

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 REQUIREMENTS FOR
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

SR.NO	Description	Page Number	Date
1	Develop an ASP.NET Core MVC based Stateless Web App.	1	
2	Develop a Spring Boot API.	17	
3	a. Create an Azure Kubernetes Service Cluster b. Configure Visual Studio to Work with an Azure Kubernetes Service Cluster c. Configure Visual Studio Code to Work with an Azure Kubernetes Service Cluster	27	
4	Create an AKS cluster a. From the portal b. With Azure CLI	38	
5	Create an API management service	45	
6	AWS API Gateway Authorizer	51	
7	Create a serverless API using Azure functions	60	
8	Create an AWS Lambda function	70	
9	Build AWS Lambda with AWS API gateway	76	

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.

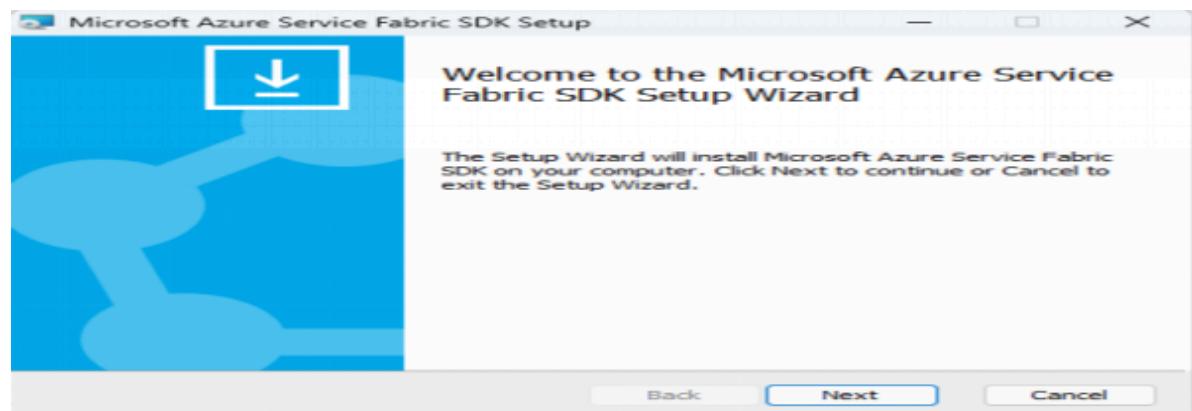
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

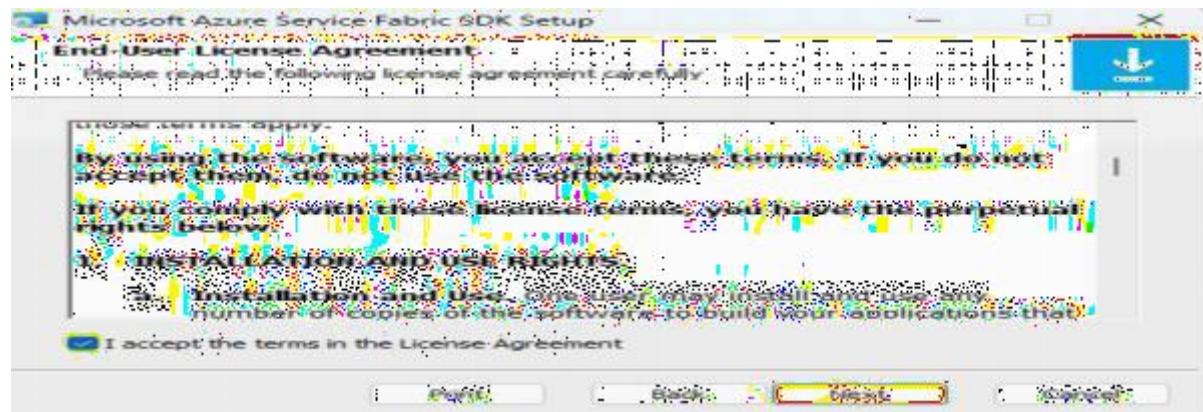
PS C:\Users\INDIA> cd d:
PS D:\> .\MicrosoftServiceFabric.10.0.1816.9590.exe /force /accepteula
PS D:\>
```

