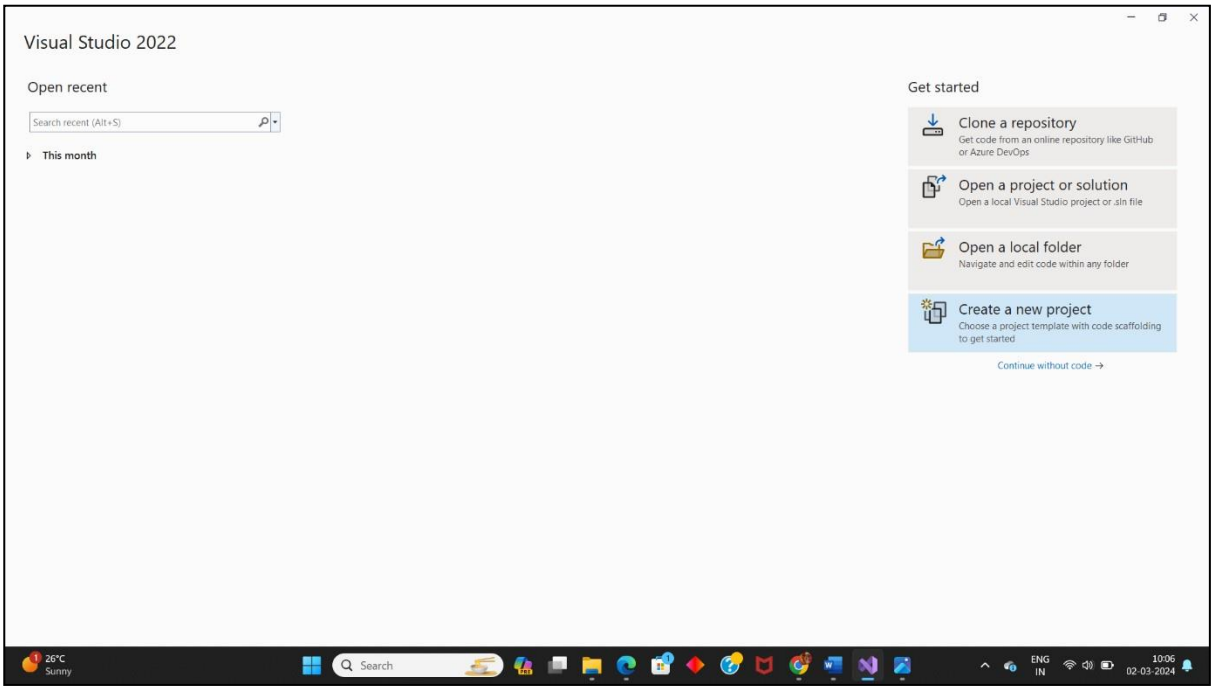


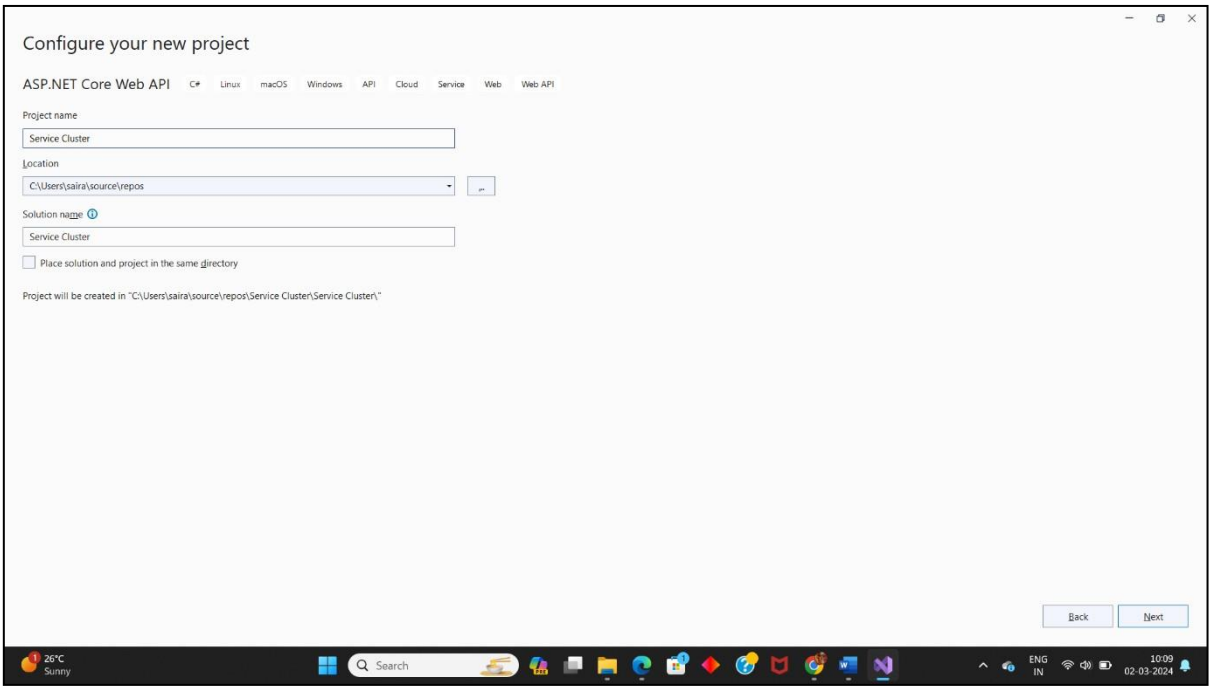
3B. Configure Visual Studio to work with an azure Kubernetes services Cluster

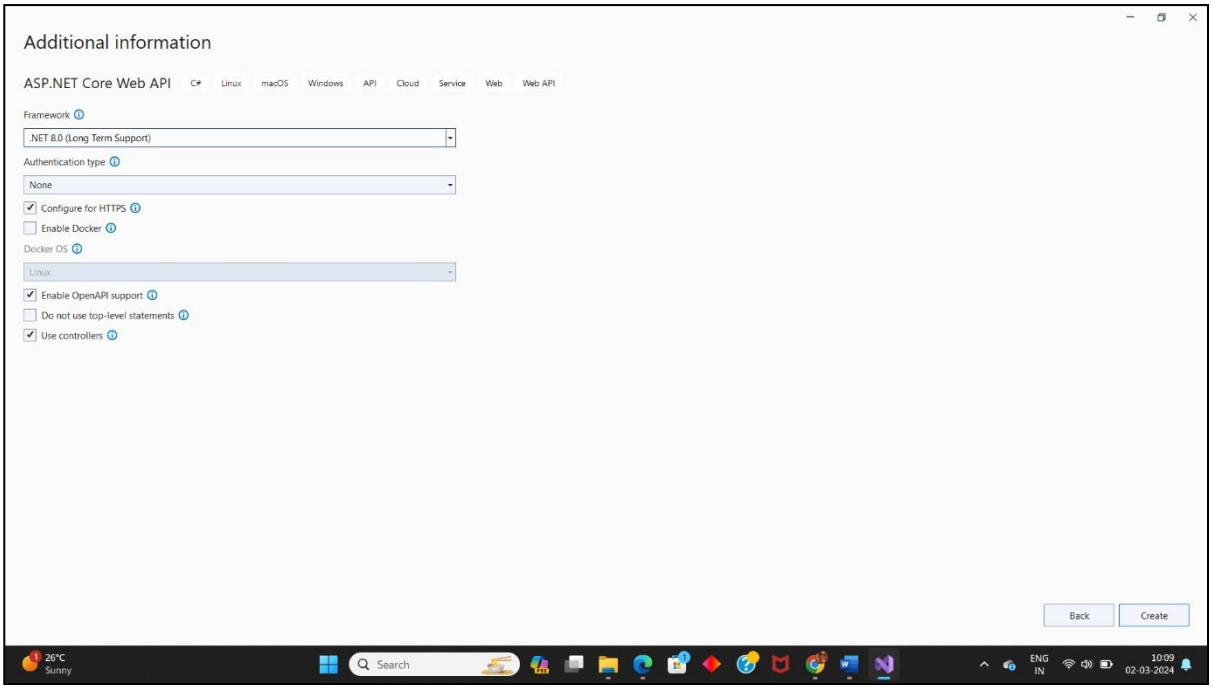
Step 1: Install Visual Studio Community 2022

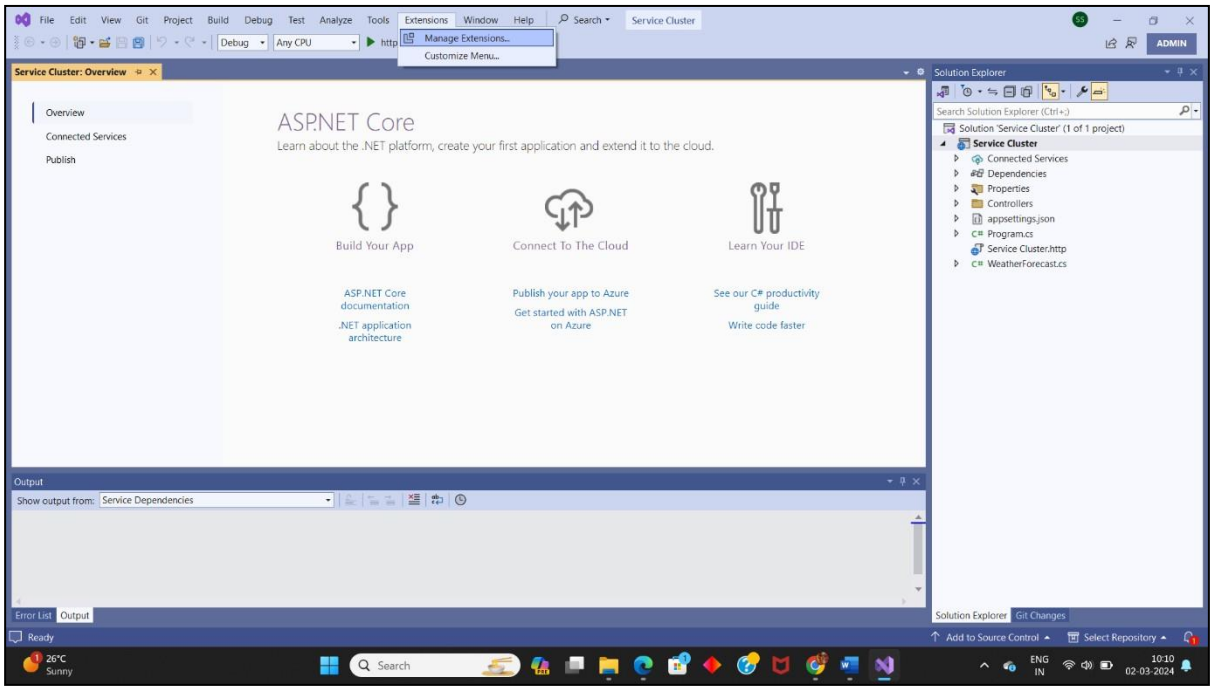
Step 2: Open Visual Studio 2022 >create a Project >Extensions>Manage Extensions





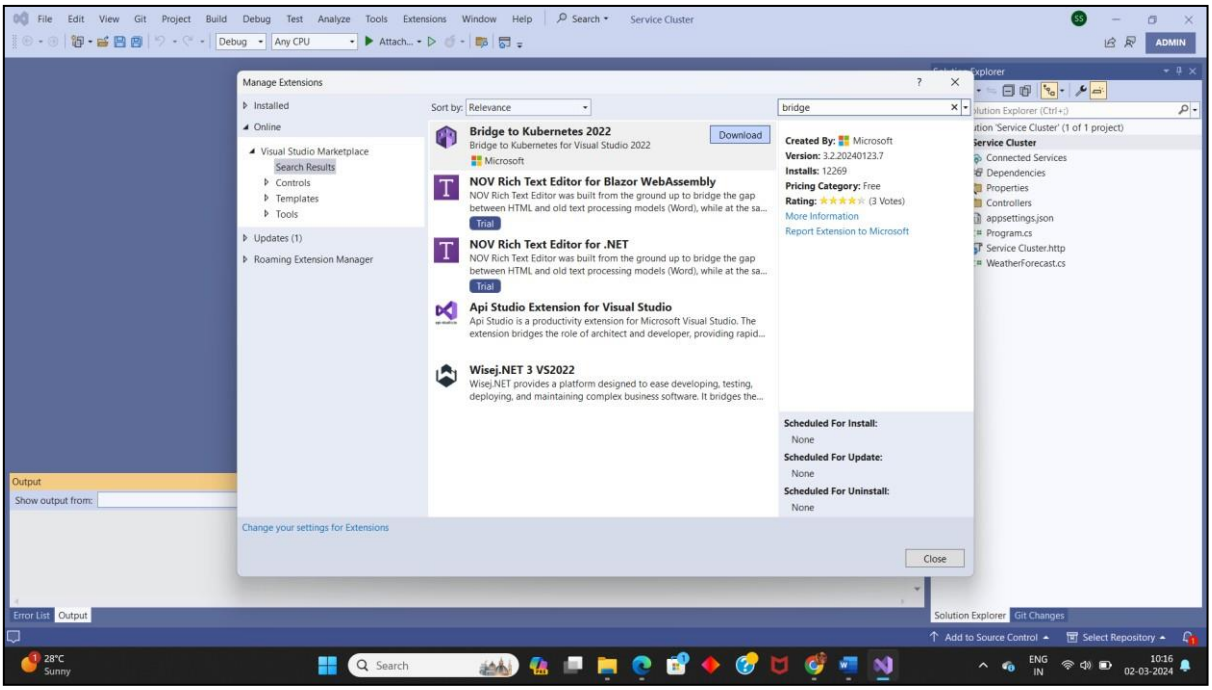

Configure your new project
ASP.NET Core Web API
C#
Linux
macOS
Windows
API
Cloud
Service
Web
Web API
Project name
Service Cluster
Location
C:\Users\saira\source\repos
Solution name
Service Cluster
Place solution and project in the same directory
Project will be created in "C:\Users\saira\source\repos\Service Cluster\Service Cluster\"
Back
Next
26°C
Sunny
Search
ENG
IN
10:09
02-03-2024

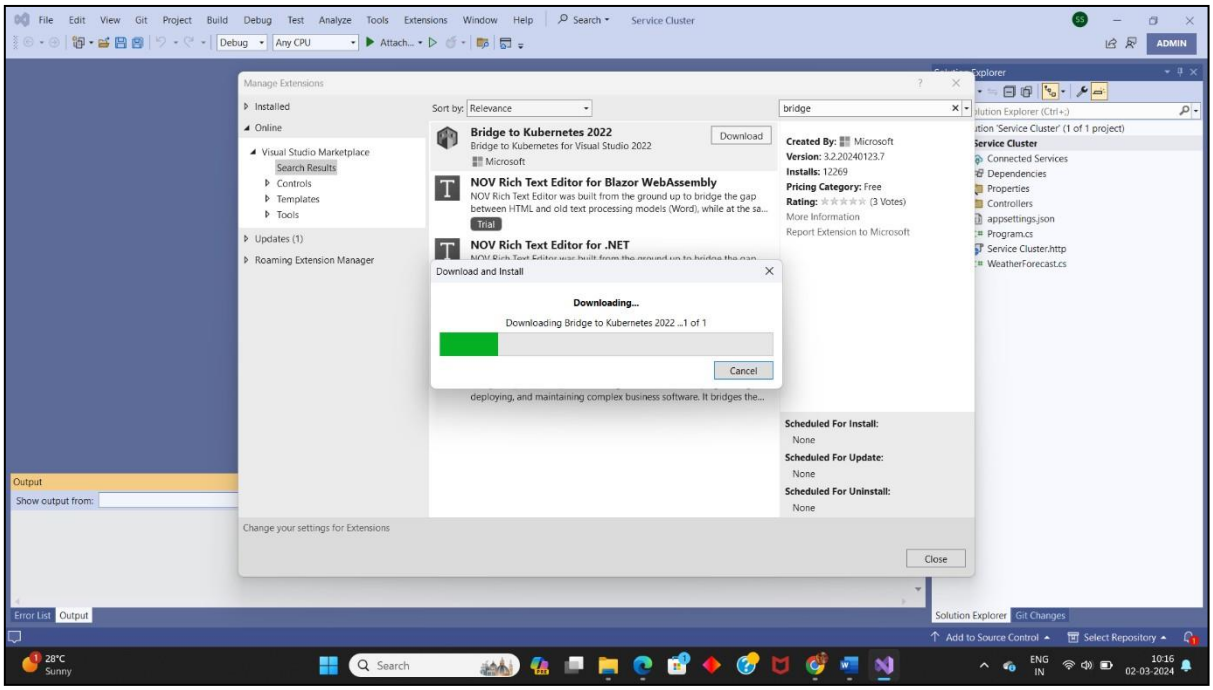

Additional information
ASP.NET Core Web API
C#
Linux
macOS
Windows
API
Cloud
Service
Web
Web API
Framework
.NET 8.0 (Long Term Support)
Authentication type
None
Configure for HTTPS
Enable Docker
Docker OS
Linux
Enable OpenAPI support
Do not use top-level statements
Use controllers
Back
Create
26°C
Sunny
Search
ENG
IN
10:09
02-03-2024



Step 3: Search for Bridge to Kubernetes 2022

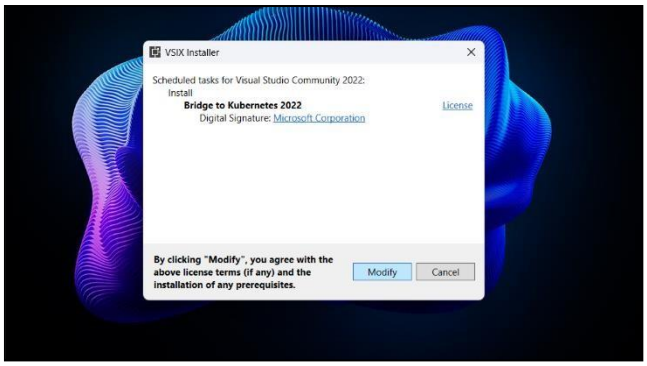
Step 4: Click on Download





Step 5: Close visual studio

Step 6: click on Modify (install Kubernetes tool extension)



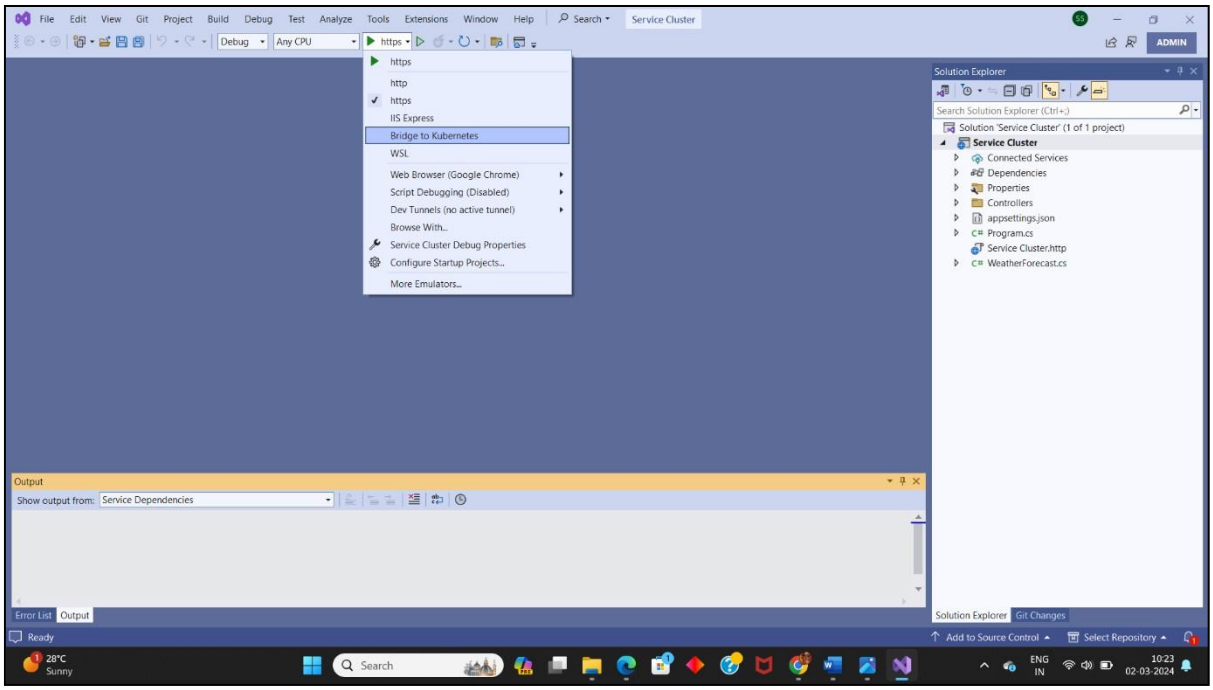
Step 7: Open Visual studio

Click on new project

Select the previous Asp.net Core Web Application

Click On Down Arrow of IIS Express

We can see option of Bridge to Kubernetes



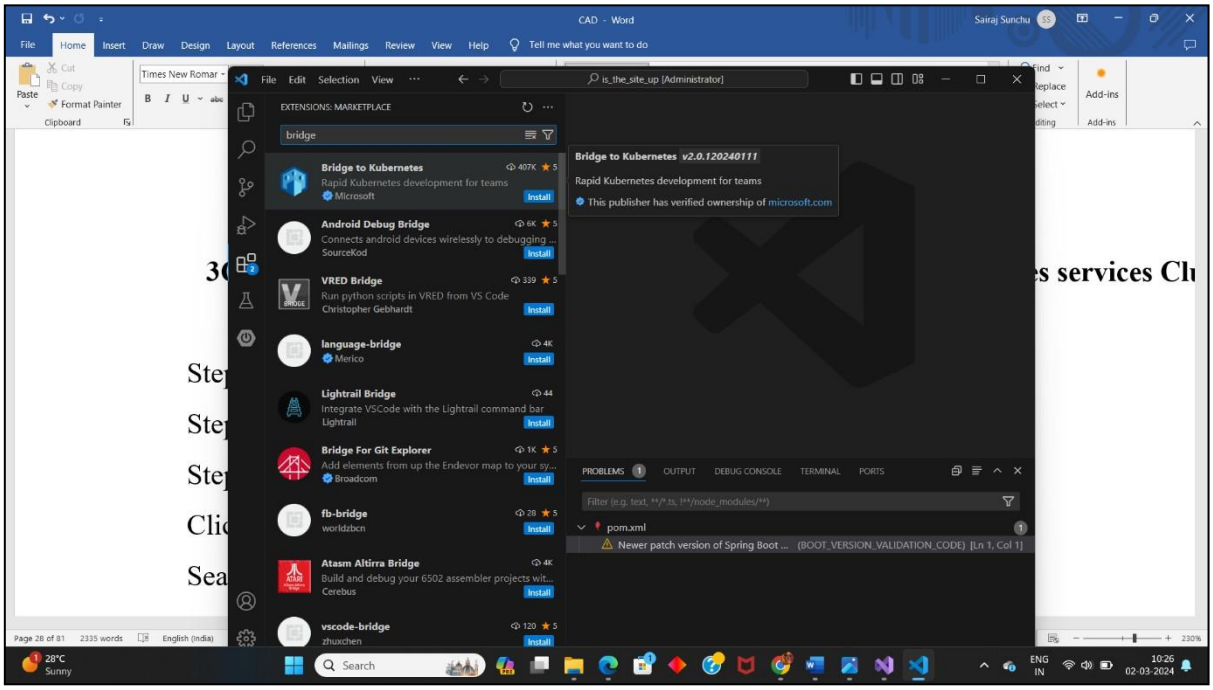
3C. Configure Visual Studio Code to work with an azure Kubernetes services Cluster

Step 1: Install visual studio code

Step 2: Open extension window

Step 3: Search for Bridge to Kubernetes 2022

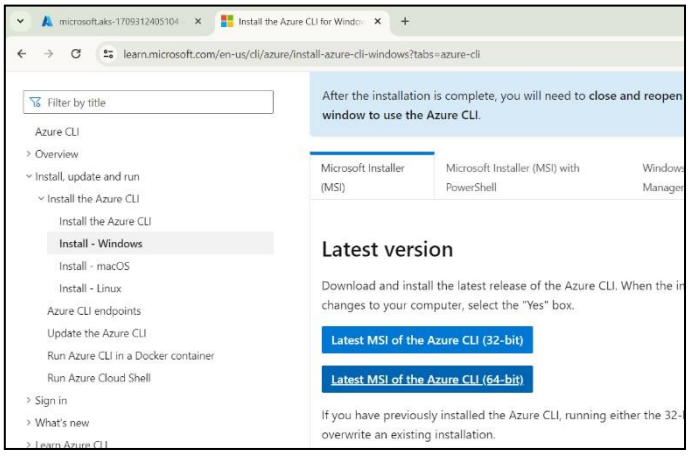
Click Install



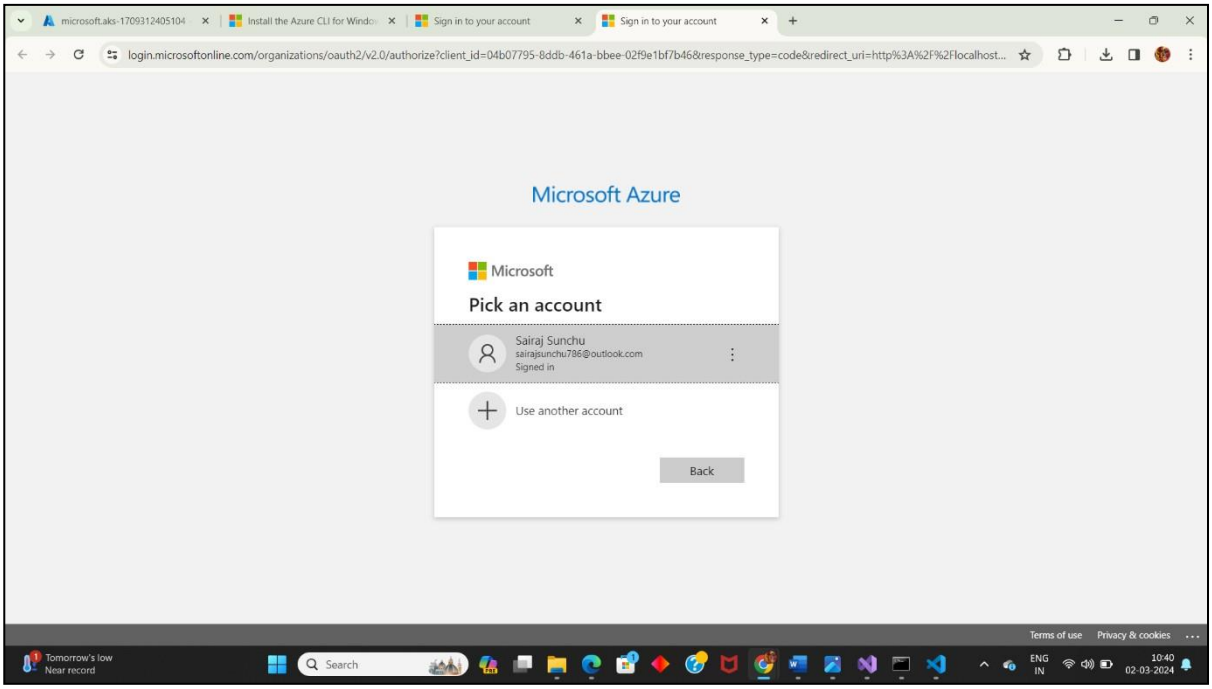
Search for Azure CLI tools and install

Go to https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli

Click on Latest release of the Azure CLI, download the msi and run

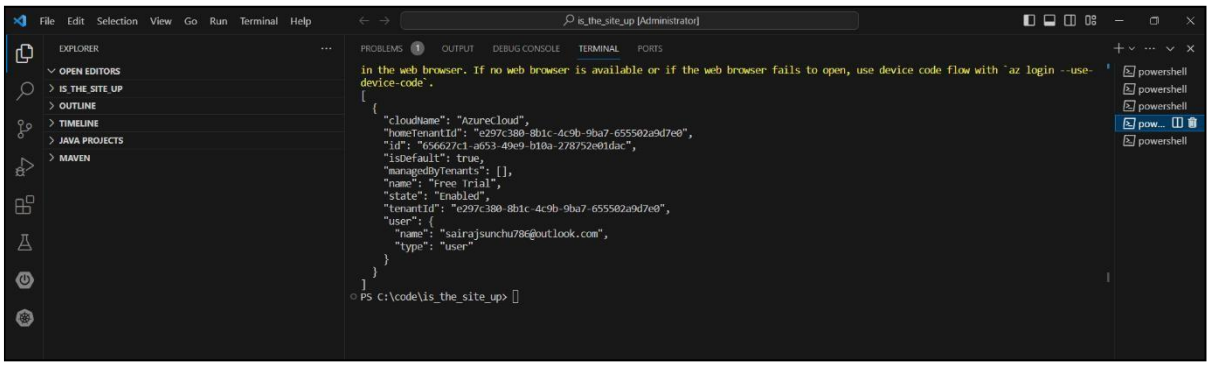


Open command prompt (Run as Administrator) and type az login



Step 4: Close the Visual Studio code and restart application.