```
D:\MicrosoftServiceFabric\10.0.1816.9590.exe
25-11-2023 05:39:30,Info,Installing Service Fabric Runtime. Logs written to C:\WINDOWS\TEMP\InstallFabricRuntime_638364587784759600.log
25-11-2023 05:39:30,Info,24-11-2023 21:39:30 Running Process: powershell.exe -NoProfile -Command Get-ExecutionPolicy
25-11-2023 05:39:30,Info,CurrentUser with timeout 00:01:00
25-11-2023 05:39:30,Info,Running extract in parallel.
25-11-2023 05:39:31,Info,Unpackingaging autoextractor...
25-11-2023 05:39:31,Info,Creating temporary runtime directory C:\Program Files\Microsoft Service Fabric.dd1lhgam.rf2.
25-11-2023 05:39:31,Info,Extracting runtime cab to C:\Program Files\Microsoft Service Fabric.dd1lhgam.rf2...
25-11-2023 05:39:31,Info,24-11-2023 21:39:31 Running Process: C:\WINDOWS\TEMP\MicrosoftServiceFabricAutoExtractor.exe
25-11-2023 05:39:32,Info,Current Powershell Execution Policy: RemoteSigned
25-11-2023 05:39:32,Info,Service Fabric is already installed, try uninstall...
25-11-2023 05:39:32,Info,24-11-2023 21:39:32 Running Process: cmd.exe /c powershell.exe -File "C:\Program Files\Microsoft Service Fabric\bin\Fabric\Fabric.Code\CleanFabric.ps1" with timeout 00:05:00
25-11-2023 05:39:33,Info,Successfully extracted cab file to C:\Program Files\Microsoft Service Fabric.dd1lhgam.rf2
25-11-2023 05:39:51,Info,Successfully uninstalled...
25-11-2023 05:39:51,Info,Moving FabricRoot files from 'C:\Program Files\Microsoft Service Fabric.dd1lhgam.rf2' to 'C:\Program Files\Microsoft Service Fabric'.
25-11-2023 05:39:51,Info,Moved FabricRoot files 'C:\Program Files\Microsoft Service Fabric' successfully.
25-11-2023 05:39:51,Info,Installing Service Fabric Runtime... Logs written to: C:\WINDOWS\TEMP\FabricSetupLog_638364587784759600.log
25-11-2023 05:39:51,Info,Executing powershell.exe with parameters: -NoProfile -Command & 'C:\Program Files\Microsoft Service Fabric\bin\Fabric\Fabric.Code\InstallFabric.ps1' -FabricRootIsPreInstalled -AcceptEULA -LogPath C:\WINDOWS\TEMP\FabricSetupLog_638364587784759600.log
25-11-2023 05:39:51,Info,24-11-2023 21:39:51 Running Process: powershell.exe -NoProfile -Command & 'C:\Program Files\Microsoft Service Fabric\bin\Fabric\Fabric.Code\InstallFabric.ps1' -FabricRootIsPreInstalled -AcceptEULA -LogPath C:\WINDOWS\TEMP\FabricSetupLog_638364587784759600.log with timeout 00:10:00
```

Now install Service Fabric SDK.

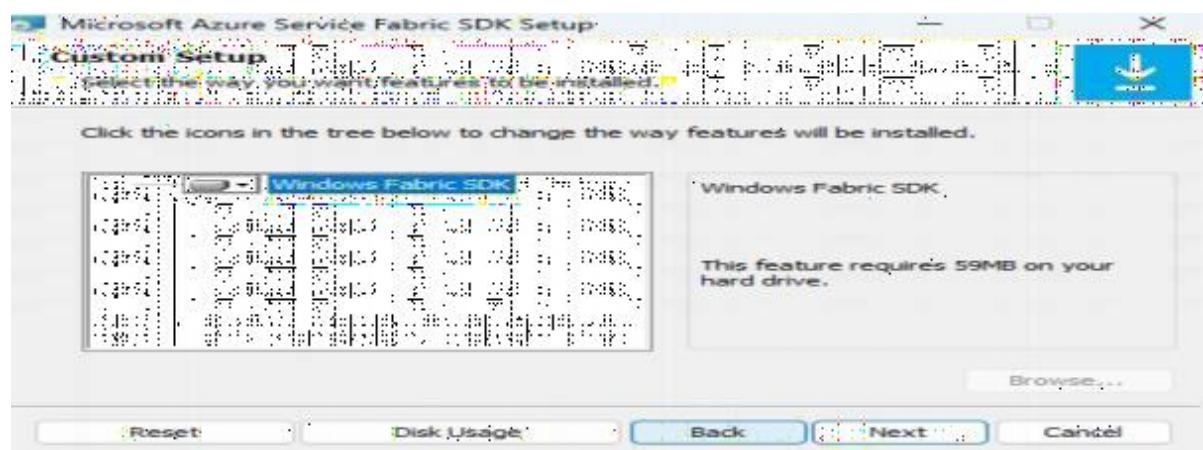
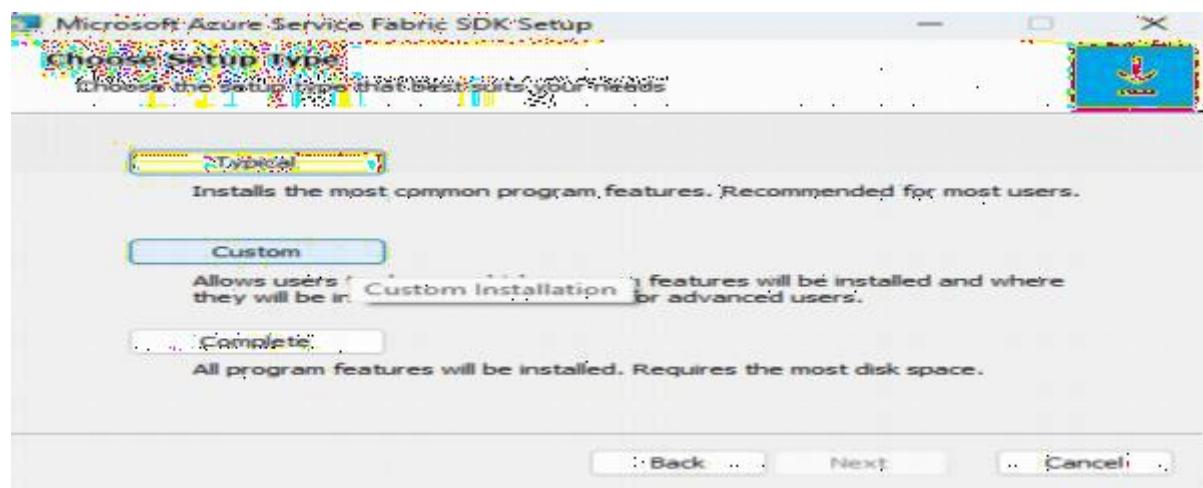


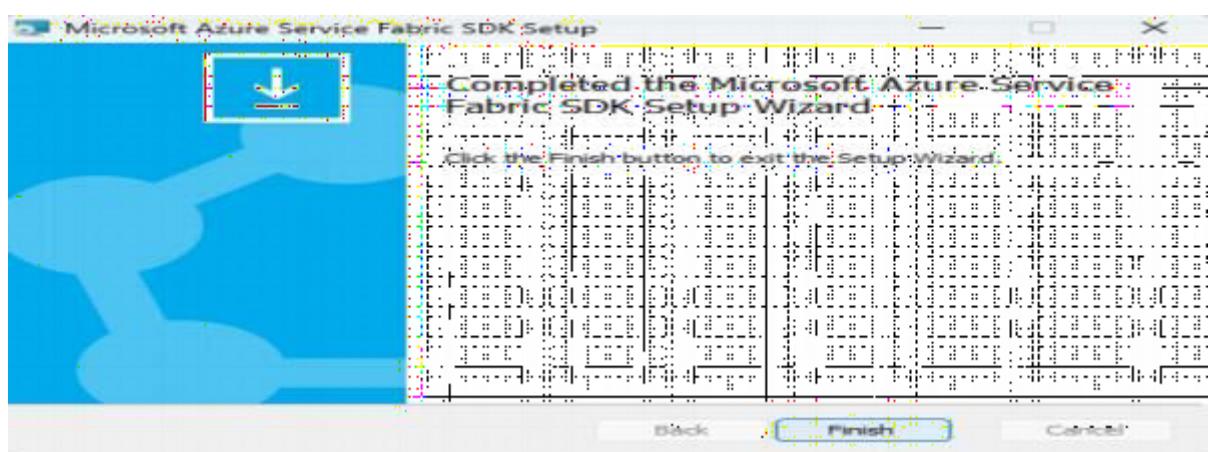
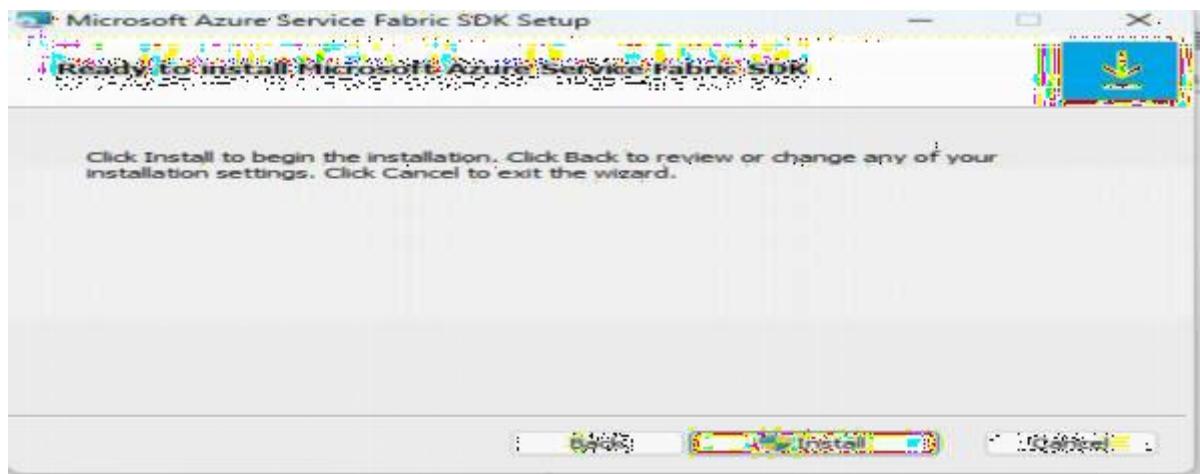
Accept the Liscence agreement.



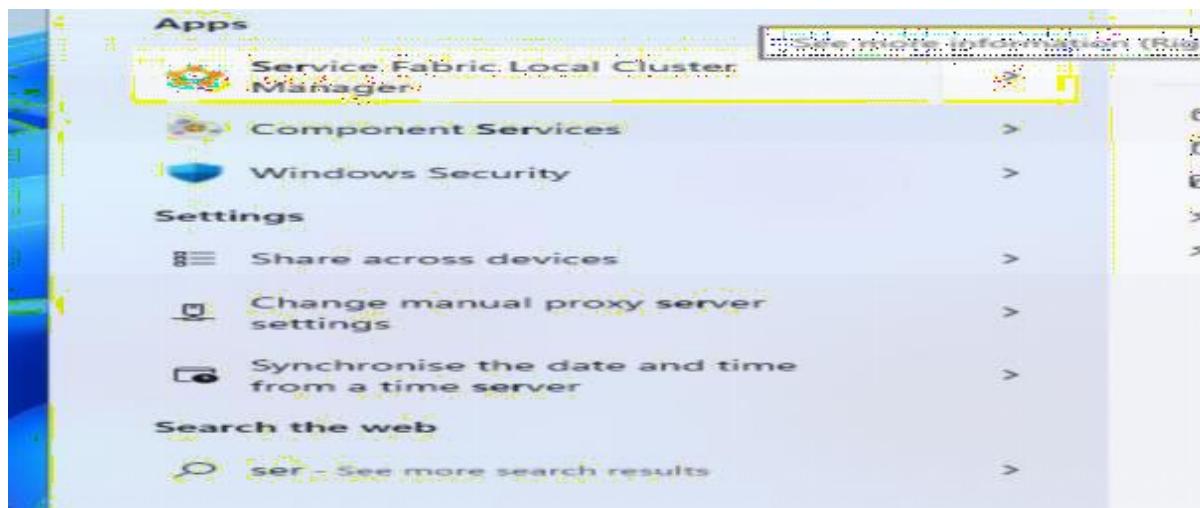
Choose Custom Setup type and click on next.

And install.

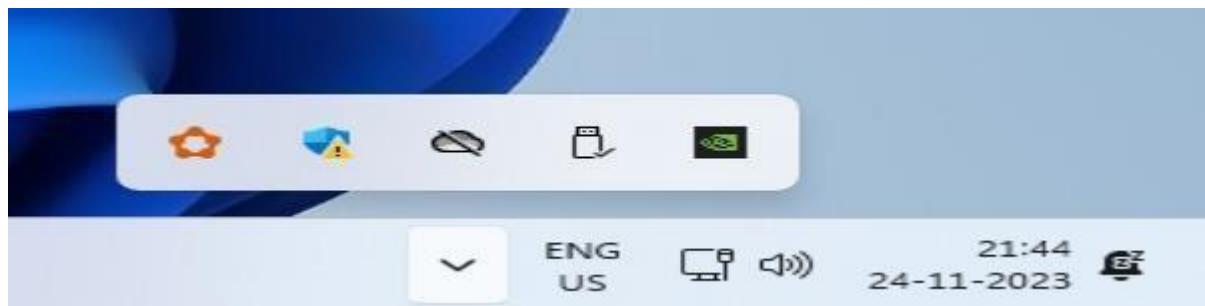




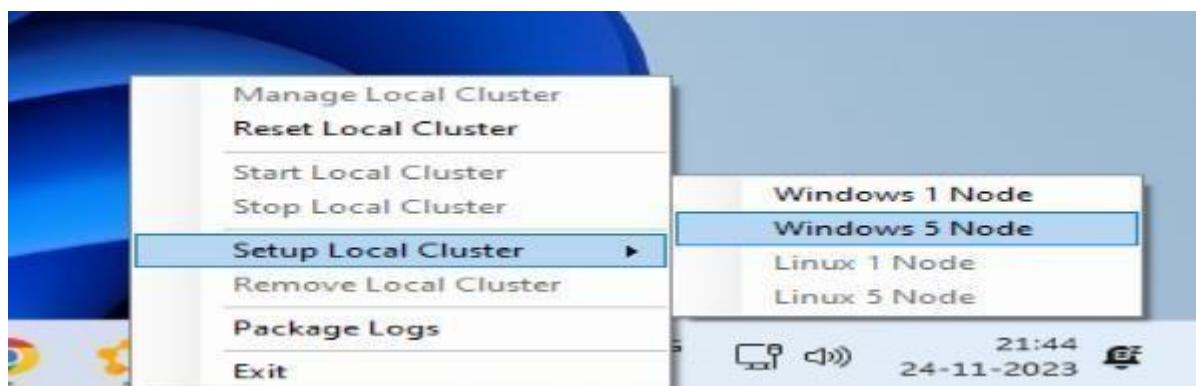
Open search Bar on desktop and search for Service fabric Local Cluster Manager and click on it.



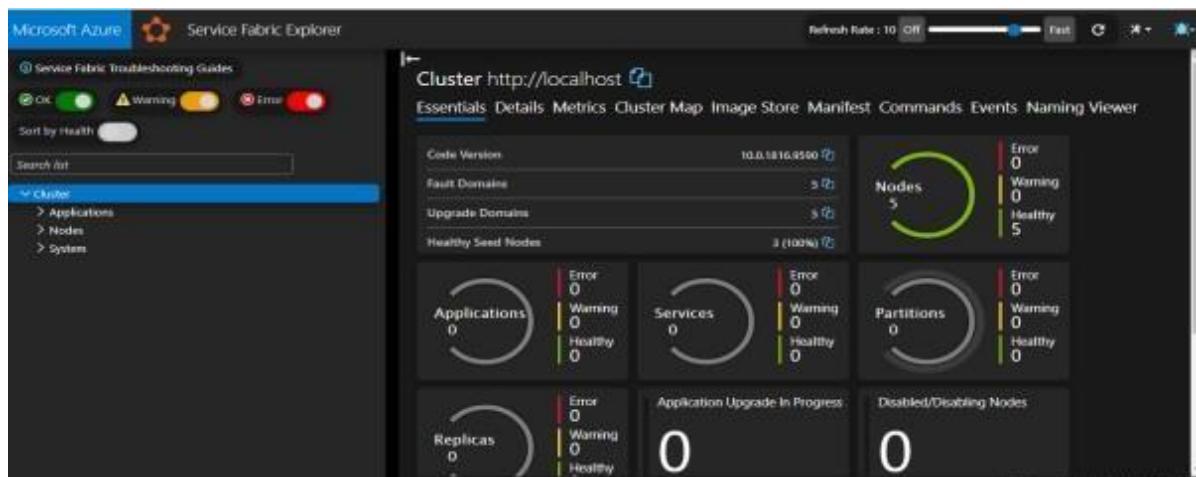
Now Service Fabric Cluster icon is available on the right-hand side icon bar.



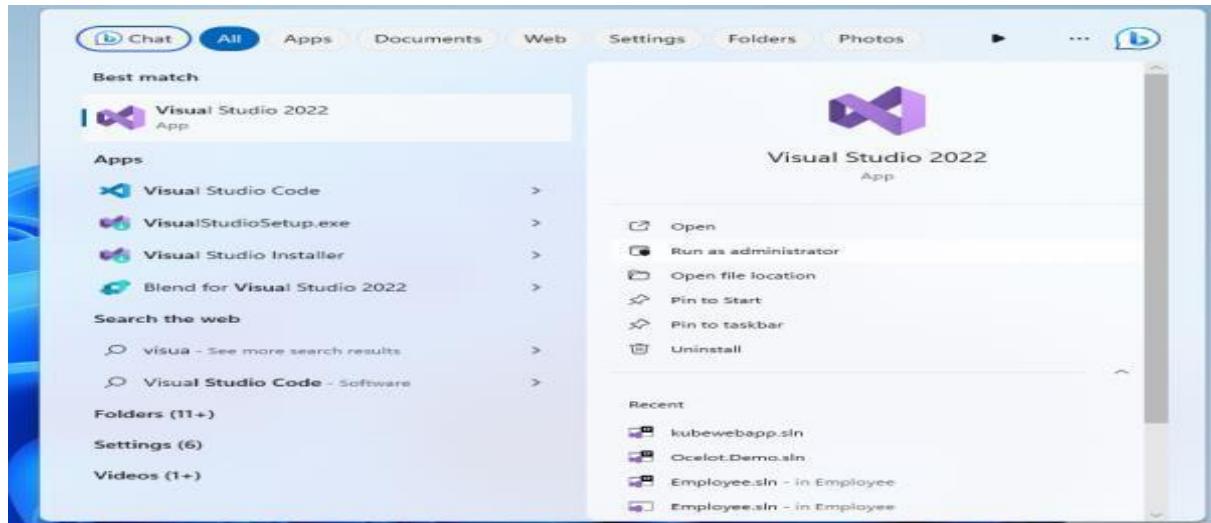
Right click on it and click on Setup local Cluster to windows 5 node. And start local cluster.



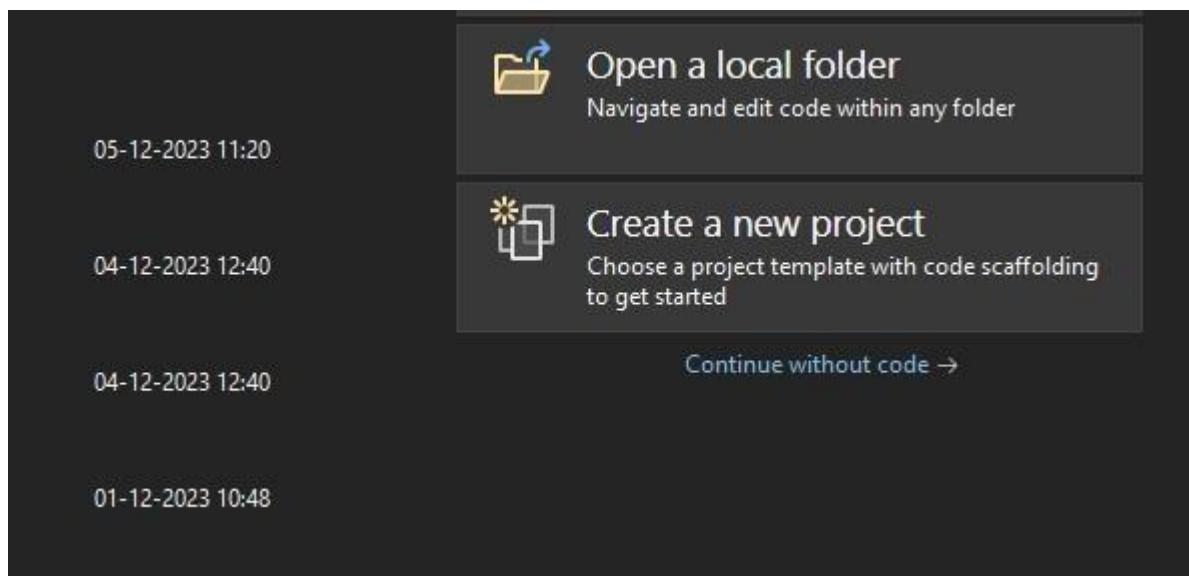
After starting local cluster wait for some time and after that click on Manage Local Cluster. And then following window will open which is Service Fabric Explorer.



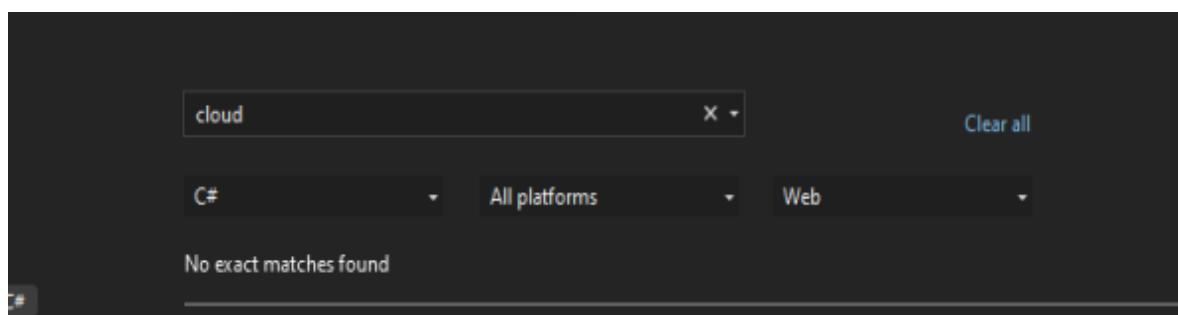
Now, open visual studio 2022 and run as administrator.



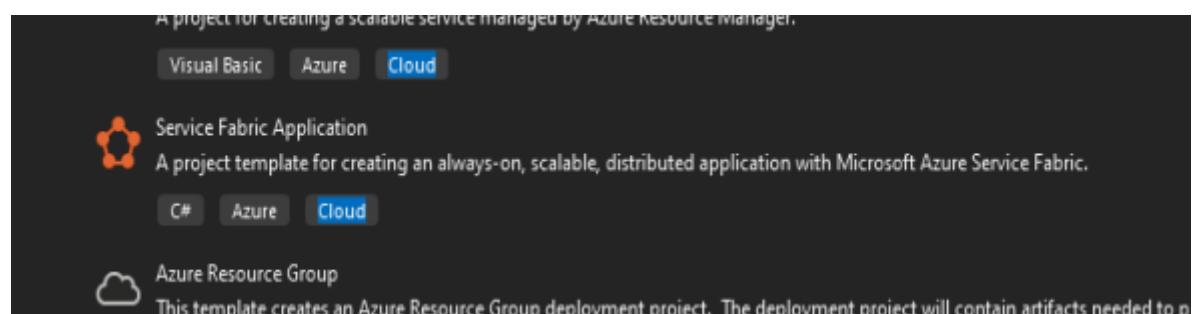
Now click on create a new project.



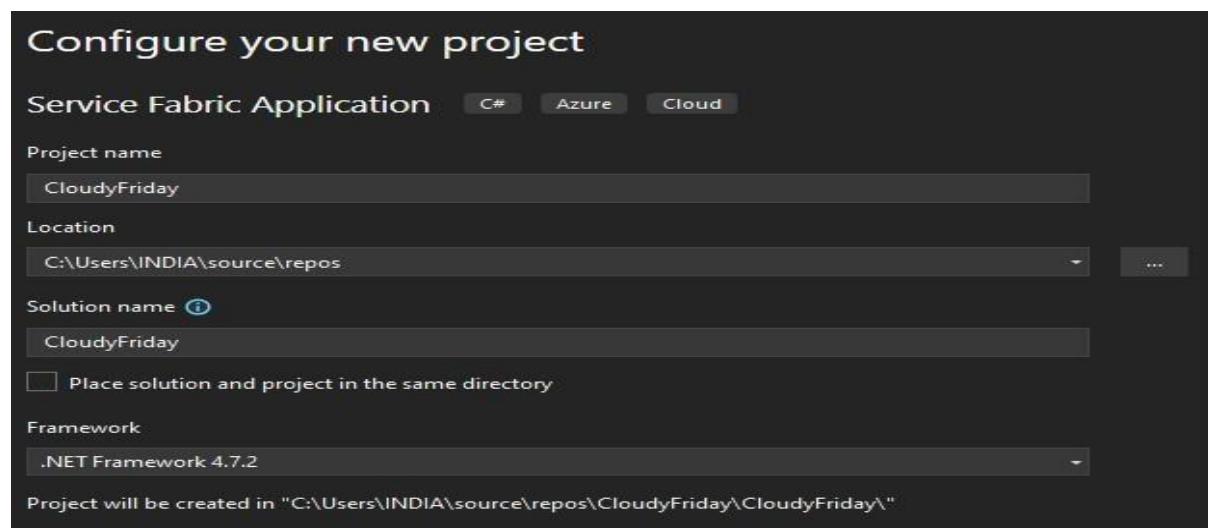
Search for cloud, choose project type c#.



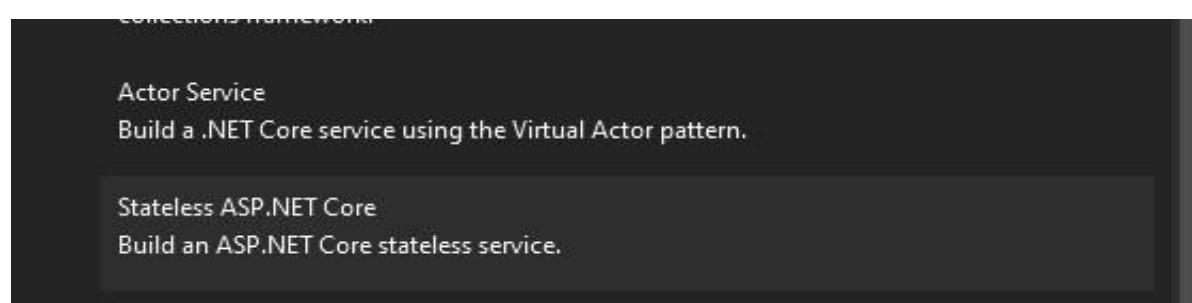
Select Service Fabric Application.



Give project Name and click on next.

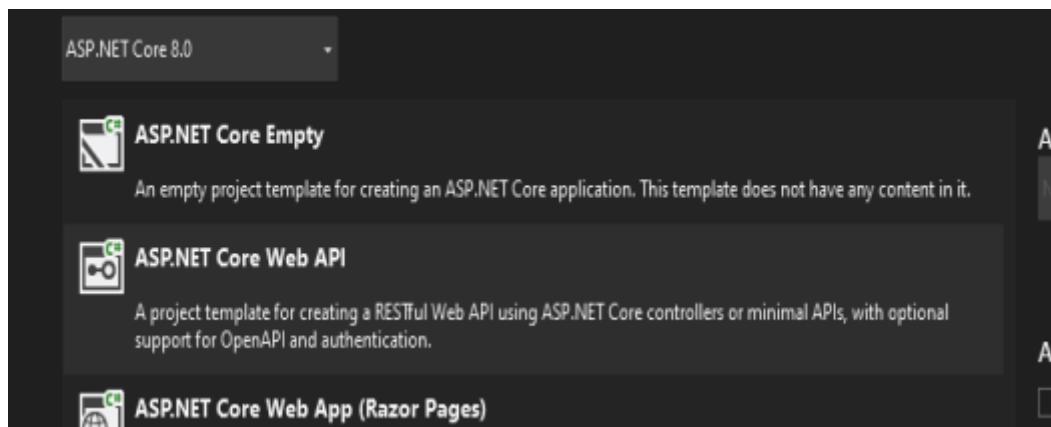


Choose stateless ASP.NET Core and give service name GeographyService.

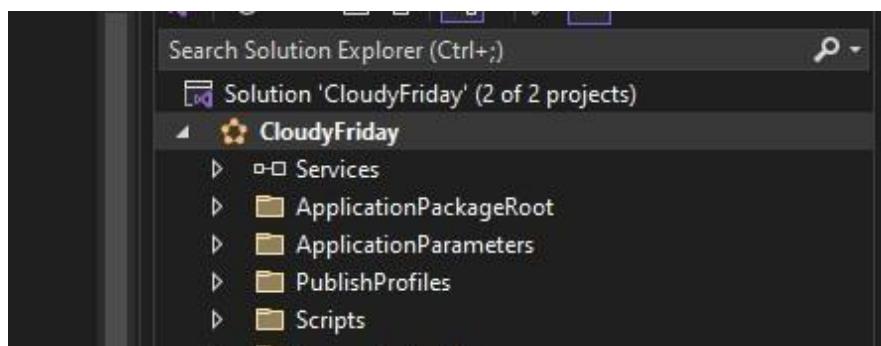




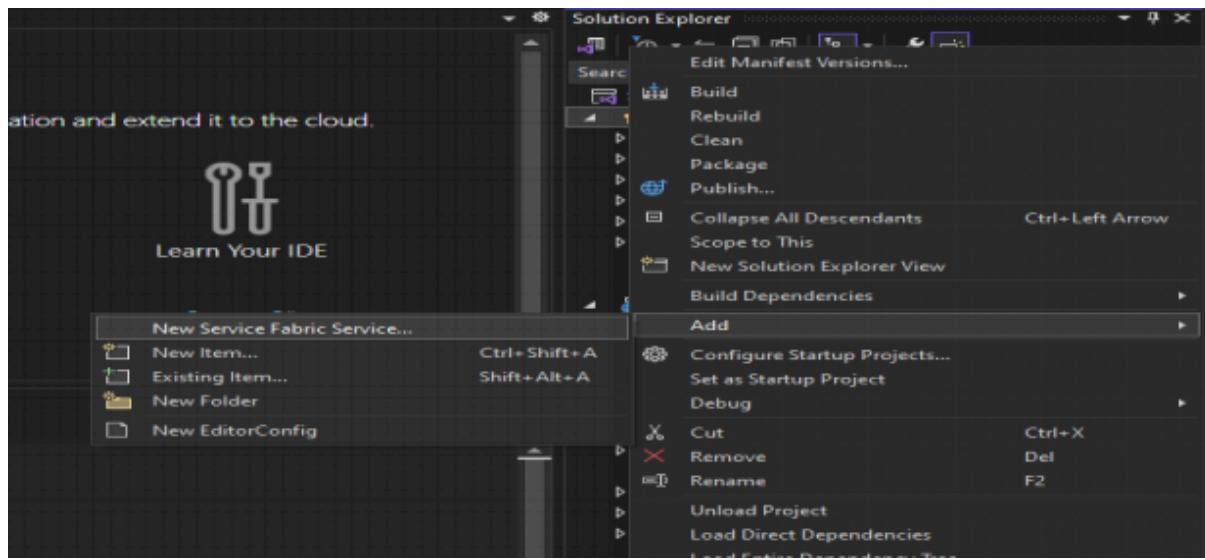
Choose ASP.NET Core Web API and create project.



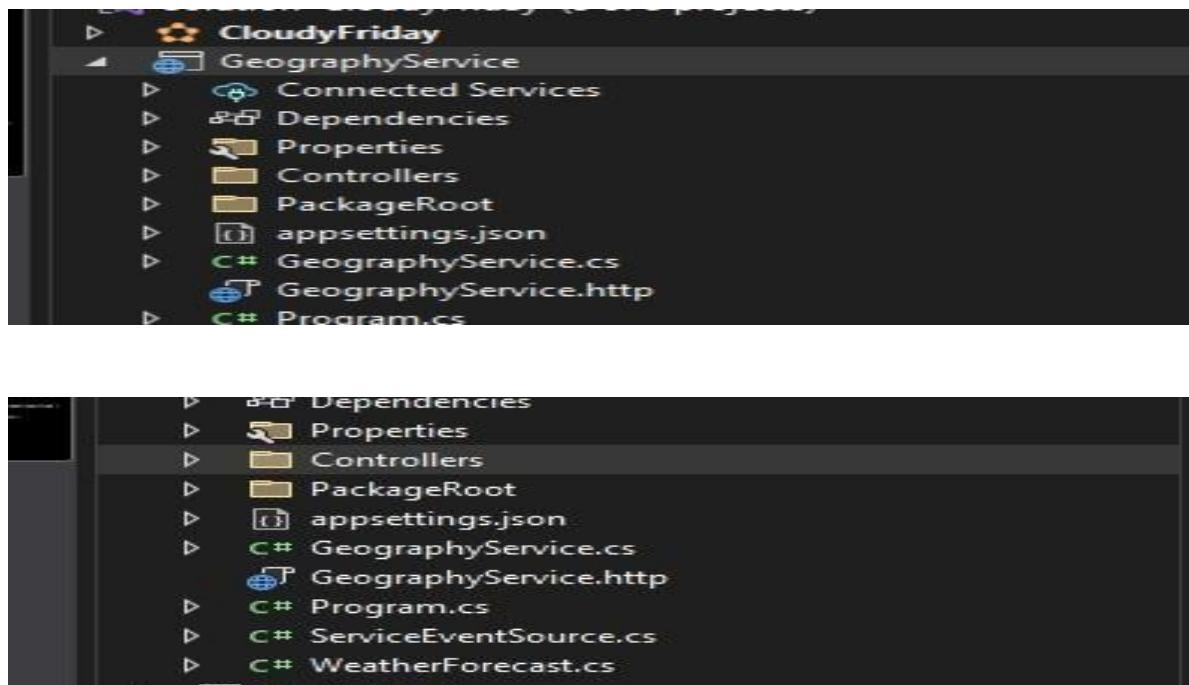
Right click on CloudyFriday.



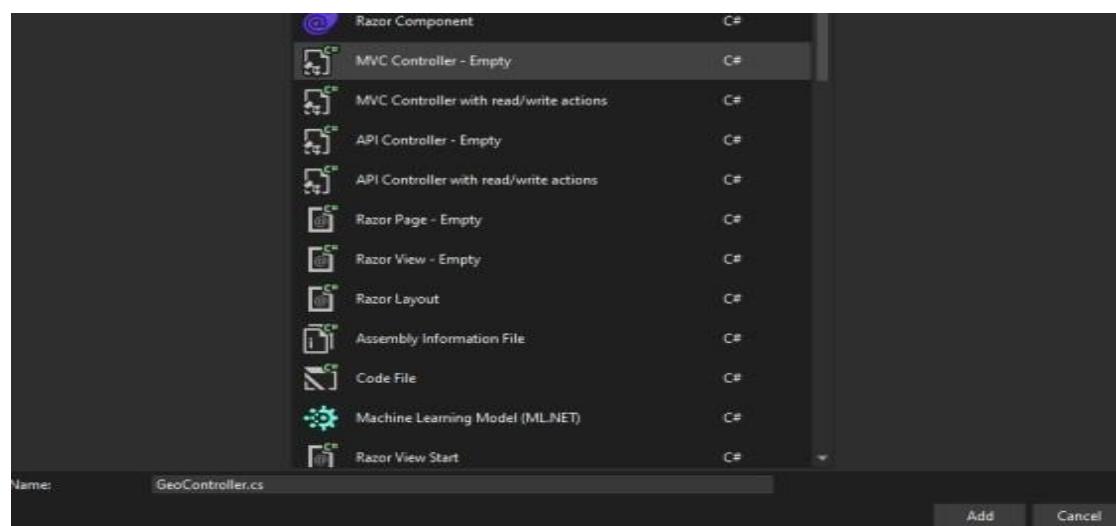
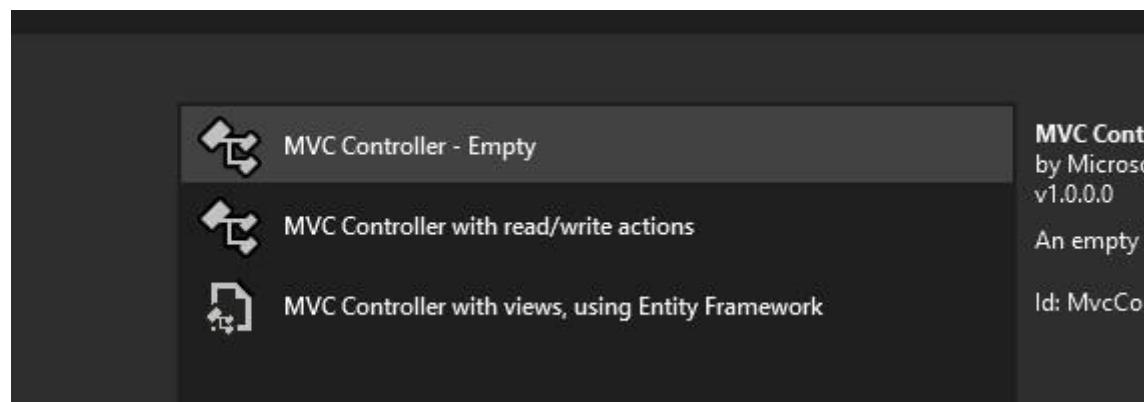
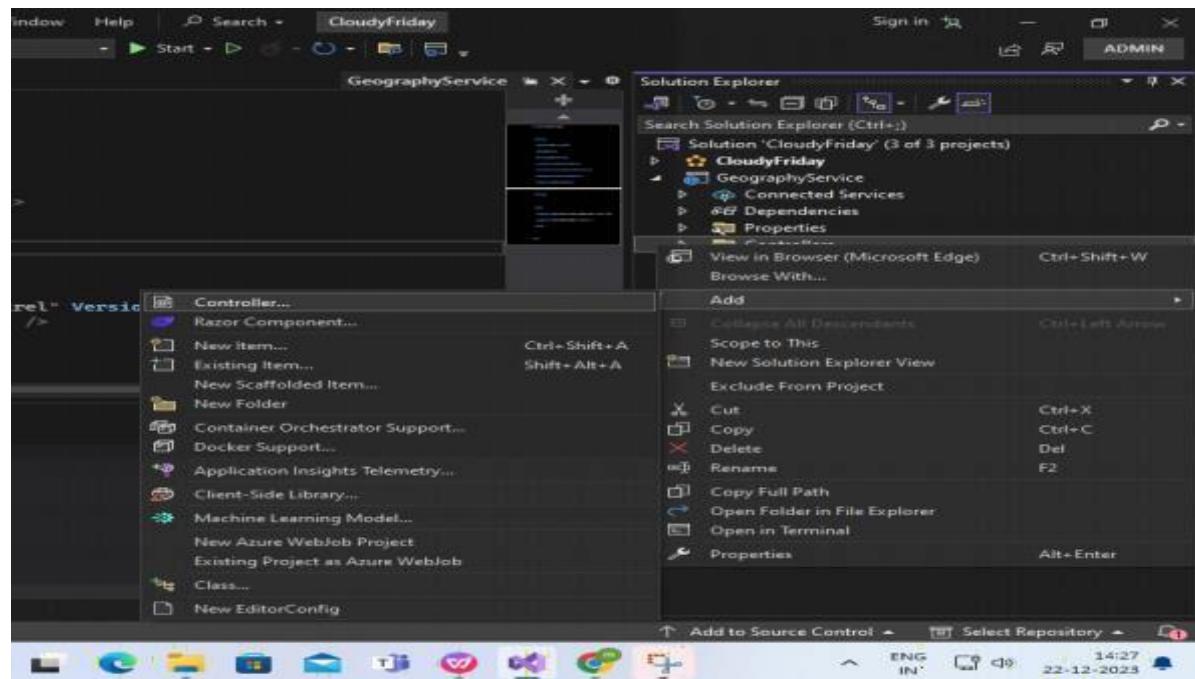
Add a new Service Fabric Service.



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;

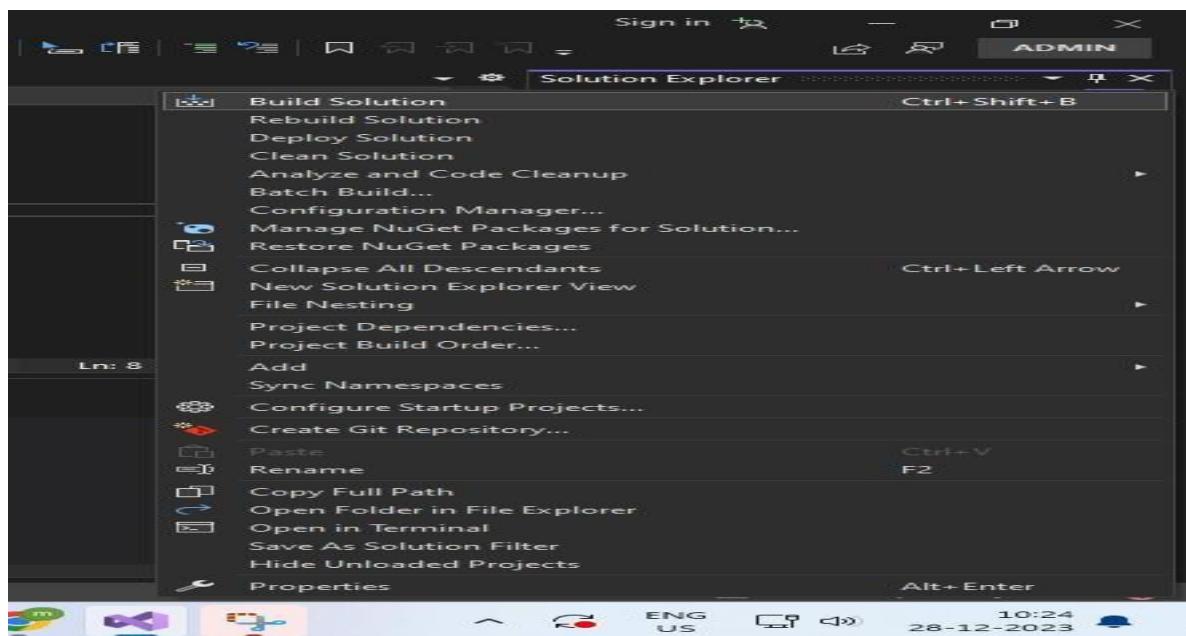
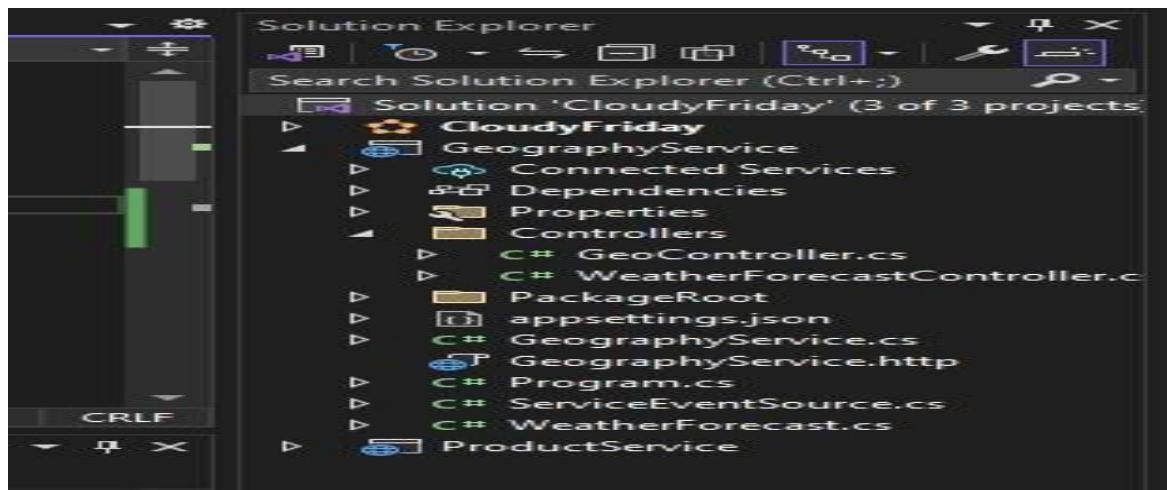
namespace GeographyService.Controllers
{
    class Geography
    {
        public String Name { get; set; }

        public int ID { get; set; }
    }

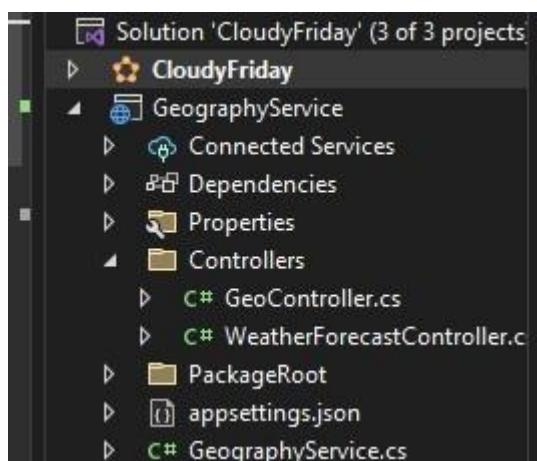
    [Route("api/[controller]")]
    public class GeoController : Controller
    {

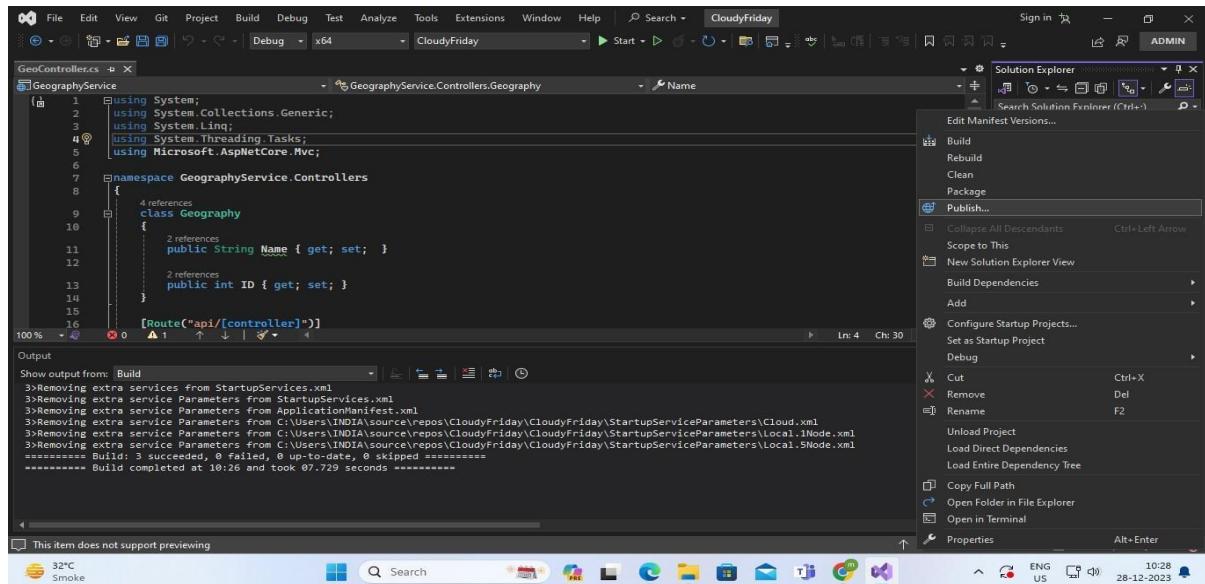
        List<Geography> geographies = new List<Geography>();
        public IActionResult Index()
        {
            geographies.Add(new Geography() { ID = 1, Name = "GEO1" });
            geographies.Add(new Geography() { ID = 2, Name = "GEO2" });
            return new JsonResult(geographies);
        }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

Right click on solution and click on build solution.

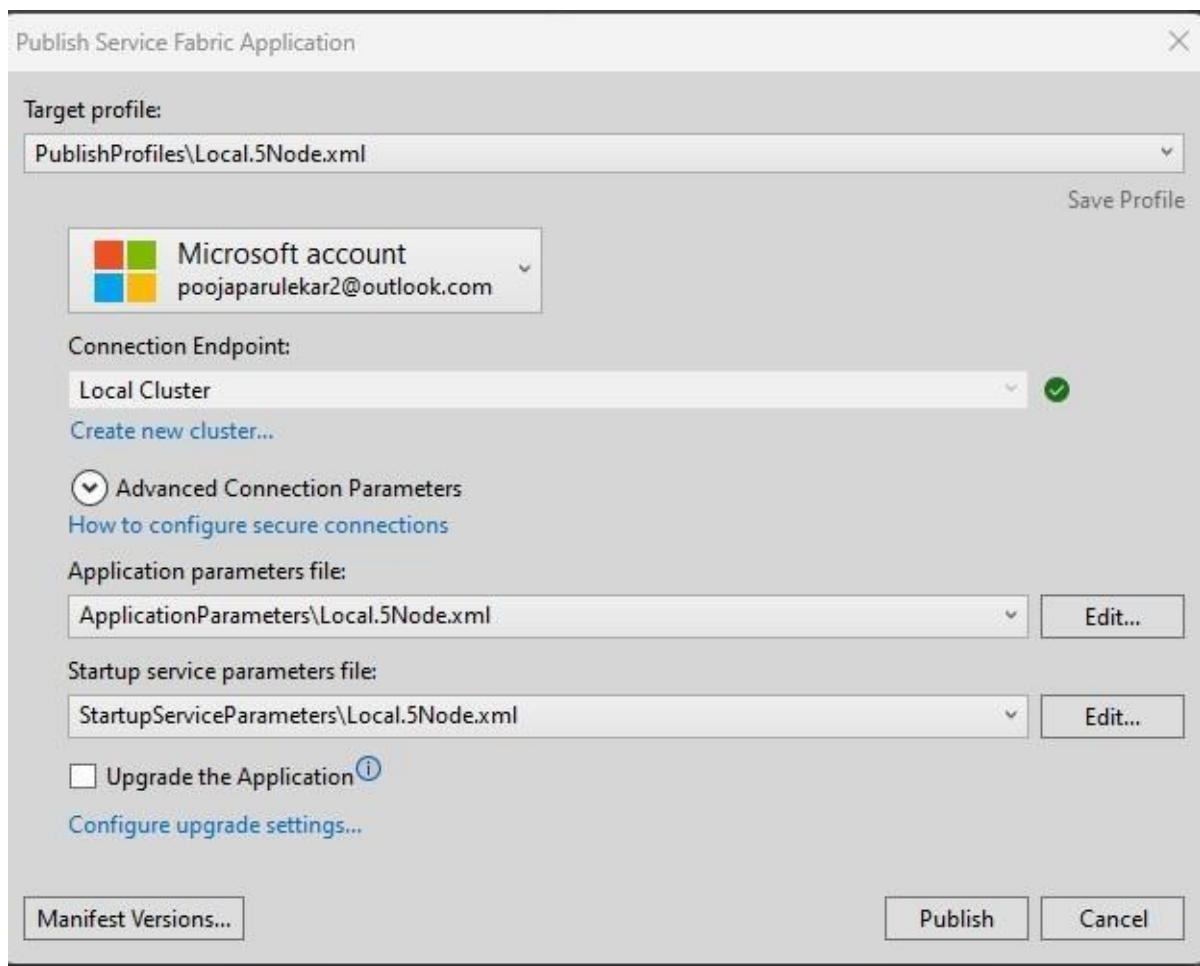


Right click on Cloudy Friday and click on publish.

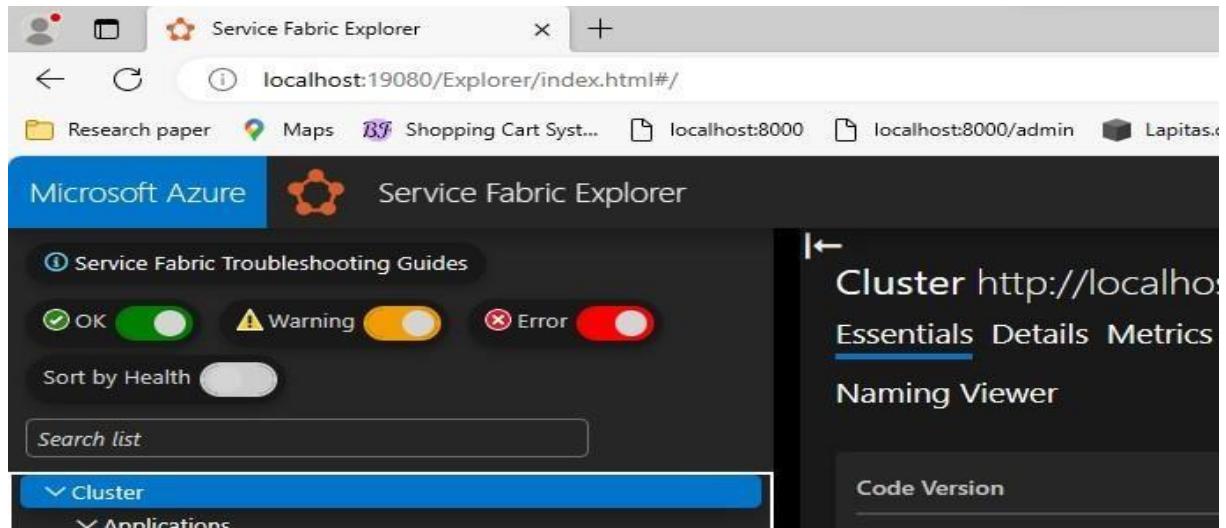




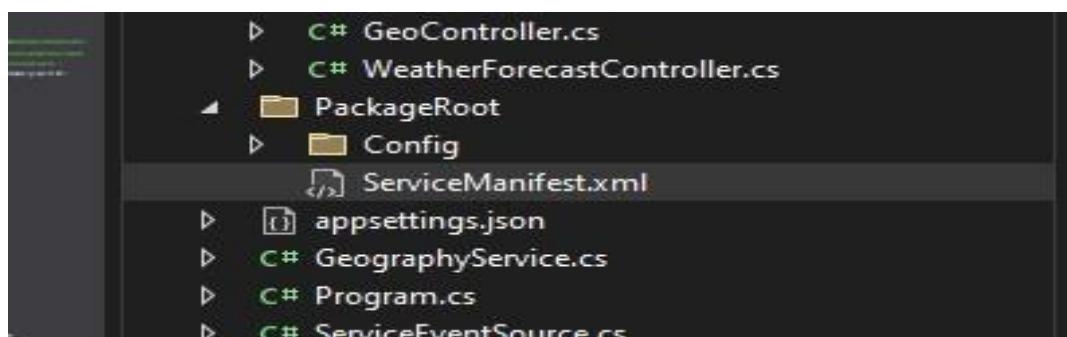
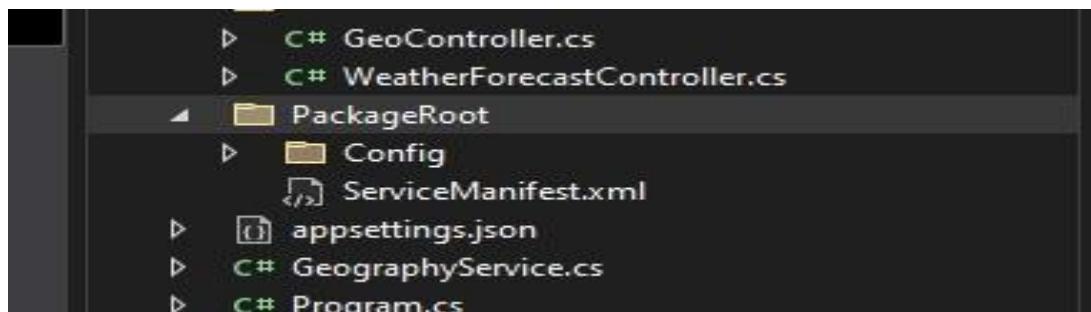
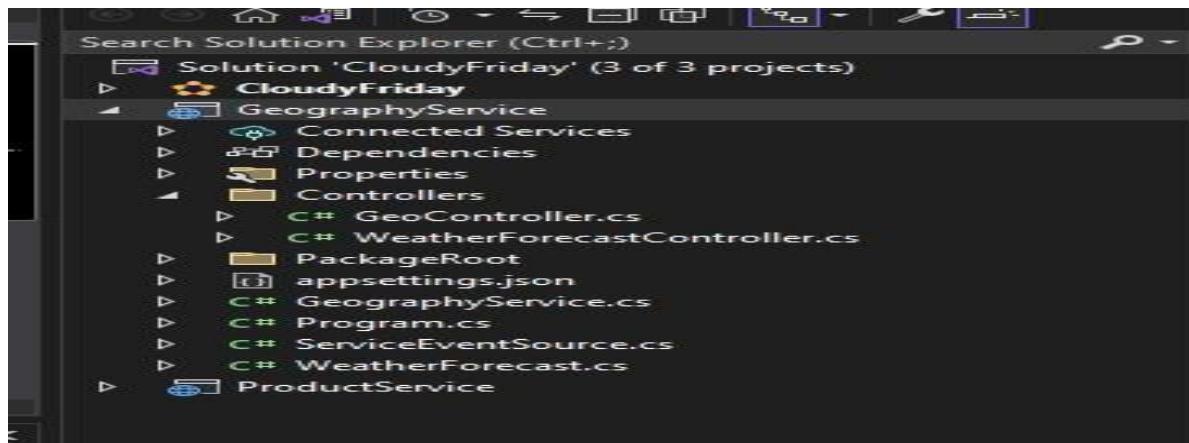
Choose Local.5Node.xml because we are using 5 node cluster. Choose Microsoft account and if not then sign in to it. And publish.



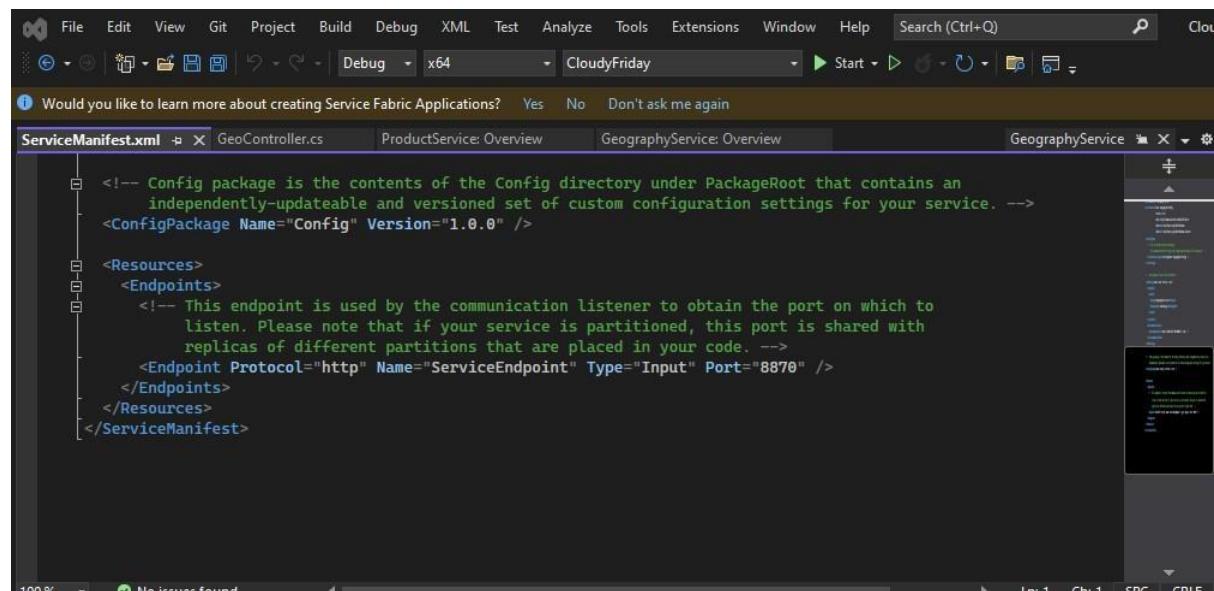
Now our project is deployed on Service Fabric .



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

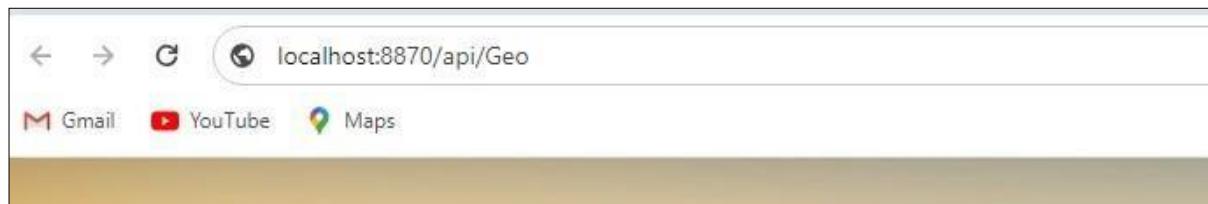


At the End of File check the port number . and go to browser and type localhost:8870/ api/Geo and enter. 8870 is a port number and Geo is a controller name.