Type the following command on the terminal. Output on Visual Studio Code

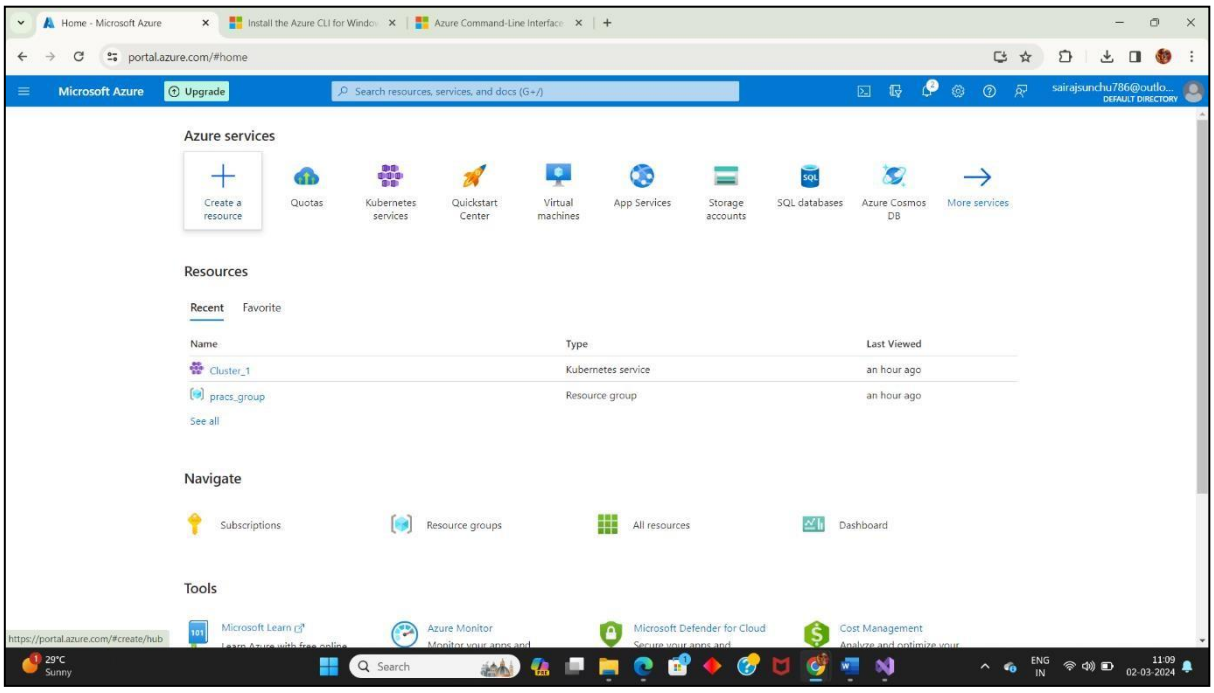


Practical No: 4

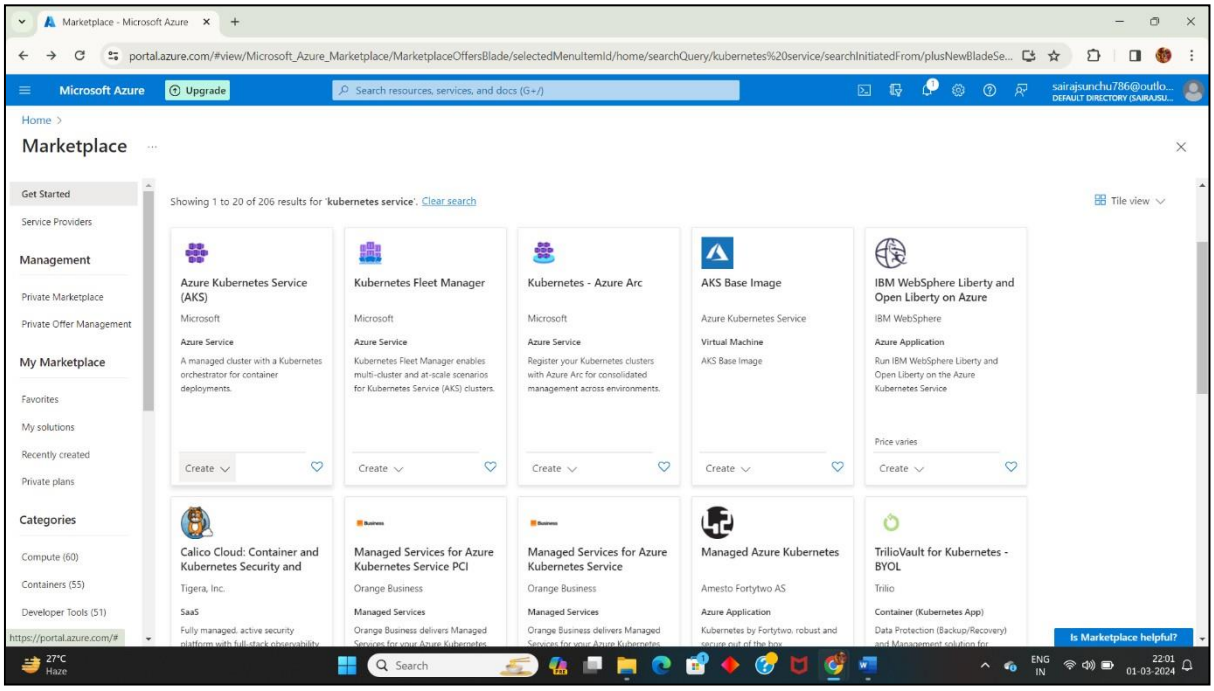
4A. Create an AKS Cluster from the Portal

Search Azure portal login on browser

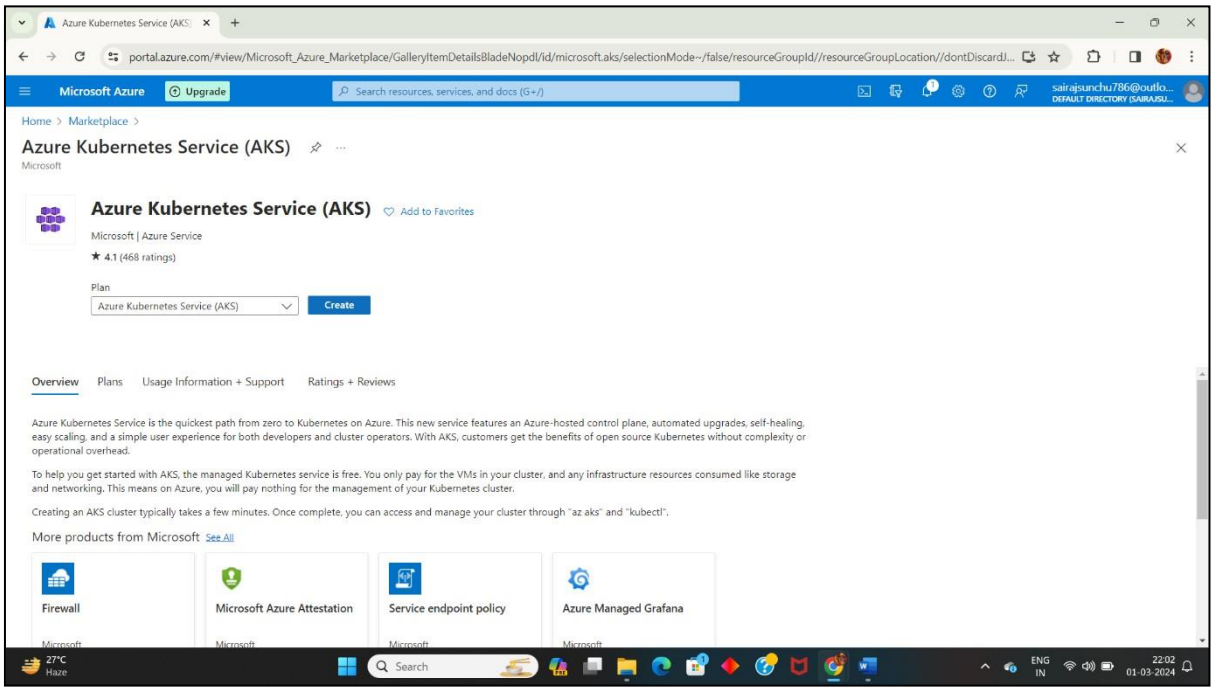
Create a resource



Search Azure Kubernetes Service



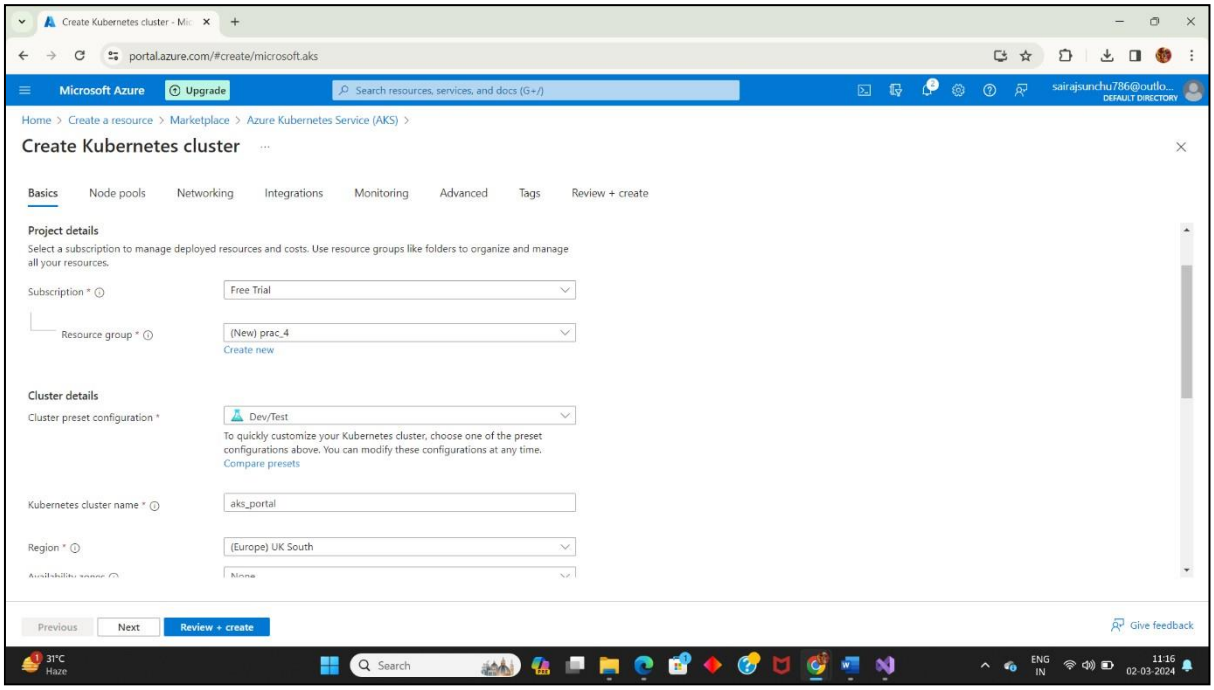
Click on create



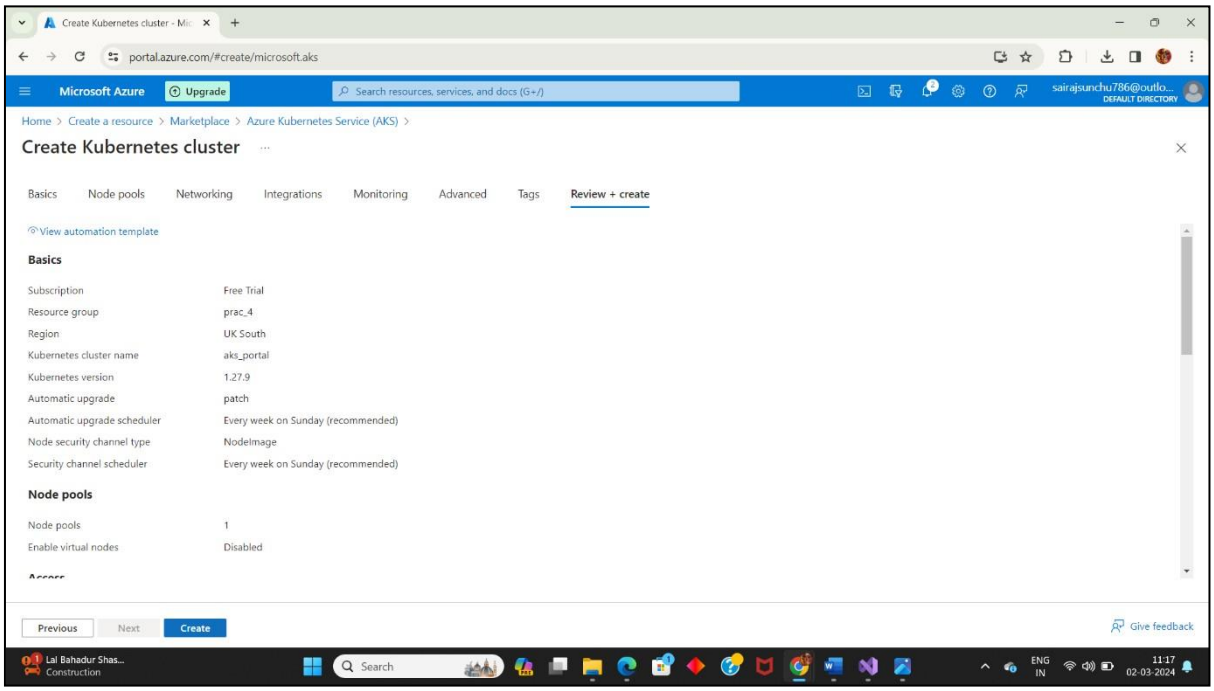
Select Subscription, create resource group, select region and add Kubernetes cluster name

Keep all other tab settings default

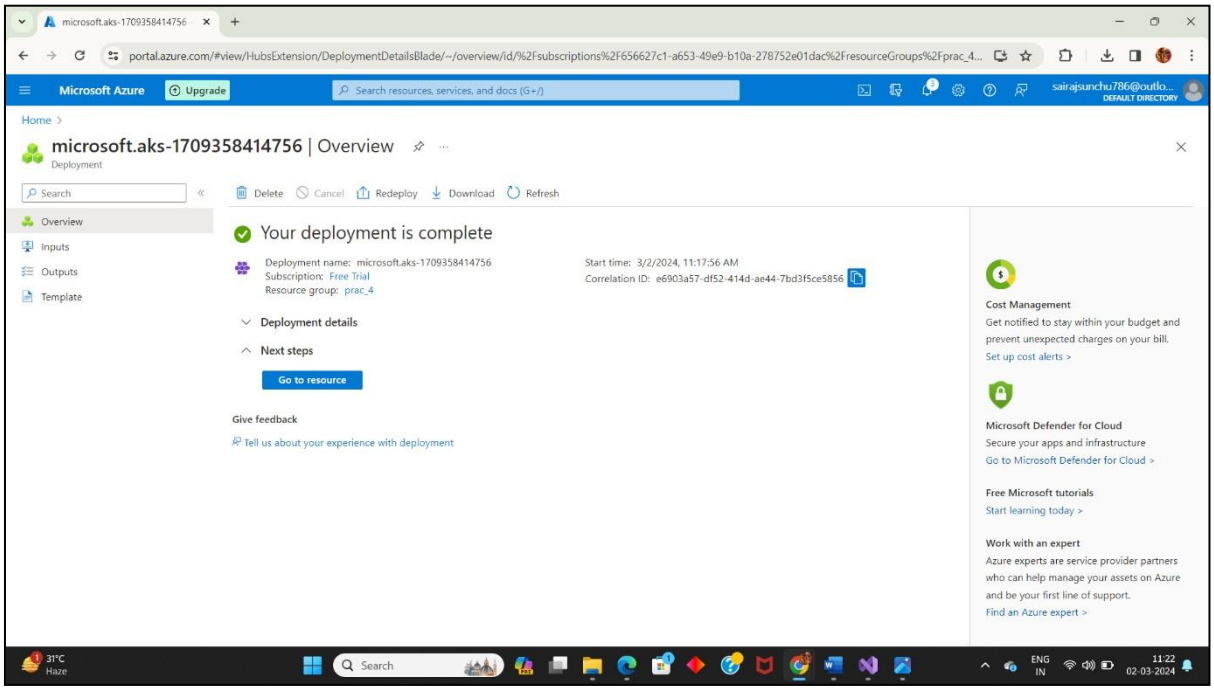
Note : If the node pools shows an VCPUs error then change the region in basic tab



Click on review + create tab



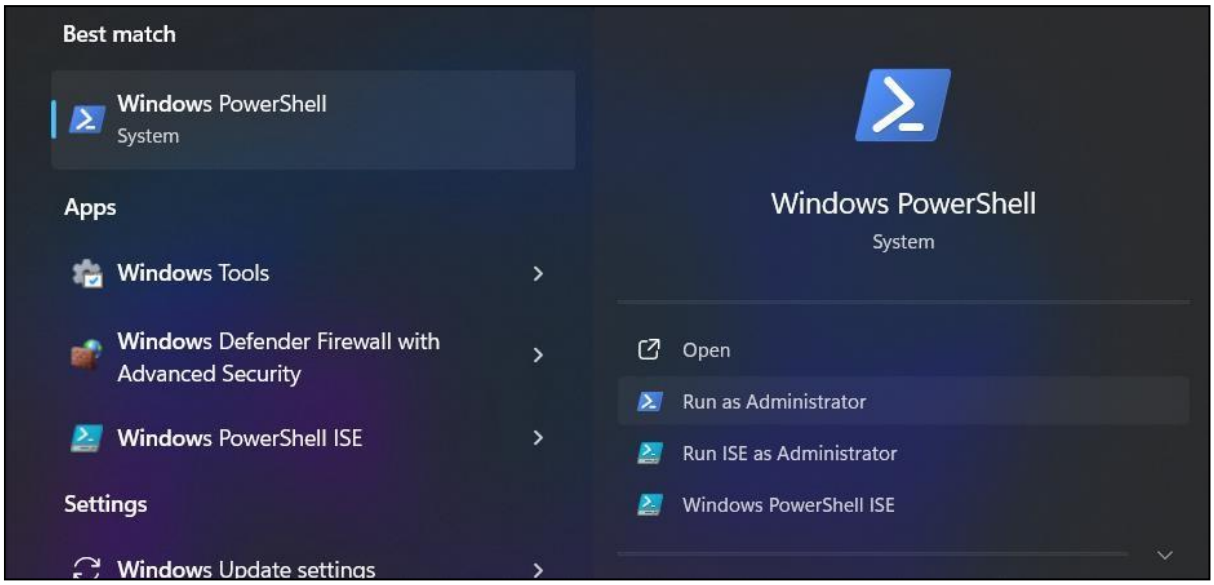
Click on create



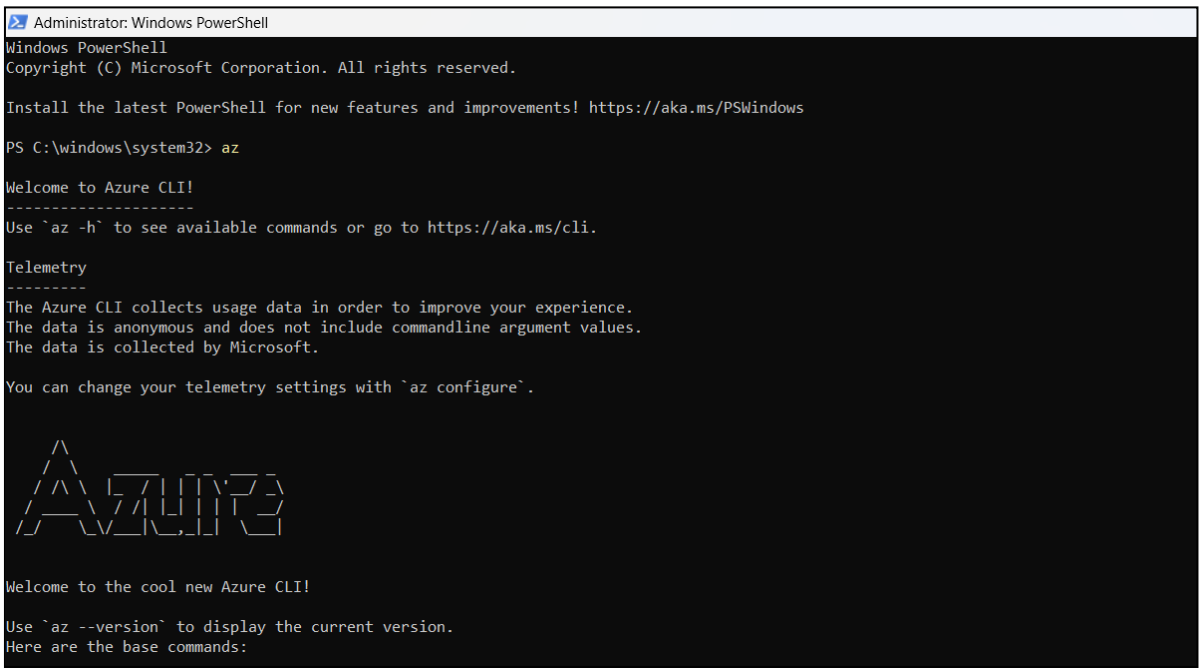
4B. Create an AKS cluster – with Azure CLI

Step 1: Install Azure CLI for windows system.

Step 2: Open Windows PowerShell Run as Administrator

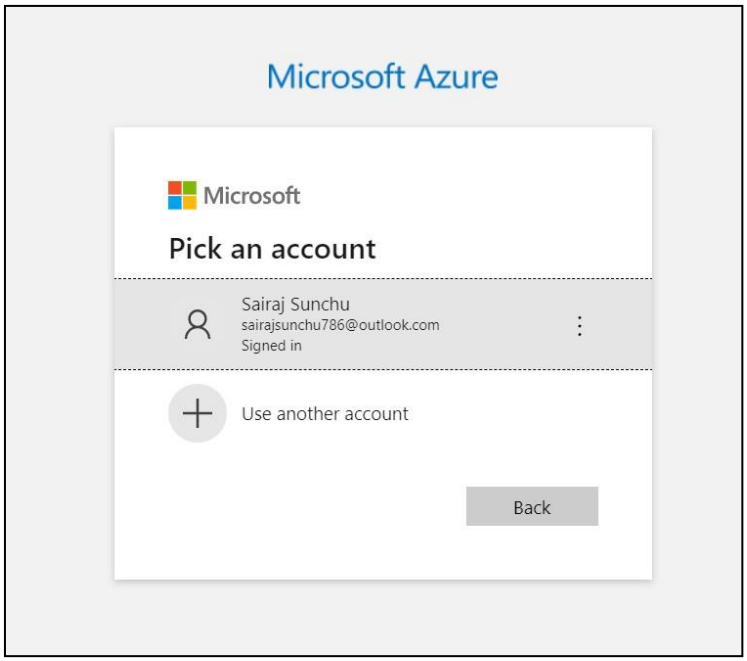


Type az



Type az login

```
    webapp              : Manage web apps.
PS C:\windows\system32> az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the logi
 open, use device code flow with `az login --use-device-code`.
```



Output on Windows PowerShell


```
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "e297c380-8b1c-4c9b-9ba7-655502a9d7e0",
    "id": "656627c1-a653-49e9-b10a-278752e01dac",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Free Trial",
    "state": "Enabled",
    "tenantId": "e297c380-8b1c-4c9b-9ba7-655502a9d7e0",
    "user": {
      "name": "sairajsunchu786@outlook.com",
      "type": "user"
    }
  }
]
PS C:\windows\system32>
```

Type az version

```
PS C:\windows\system32> az version
{
  "azure-cli": "2.57.0",
  "azure-cli-core": "2.57.0",
  "azure-cli-telemetry": "1.1.0",
  "extensions": {}
}
```

Type az group create –name akscube –location southeastasia

```
PS C:\Users\IT70> az group create --name akscube --location southeastasia
{
  "id": "/subscriptions/5037bd55-a725-439b-8977-6ce99cb02934/resourceGroups/akscube",
  "location": "southeastasia",
  "managedBy": null,
  "name": "akscube",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
PS C:\Users\IT70>
```

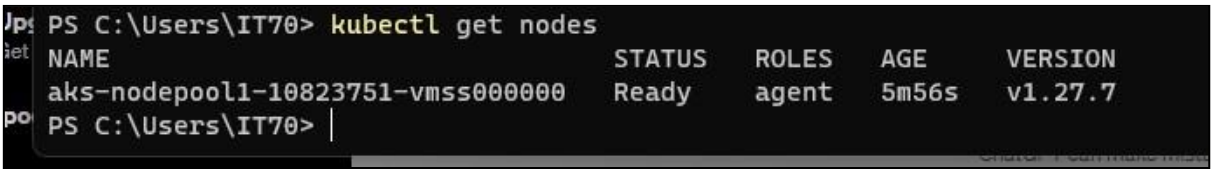
Type az aks create –resource-group akscube –name onecluster –node-count 1 –enable-addons monitoring –generate-ssh-keys

```
PS C:\Users\IT70> az aks create --resource-group akscube --name onecluster --node-count 1 --enable-addons monitoring --g
enerate-ssh-keys
SSH key files 'C:\Users\IT70\.ssh\id_rsa' and 'C:\Users\IT70\.ssh\id_rsa.pub' have been generated under ~/.ssh to allow
SSH access to the VM. If using machines without permanent storage like Azure Cloud Shell without an attached file share,
 back up your keys to a safe location
Resource provider 'Microsoft.ContainerService' used by this operation is not registered. We are registering for you.
Registration succeeded.
{
  "aadProfile": null,
  "addonProfiles": {
    "omsagent": {
      "config": {
        "logAnalyticsWorkspaceResourceID": "/subscriptions/5037bd55-a725-439b-8977-6ce99cb02934/resourceGroups/DefaultRe
sourceGroup-SEA/providers/Microsoft.OperationalInsights/workspaces/DefaultWorkspace-5037bd55-a725-439b-8977-6ce99cb02934
-SEA",
        "useAADAuth": "true"
      },
      "enabled": true,
      "identity": null
    }
  },
  "agentPoolProfiles": [
    {
      "availabilityZones": null,
      "capacityReservationGroupId": null,
      "count": 1,
```

Type az aks get-credentials –resource-group akscube –name onecluster

```
PS C:\Users\IT70> az aks get-credentials --resource-group akscube --name onecluster
Merged "onecluster" as current context in C:\Users\IT70\.kube\config
PS C:\Users\IT70>
```

Type kubectl get nodes


PS C:\Users\IT70> kubectl get nodes
NAME STATUS ROLES AGE VERSION
aks-nodepool1-10823751-vmss000000 Ready agent 5m56s v1.27.7
PS C:\Users\IT70>

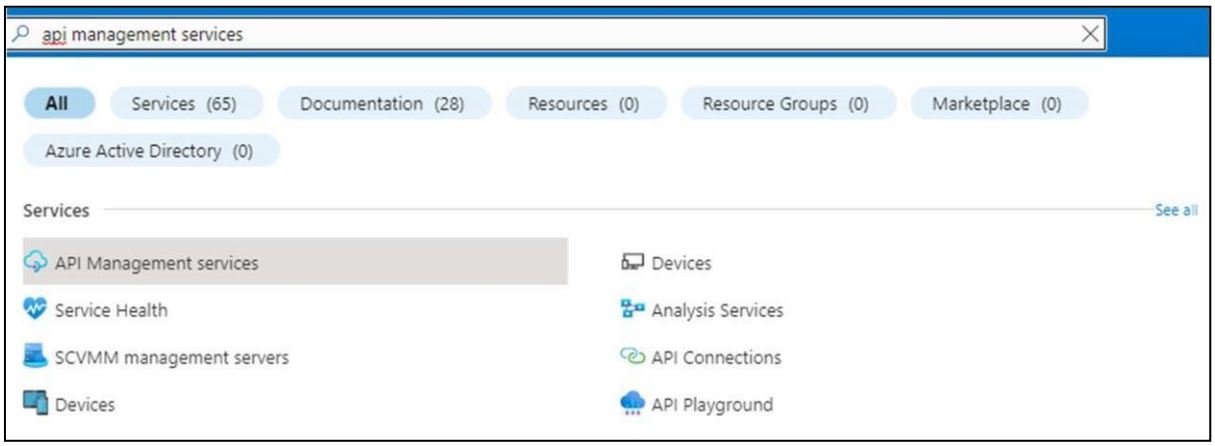
Practical No: 5

Create an API management service

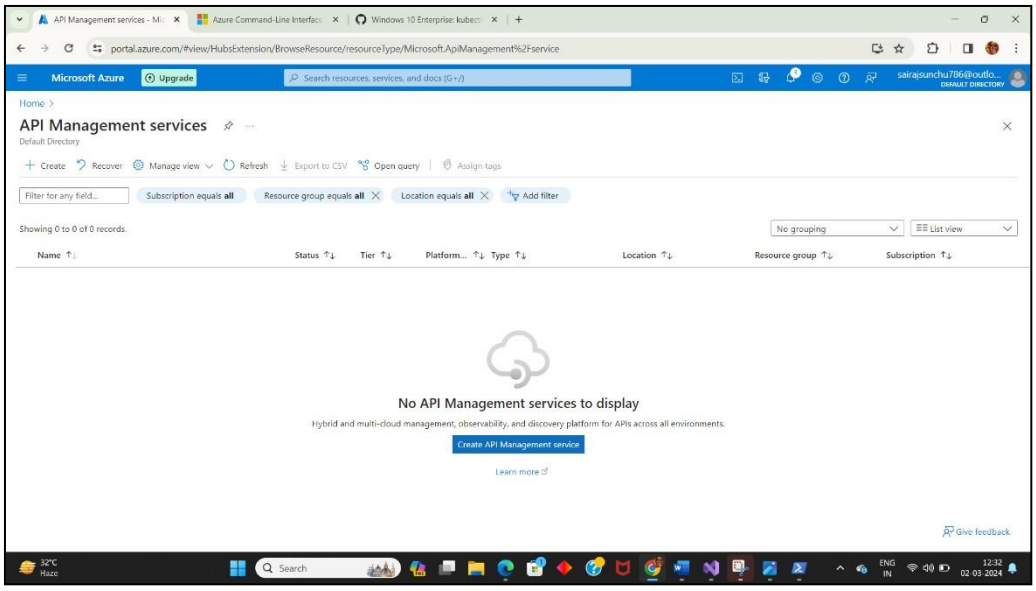
Step 1: Sign-in to your Azure Subscription Portal

Step 2: Search for "API Management", then select API Management in order to create a service instance.

Step 3: Select "API Management service"

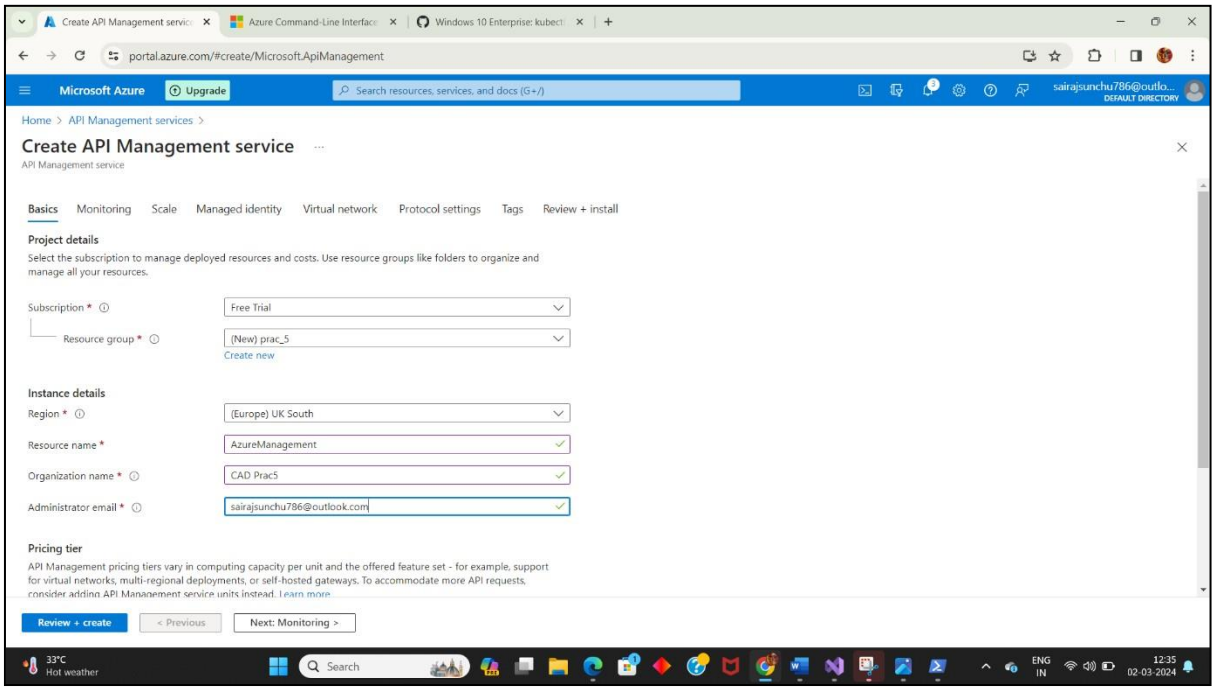


Step 4: As API management service page is loaded "Create API management service" button.