```
<!-- Config package is the contents of the Config directory under PackageRoot that contains an independently-updateable and versioned set of custom configuration settings for your service. -->
<ConfigPackage Name="Config" Version="1.0.0" />

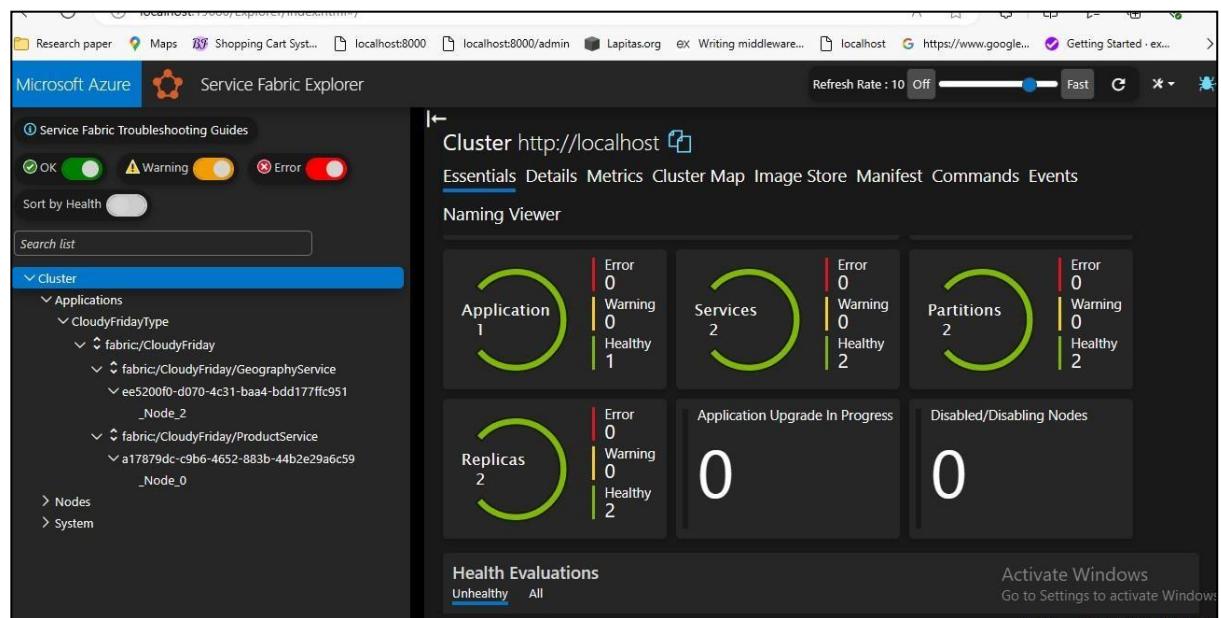
<Resources>
  <Endpoints>
    <!-- This endpoint is used by the communication listener to obtain the port on which to listen. Please note that if your service is partitioned, this port is shared with replicas of different partitions that are placed in your code. -->
    <Endpoint Protocol="http" Name="ServiceEndpoint" Type="Input" Port="8870" />
  </Endpoints>
</Resources>
</ServiceManifest>
```



And here's the Output.



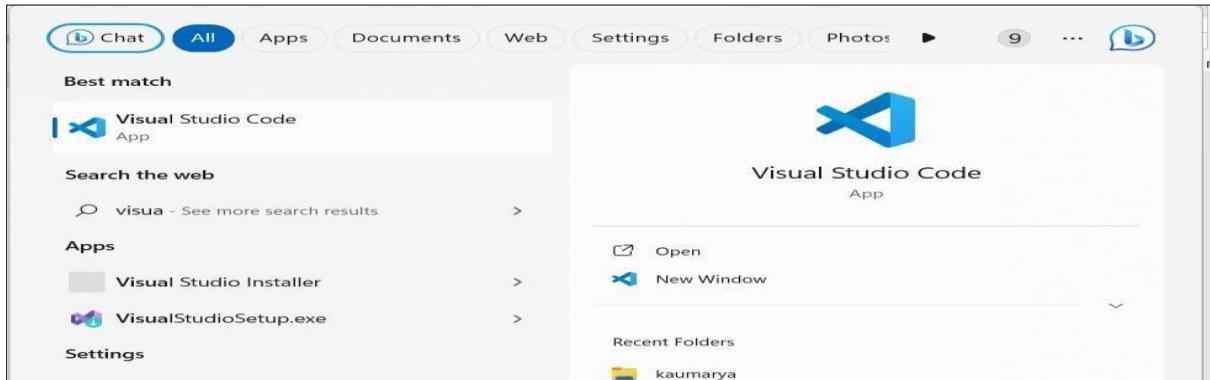
Service Fabric shows all nodes



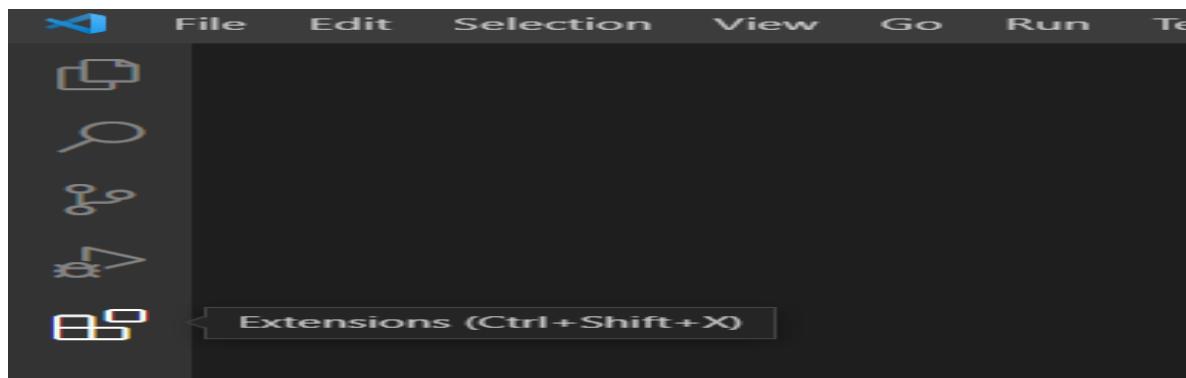
Practical No: 2

Develop a Spring Boot API.

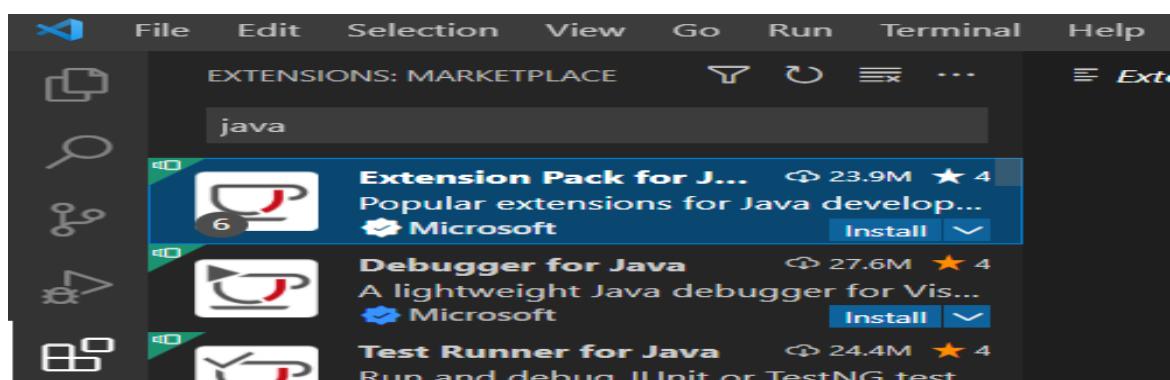
First open a visual studio code.



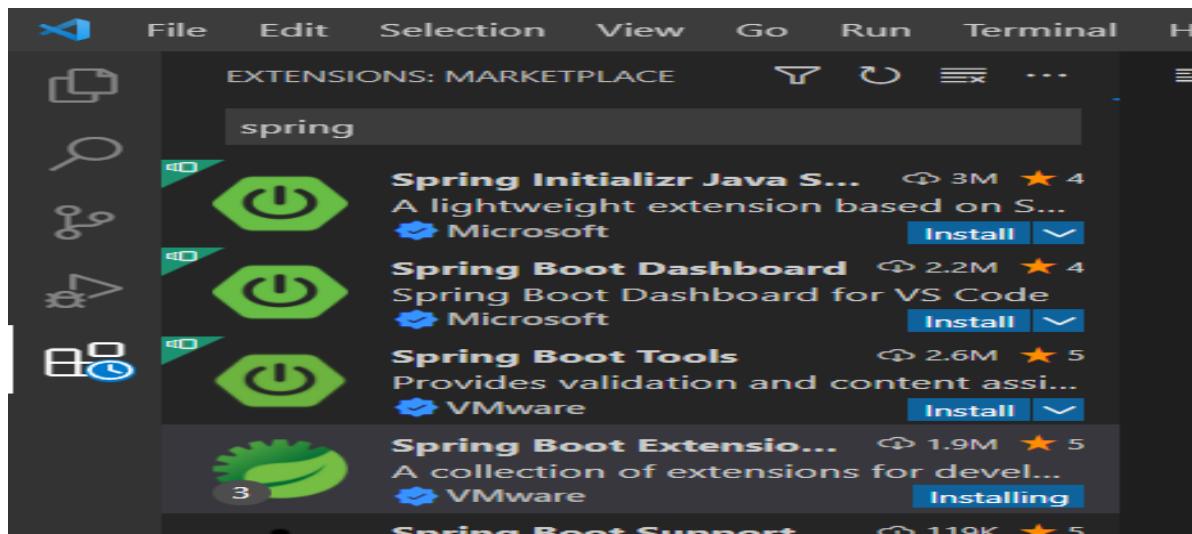
In visual studio code go to extensions.



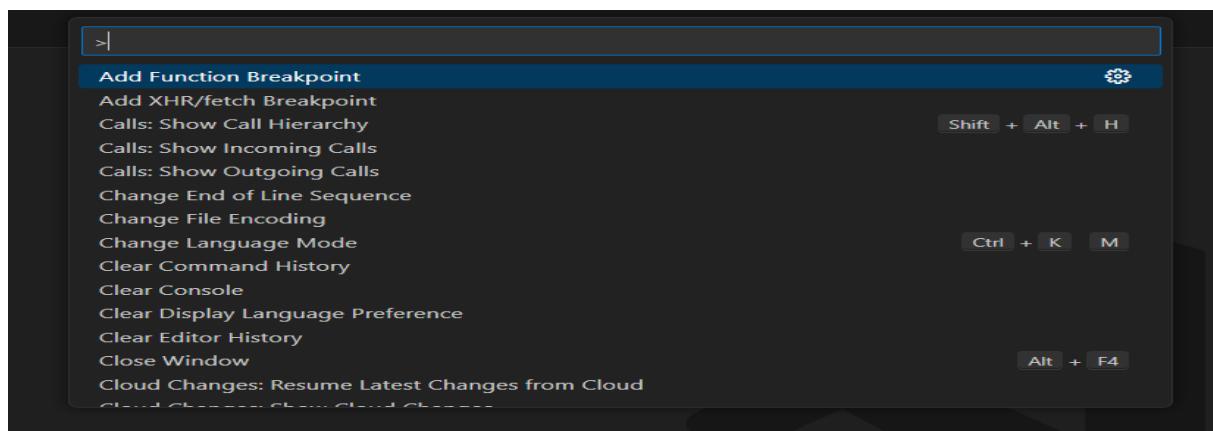
And search for java extension pack and install it.



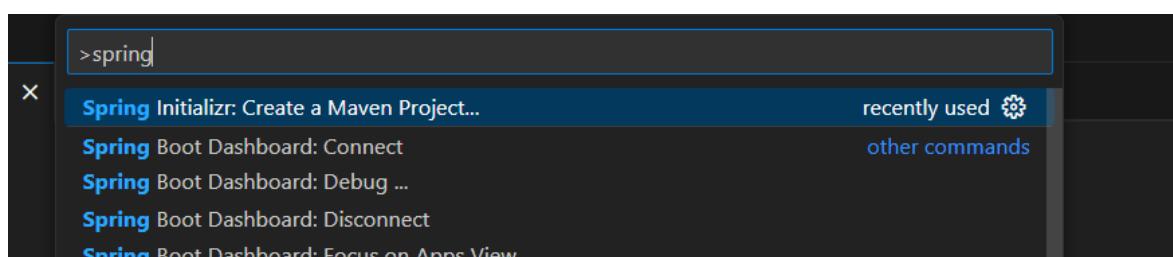
After installing java extension pack now install spring boot extension pack .



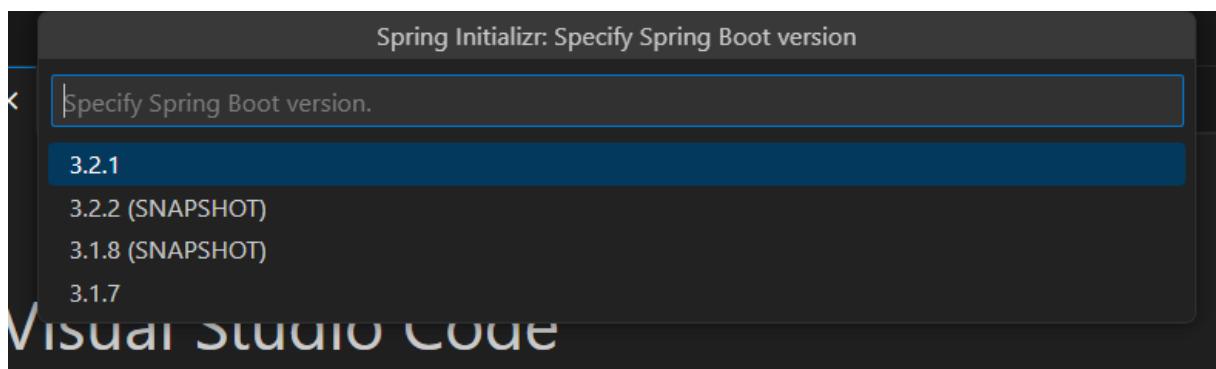
After installing extensions, we are going to implement our spring boot application . for that first of all open command palette using **ctrl + shift + p** .



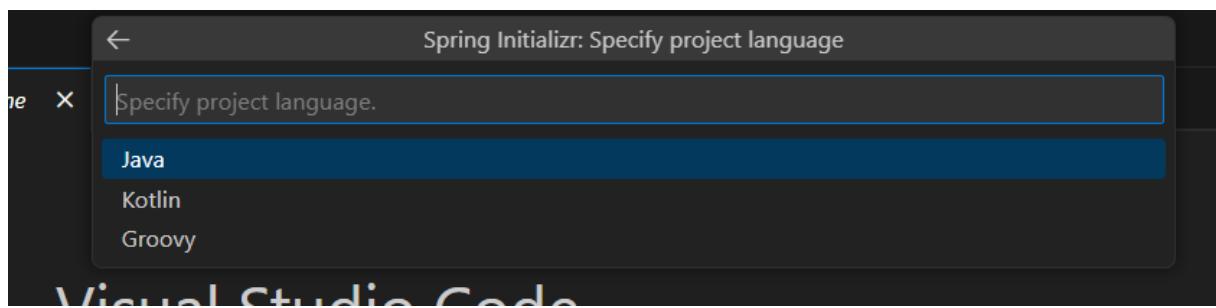
Search for spring and select Spring initializr : create a maven project.



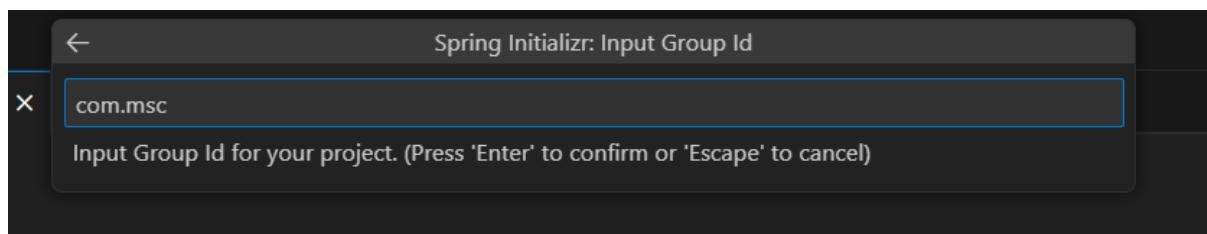
Now , select a spring boot version 3.2.1 .



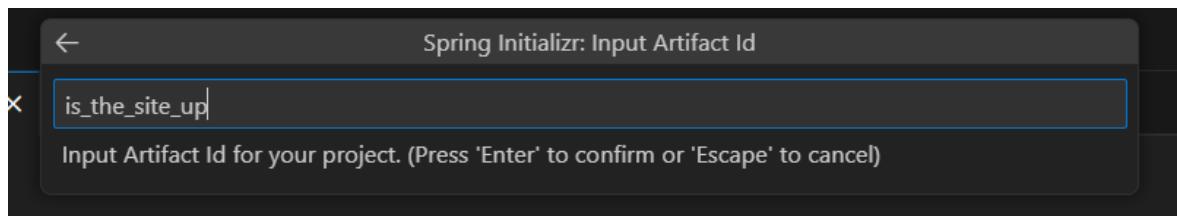
Select project language java.



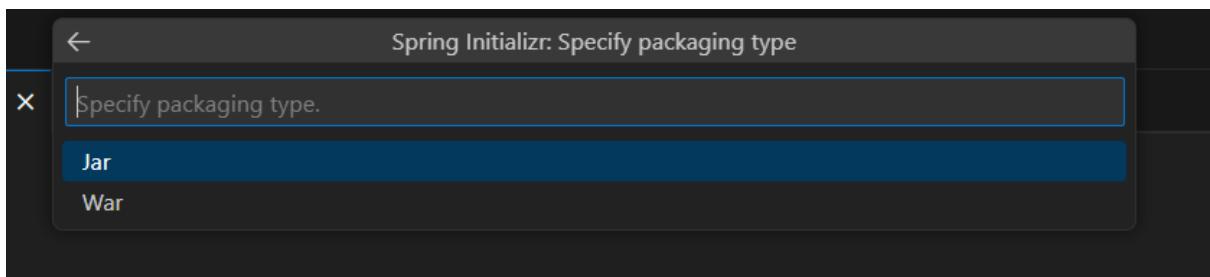
Give Group id com.msc and press enter.



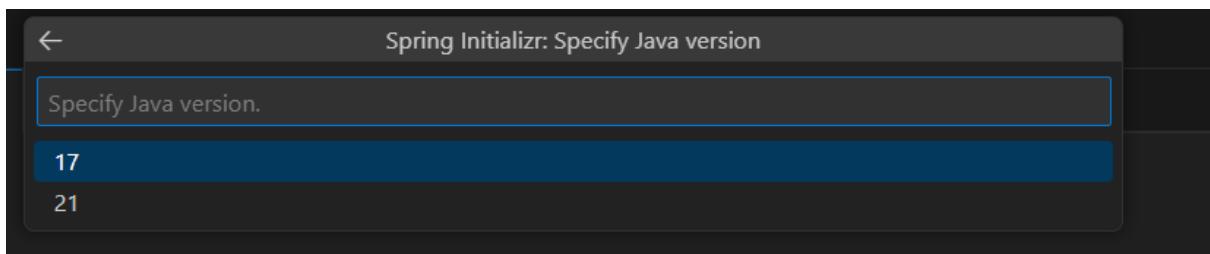
Give artifact id is_the_site_up and press enter.



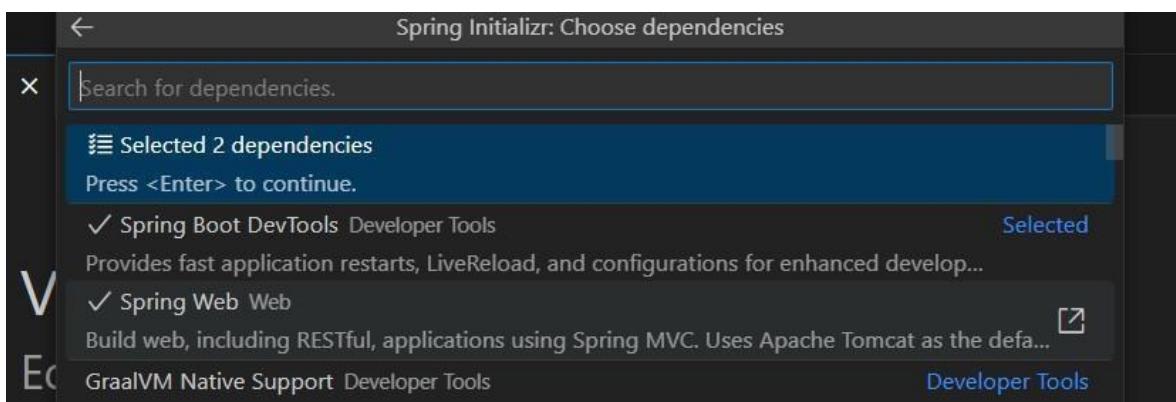
Select packaging type jar.



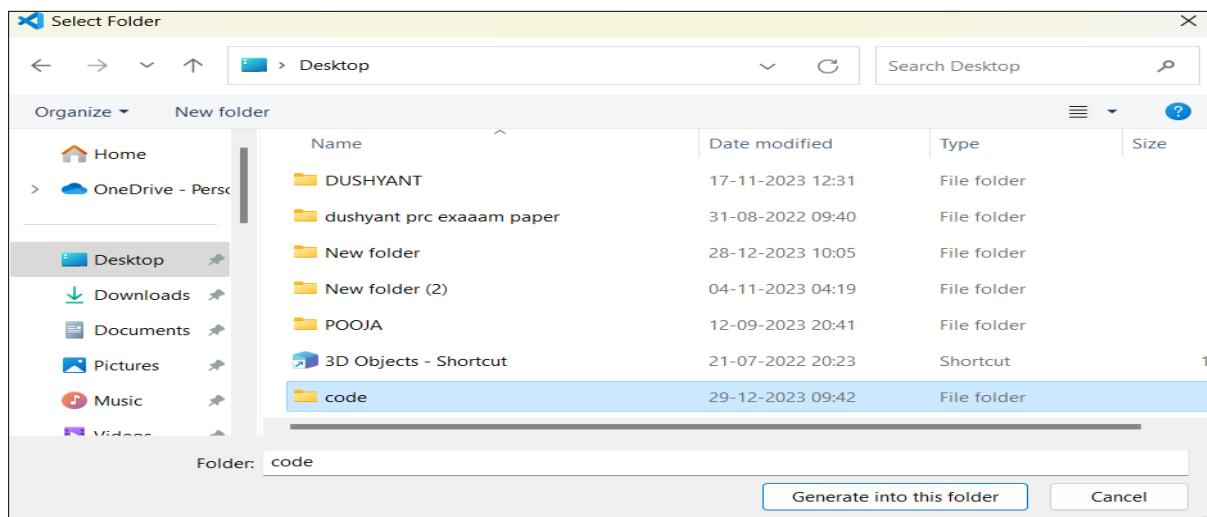
Select java version 17.



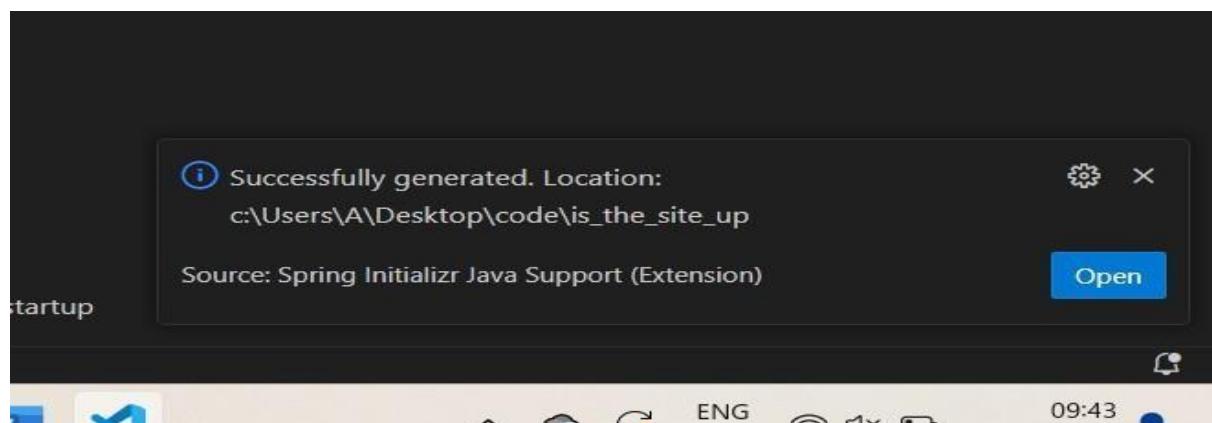
Select the dependencies spring boot devtools and spring web.



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

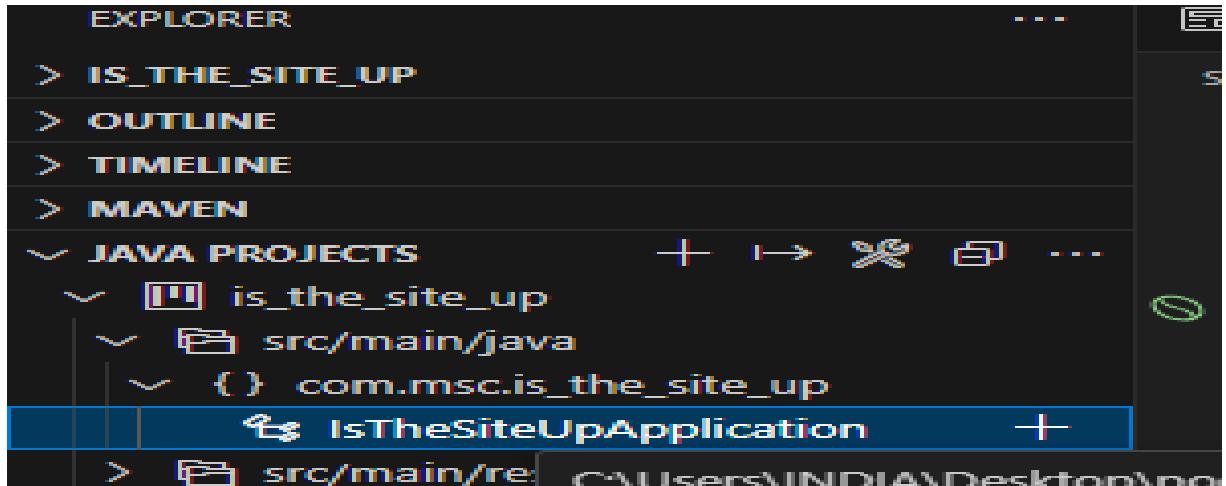


On the right-hand side notification will pop up for open a project .



In application got to java projects - src/main/java

- com.msc.is_the_site_up and open the application.



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

```

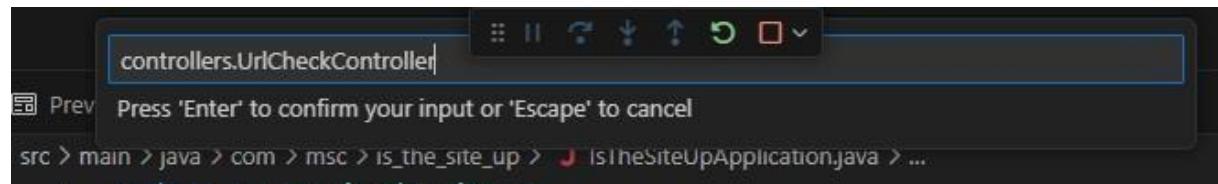
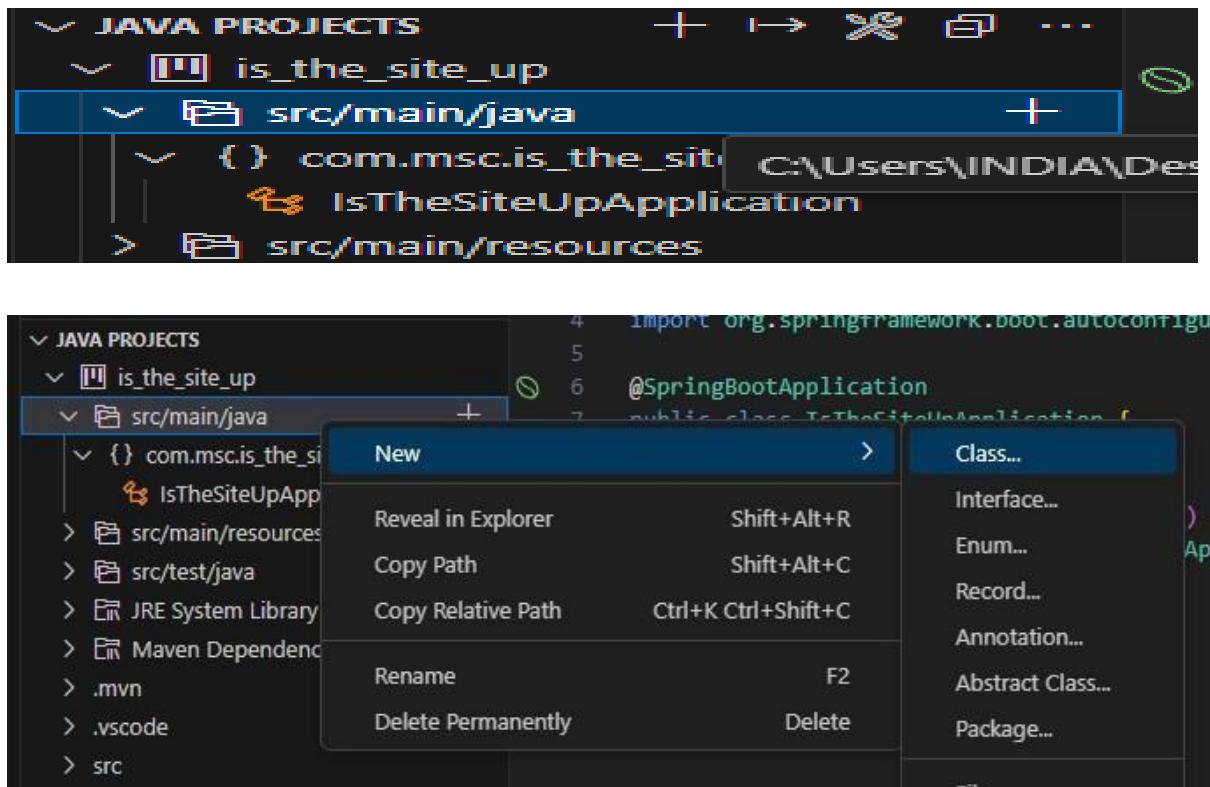
src > main > java > com > msc > is_the_site_up > IsTheSiteUpApplication.java > Language Support for Java(TM) by Red Hat > {} com.msc.is_the_site_up
1 package com.msc.is_the_site_up;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class IsTheSiteUpApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(IsTheSiteUpApplication.class, args);
11     }
12 }
13
14 }
```

```

7 public class IsTheSiteUpApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(IsTheSiteUpApplication.class, args);
11     }
12 }
13
14 }
```

```
2023-12-30T09:44:51.324+05:30 INFO 10148 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization 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 port 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.



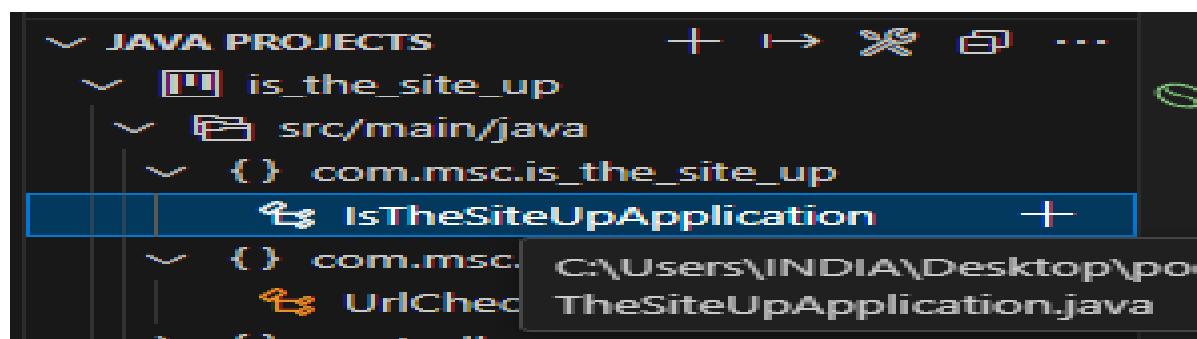
Write a following code in UrlCheckController.