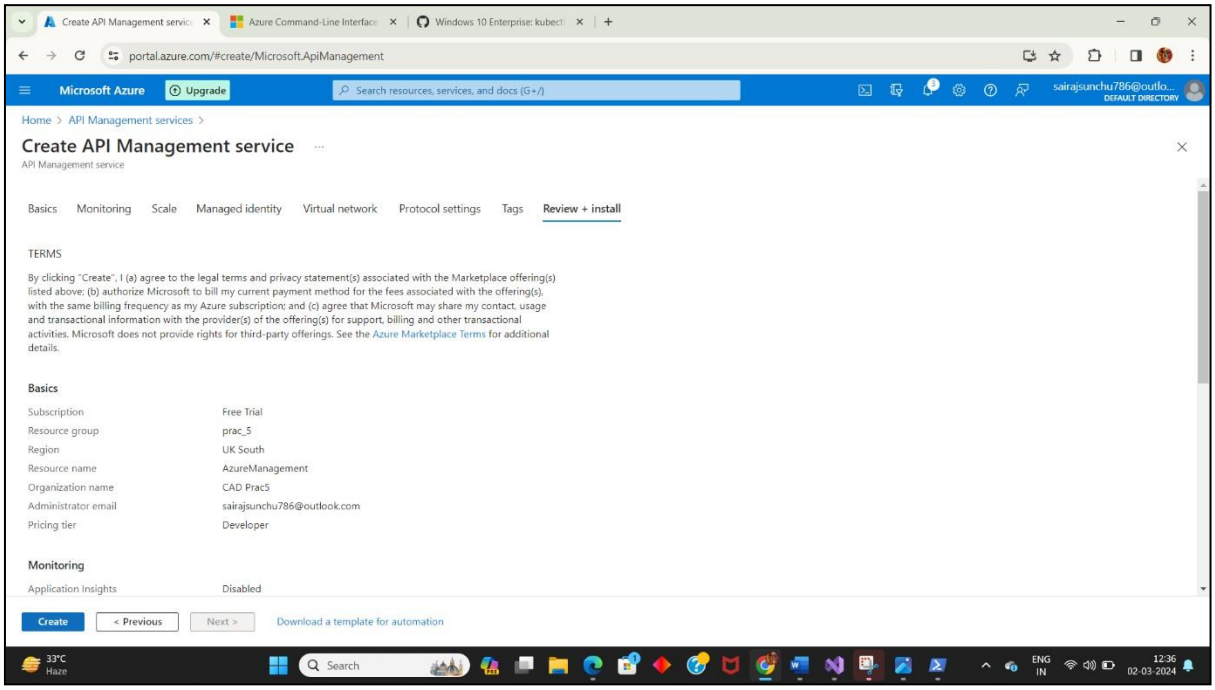


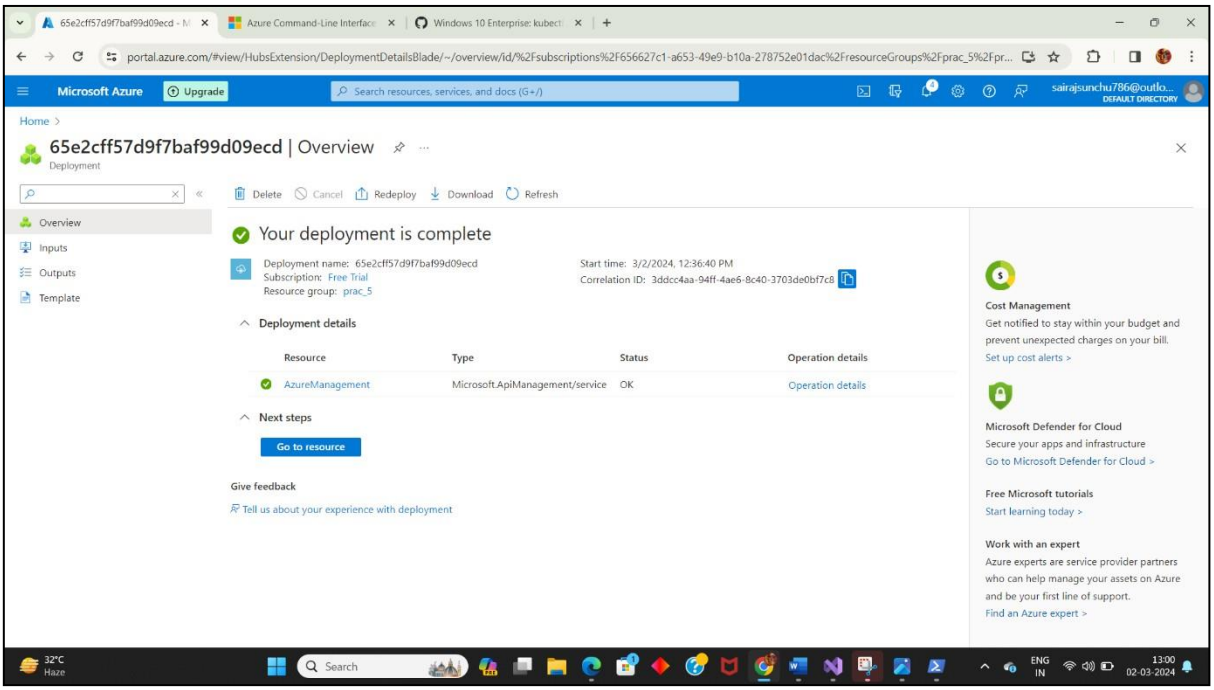
Select subcription, create new resource group

Enter region, resource name, organization name and administrator email

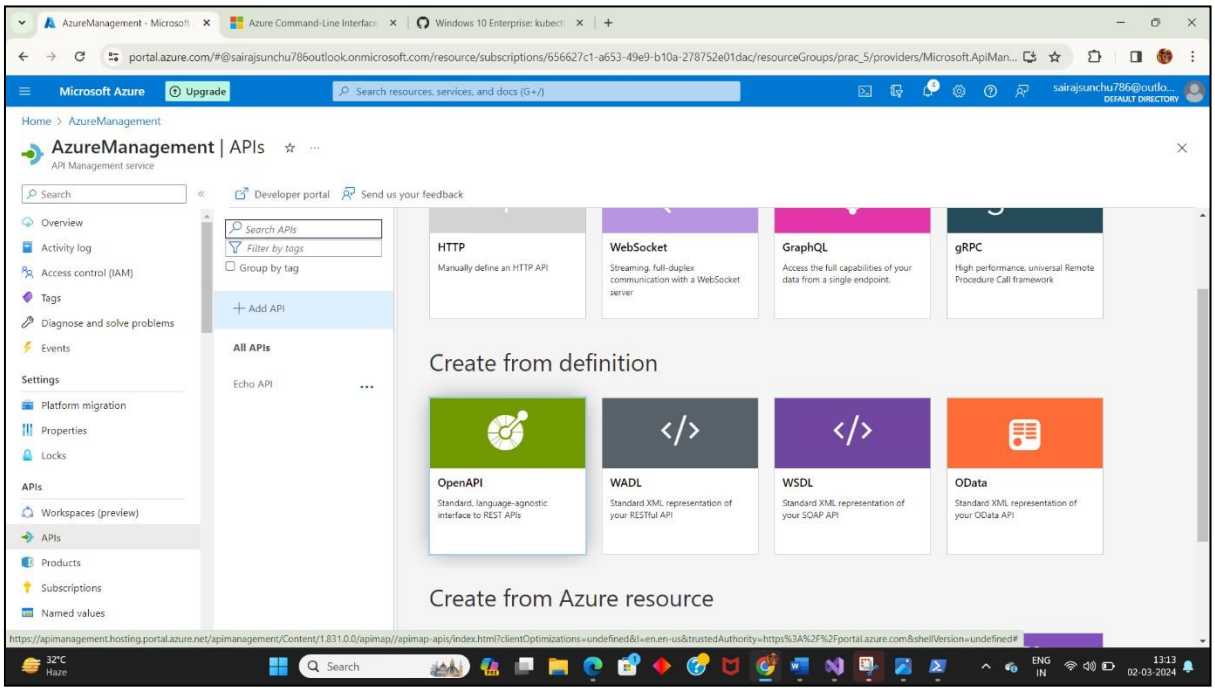


Click on review + install tab and click on create button.





At the left side click on APIs and click on Add on and Select OpenAPI

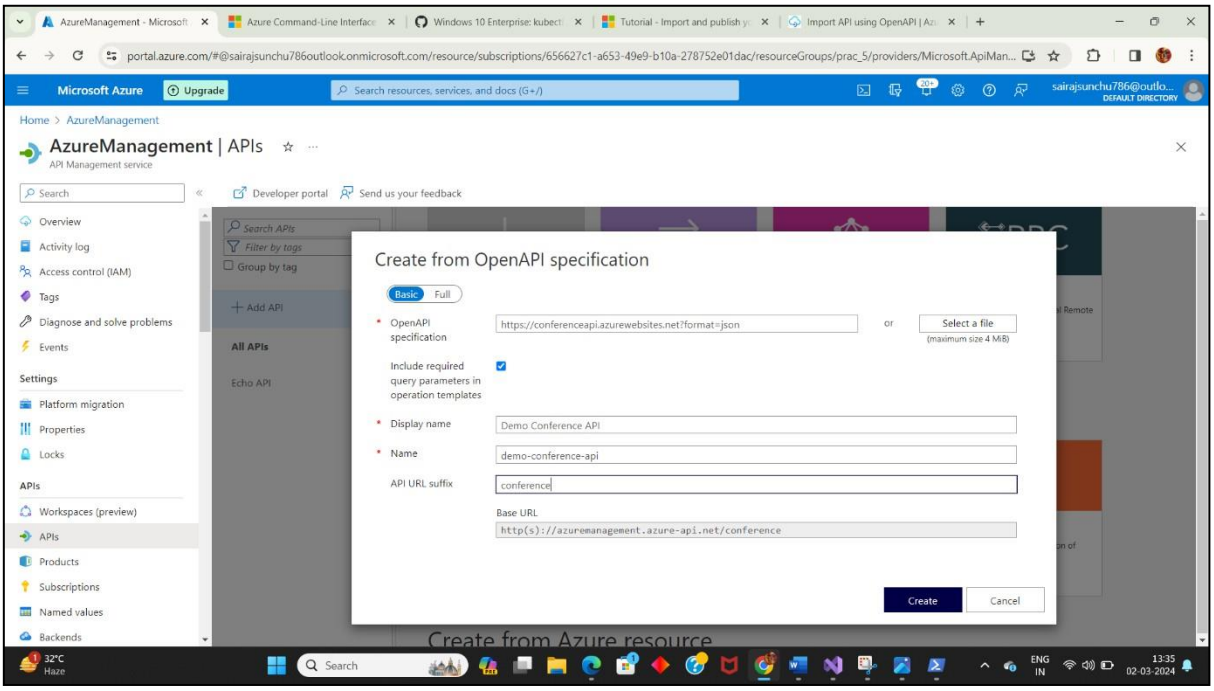


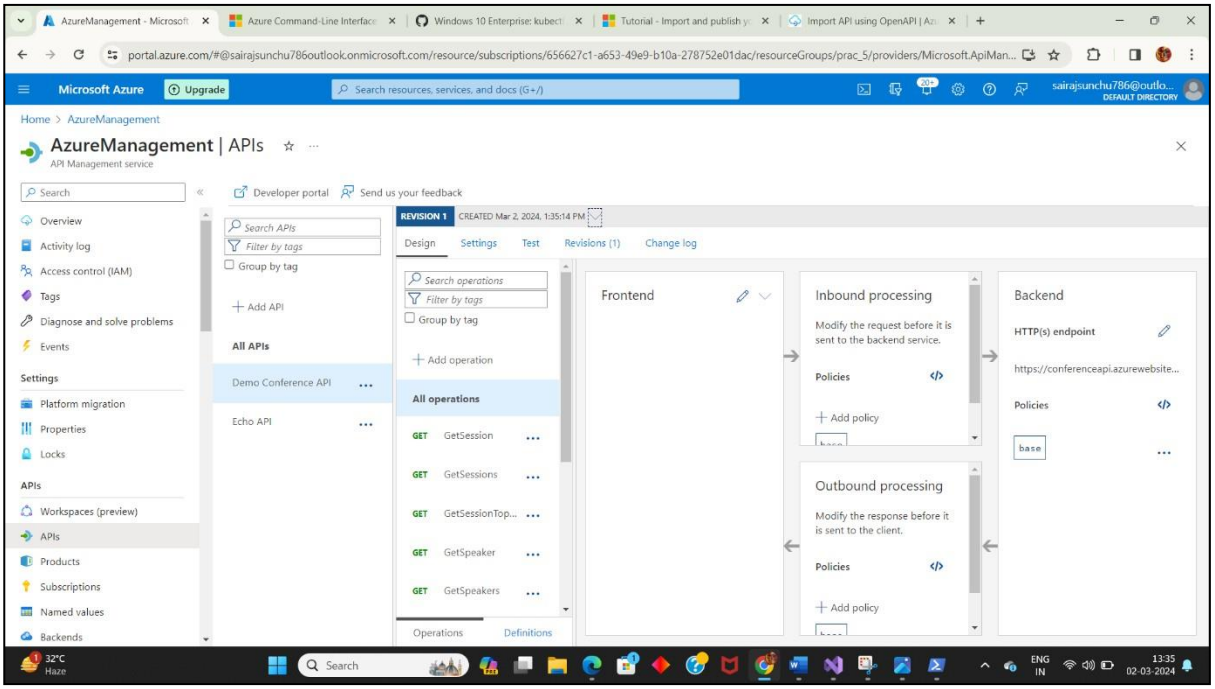
Enter details in OpenAPI specification

https://conferenceapi.azurewebsites.net?format=json

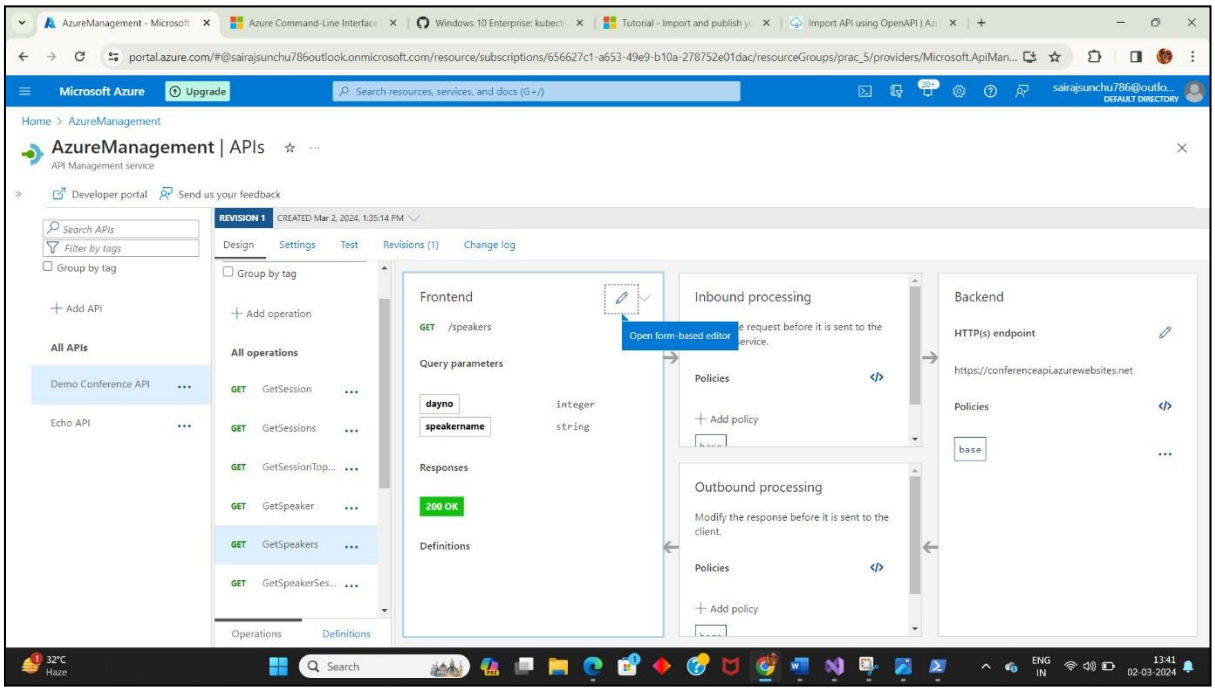
Display name Demo Conference API

Click on create

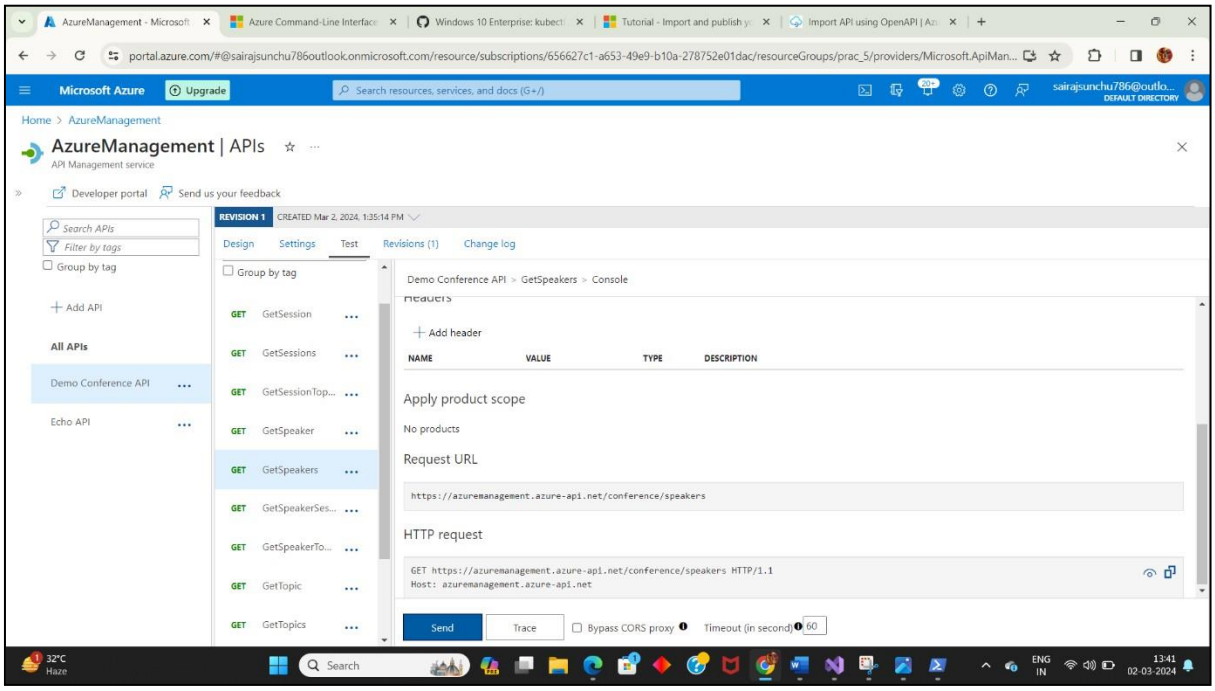




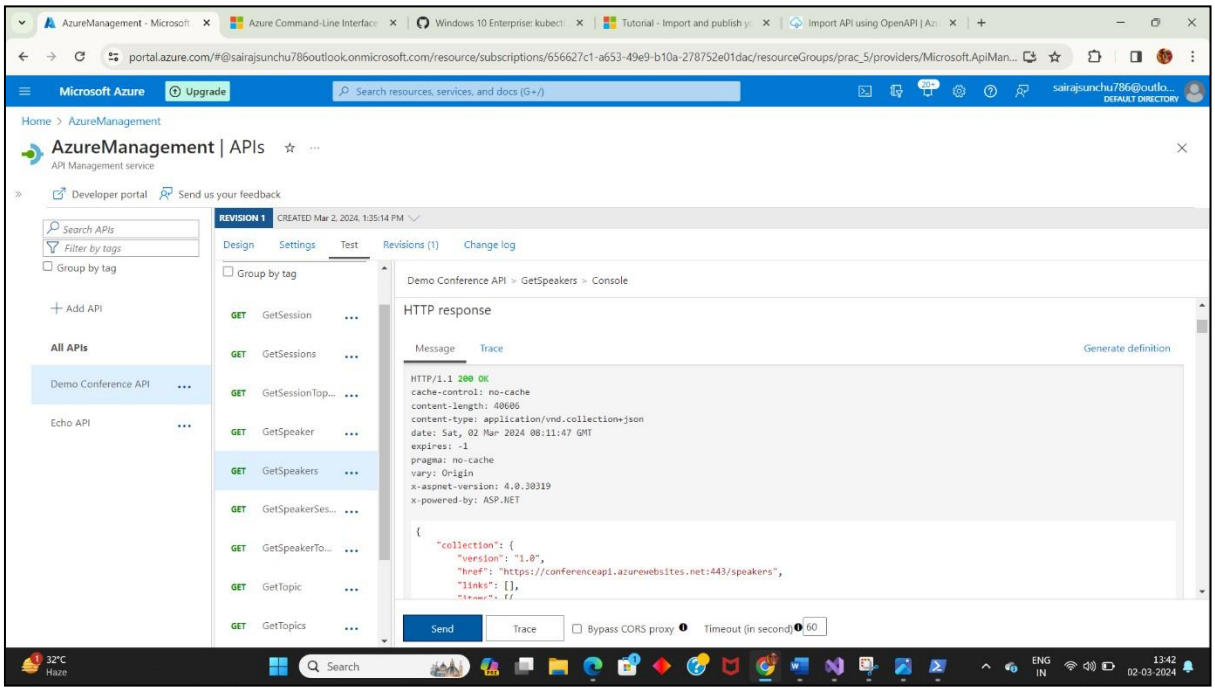
Select GetSpeakers



Click on send button



Check the Http response

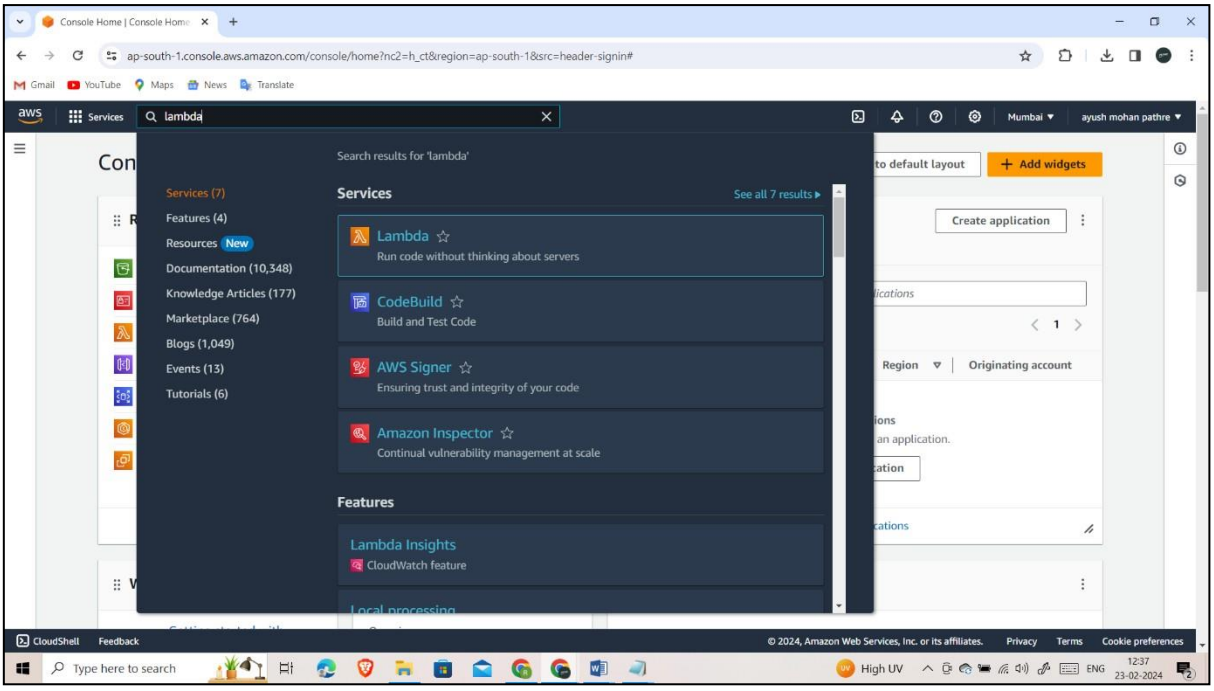


Practical No: 6

AWS API Gateway Authorizer

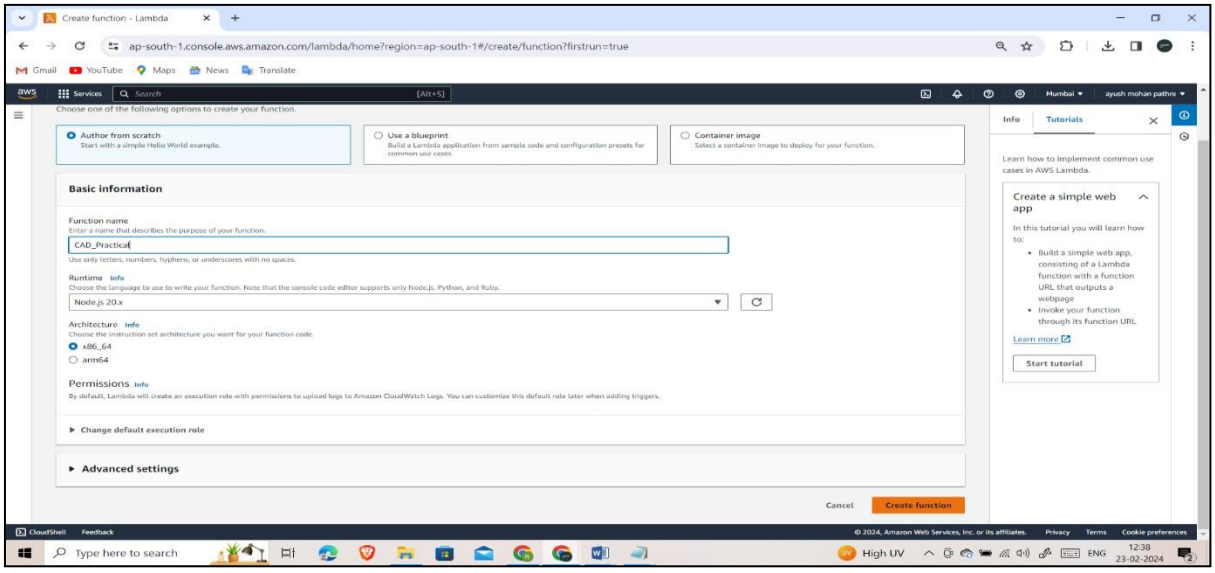
Step 1 – Login to aws.amazon.com and Search for “Lambda”

Step 2 – Click on Create a Function

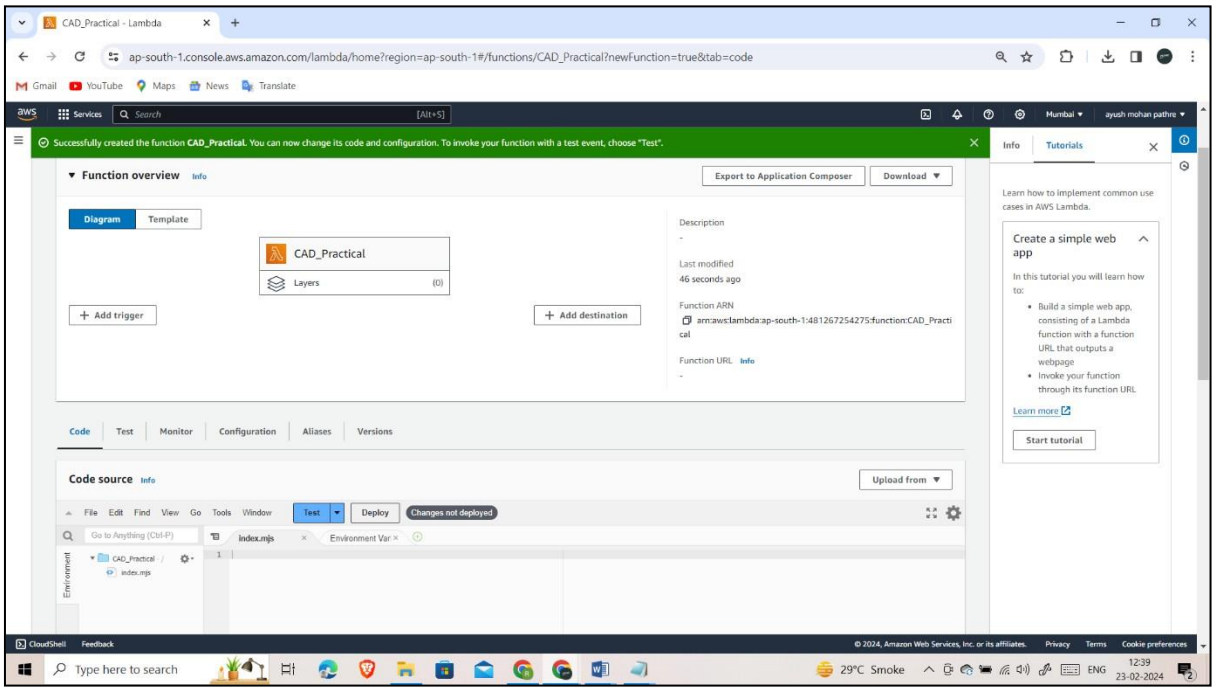


Name – CAD_Practical

Leave all others fields as it is and click on Create a function



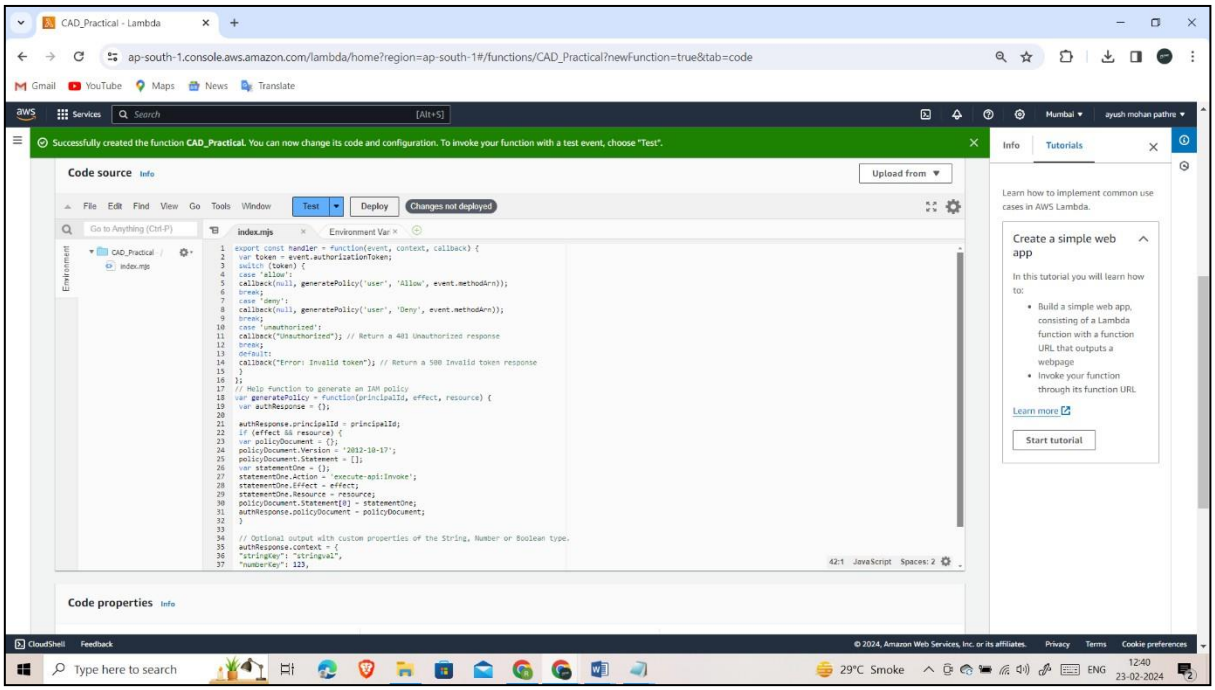
Step 3 – Double click on index.mjs file



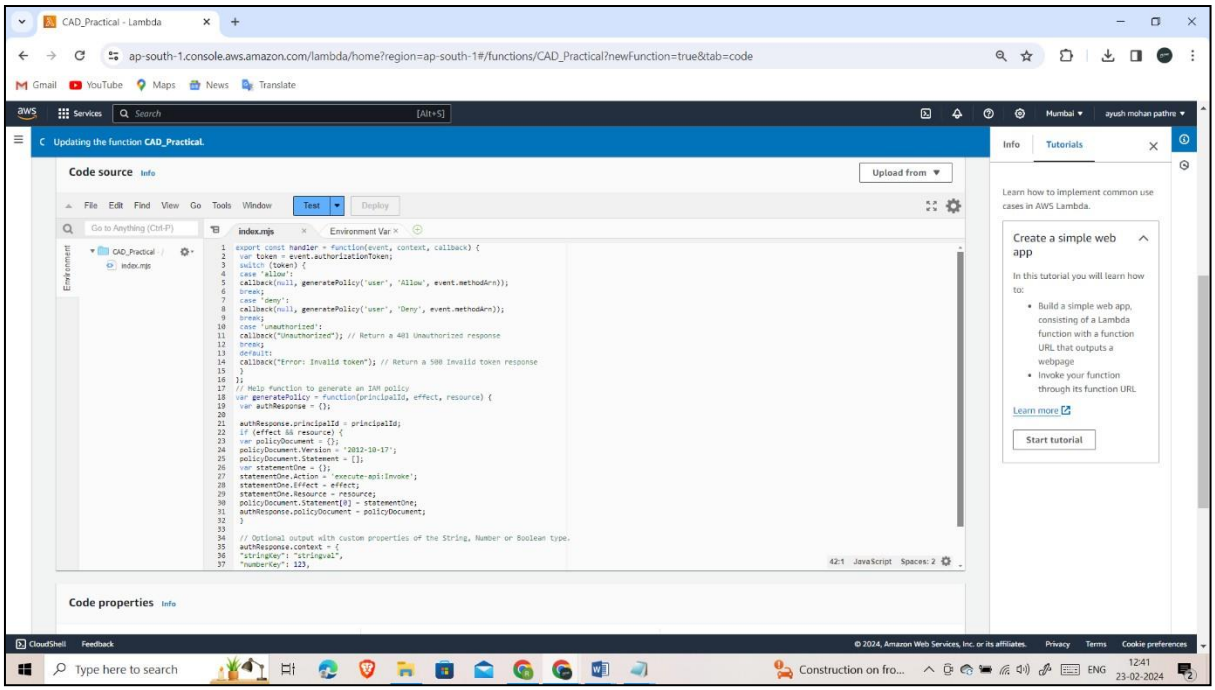
Step 4 – Copy the below and paste the same in the index.mjs editor section.

```
export const handler = function(event, context, callback) {
var token = event.authorizationToken;
switch (token) {
case 'allow':
callback(null, generatePolicy('user', 'Allow', event.methodArn));
break;
case 'deny':
callback(null, generatePolicy('user', 'Deny', event.methodArn));
break;
case 'unauthorized':
callback("Unauthorized"); // Return a 401 Unauthorized response
break;
default:
callback("Error: Invalid token"); // Return a 500 Invalid token response
}
};
```

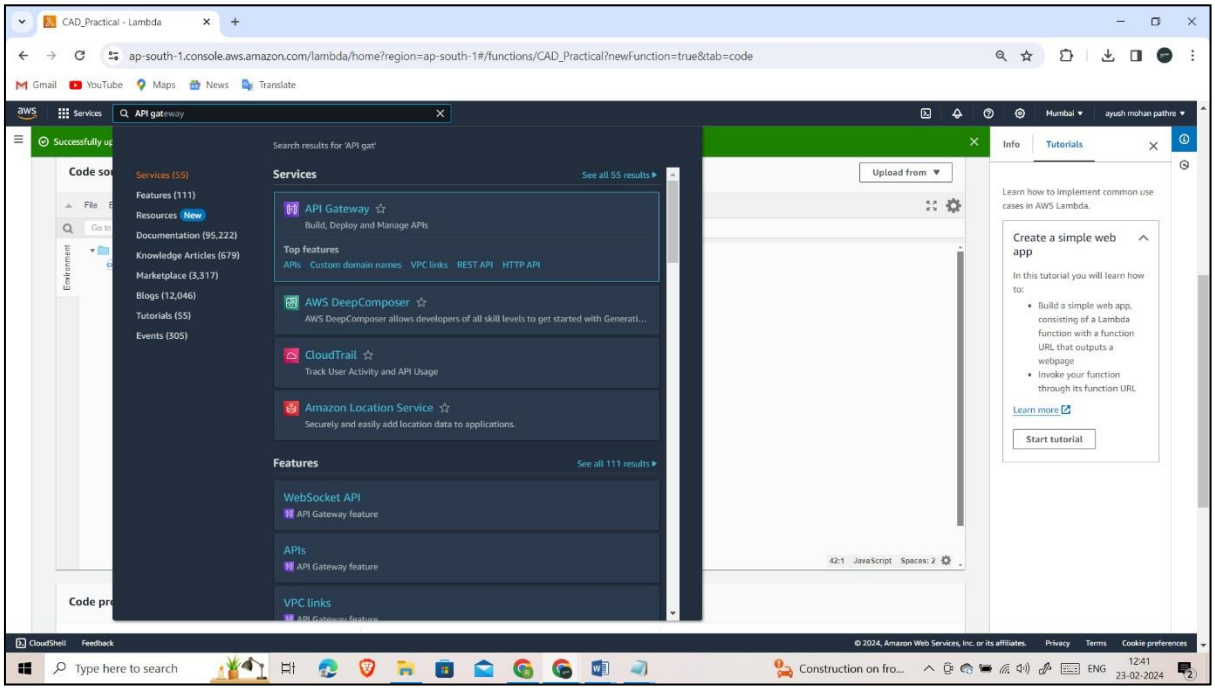
```
                                        // Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
var authResponse = {};

authResponse.principalId = principalId;
if (effect && resource) {
var policyDocument = {};
policyDocument.Version = '2012-10-17';
policyDocument.Statement = [];
var statementOne = {};
statementOne.Action = 'execute-api:Invoke';
statementOne.Effect = effect;
statementOne.Resource = resource;
policyDocument.Statement[0] = statementOne;
authResponse.policyDocument = policyDocument;
}

// Optional output with custom properties of the String, Number or Boolean type.
authResponse.context = {
"stringKey": "stringval",
"numberKey": 123,
"booleanKey": true
};
return authResponse;
}
```



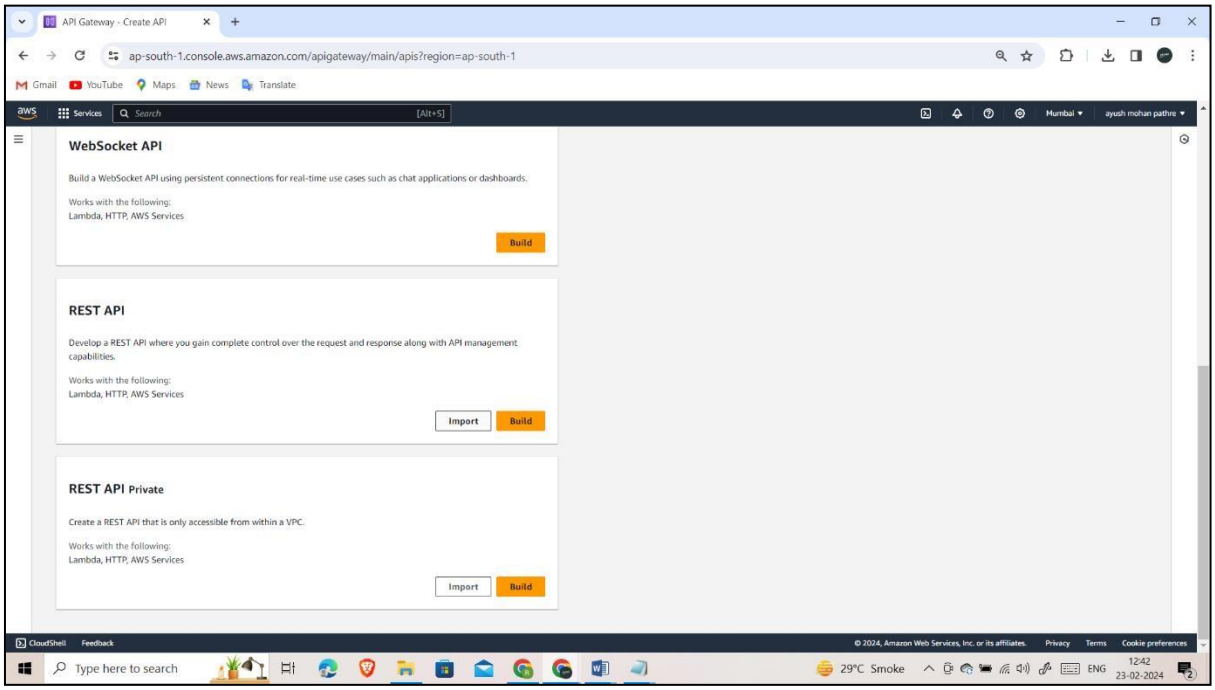
Step 5 – Click on deploy once the code is pasted. The code is successfully deployed



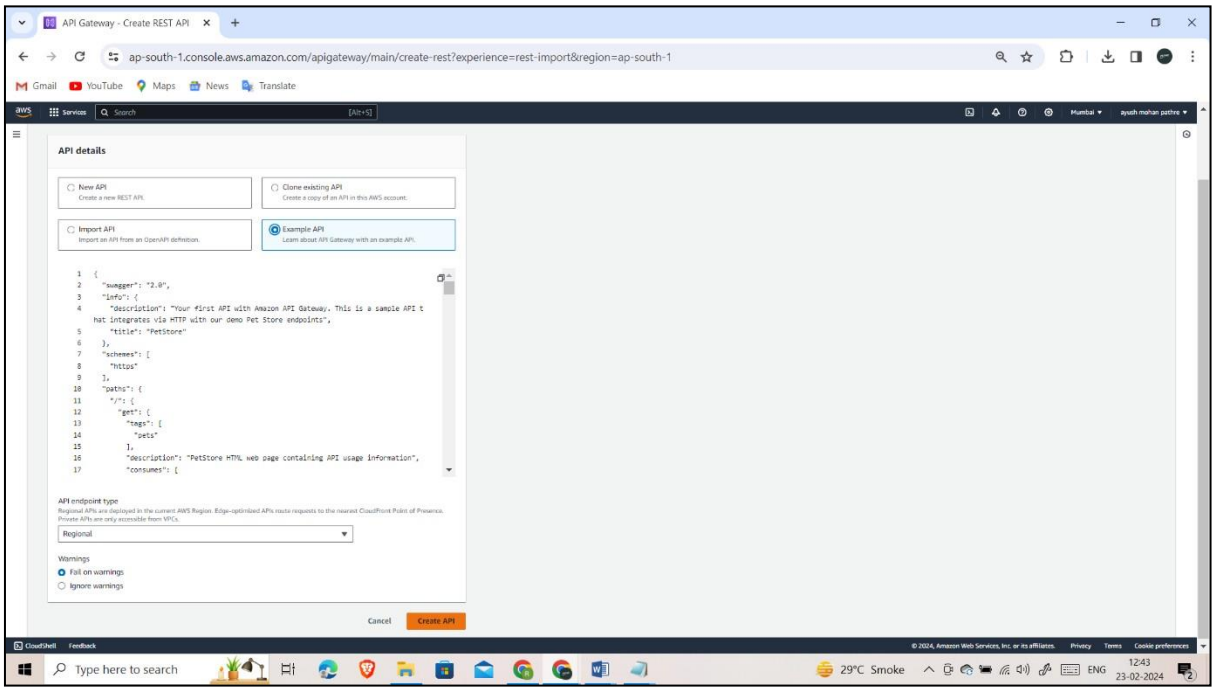
Step 6 – Now search API Gateway in the search bar and open in the new tab



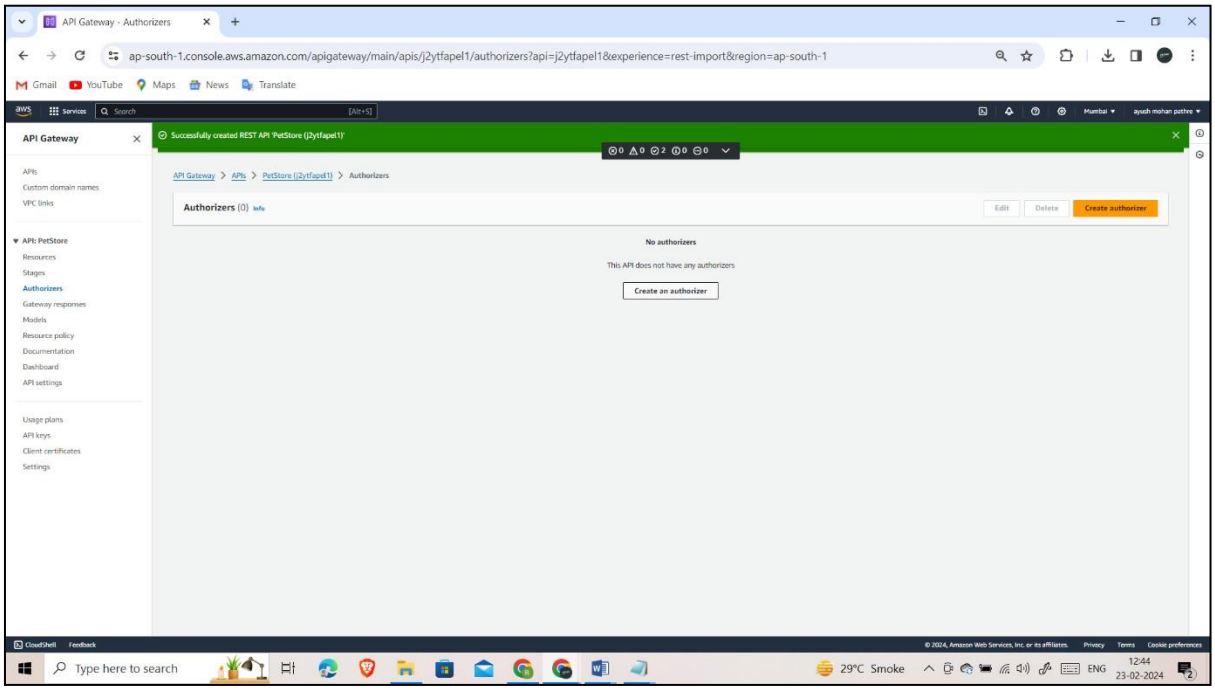
Step 7 – Select Choose an API type - > REST API -> Build



Step 8 – Leave all the fields as it and scroll down to the bottom and click on Import



Step 9 – Now click on the Authorizers on the left-hand side as mentioned in the screenshot.



Step 10 – Click on the button - Create New Authorizer and fill in the details as below

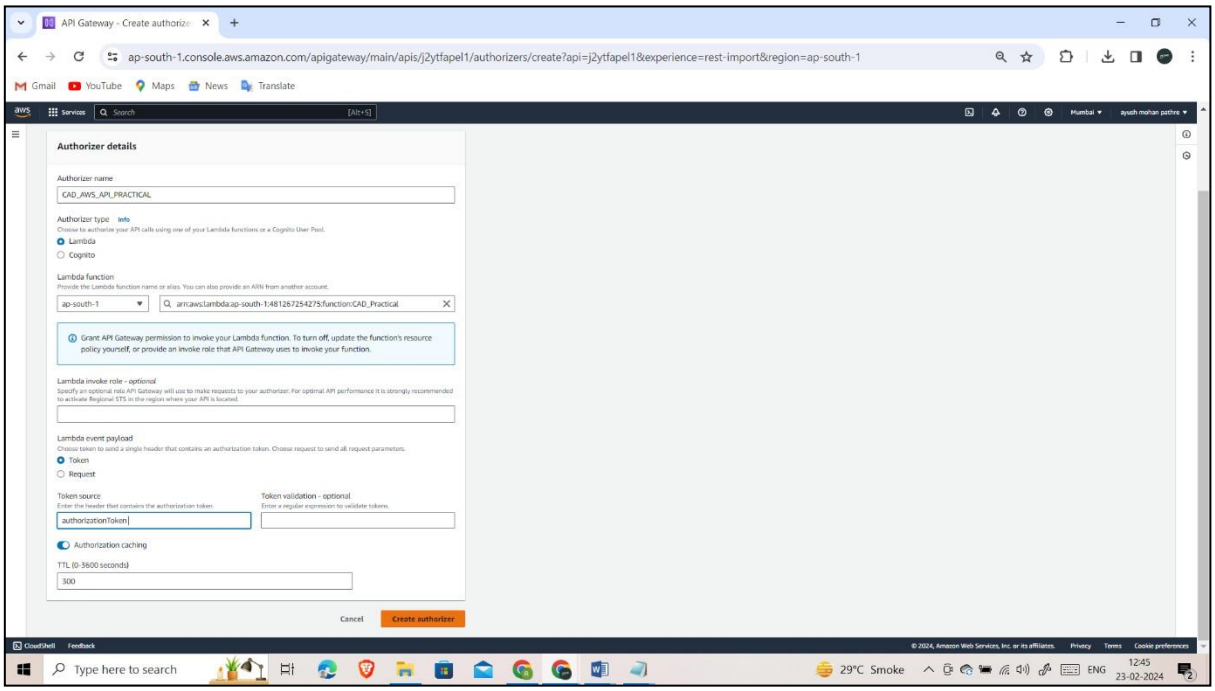
Name – CAD_AWS_API_PRACTICAL

Lambda Function – Automatically CAD_Practical will be displayed and select the same

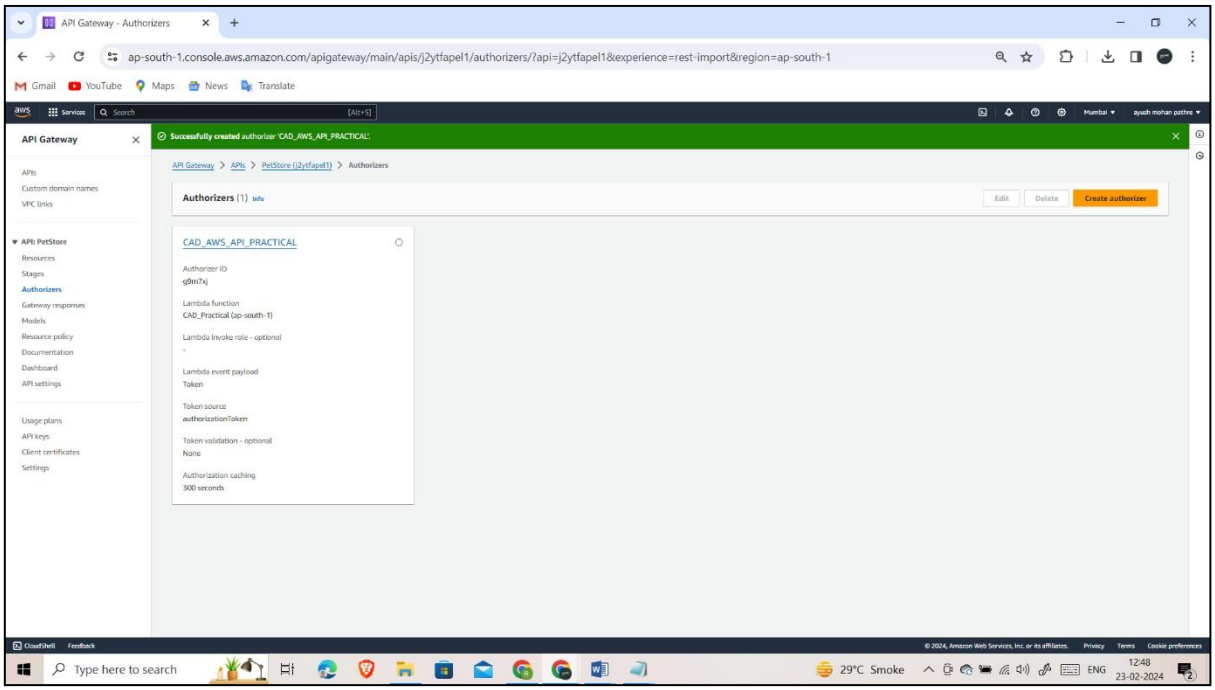
Lambda Invoke Role – Leave it Blank

Token Source – authorizationToken

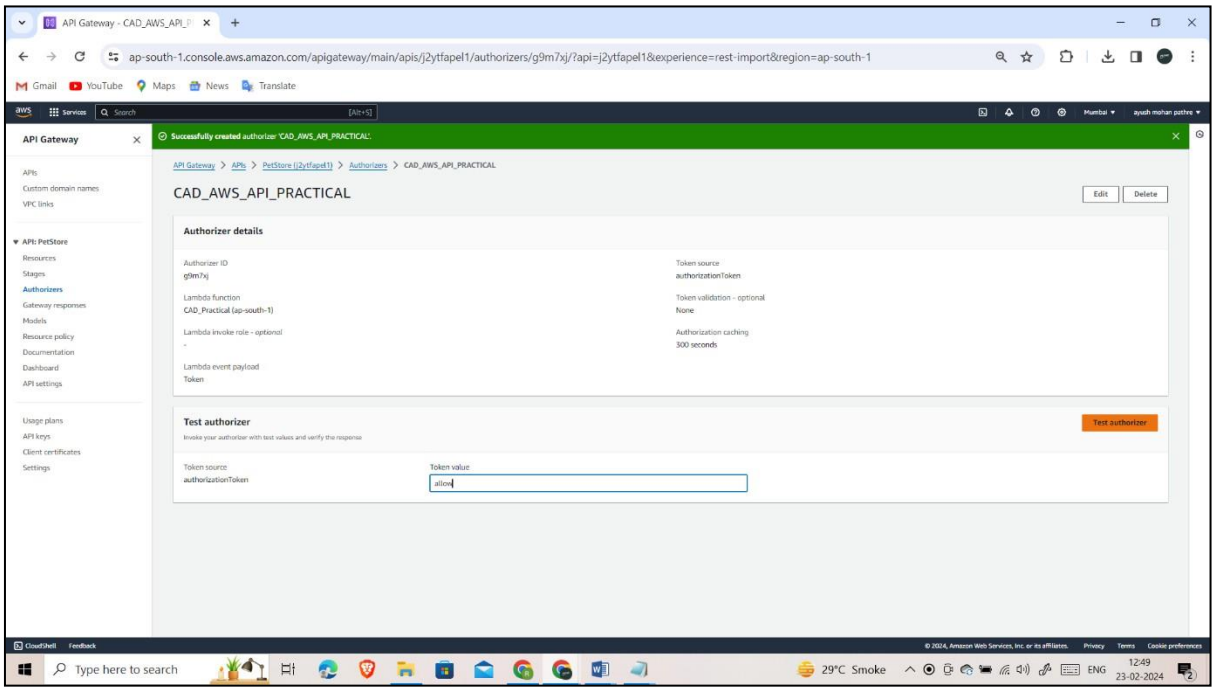
And then click on Create



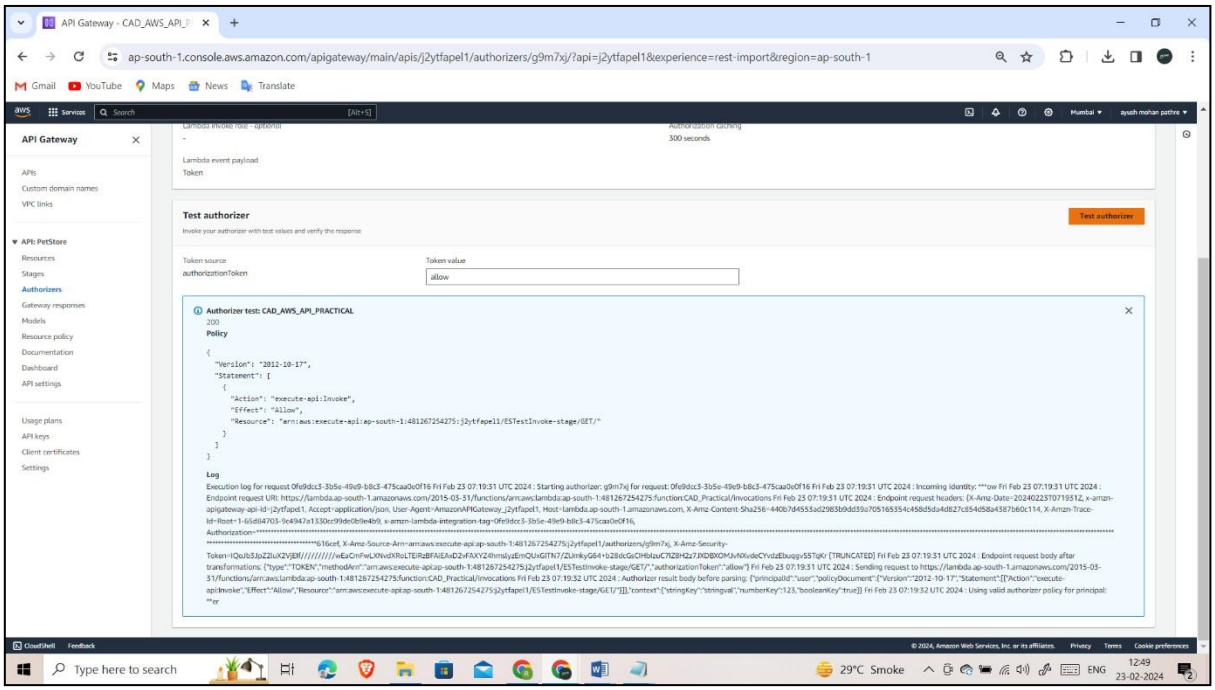
Step 11 – Click on Create



Step 12 – Now click on the test button



Step 13 – Type allows in the box as mentioned in the screenshot

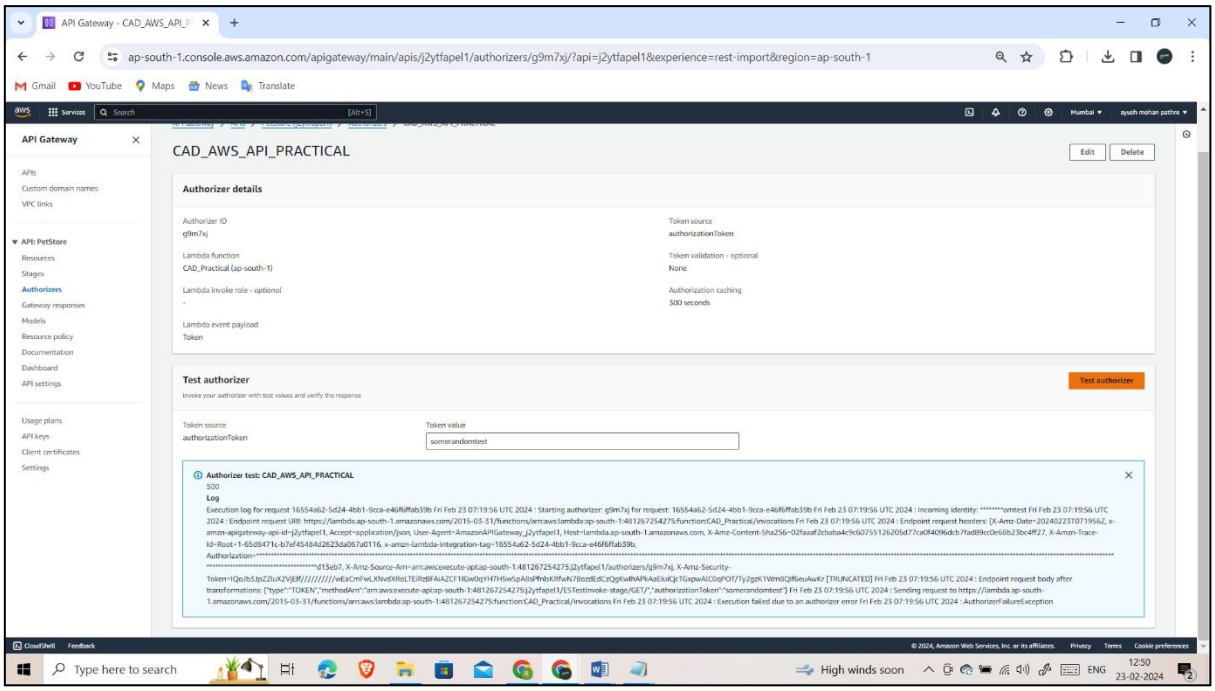


We can see the that function that we created is tokenized. When we input allow, it will check the IAM policy and return policy output of the same.