Code :

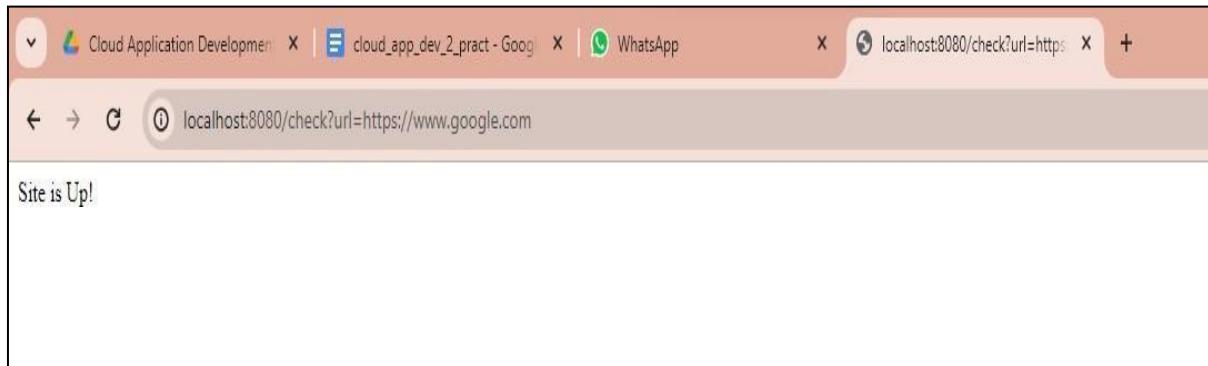
```
package com.msc.is_the_site_up.controllers;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
@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;
             * }
             */
        } catch (IOException e) {
            returnMsg = URL_INCORRECT;
        }
        return returnMsg;
    }
}
```

```
*/  
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:



Practical No: 3

3A. Create an Azure Kubernetes Service Cluster

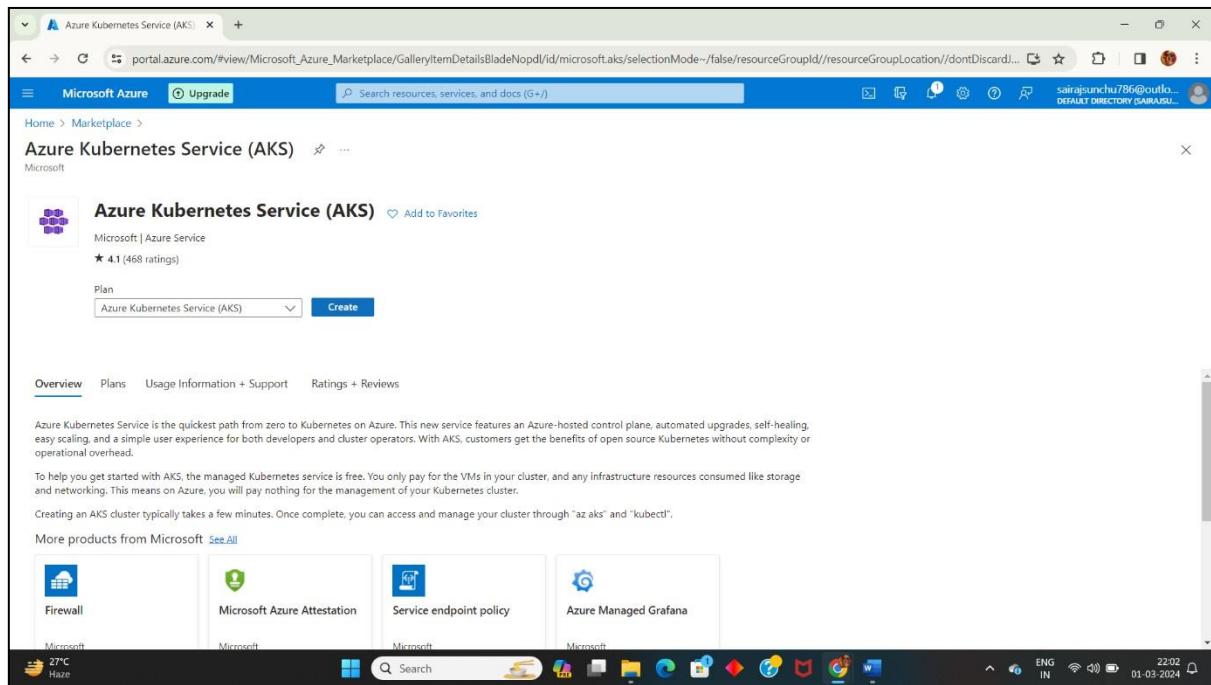
Step 1: Sign in Azure Portal (<https://portal.azure.com/>)

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

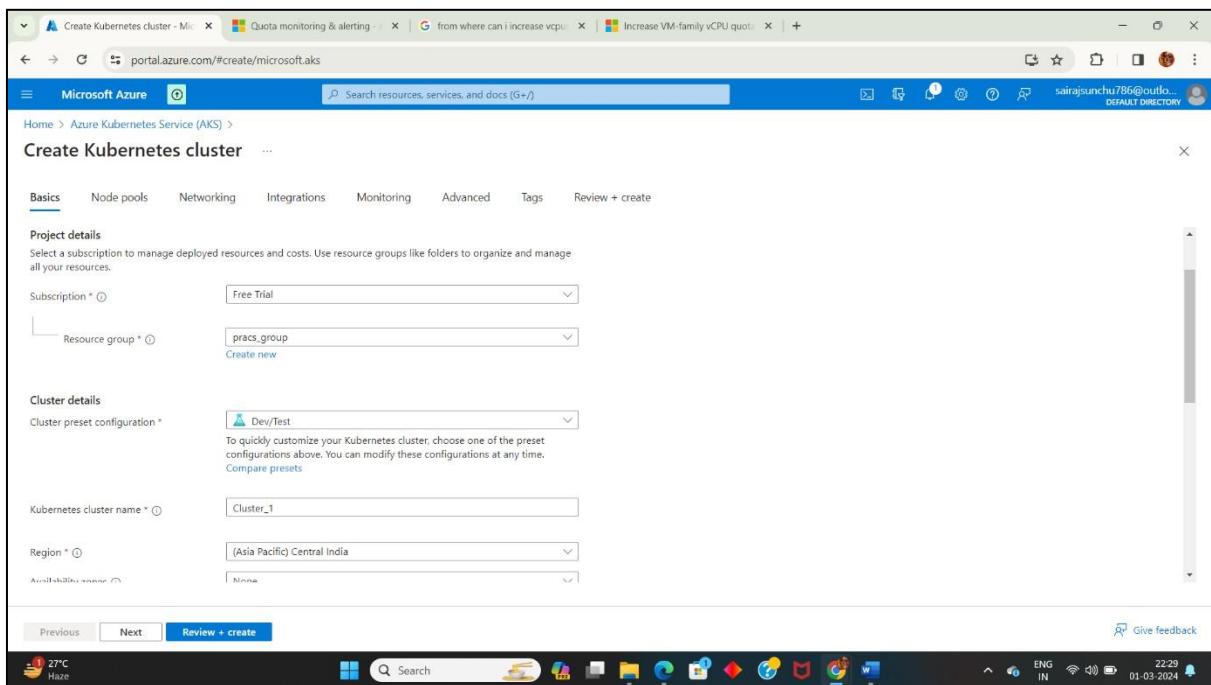
The screenshot shows the Microsoft Azure portal homepage. At the top, there's a search bar and a 'Create a resource' button. Below the search bar, there are icons for different services: Kubernetes services, Quickstart Center, Virtual machines, App Services, Storage accounts, SQL databases, Azure Cosmos DB, Function App, and More services. A message 'No resources have been viewed recently' is displayed with a 'View all resources' button. In the 'Resources' section, there are tabs for 'Recent' and 'Favorite'. The 'Recent' tab is selected, showing an empty list. At the bottom, there are navigation links for Subscriptions, Resource groups, All resources, and Dashboard. The status bar at the bottom right shows the date as 01-03-2024 and the time as 22:00.

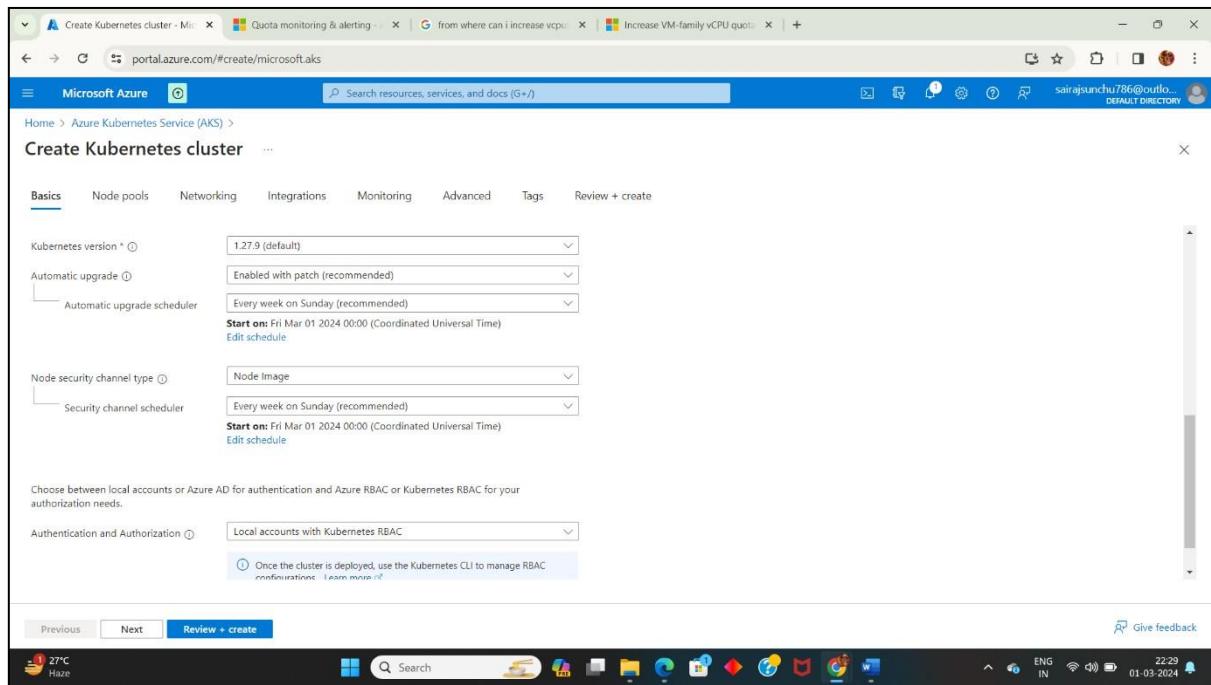
Step 3: Select Kubernetes Services

The screenshot shows the Microsoft Azure Marketplace search results for 'kubernetes service'. On the left, there's a sidebar with 'Get Started', 'Service Providers', 'Management', 'My Marketplace', 'Favorites', 'My solutions', 'Recently created', 'Private plans', 'Categories', and a link to the Azure portal. The main area displays a grid of service cards. The first card is 'Azure Kubernetes Service (AKS)' by Microsoft, described as a managed cluster with a Kubernetes orchestrator for container deployments. The second card is 'Kubernetes Fleet Manager' by Microsoft, described as enabling multi-cluster and at-scale scenarios for Kubernetes Service (AKS) clusters. The third card is 'Kubernetes - Azure Arc' by Microsoft, described as Register your Kubernetes clusters with Azure Arc for consolidated management across environments. The fourth card