If we input as somerandomtest, then following is the output of the same. An error response will be thrown by the same. As per the function created, it should only be one of the following inputs 'allow':

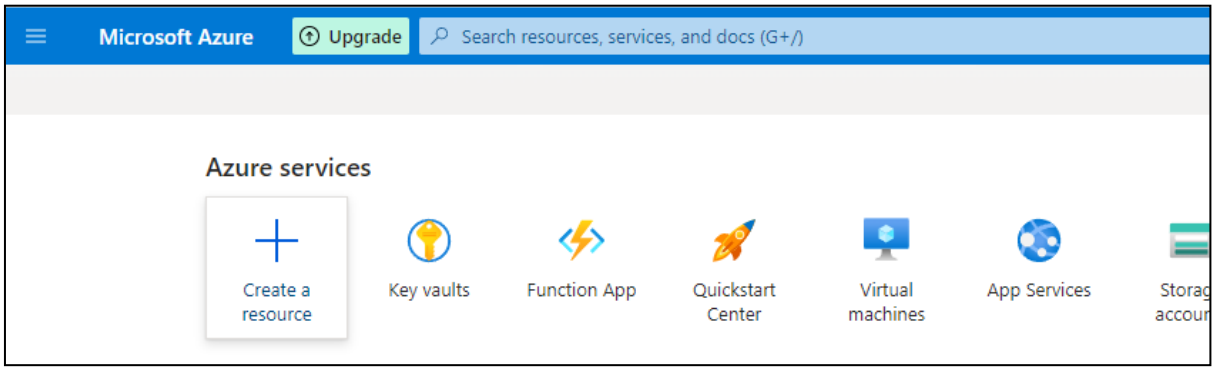
'deny':

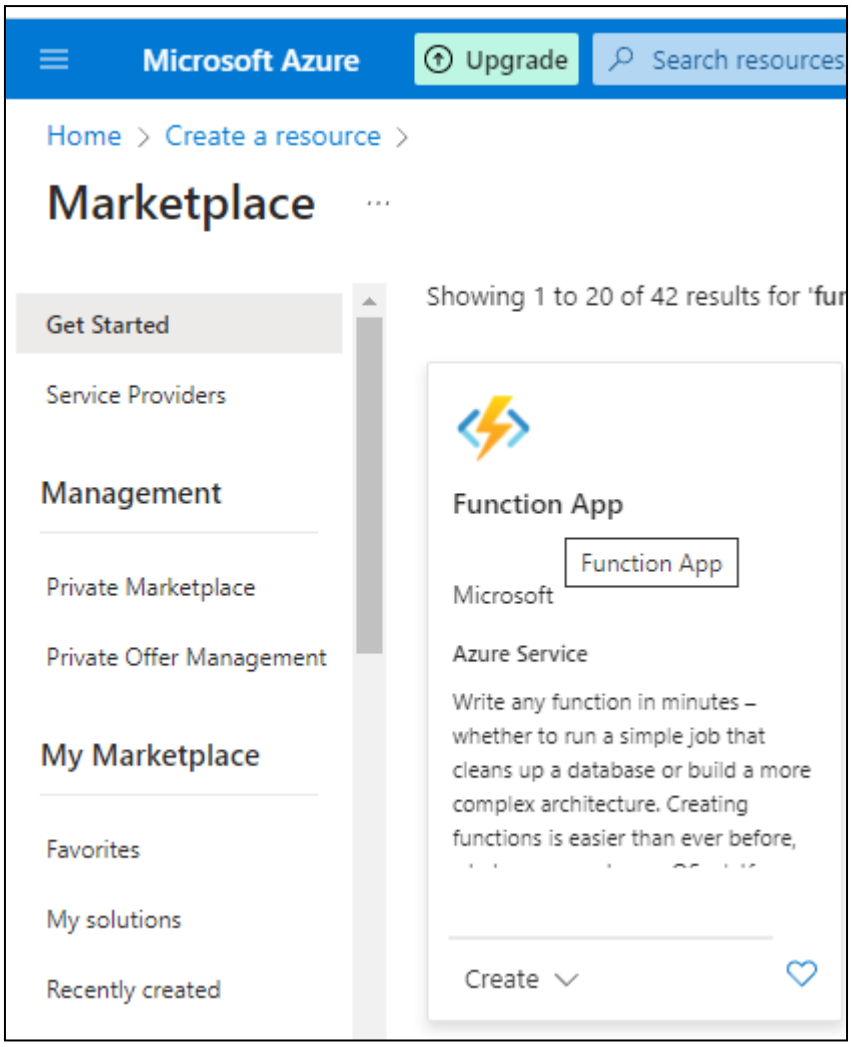
'unauthorized':

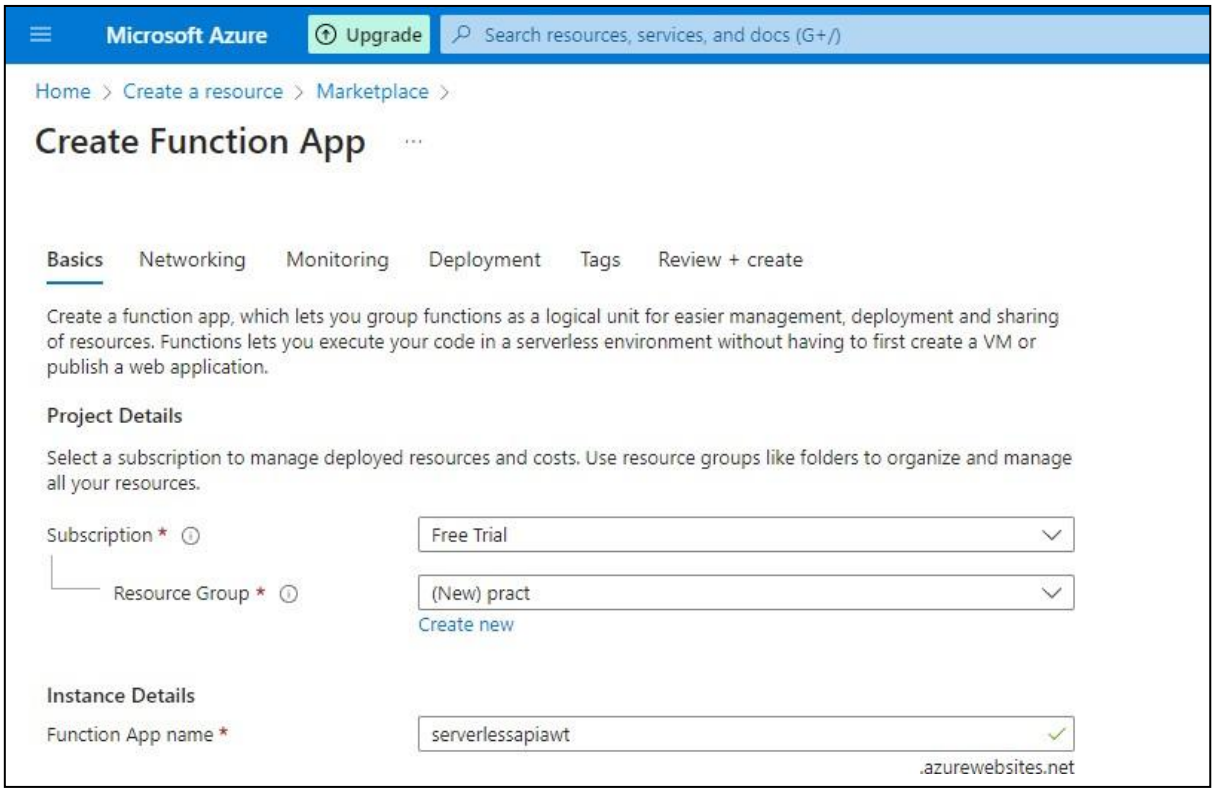


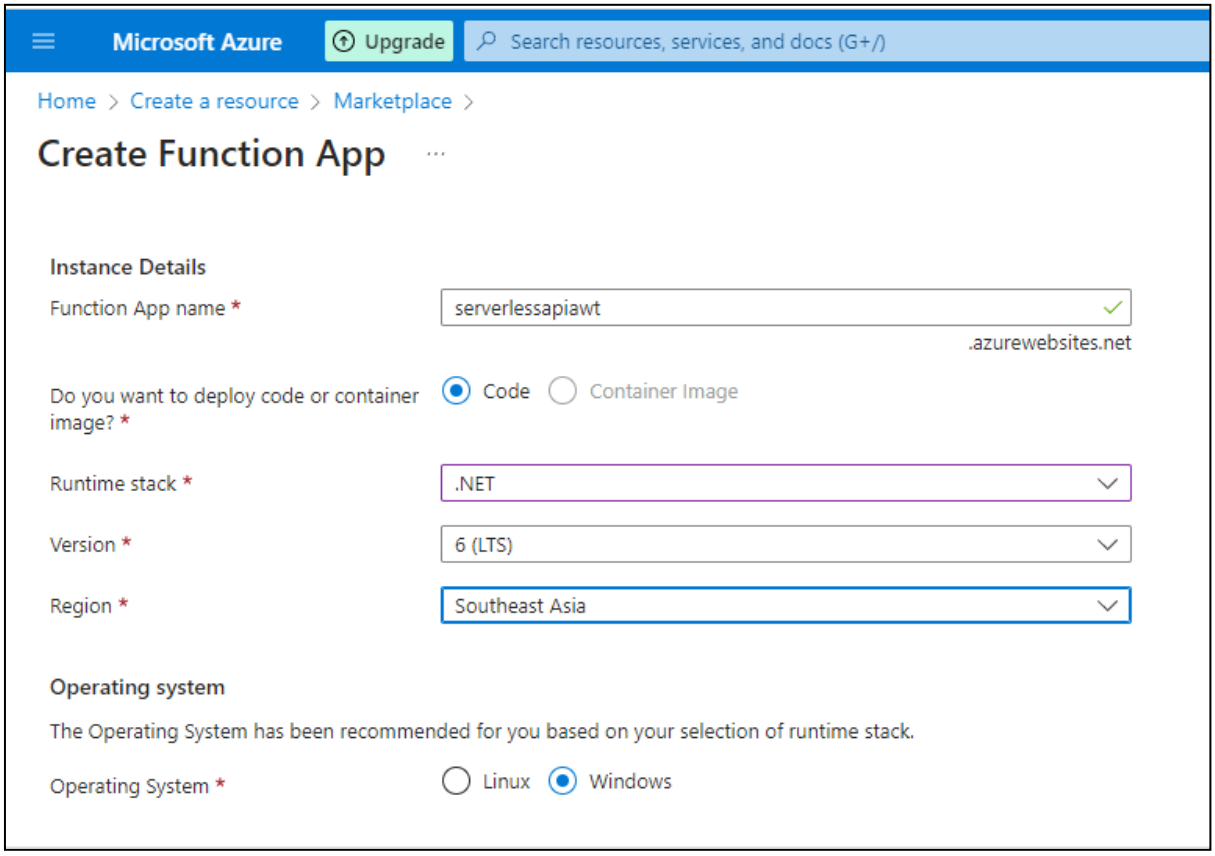
Practical No: 7

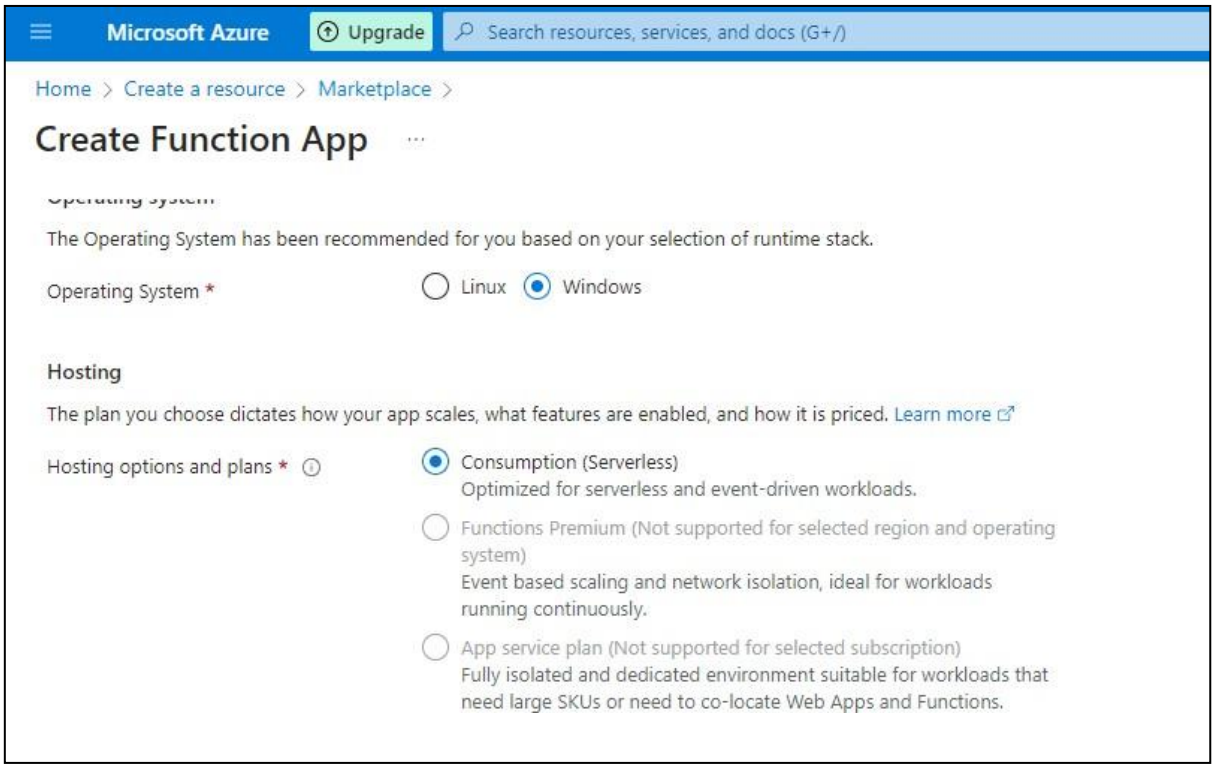
Create a serverless API using Azure functions

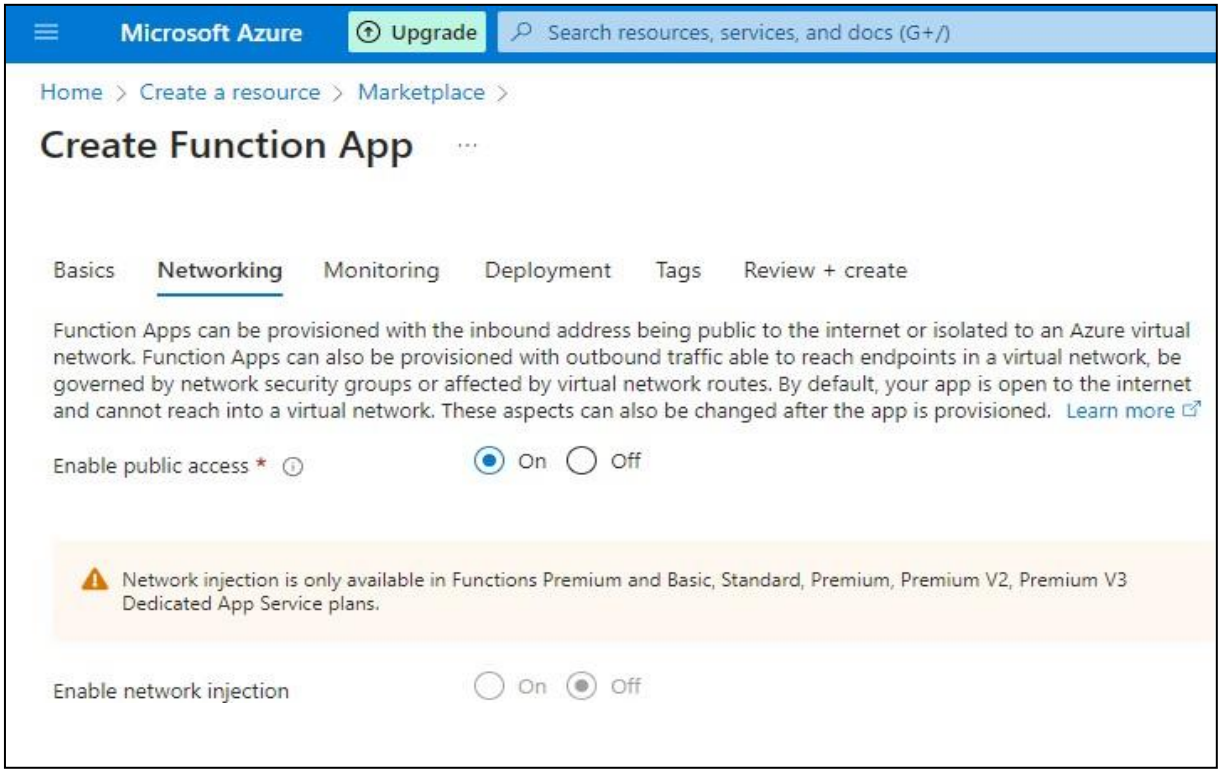


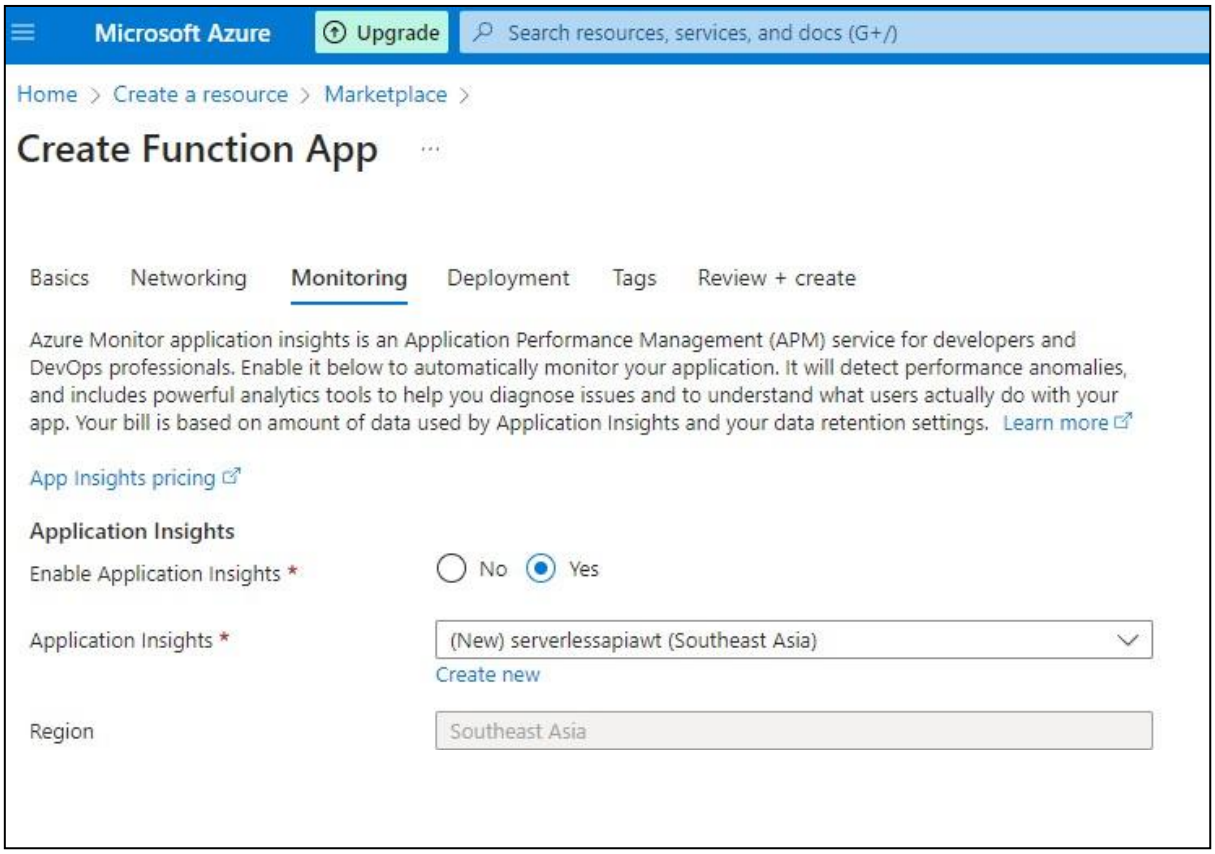


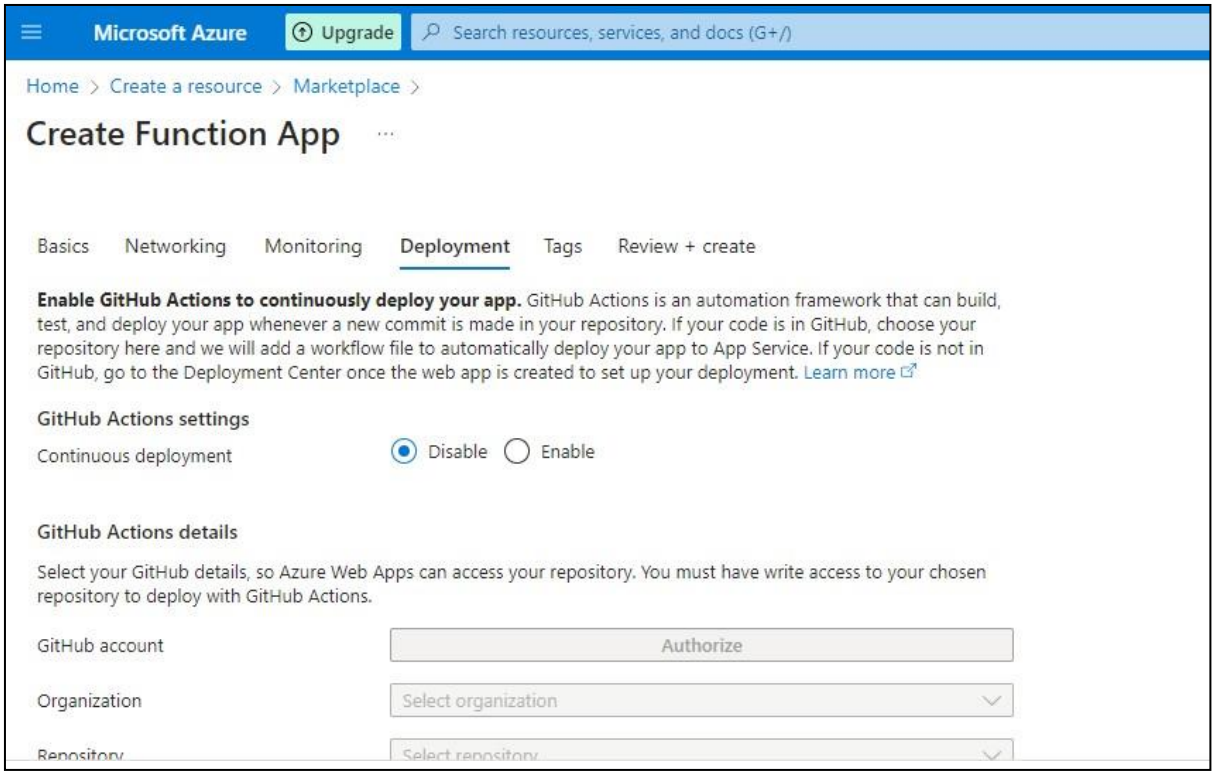

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > Create a resource > Marketplace >
Create Function App
Basics
Networking
Monitoring
Deployment
Tags
Review + create
Create a function app, which lets you group functions as a logical unit for easier management, deployment and sharing of resources. Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application.
Project Details
Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription *
Free Trial
Resource Group *
(New) pract
Create new
Instance Details
Function App name *
serverlessapiawt
.azurewebsites.net

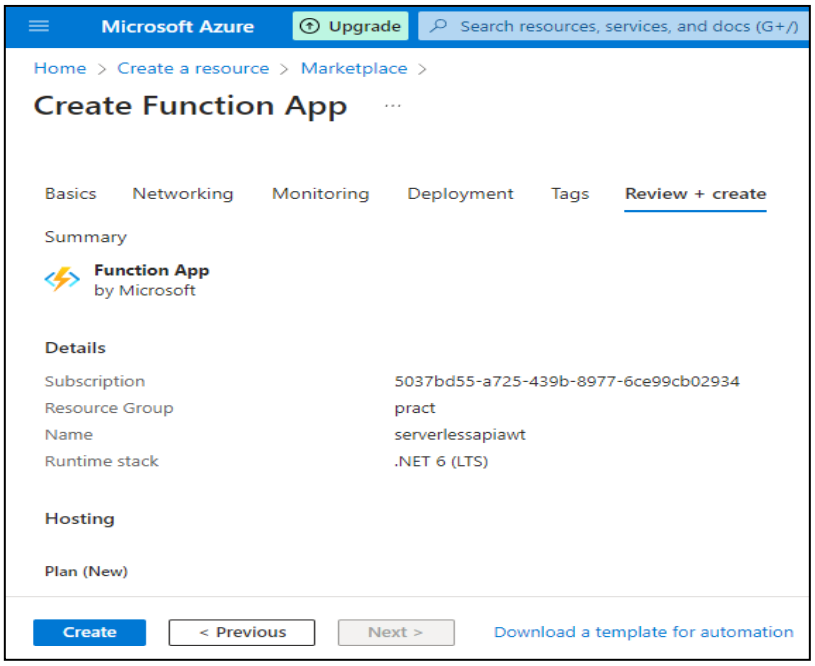

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > Create a resource > Marketplace >
Create Function App
Instance Details
Function App name *
serverlessapiawt
.azurewebsites.net
Do you want to deploy code or container image? *
Code
Container Image
Runtime stack *
.NET
Version *
6 (LTS)
Region *
Southeast Asia
Operating system
The Operating System has been recommended for you based on your selection of runtime stack.
Operating System *
Linux
Windows

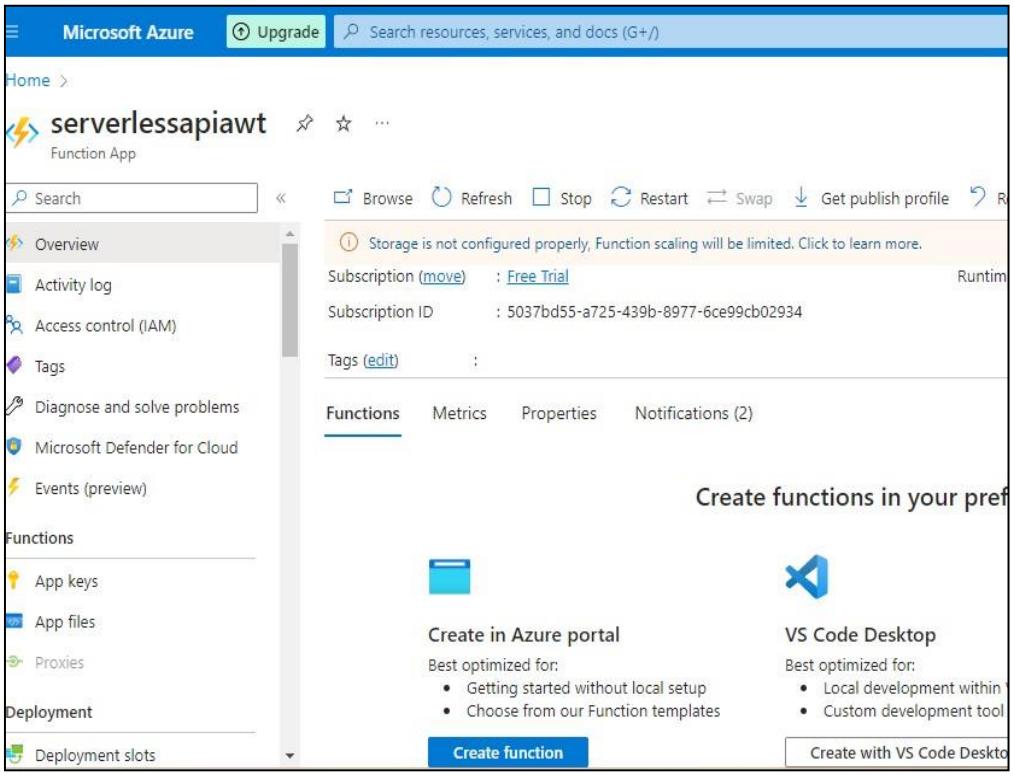

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > Create a resource > Marketplace >
Create Function App
The Operating System has been recommended for you based on your selection of runtime stack.
Operating System *
Linux
Windows
Hosting
The plan you choose dictates how your app scales, what features are enabled, and how it is priced. Learn more
Hosting options and plans *
Consumption (Serverless)
Optimized for serverless and event-driven workloads.
Functions Premium (Not supported for selected region and operating system)
Event based scaling and network isolation, ideal for workloads running continuously.
App service plan (Not supported for selected subscription)
Fully isolated and dedicated environment suitable for workloads that need large SKUs or need to co-locate Web Apps and Functions.

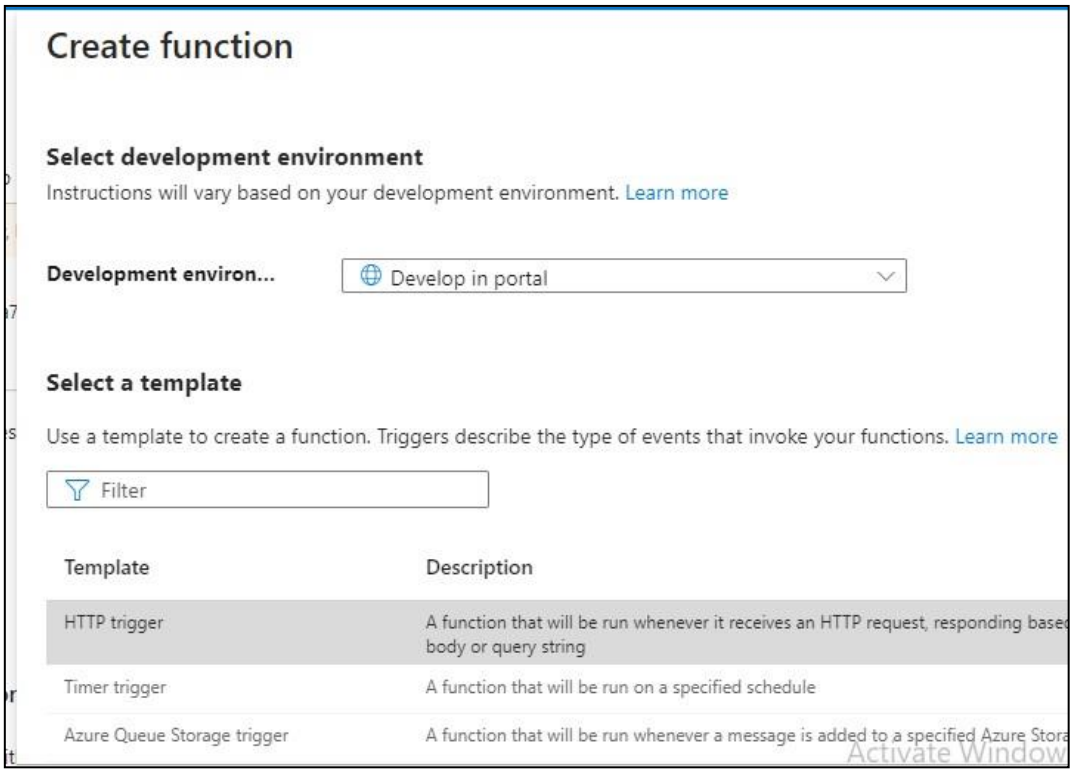

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > Create a resource > Marketplace >
Create Function App
Basics
Networking
Monitoring
Deployment
Tags
Review + create
Function Apps can be provisioned with the inbound address being public to the internet or isolated to an Azure virtual network. Function Apps can also be provisioned with outbound traffic able to reach endpoints in a virtual network, be governed by network security groups or affected by virtual network routes. By default, your app is open to the internet and cannot reach into a virtual network. These aspects can also be changed after the app is provisioned. Learn more
Enable public access *
On
Off
Network injection is only available in Functions Premium and Basic, Standard, Premium, Premium V2, Premium V3 Dedicated App Service plans.
Enable network injection
On
Off

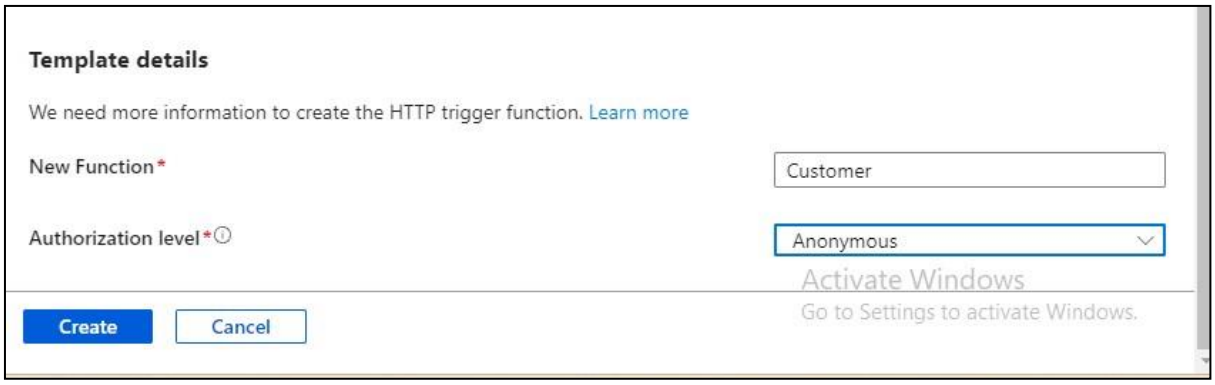

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > Create a resource > Marketplace >
Create Function App
Basics Networking Monitoring Deployment Tags Review + create
Azure Monitor application insights is an Application Performance Management (APM) service for developers and DevOps professionals. Enable it below to automatically monitor your application. It will detect performance anomalies, and includes powerful analytics tools to help you diagnose issues and to understand what users actually do with your app. Your bill is based on amount of data used by Application Insights and your data retention settings. Learn more
App Insights pricing
Application Insights
Enable Application Insights *
No
Yes
Application Insights *
(New) serverlessapiawt (Southeast Asia)
Create new
Region
Southeast Asia

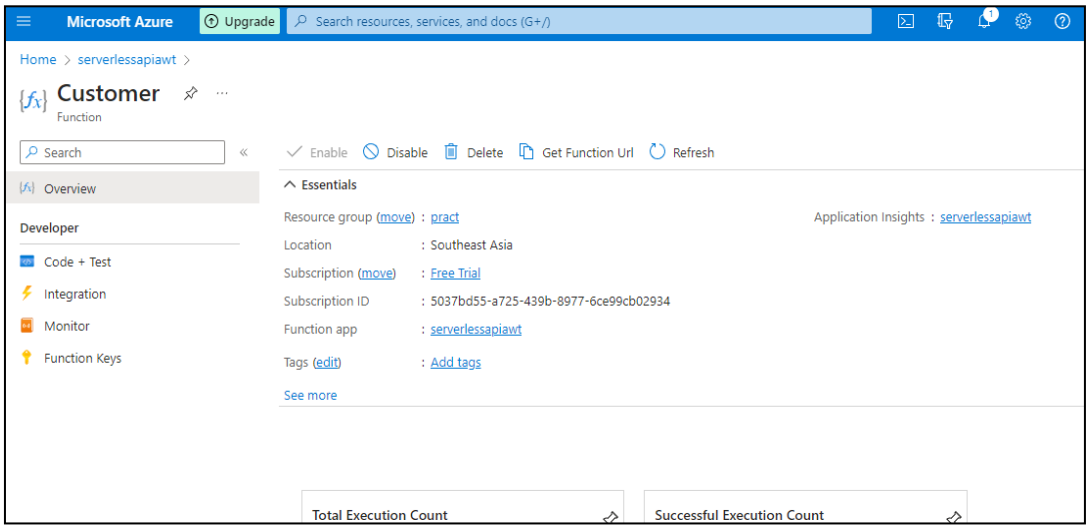

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > Create a resource > Marketplace >
Create Function App
Basics Networking Monitoring Deployment Tags Review + create
Enable GitHub Actions to continuously deploy your app. GitHub Actions is an automation framework that can build, test, and deploy your app whenever a new commit is made in your repository. If your code is in GitHub, choose your repository here and we will add a workflow file to automatically deploy your app to App Service. If your code is not in GitHub, go to the Deployment Center once the web app is created to set up your deployment. Learn more
GitHub Actions settings
Continuous deployment
Disable
Enable
GitHub Actions details
Select your GitHub details, so Azure Web Apps can access your repository. You must have write access to your chosen repository to deploy with GitHub Actions.
GitHub account
Authorize
Organization
Select organization
Repository
Select repository

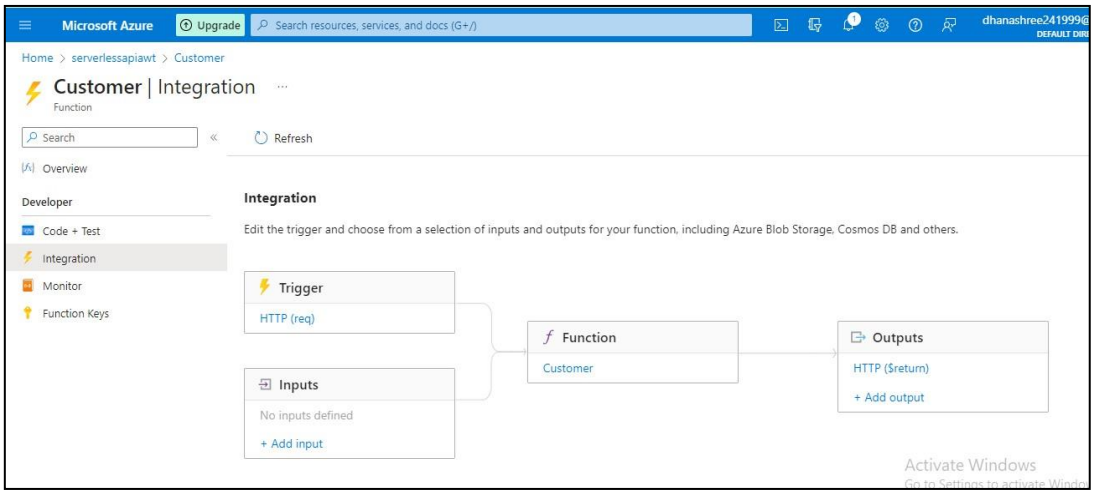

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > Create a resource > Marketplace >
Create Function App
Basics
Networking
Monitoring
Deployment
Tags
Review + create
Summary
Function App
by Microsoft
Details
Subscription
5037bd55-a725-439b-8977-6ce99cb02934
Resource Group
pract
Name
serverlessapiawt
Runtime stack
.NET 6 (LTS)
Hosting
Plan (New)
Create
< Previous
Next >
Download a template for automation

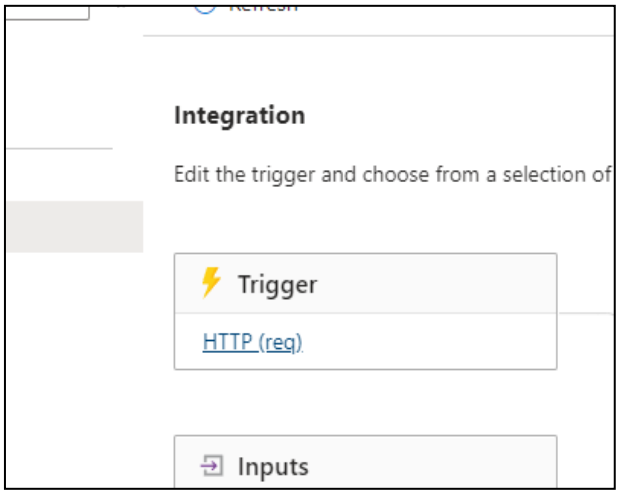

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home >
serverlessapiawt
Function App
Search
Browse
Refresh
Stop
Restart
Swap
Get publish profile
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Microsoft Defender for Cloud
Events (preview)
Functions
App keys
App files
Proxies
Deployment
Deployment slots
Storage is not configured properly, Function scaling will be limited. Click to learn more.
Subscription (move) : Free Trial
Subscription ID : 5037bd55-a725-439b-8977-6ce99cb02934
Tags (edit) :
Functions
Metrics
Properties
Notifications (2)
Create functions in your pref
Create in Azure portal
Best optimized for:
Getting started without local setup
Choose from our Function templates
Create function
VS Code Desktop
Best optimized for:
Local development within
Custom development tool
Create with VS Code Deskto

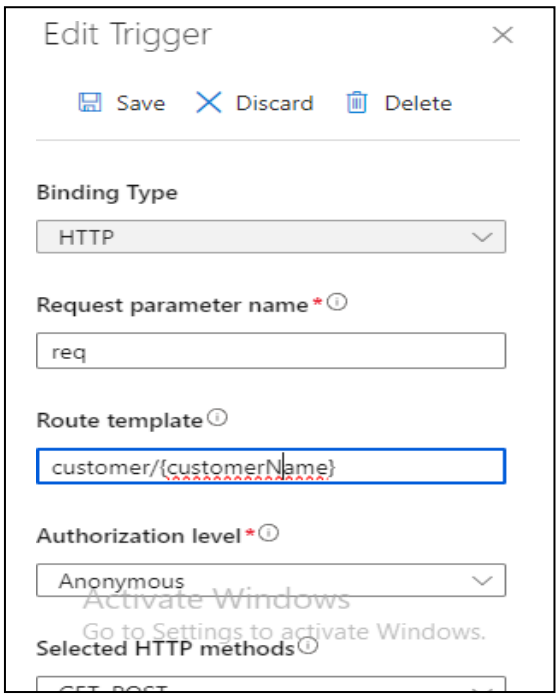

Create function
Select development environment
Instructions will vary based on your development environment. Learn more
Development environ...
Develop in portal
Select a template
Use a template to create a function. Triggers describe the type of events that invoke your functions. Learn more
Filter
Template
Description
HTTP trigger
A function that will be run whenever it receives an HTTP request, responding base
body or query string
Timer trigger
A function that will be run on a specified schedule
Azure Queue Storage trigger
A function that will be run whenever a message is added to a specified Azure Stora
Activate Window

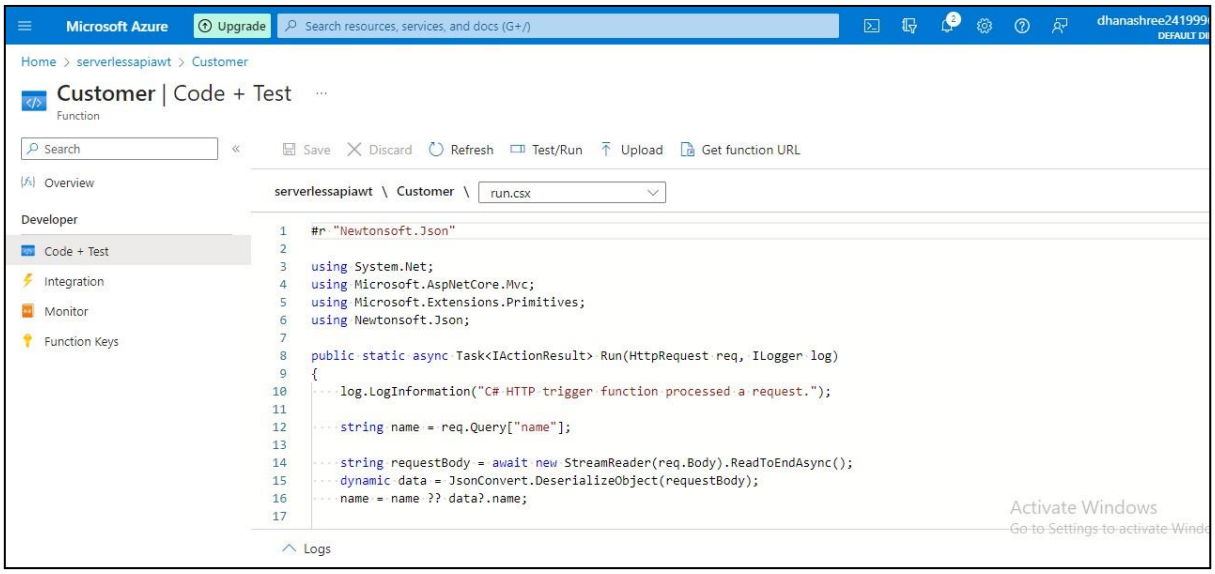

Template details
We need more information to create the HTTP trigger function. Learn more
New Function *
Customer
Authorization level *
Anonymous
Activate Windows
Go to Settings to activate Windows.
Create
Cancel

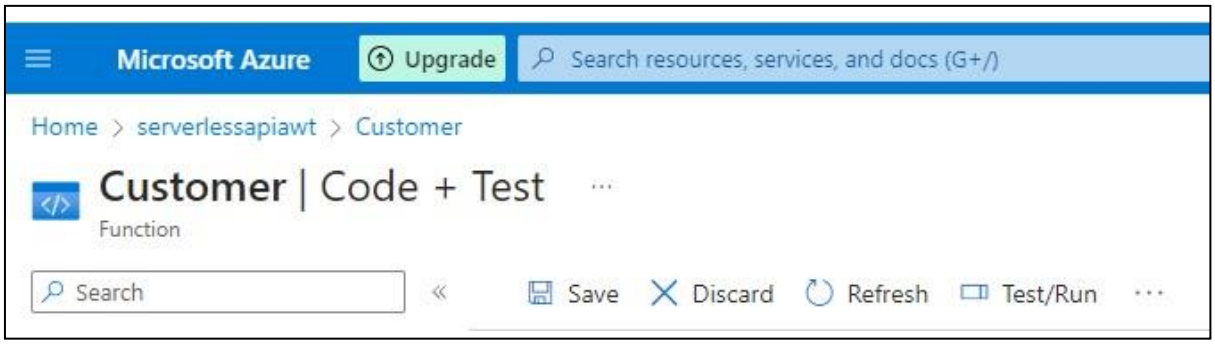

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > serverlessapiawt >
Customer
Function
Search
Overview
Developer
Code + Test
Integration
Monitor
Function Keys
Enable
Disable
Delete
Get Function Url
Refresh
Essentials
Resource group (move) : pract
Location : Southeast Asia
Subscription (move) : Free Trial
Subscription ID : 5037bd55-a725-439b-8977-6ce99cb02934
Function app : serverlessapiawt
Tags (edit) : Add tags
Application Insights : serverlessapiawt
See more
Total Execution Count
Successful Execution Count


Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > serverlessapiawt > Customer
Customer | Integration
Function
Search
Refresh
Overview
Developer
Code + Test
Integration
Monitor
Function Keys
Integration
Edit the trigger and choose from a selection of inputs and outputs for your function, including Azure Blob Storage, Cosmos DB and others.
Trigger
HTTP (req)
Inputs
No inputs defined
+ Add input
Function
Customer
Outputs
HTTP ($return)
+ Add output
Activate Windows


Integration
Edit the trigger and choose from a selection of
Trigger
HTTP (req)
Inputs

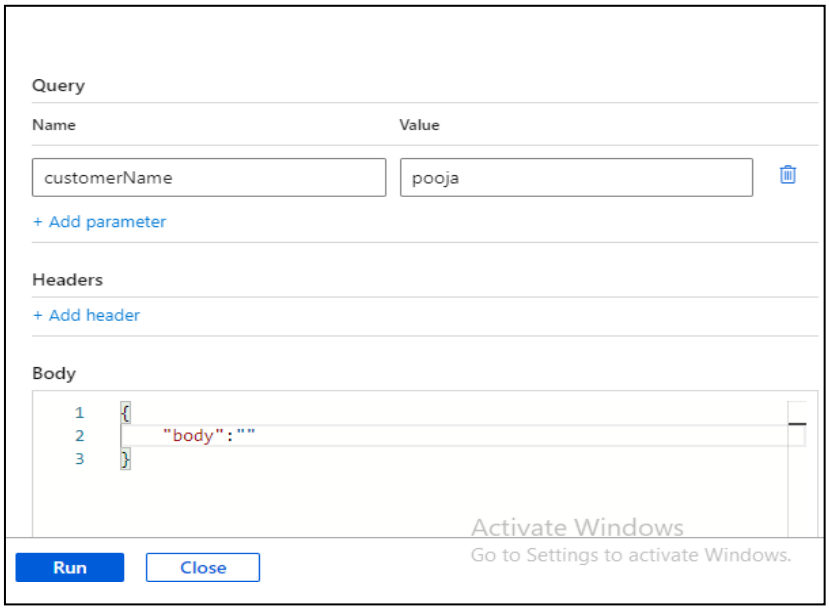

Edit Trigger
Save
Discard
Delete
Binding Type
HTTP
Request parameter name *
req
Route template
customer/{customerName}
Authorization level *
Anonymous
Activate Windows
Go to Settings to activate Windows.
Selected HTTP methods

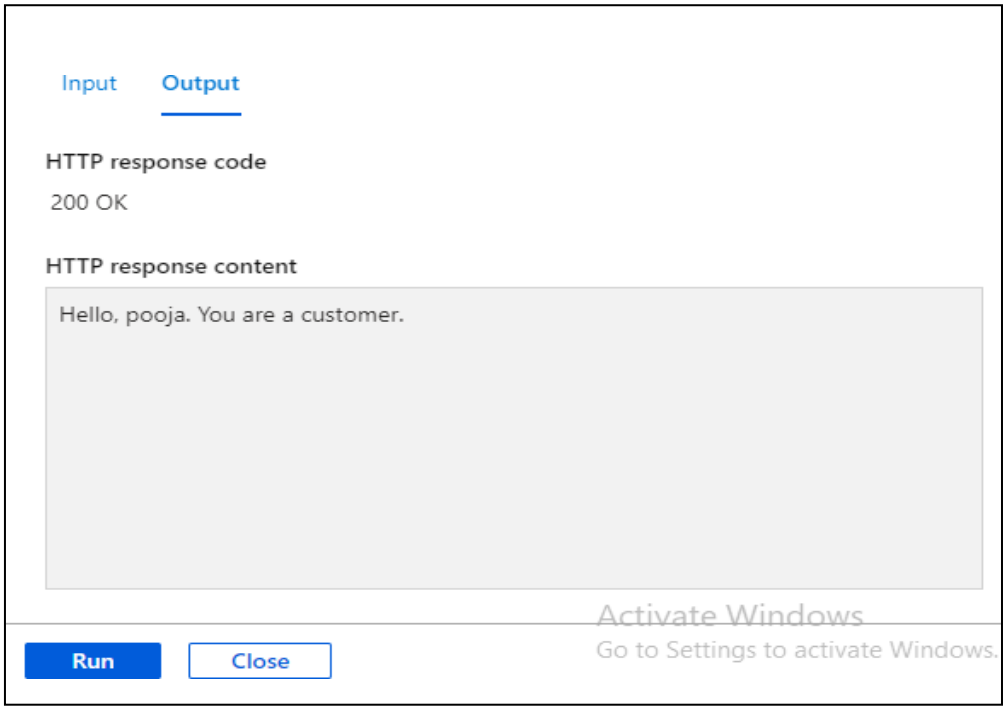


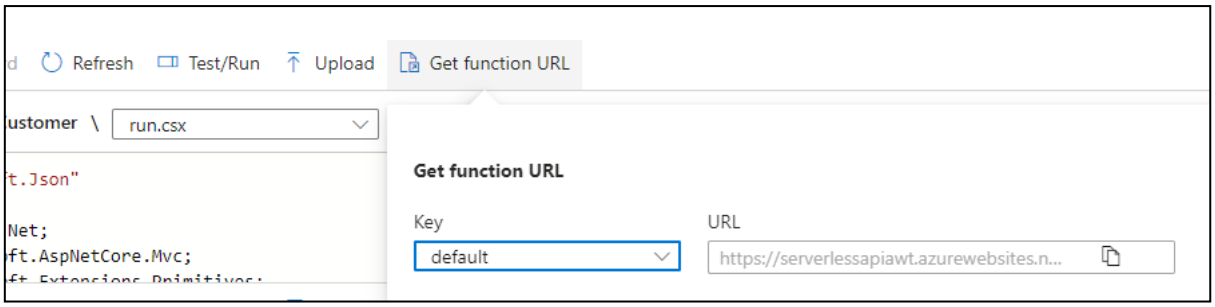



```
#r "Newtonsoft.Json"

using System.Net;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using Newtonsoft.Json;

public static async Task<IActionResult> Run(HttpRequest req, string
customerName, ILogger log)
{
    log.LogInformation("C# HTTP trigger function processed a
request.");

    string name = customerName;

    string requestBody = await new
StreamReader(req.Body).ReadToEndAsync();
    dynamic data = JsonConvert.DeserializeObject(requestBody);
    name = name ?? data?.name;
```

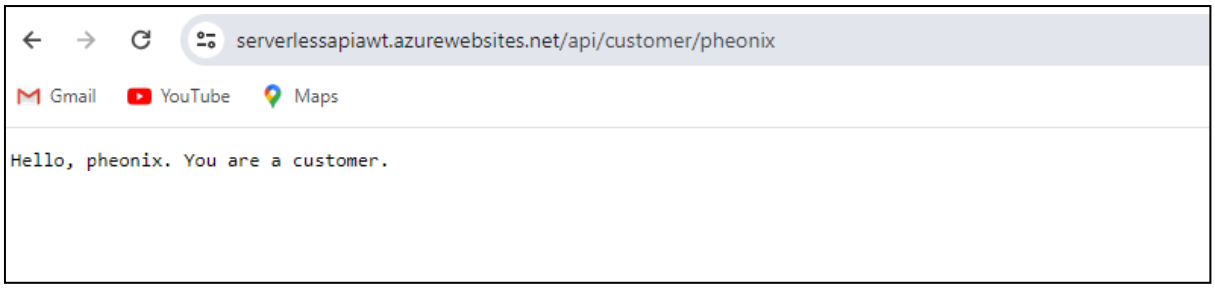
```
                                        string responseMessage =
string.IsNullOrEmpty(name)
        ? "This HTTP triggered function executed successfully. Pass a
name in the query string or in the request body for a personalized
response."
                : $"Hello, {name}. You are a customer.";

            return new OkObjectResult(responseMessage);
}
```






Refresh
Test/Run
Upload
Get function URL
ustomer \
run.csx
t.Json"
Net;
ft.AspNetCore.Mvc;
Get function URL
Key
default
URL
https://serverlessapiawt.azurewebsites.n...


https://serverlessapiawt.azurewebsites.net/api/customer/{customerName}
Gmail
https://serverlessapiawt.azurewebsites.net/api/customer/%7BcustomerName%7D
https://serverlessapiawt.azurewebsites.net/api/customer/{customerName} - Google Search


serverlessapiawt.azurewebsites.net/api/customer/pheonix
Gmail
YouTube
Maps
Hello, pheonix. You are a customer.

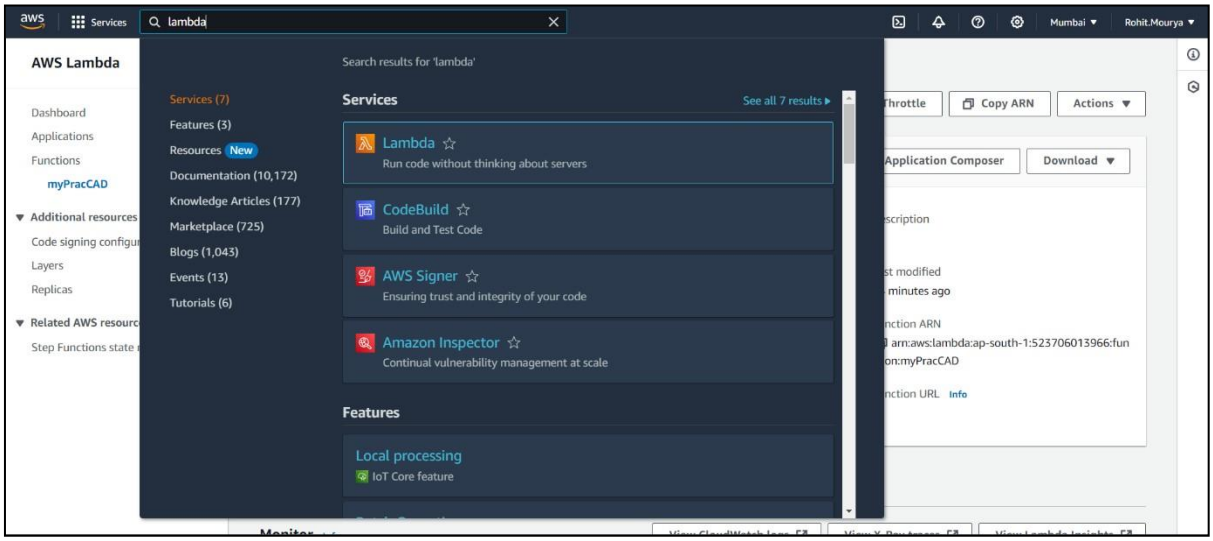
Practical No: 8

Create AWS Lambda Function

Steps to be followed:

Step: 01

To Create a Lambda function with the console, search Lambda in the search bar and open it.



Step: 02

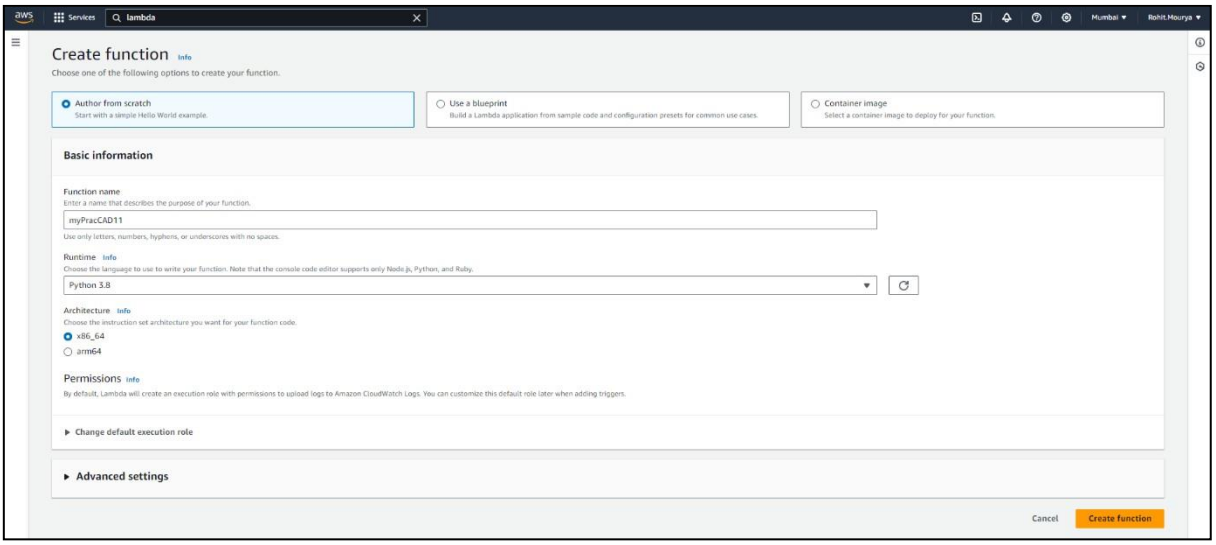
Choose create Function.

Select **Author from scratch**.

Give function name as ‘ **mypracCAD11**’.

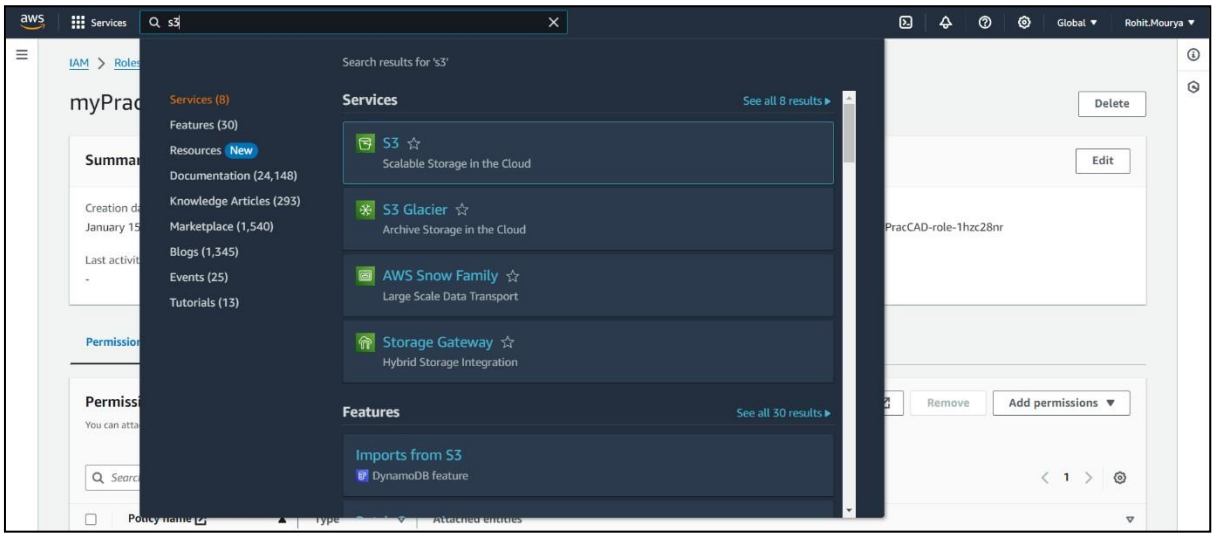
Select **Runtime** as **Python 3.8**.

Then, click create function.