is 'AKS Base Image' by Azure, described as a Virtual Machine AKS Base Image. The fifth card is 'IBM WebSphere Liberty and Open Liberty on Azure' by IBM WebSphere, described as running IBM WebSphere Liberty and Open Liberty on the Azure Kubernetes Service. Other visible cards include 'Calico Cloud: Container and Kubernetes Security and Compliance' by Tigera, Inc., 'Managed Services for Azure Kubernetes Service PCI' by Orange Business, 'Managed Services for Azure Kubernetes Service' by Orange Business, 'Managed Azure Kubernetes' by Amesto Fortytwo AS, and 'TrilioVault for Kubernetes - BYOL' by Trilio. The status bar at the bottom right shows the date as 01-03-2024 and the time as 22:01.

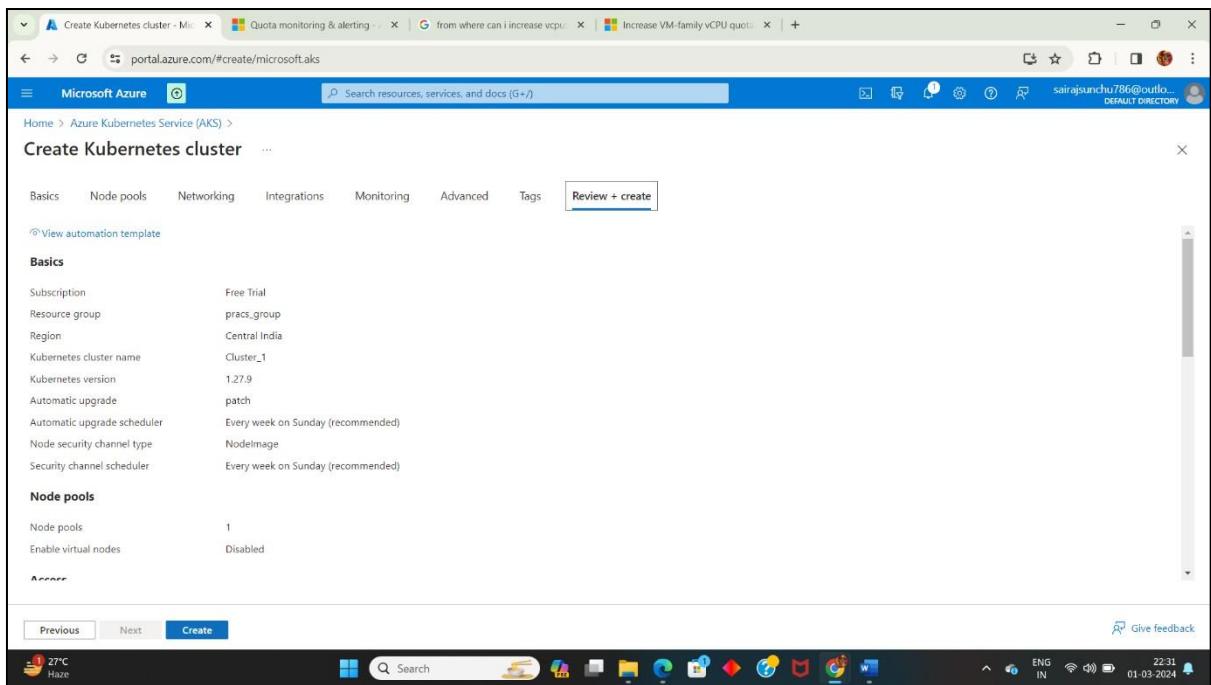


Step 4: Enter the subscription, resource group, Kubernetes Cluster name, Region, Kubernetes version, & DNS Name prefix

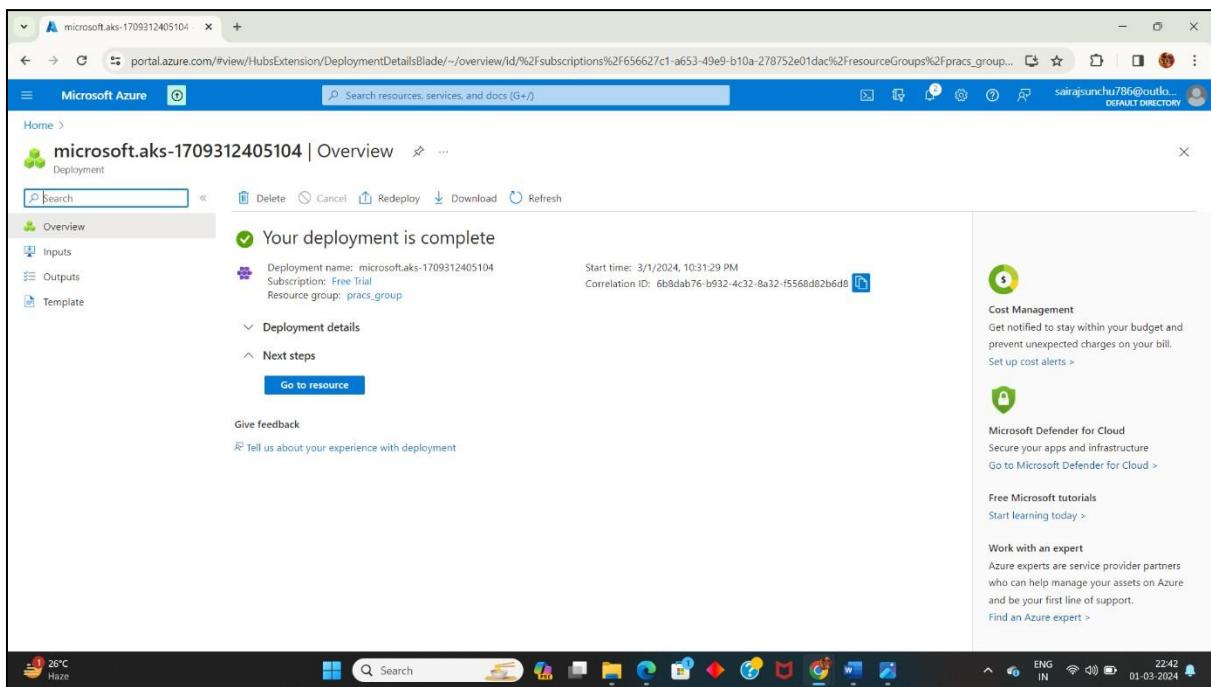




Step 5: Click on Review + Create Button



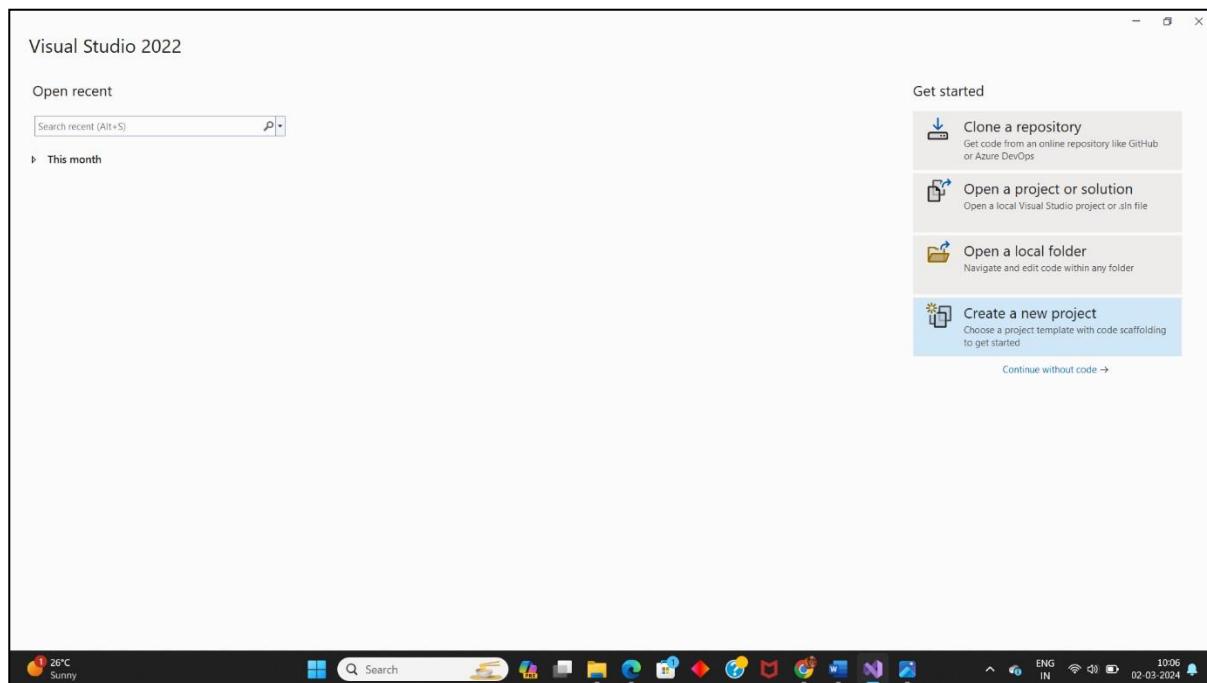
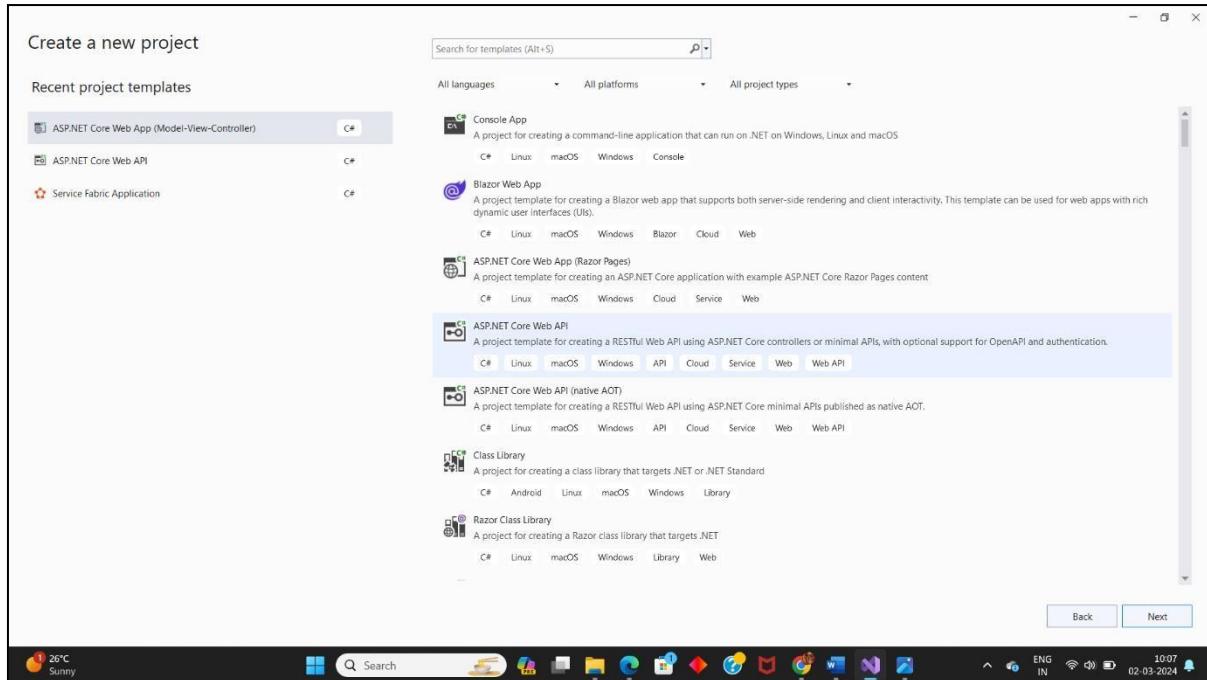
Step 6: Click on Create Button

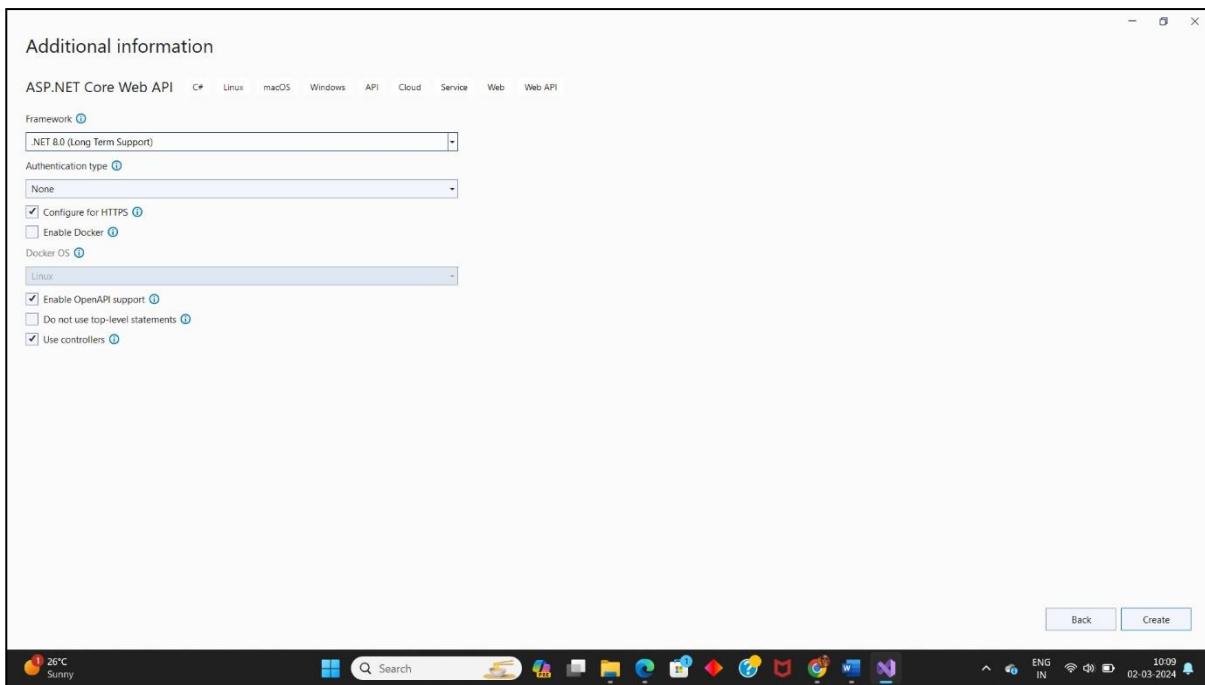
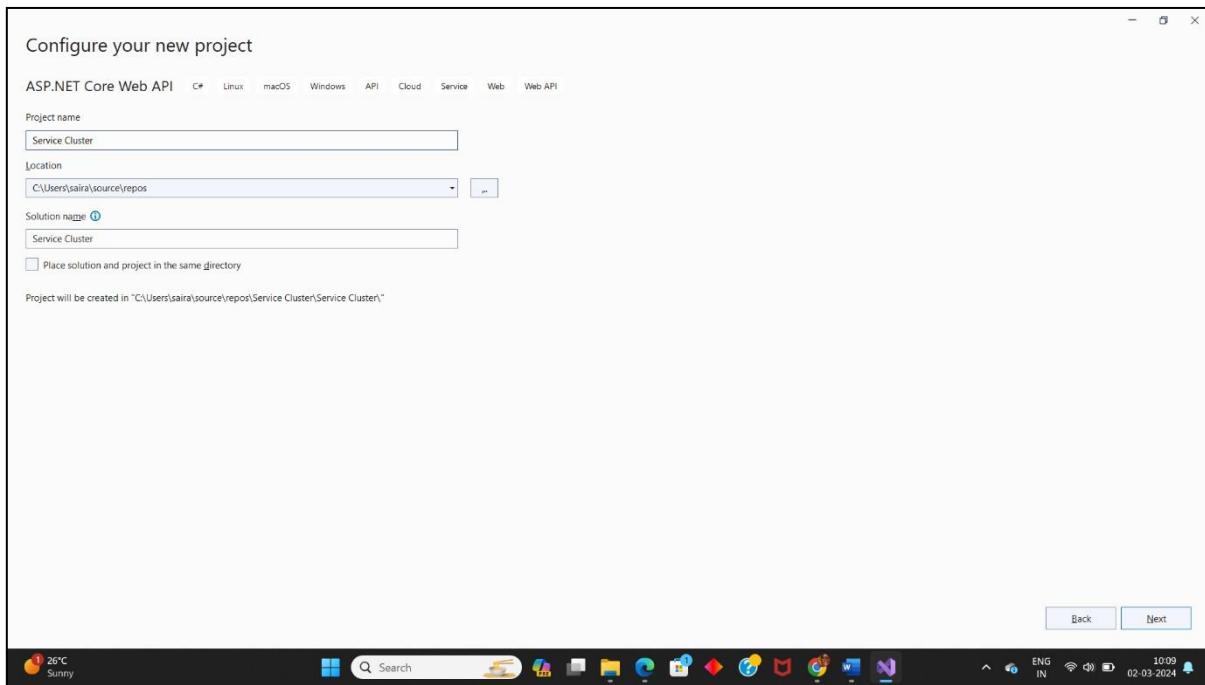


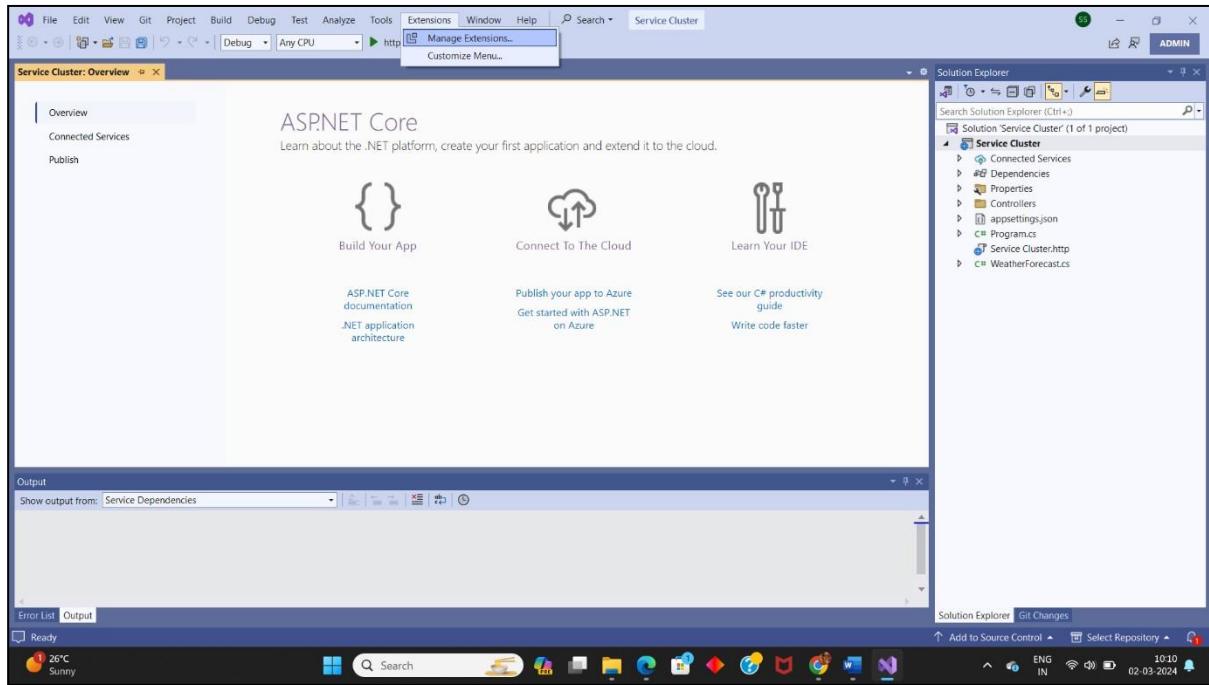
3B. Configure Visual Studio to work with an azure Kubernetes services Cluster

Step 1: Install Visual Studio Community 2022

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

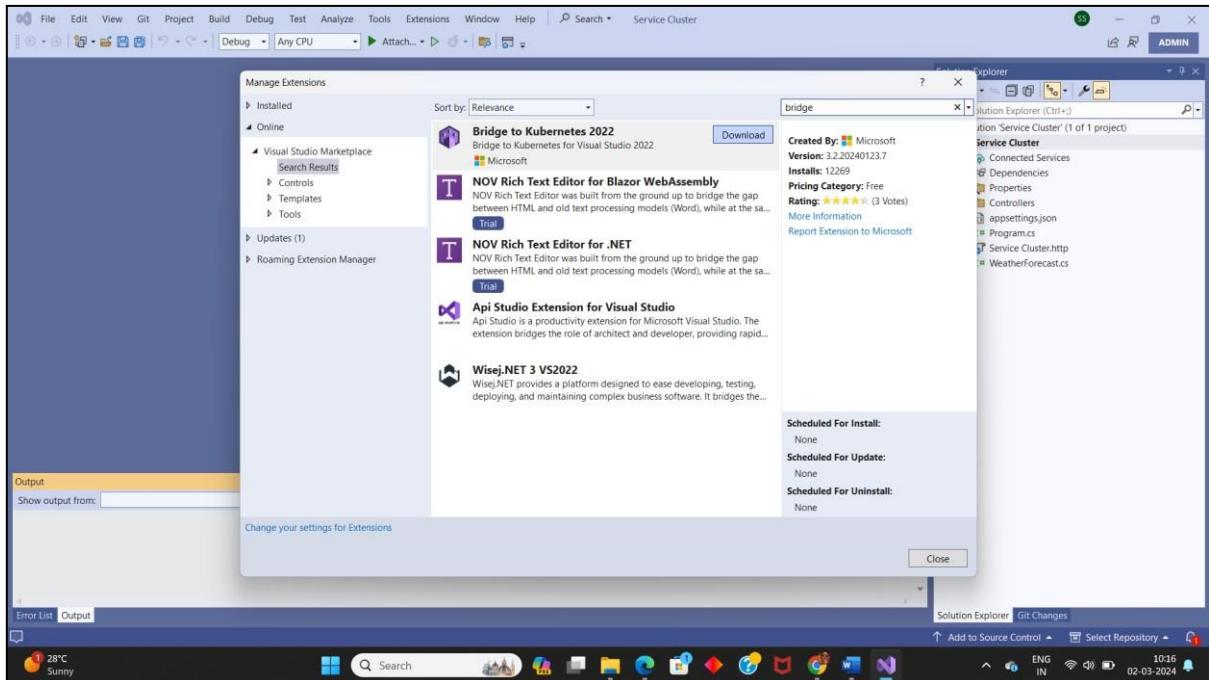


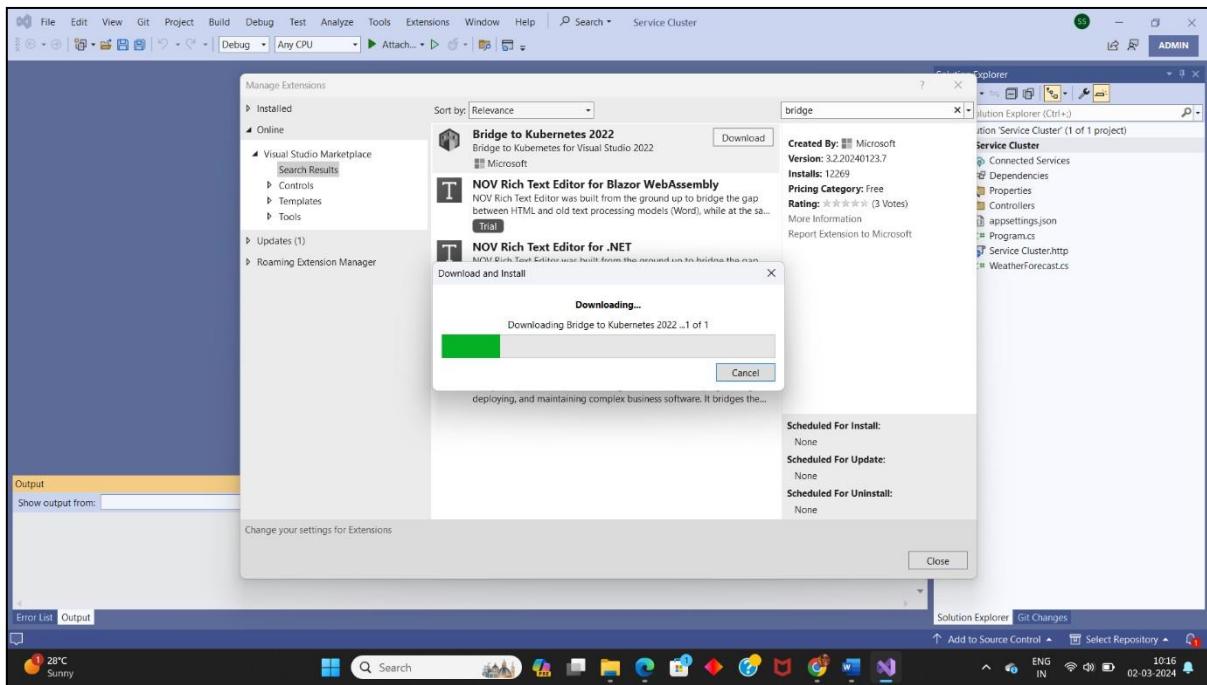




Step 3: Search for Bridge to Kubernetes 2022

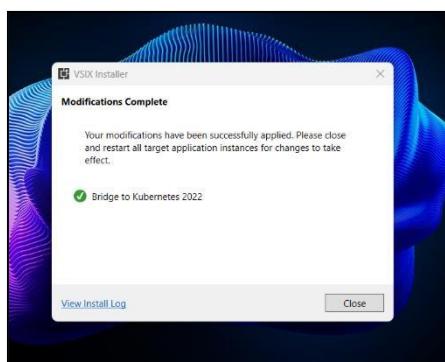
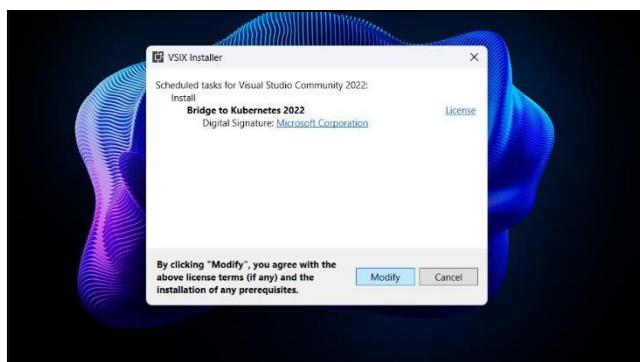
Step 4: Click on Download





Step 5: Close visual studio

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



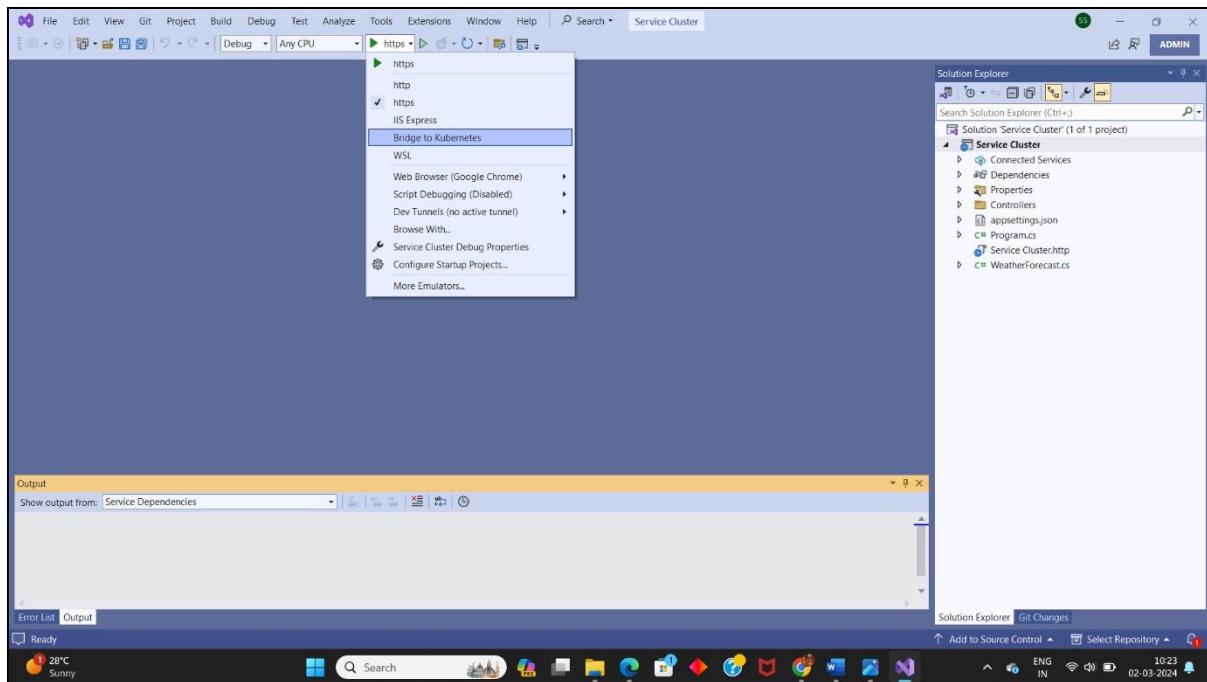
Step 7: Open Visual studio

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



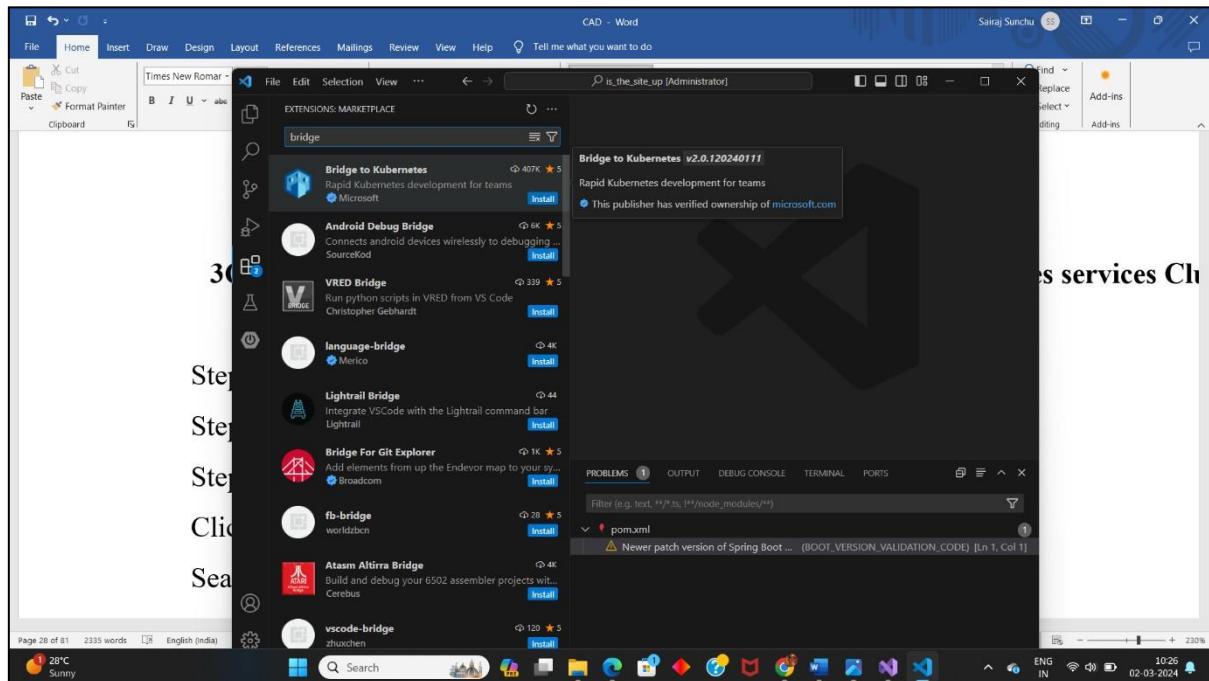
3C. Configure Visual Studio Code to work with an azure Kubernetes services Cluster

Step 1: Install visual studio code

Step 2: Open extension window

Step 3: Search for Bridge to Kubernetes 2022

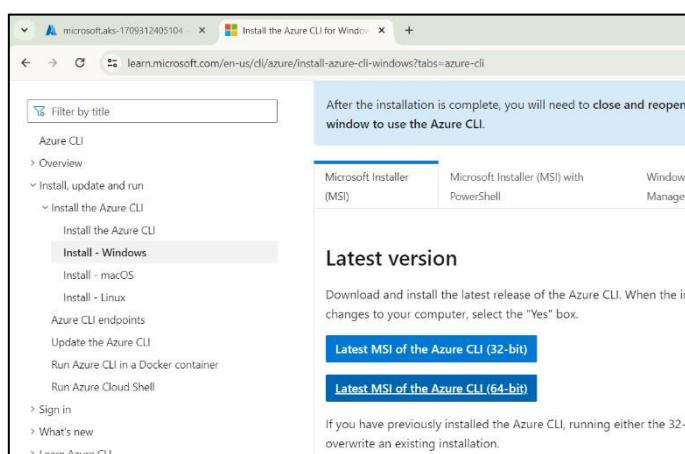
Click Install



Search for Azure CLI tools and install

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

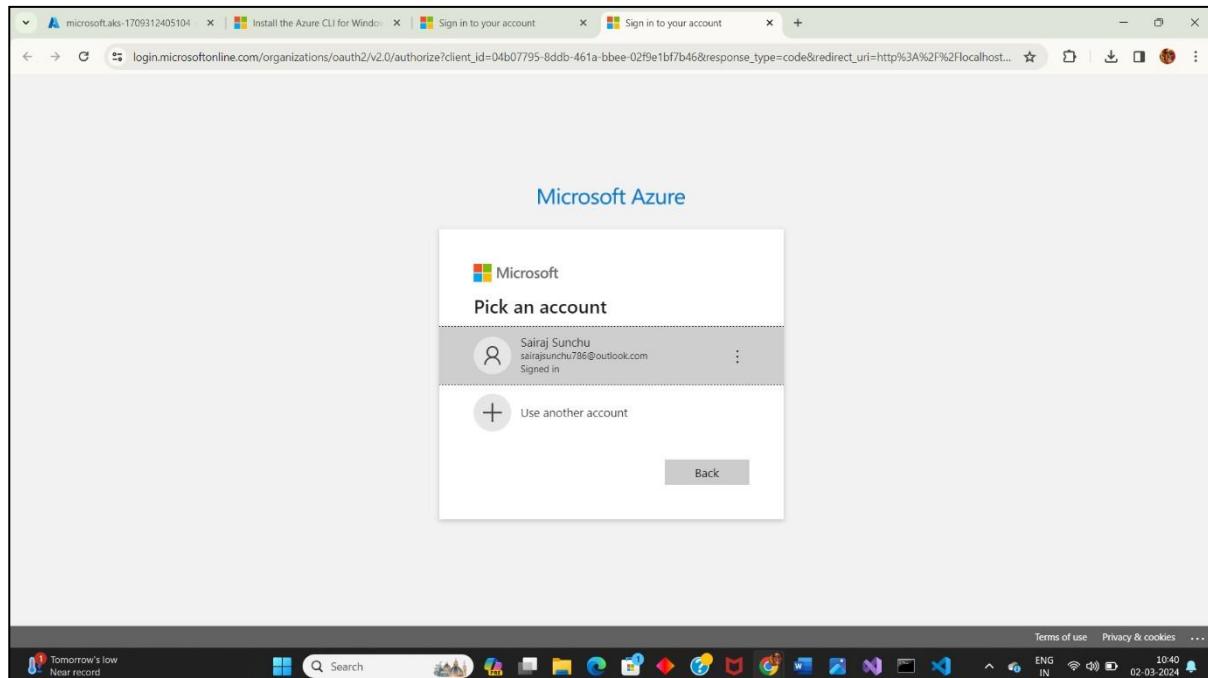
Click on Latest release of the Azure CLI, download the msi and run



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