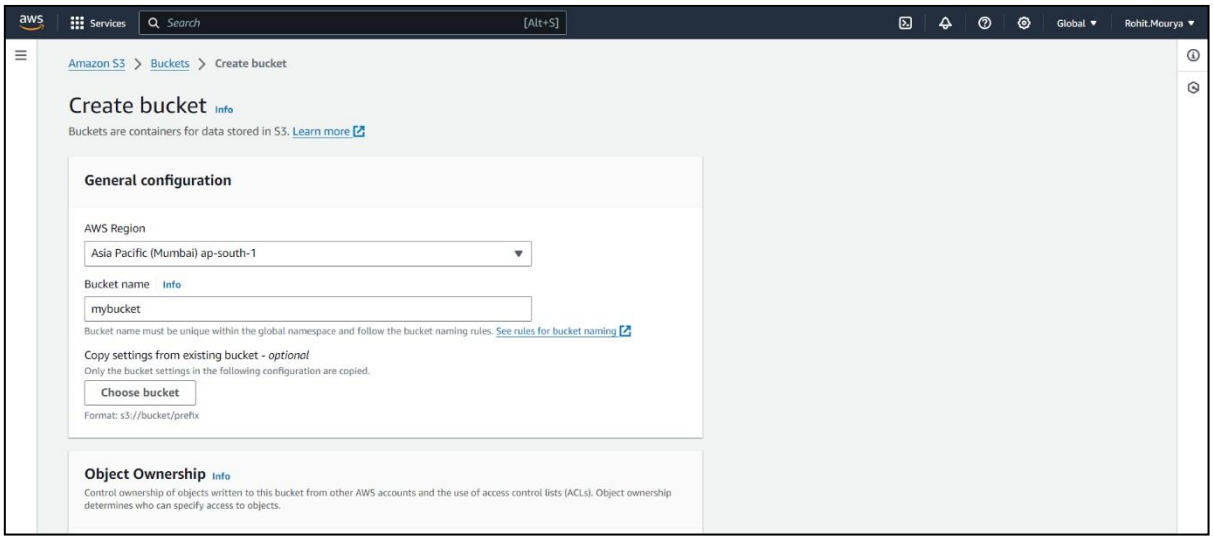


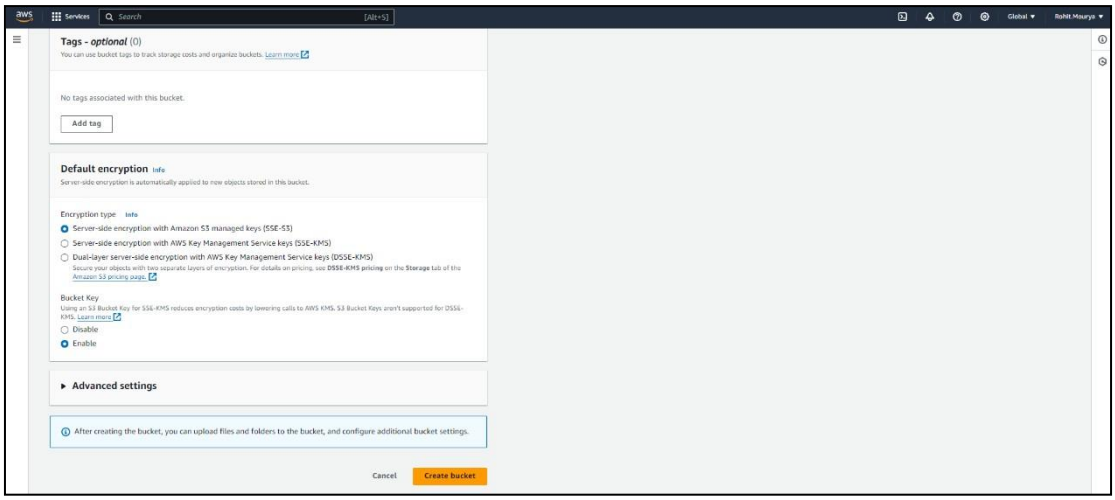
Step: 03

Now search “S3” int the search bar and open it.



Create a bucket with appropriate bucket name, **AWS region** and **encryption**.





Step: 04

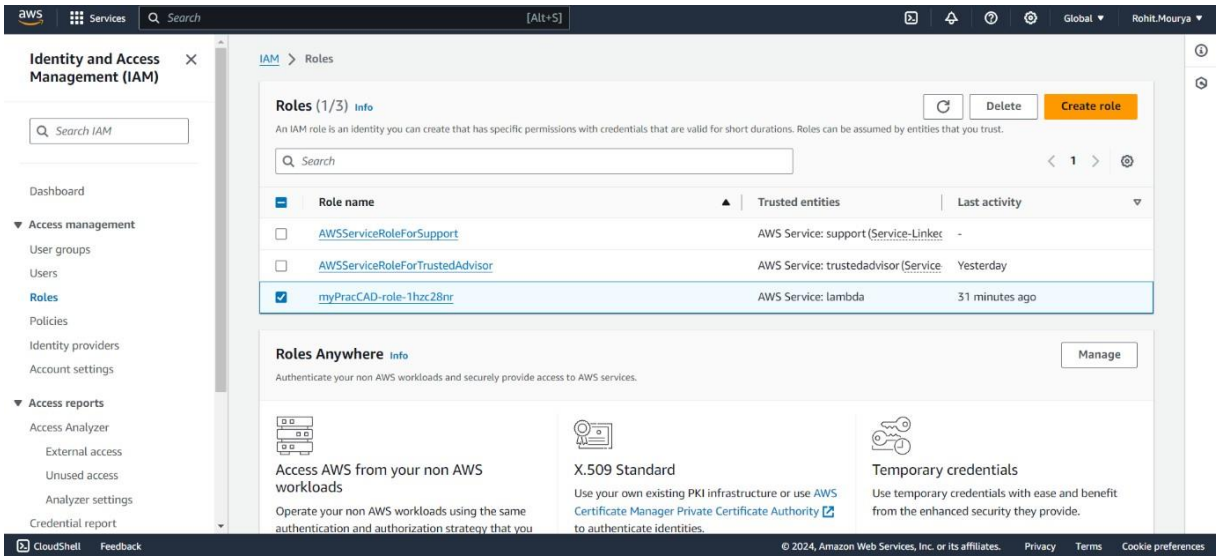
Now go the roles under IAM and select the role and attach policies for S3 bucket access.

Then, create a **Test case** and **deploy** the code.

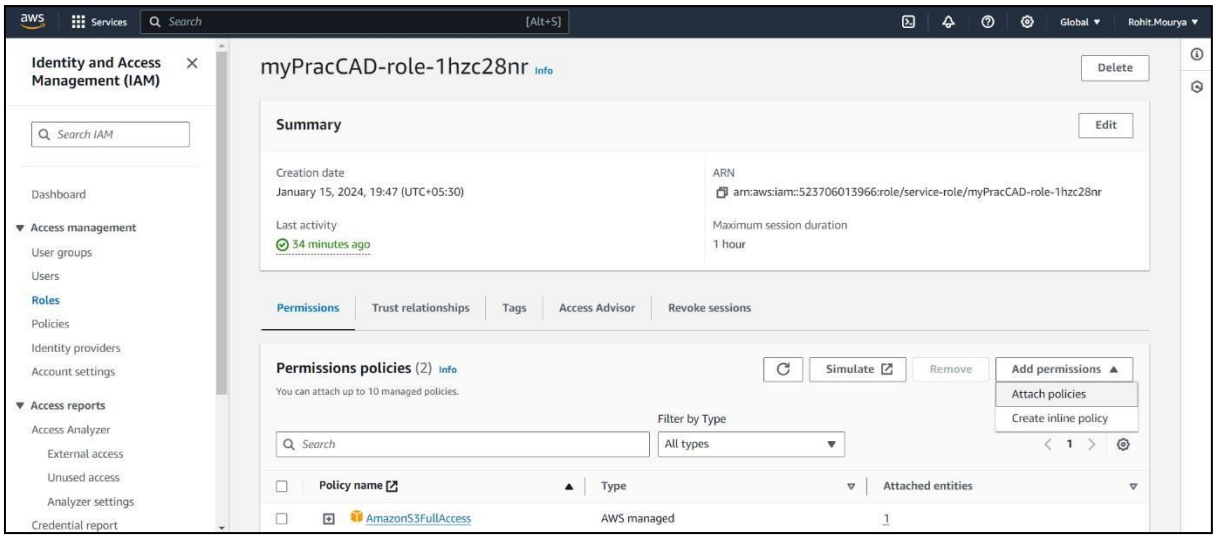
Code:

```
import json
import boto3
s3 = boto3.client('s3')
def lambda_handler(event, context):
bucket='mycadbuckets'        //enter filename that you save as S3 bucket name.
transactionToUpload = {}
transactionToUpload['transactionId'] = '12345'
transactionToUpload['type'] = 'PURCHASE'
transactionToUpload['amount'] = 20
```

```
transactionToUpload['customerID'] = 'CID-11111'
filename = 'CID-11111' + '.json'
uploadByteStream = bytes(json.dumps(transactionToUpload).encode('UTF-8'))
s3.put_object(Bucket=bucket, Key=filename, Body=uploadByteStream)
print('Put Completed')
```



Add the "**AmazonS3FullAccess**" policy the IAM role.

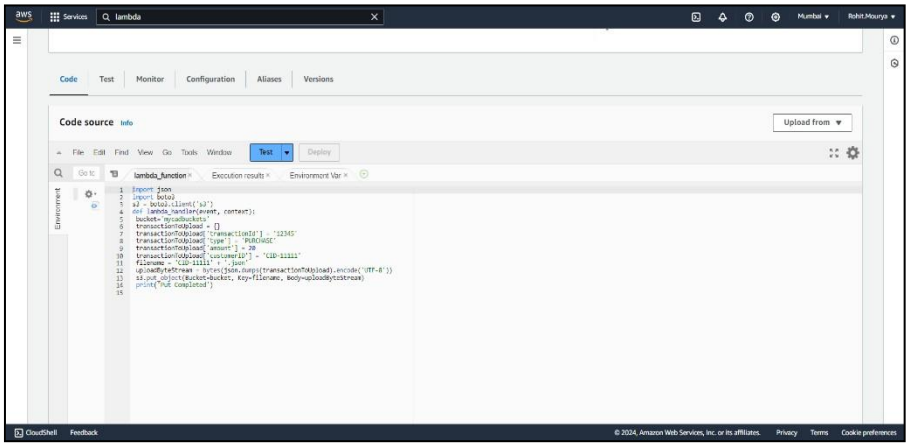


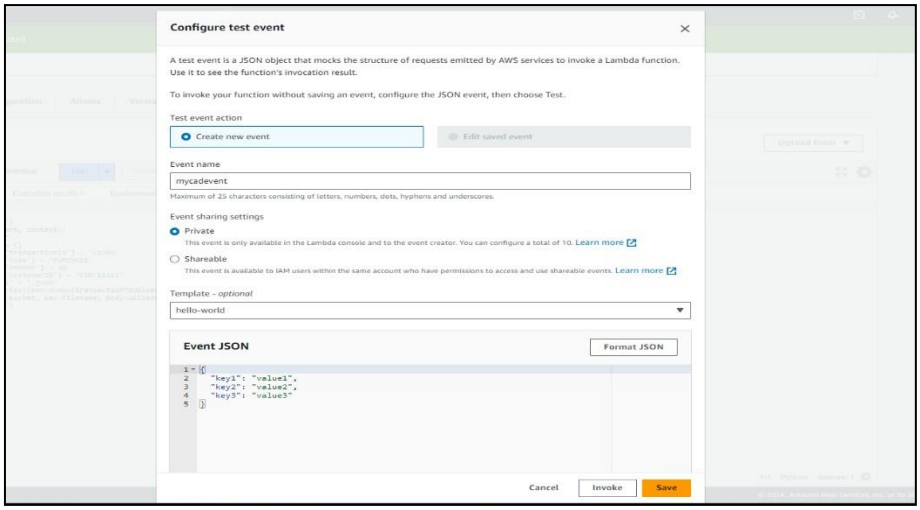
Step: 05

Now copy the paste the below code in the lambda Function which we have created earlier.

Then, create a **Test case** and **deploy** the code.

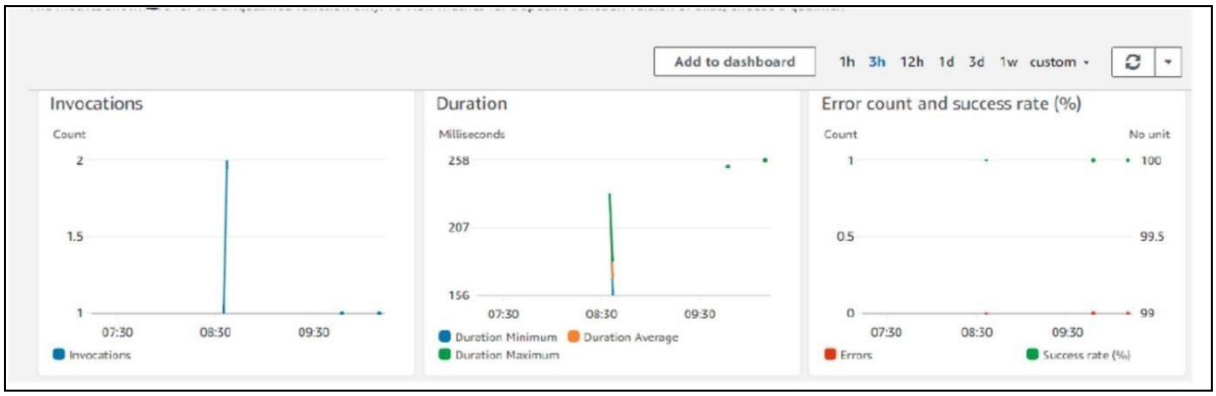
Code:



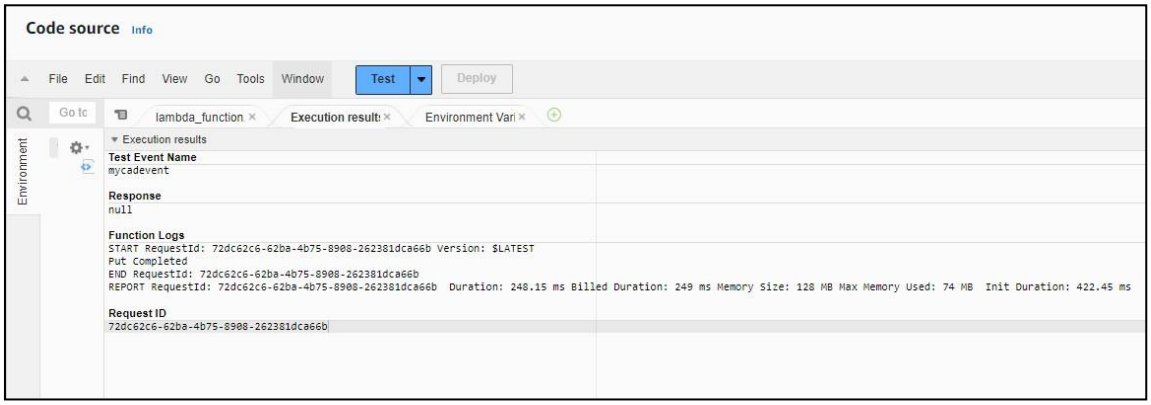


OUTPUT:-

Monitoring Result:



Execution Result:



Practical No: 9

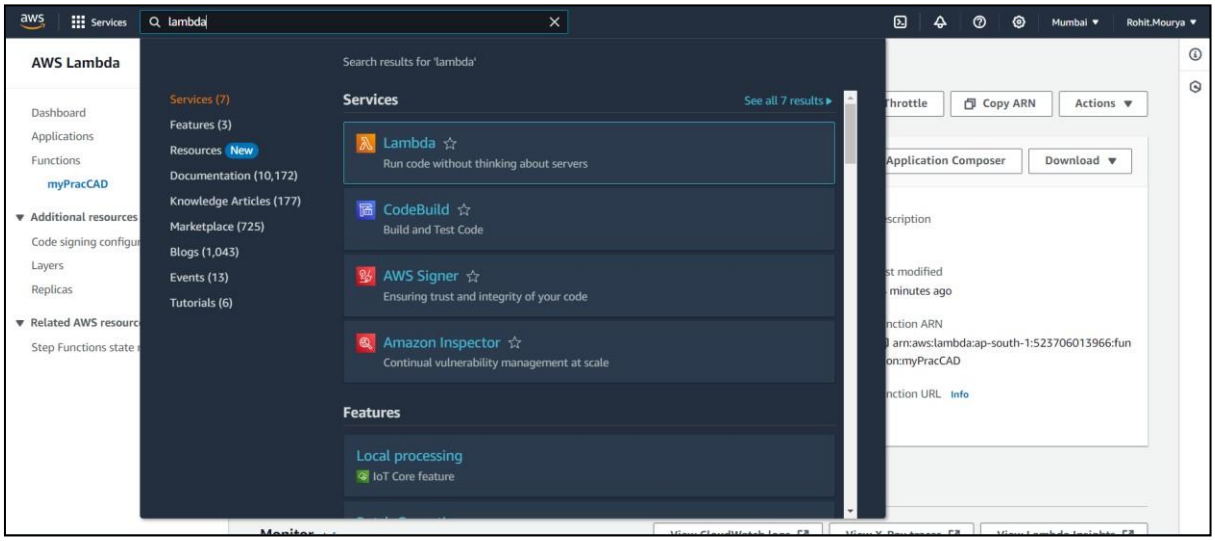
Build AWS Lambda with AWS API Gateway

Steps to be followed:-

Step: 01

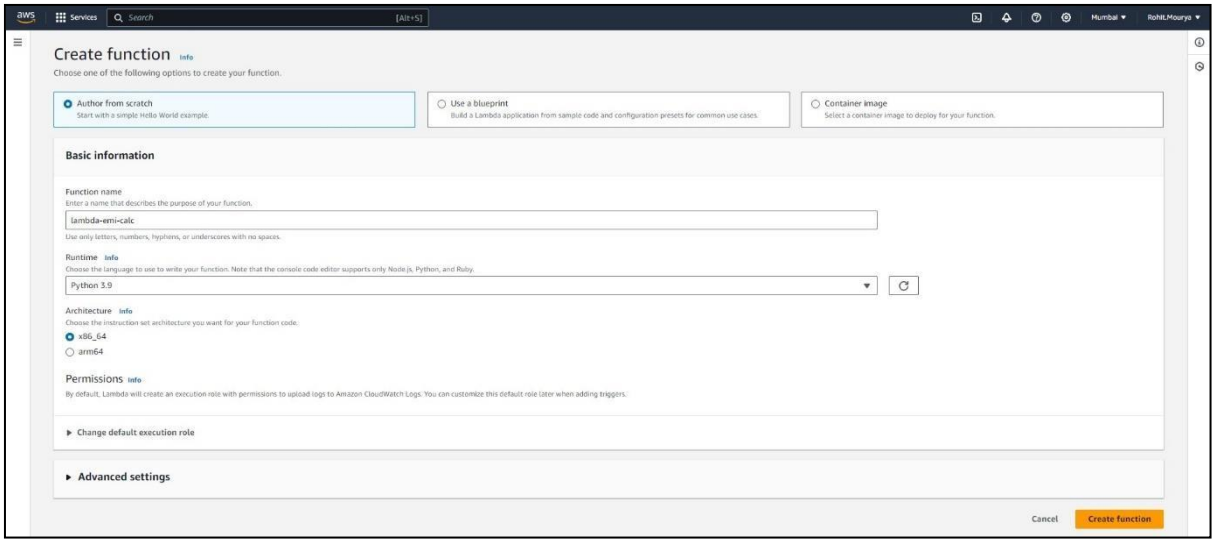
Login into the AWS console.

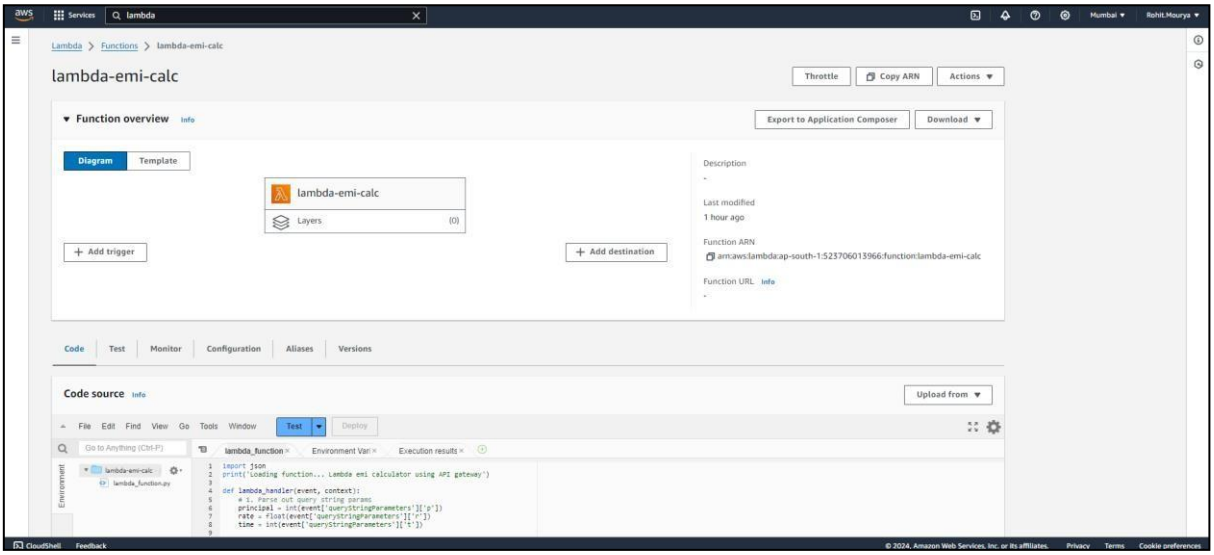
Search for lambda on the service bar.



Then, create a lambda function of name **lambda-emi-calc**.

And select the Runtime of function as **Python 3.9**, then click on create function.





Step: 02

Next step is to clear existing code and write emi calculator python code as below :

Code:

```
import json
print('Loading function... Lambda emi calculator using API gateway')
def lambda_handler(event, context):
# 1. Parse out query string params
principal = int(event['queryStringParameters']['p'])
rate = float(event['queryStringParameters']['r'])
time = int(event['queryStringParameters']['t'])
emi = emi_calculator(principal, rate, time)
# 2. Construct the body of the response object
transactionResponse = {
'p': principal,
'r': rate,
't': time,
'emi': emi
}


# 3. Construct http response object
```

responseObject = {

'statusCode': 200,

'headers': {

'Content-Type': 'application/json'

},

'body': json.dumps(transactionResponse)

}

4. Return the response object

return responseObject

def emi_calculator(p, r, t):

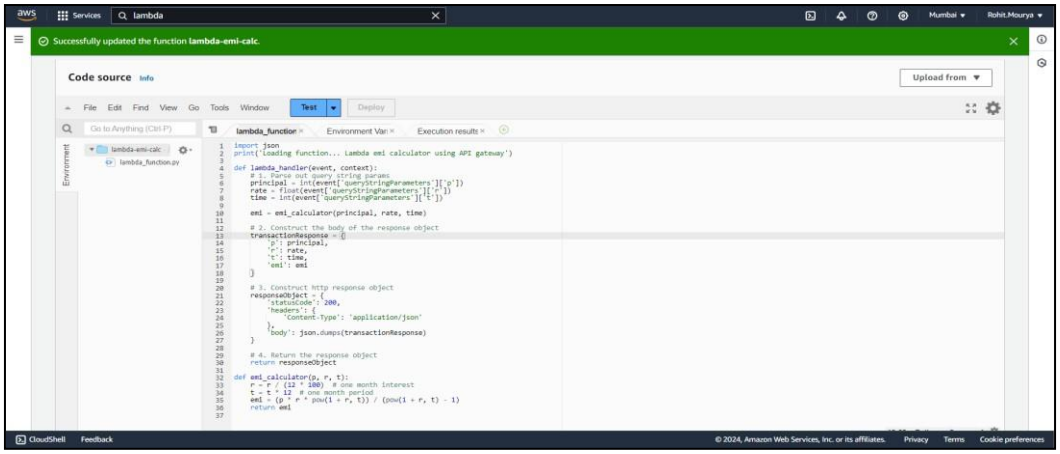
r = r / (12 * 100) # one month interest

t = t * 12 # one month period

emi = (p * r * pow(1 + r, t)) / (pow(1 + r, t) - 1)

return emi

Deploy the lambda function



Step: 03

Then go to dropdown besides test and configure test event with sample input values for testing lambda function code.

Code:

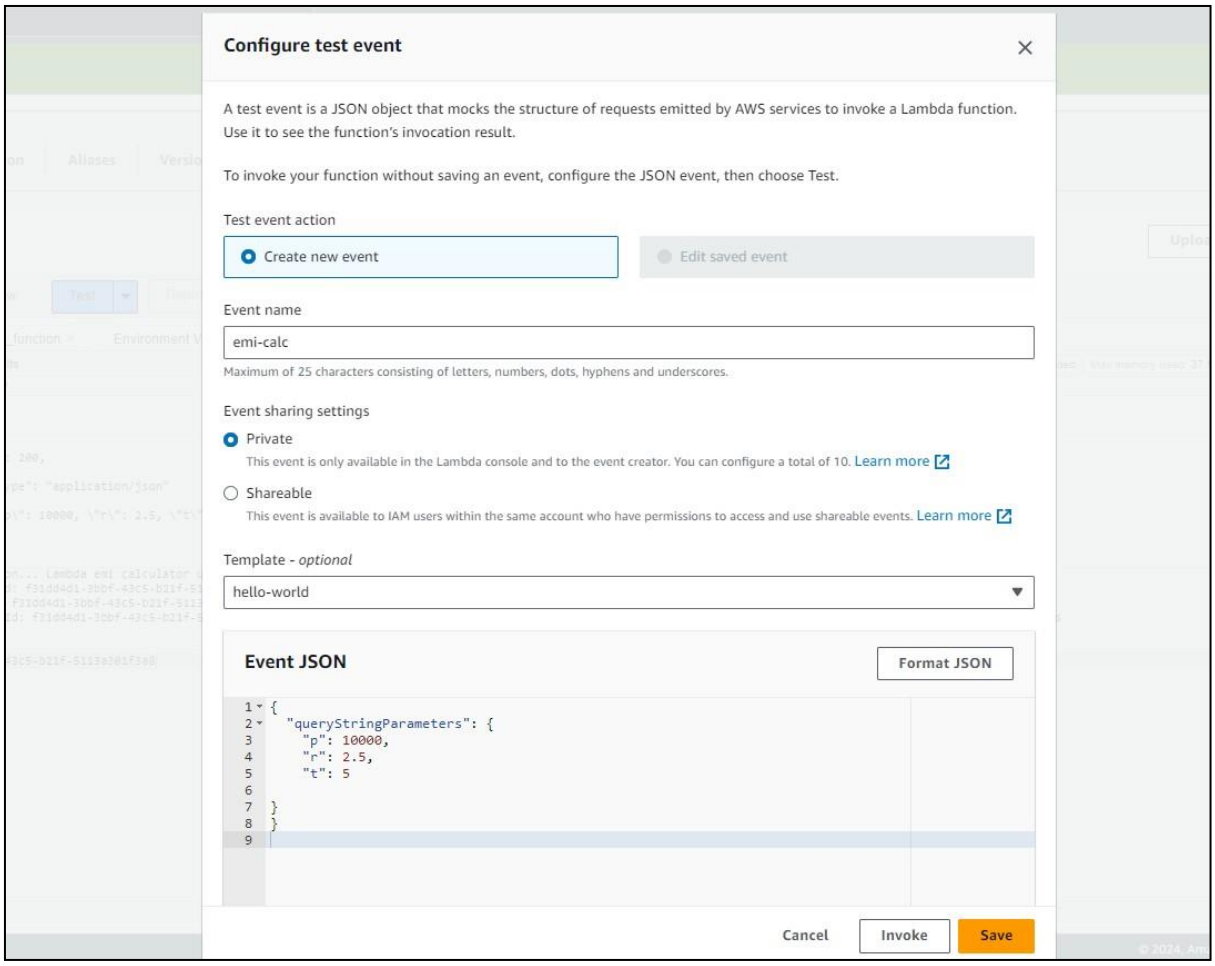
{

"queryStringParameters": {

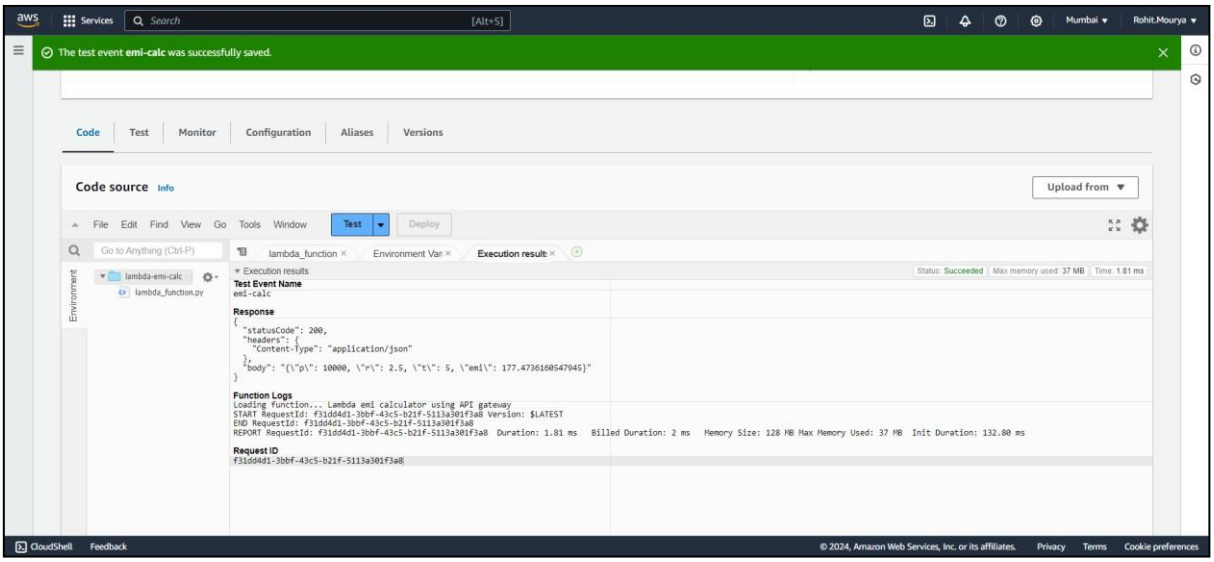
"p": 10000,

"r": 2.5,

```
"t": 5
}
}
```

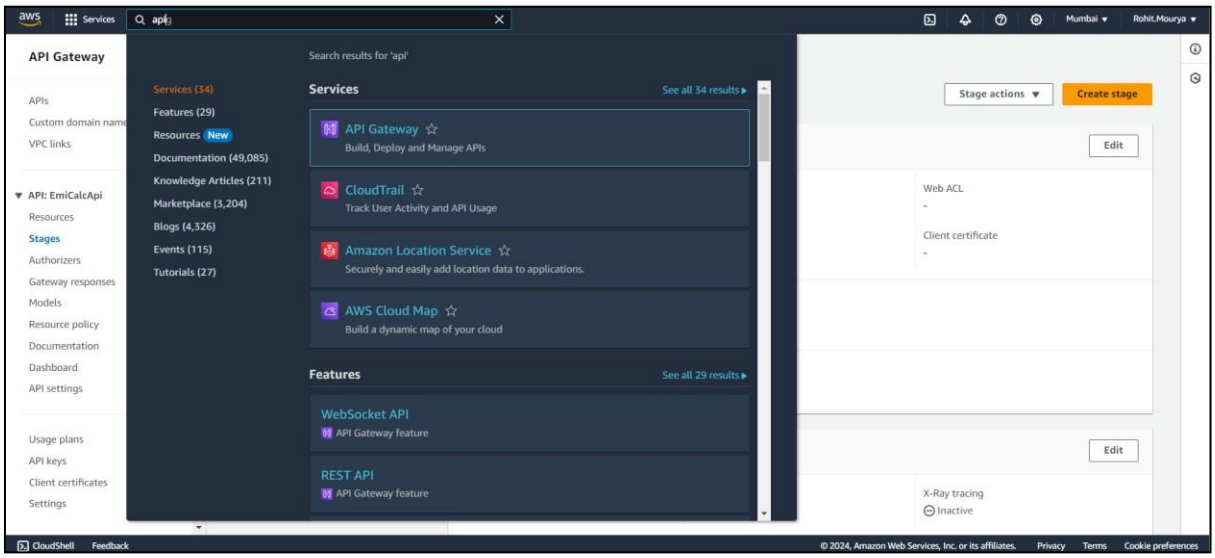


Test the code and get output without any errors.



Step: 04

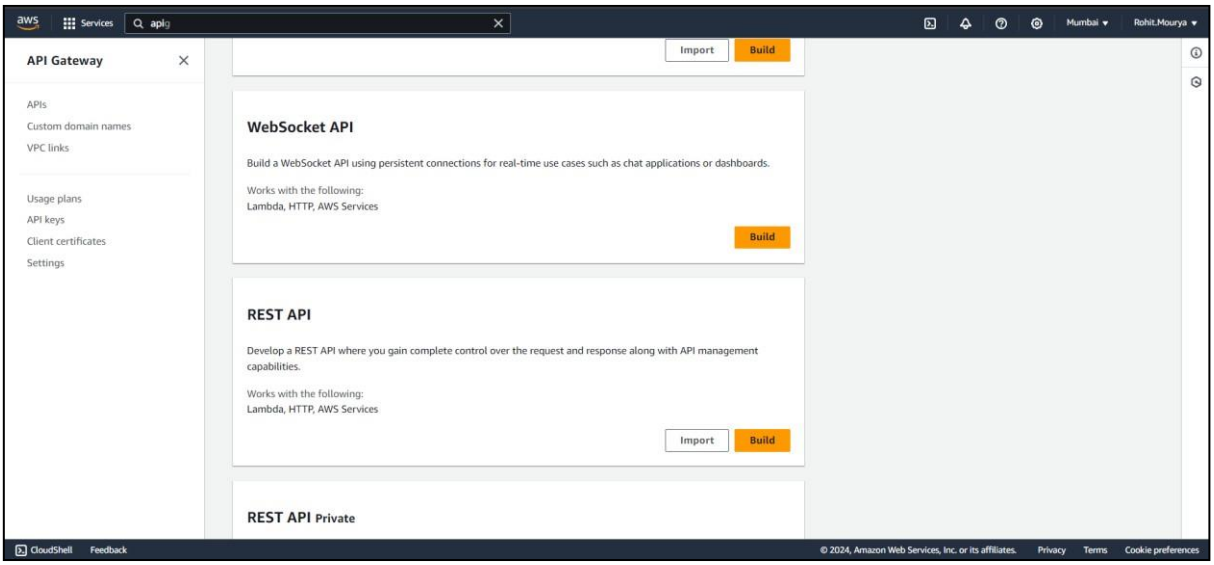
Go above and search for API gateway on the service bar and open it.



Click on create API gateway

Choose Rest API

Click on import



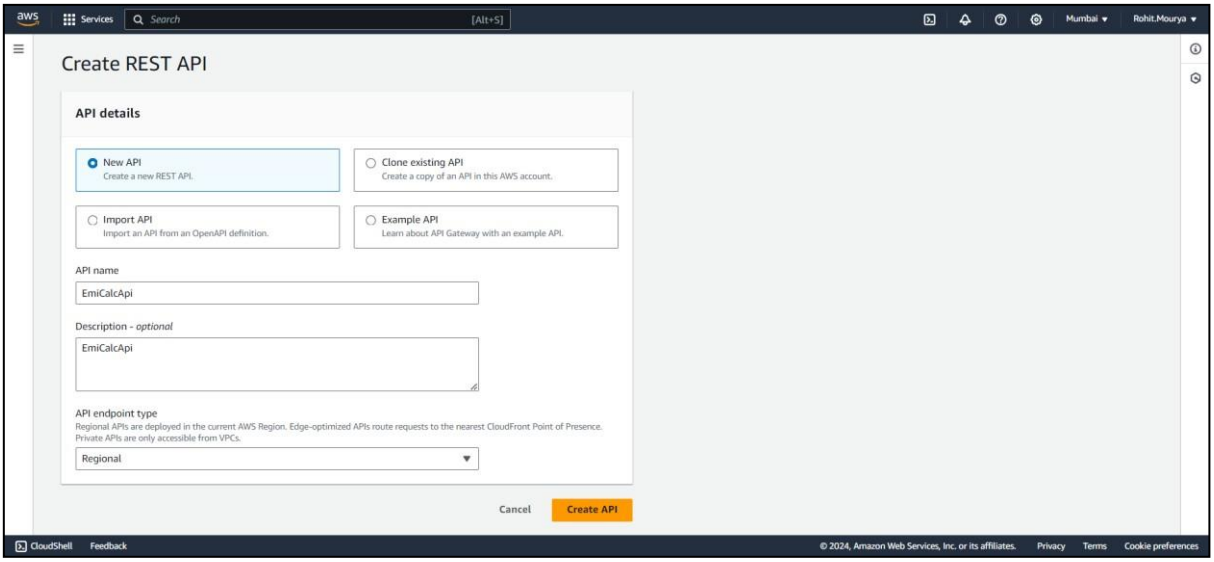
Create new API

API name = EmiCalcApi

Description = EmiCalcApi

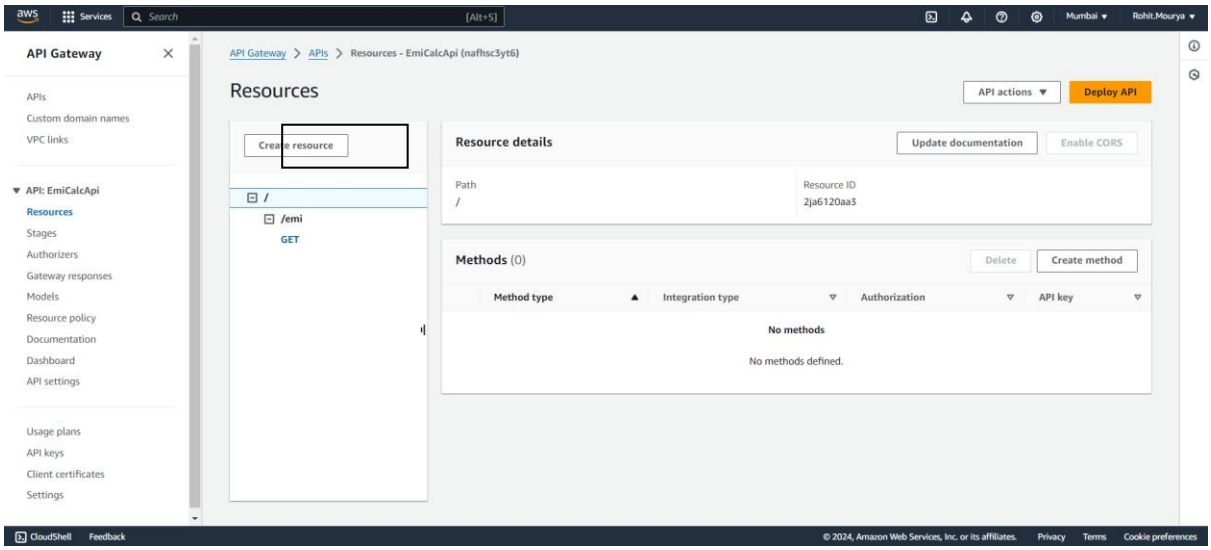
Endpoint type = Regional

Click on create tab

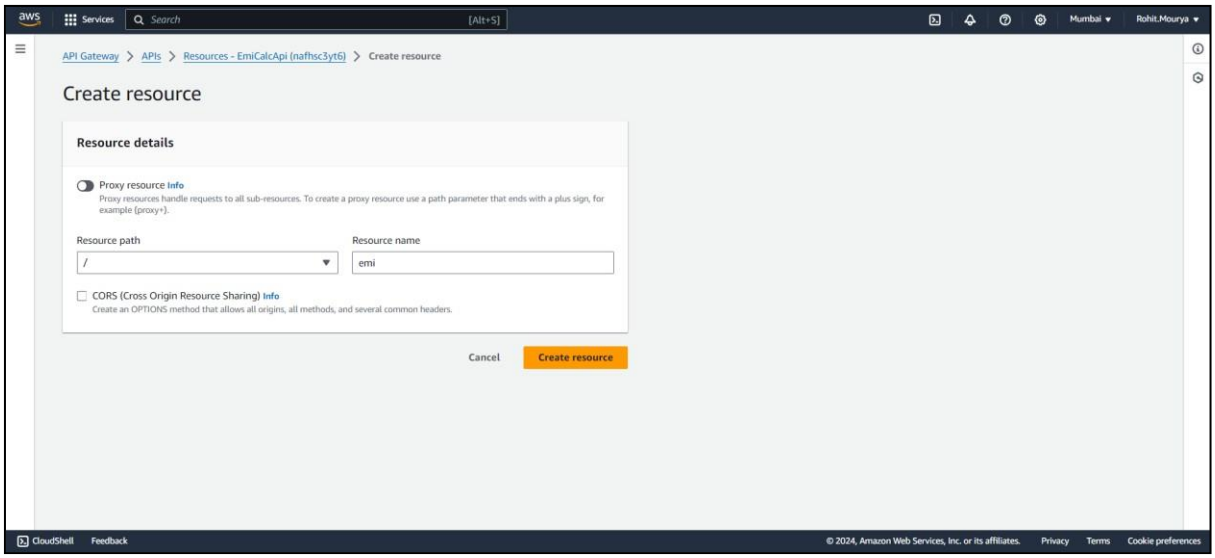


Step: 05

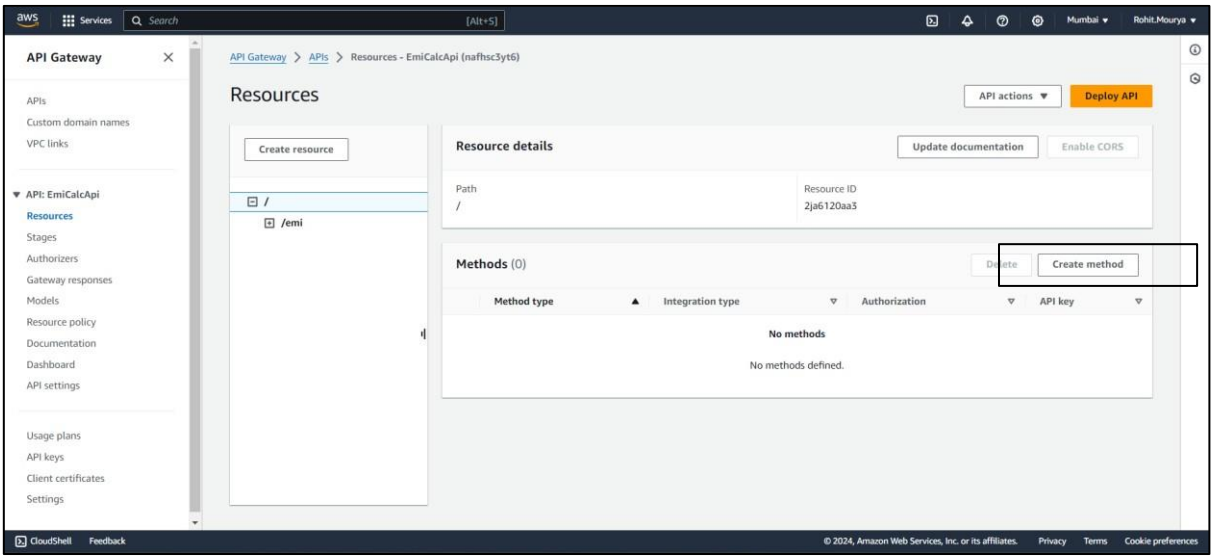
Next step clicks on actions then create resource



Enter resource name as **emi** and click create resource



Now click on create method.



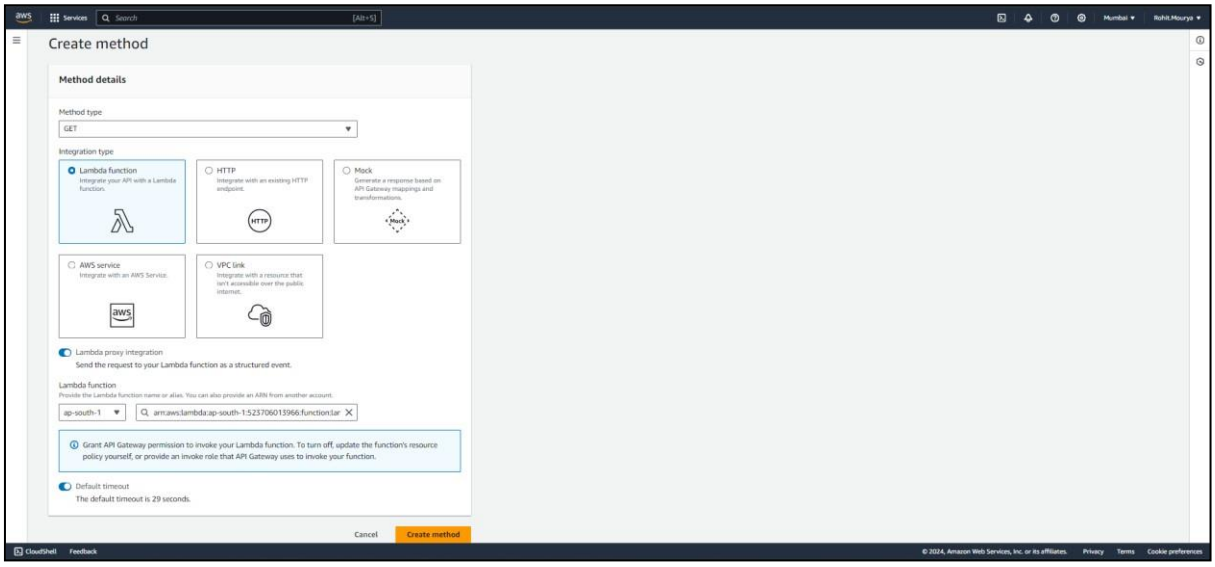
Select Method as GET.

Integration type as Lambda Function.

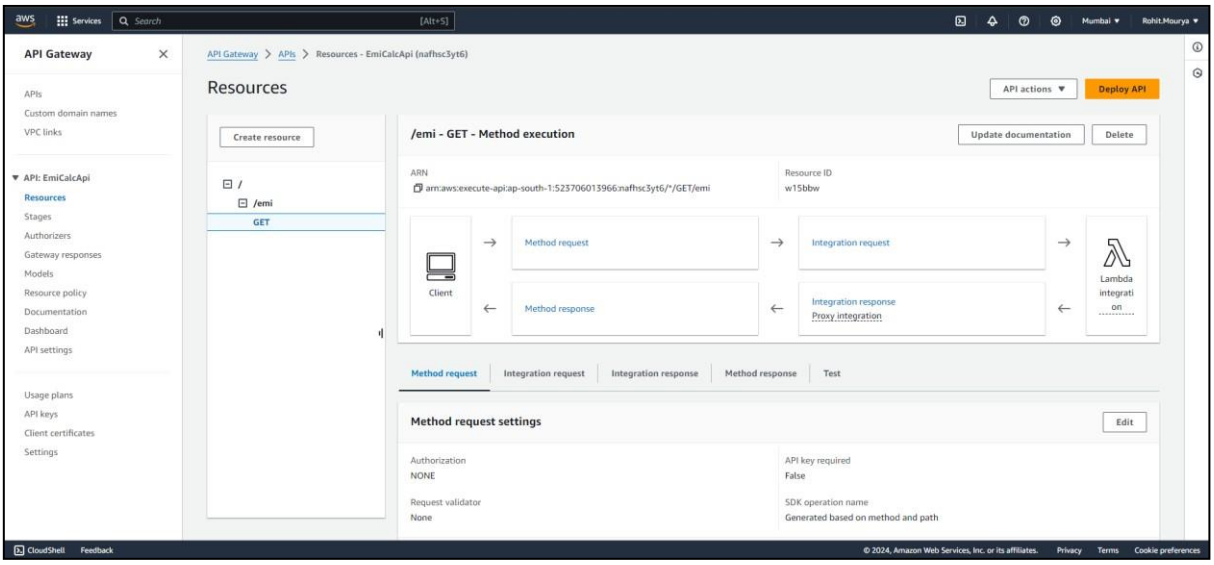
Enable the Lambda proxy integration.

Then, select the Lambda function which we have created previously.

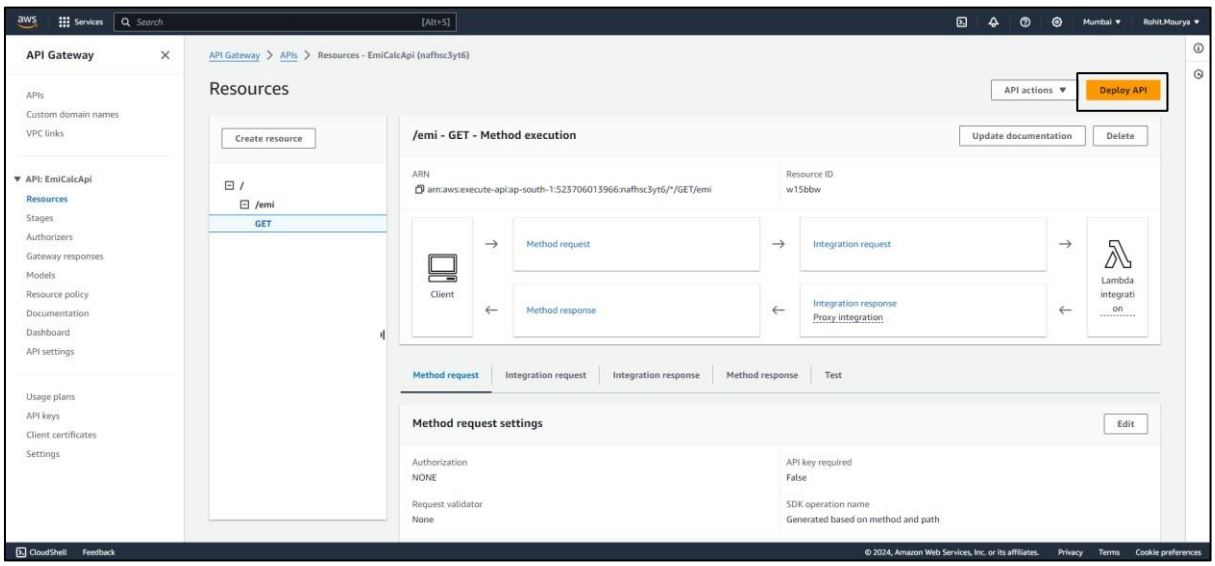
And click on create method.



The GET method will look below.

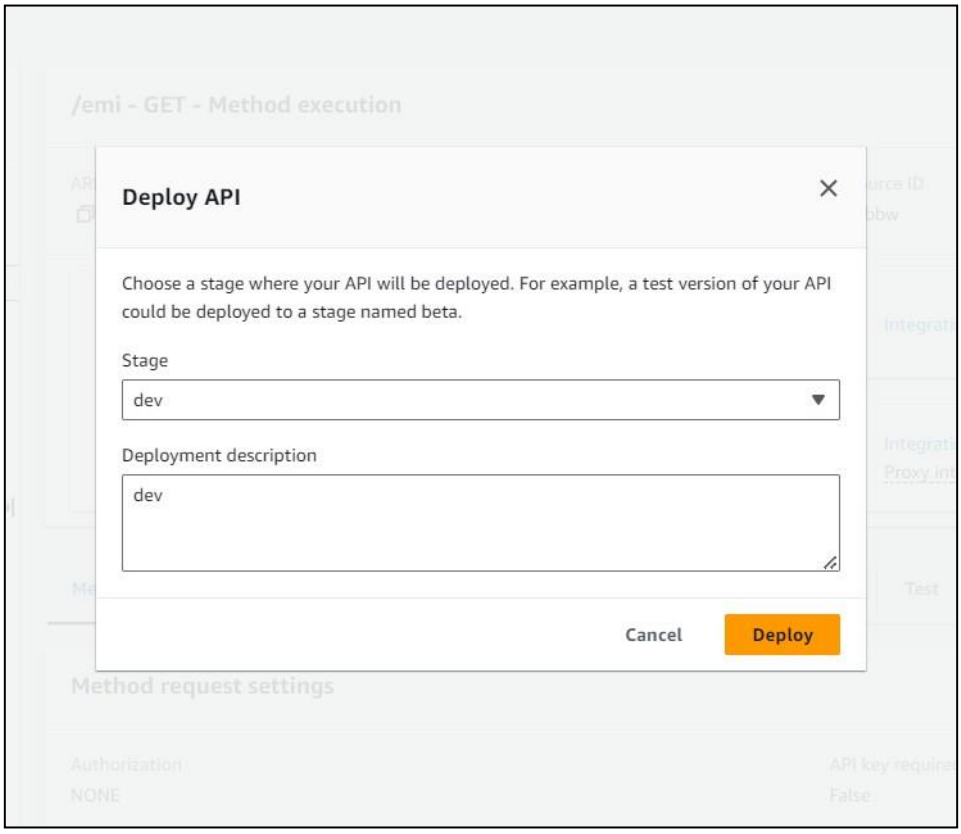


Now deploy the method by click on **Deploy API** button

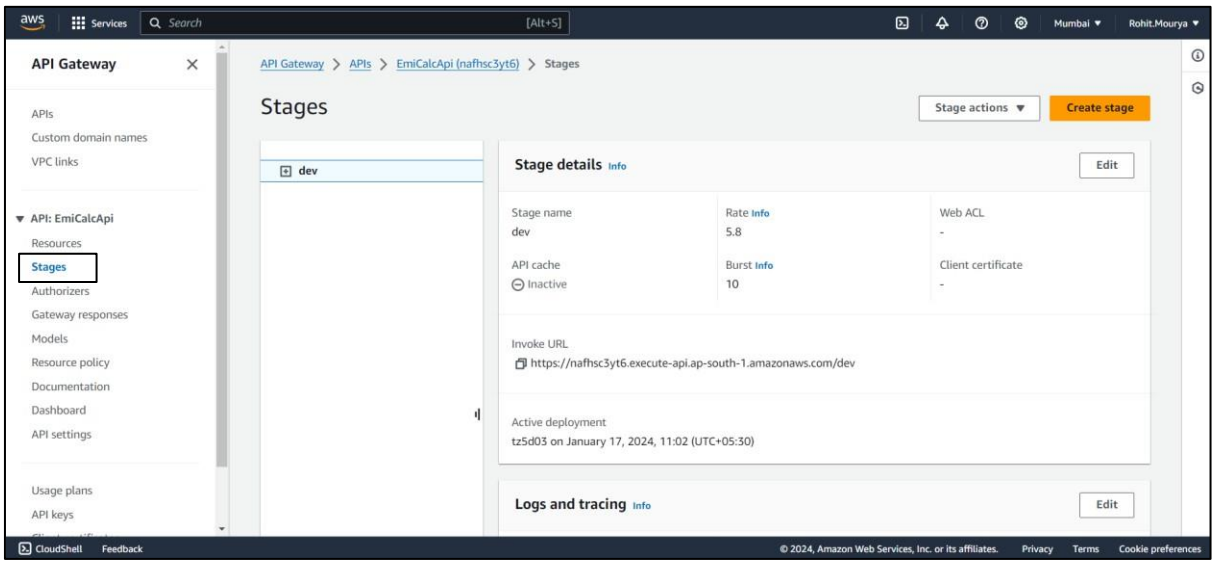


Enter stage name as **dev**.

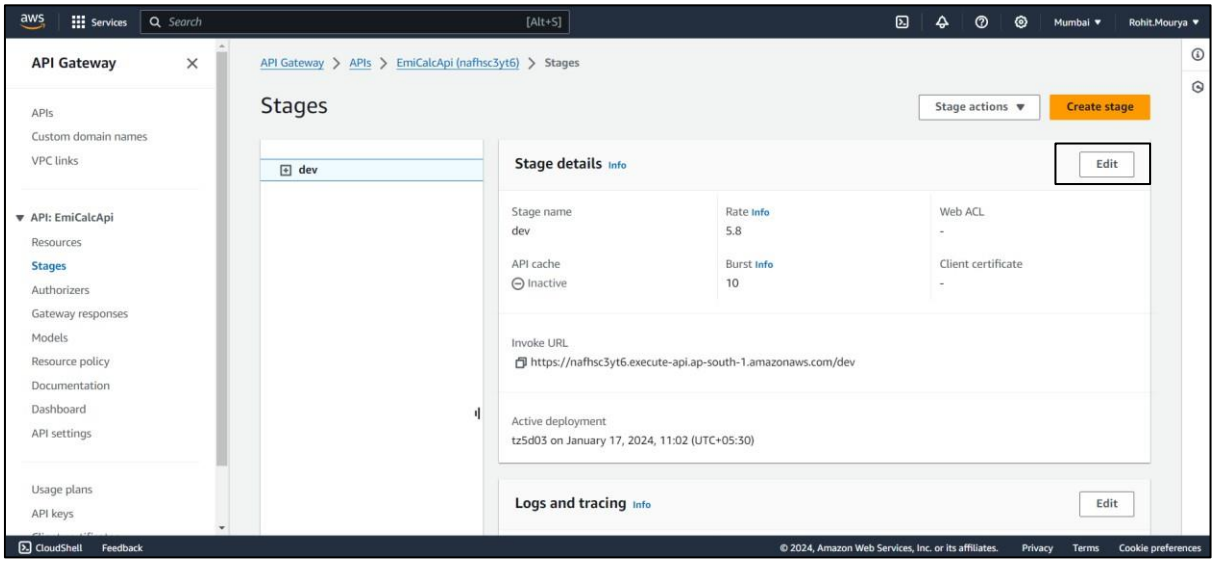
Deployment description as **dev**.



After deploying go to the Stage option from Api gateway column.



Then click on the edit button from the stage details tab.

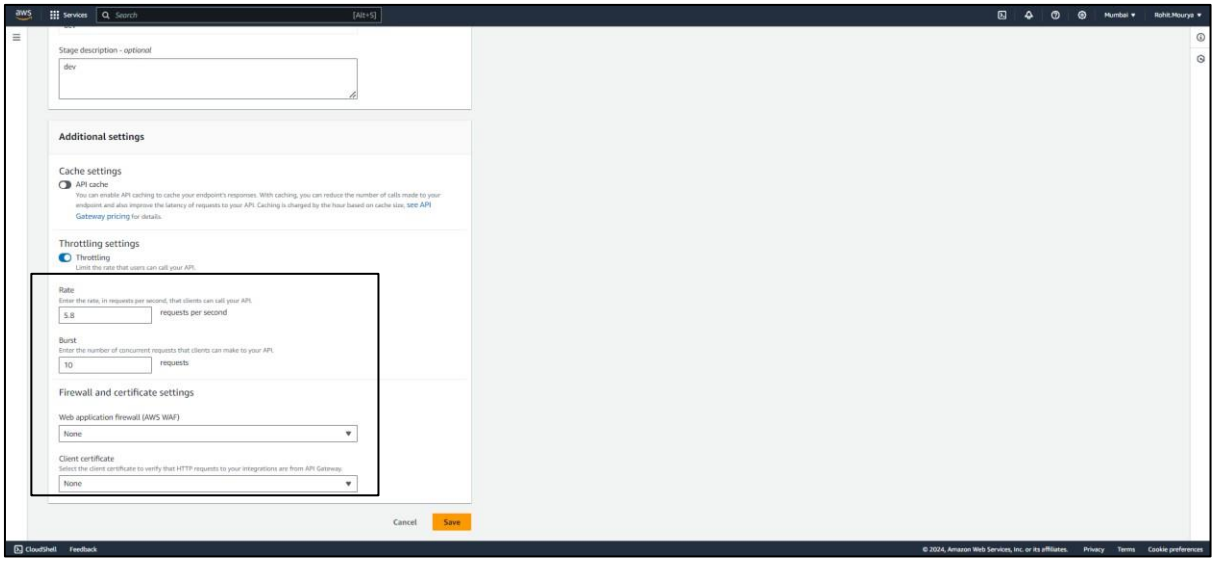


After clicking on edit now we have to change the rate and burst for calculating EMI.

To calculate the EMI, we have to place valid parameters

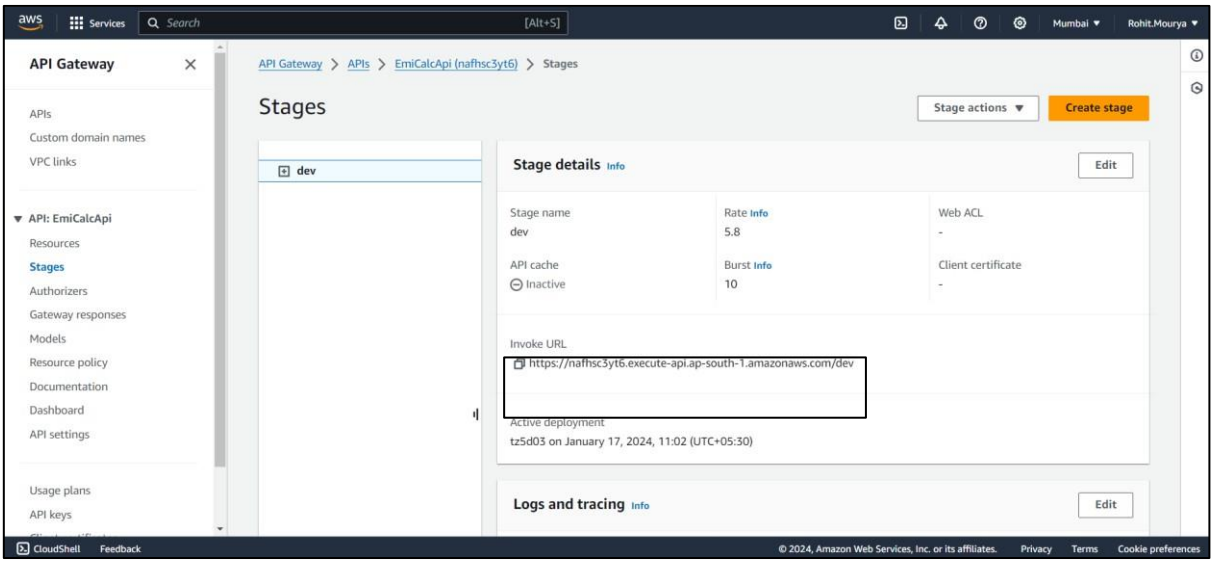
Principle = p , Rate = r and Time = t

Then add rate (r) as certain value 5.8 and the final value time as 10



Then copy the Invoke URL form the stage details and paste it in the browser.

Need to add after url - /emi?p=10000&r=5.8&t=10



Now copy and paste the whole url in the browser and you will have the required EMI.

OUTPUT:-