```
C:\Windows\System32>az login
Microsoft Windows [Version 10.0.22631.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>
```



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

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

```
is_the_site_up [Administrator]
{
  "cloudName": "AzureCloud",
  "homeTenantId": "e297c380-8b1c-4c9b-9ba7-655502a9d7e0",
  "id": "656627c1-a653-49eb-b10a-278752e01dac",
  "isDefault": true,
  "managedByTenants": [],
  "name": "Free Trial",
  "state": "Enabled",
  "tenantId": "e297c380-8b1c-4c9b-9ba7-655502a9d7e0",
  "user": {
    "name": "sairaj.sunchu798@outlook.com",
    "type": "user"
  }
}

PS c:\code\is_the_site_up>
```

Practical No: 4

4A. Create an AKS Cluster from the Portal

Search Azure portal login on browser

Create a resource

The screenshot shows the Microsoft Azure portal interface. At the top, there are three tabs: 'Home - Microsoft Azure', 'Install the Azure CLI for Windows', and 'Azure Command-Line Interface'. The main navigation bar includes 'Microsoft Azure' (with an 'Upgrade' button), a search bar ('Search resources, services, and docs (G+)'), and a user profile ('sairajsunchu786@outlook.com'). Below the navigation is a section titled 'Azure services' with a 'Create a resource' button and icons for Quotas, Kubernetes services, Quickstart Center, Virtual machines, App Services, Storage accounts, SQL databases, Azure Cosmos DB, and More services. A 'Resources' section shows a table of recent resources: 'Cluster_1' (Kubernetes service, last viewed an hour ago) and 'pracs_group' (Resource group, last viewed an hour ago). Below this is a 'Navigate' section with links for Subscriptions, Resource groups, All resources, and Dashboard. At the bottom, there's a toolbar with various icons and a status bar showing the date and time.

Search Azure Kubernetes Service

The screenshot shows the Microsoft Azure Marketplace search results for 'kubernetes service'. The search bar at the top contains the query 'kubernetes service'. On the left, a sidebar provides navigation options like 'Get Started', 'Service Providers', 'Management', 'My Marketplace', 'Categories', and a link to the marketplace URL. The main area displays a grid of 20 results, each with a thumbnail, title, provider, and a 'Create' button. The results include: 'Azure Kubernetes Service (AKS)' by Microsoft, 'Kubernetes Fleet Manager' by Microsoft, 'Kubernetes - Azure Arc' by Microsoft, 'AKS Base Image' by Azure, 'IBM WebSphere Liberty and Open Liberty on Azure' by IBM WebSphere, 'Calico Cloud: Container and Kubernetes Security' by Tigera, Inc., 'Managed Services for Azure Kubernetes Service PCI' by Orange Business, 'Managed Services for Azure Kubernetes Service' by Orange Business, 'Managed Azure Kubernetes' by Amesto Fortytwo AS, and 'TrilioVault for Kubernetes - BYOL' by Trilio.

Click on create

The screenshot shows the Azure Marketplace page for the Azure Kubernetes Service (AKS). At the top, there's a search bar and a 'Create' button. Below the search bar, there's a section for 'Plan' where 'Azure Kubernetes Service (AKS)' is selected. The main content area includes tabs for 'Overview', 'Plans', 'Usage Information + Support', and 'Ratings + Reviews'. The 'Overview' tab is active, displaying a brief description of AKS as the quickest path to Kubernetes on Azure. Below the description, there are links to related services like Firewall, Microsoft Azure Attestation, Service endpoint policy, and Azure Managed Grafana. The bottom of the screen shows a Windows taskbar with various icons and system status.

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

The screenshot shows the 'Create Kubernetes cluster' wizard in the Azure portal. The 'Basics' tab is selected. In the 'Project details' section, 'Subscription' is set to 'Free Trial' and 'Resource group' is set to '(New) prac_4'. In the 'Cluster details' section, 'Cluster preset configuration' is set to 'Dev/Test', 'Kubernetes cluster name' is 'aks_portal', and 'Region' is '(Europe) UK South'. The bottom of the screen shows a Windows taskbar with various icons and system status.

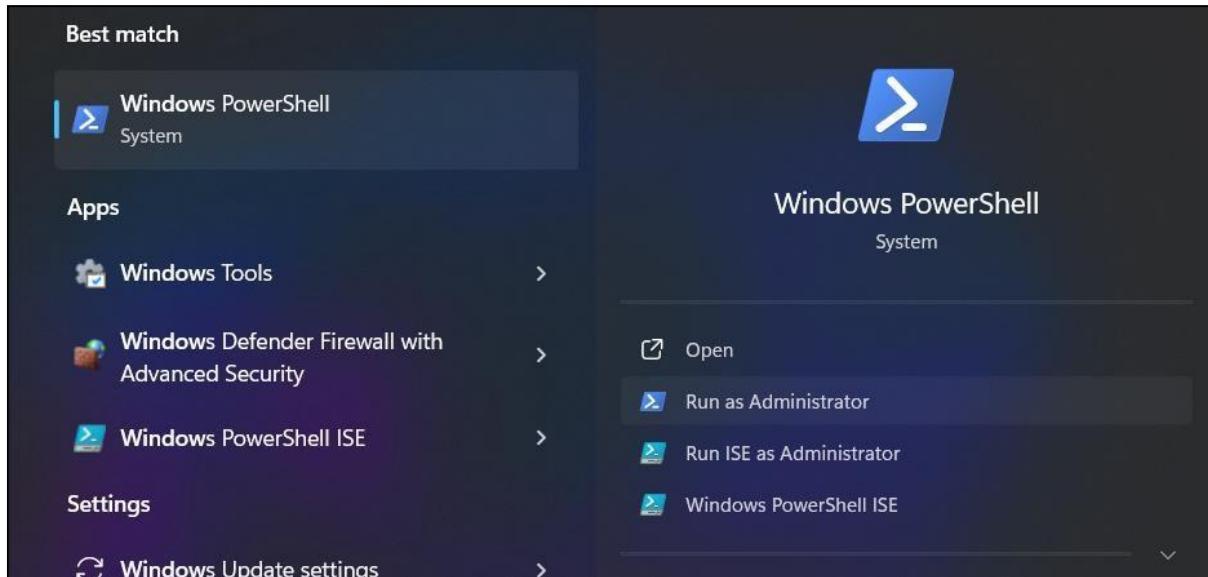
Click on review + create tab

Click on create

4B. Create an AKS cluster – with Azure CLI

Step 1: Install Azure CLI for windows system.

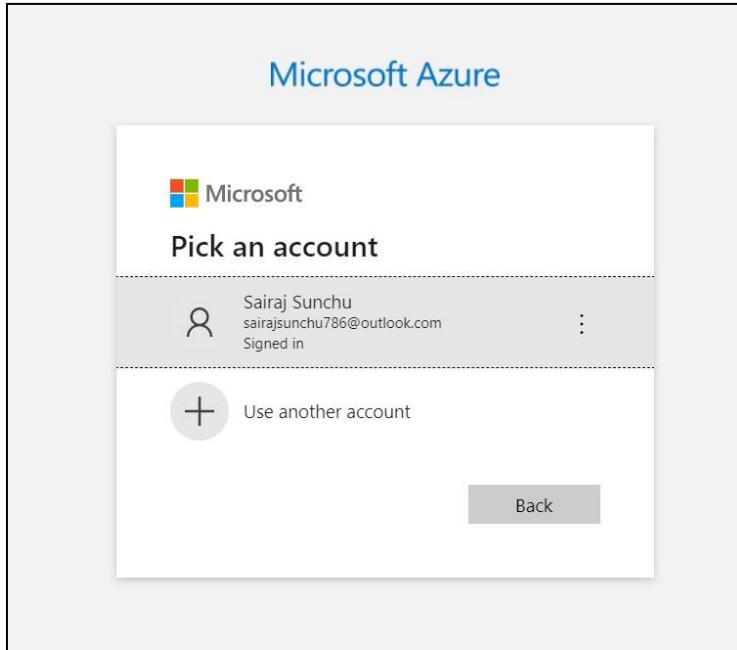
Step 2: Open Windows PowerShell Run as Administrator



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 --generate-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,
      "osType": "Linux"
    }
  ],
  "dnsPrefix": "onecluster",
  "nodeCount": 1,
  "nodePools": [
    {
      "count": 1,
      "osType": "Linux",
      "vmSize": "Standard_DS2_v2"
    }
  ],
  "serviceEndpoint": "Microsoft.ContainerService/managedClusters"
}
```

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.

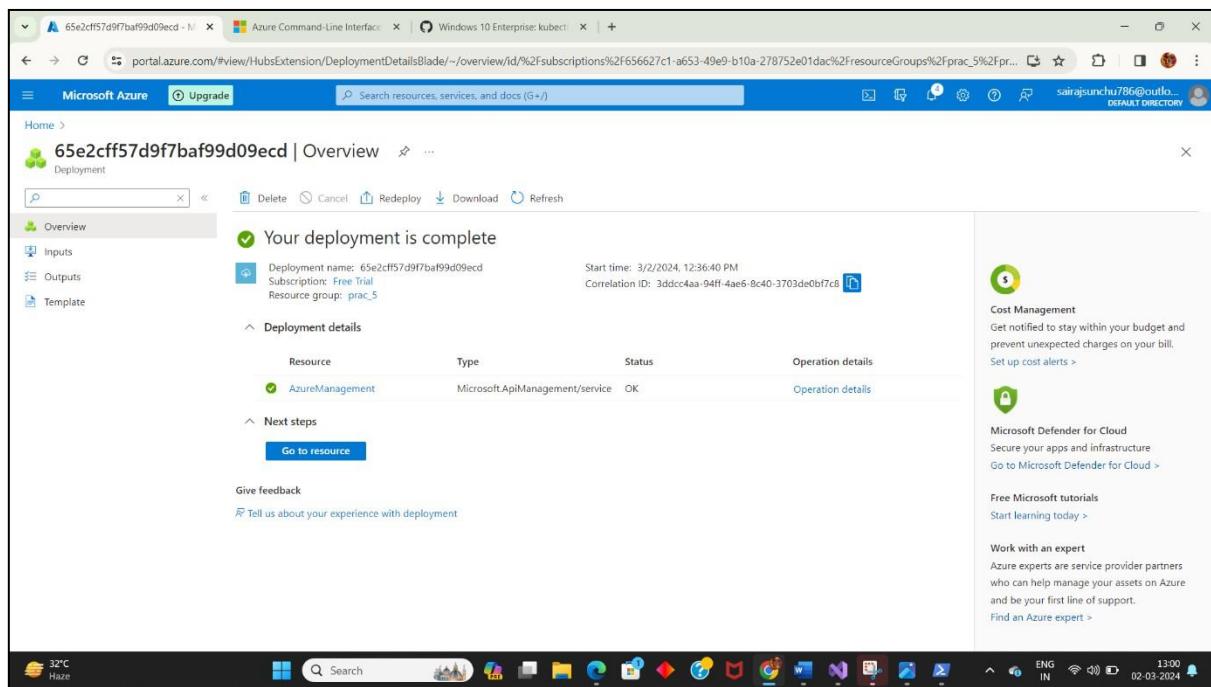
Step 3: Select “API Management service”

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

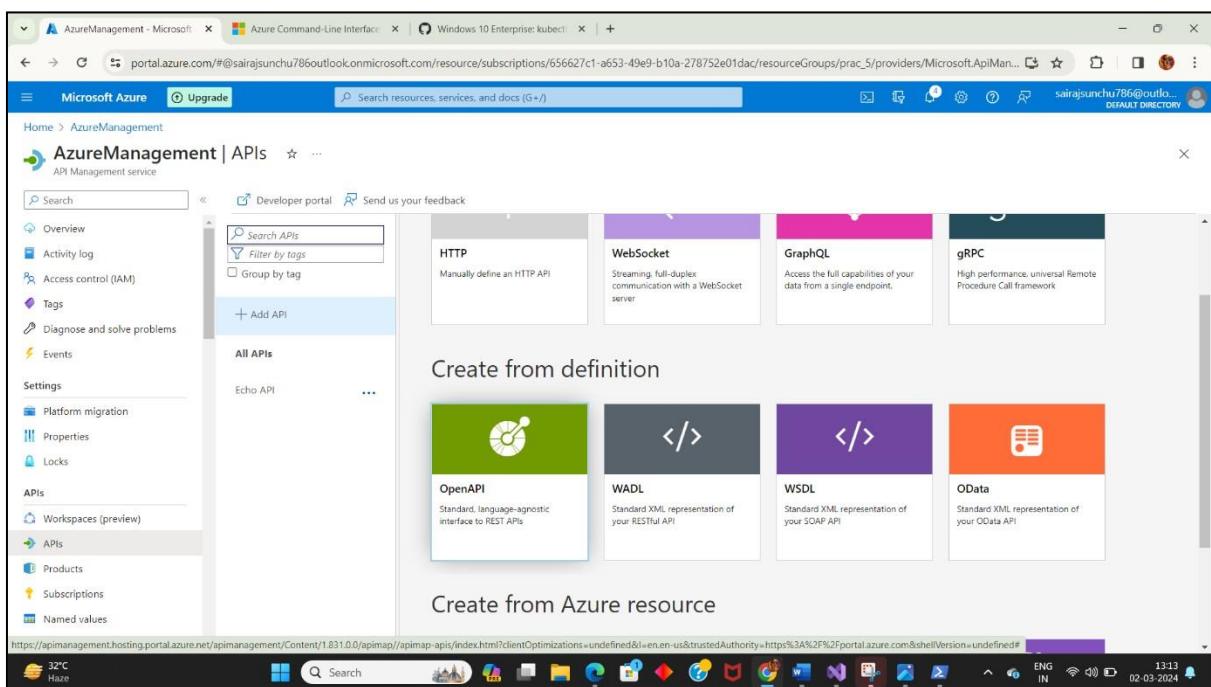
Select subscription, create new resource group

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

The screenshot shows the Azure Management Portal's API Management service. A modal window titled 'Create from OpenAPI specification' is open. In the 'Basic' tab, the URL 'https://conferenceapi.azurewebsites.net?format=json' is entered. The 'Display name' is 'Demo Conference API', 'Name' is 'demo-conference-api', and 'API URL suffix' is 'conference'. The 'Base URL' is 'http://azuremanagement.azure-api.net/conference'. At the bottom right of the modal are 'Create' and 'Cancel' buttons.

The screenshot shows the 'Design' tab for the 'Demo Conference API'. The 'Frontend' section lists operations: GET GetSession, GET GetSessions, GET GetSessionTop..., GET GetSpeaker, and GET GetSpeakers. The 'Backend' section shows an 'HTTP(s) endpoint' at 'https://conferenceapi.azurewebsite...' and a 'Policies' section with 'base'. Below the frontend, there is an 'Outbound processing' section with its own 'Policies' section containing 'base'.

Select GetSpeakers

The screenshot shows the Azure Management API portal interface. A specific API operation, 'GetSpeakers', is selected for configuration. The 'Frontend' panel details the API endpoint and its parameters. The 'Backend' panel specifies the service endpoint. Policies are being configured for both the incoming and outgoing requests.

Click on send button

The screenshot shows the 'Test' tab of the API configuration screen. It displays the request configuration, including the URL and the full HTTP request header. The 'Send' button is prominently displayed at the bottom of the request pane.

Check the Http response

The screenshot shows the Azure Management API Test tool interface. On the left, there's a sidebar with 'All APIs' and two entries: 'Demo Conference API' and 'Echo API'. The 'Demo Conference API' entry is selected. In the main area, under 'REVISION 1' (CREATED Mar 2, 2024, 1:35:14 PM), there are tabs for 'Design', 'Settings', 'Test', 'Revisions (1)', and 'Change log'. The 'Test' tab is active, showing a list of API methods: 'GET GetSession', 'GET GetSessions', 'GET GetSessionTop...', 'GET GetSpeaker', 'GET GetSpeakers' (which is highlighted in blue), 'GET GetSpeakerSes...', 'GET GetSpeakerTo...', 'GET GetTopic', and 'GET GetTopics'. Below this list, the 'HTTP response' section is expanded, showing the 'Message' tab. The message content is as follows:

```
HTTP/1.1 200 OK
cache-control: no-cache
content-length: 40606
content-type: application/vnd.collection+json
date: Sat, 02 Mar 2024 08:11:47 GMT
expires: -1
pragma: no-cache
vary: Origin
x-aspnet-version: 4.0.30319
x-powered-by: ASP.NET

{
  "collection": {
    "version": "1.0",
    "href": "https://conferenceapi.azurewebsites.net:443/speakers",
    "links": []
  }
}
```

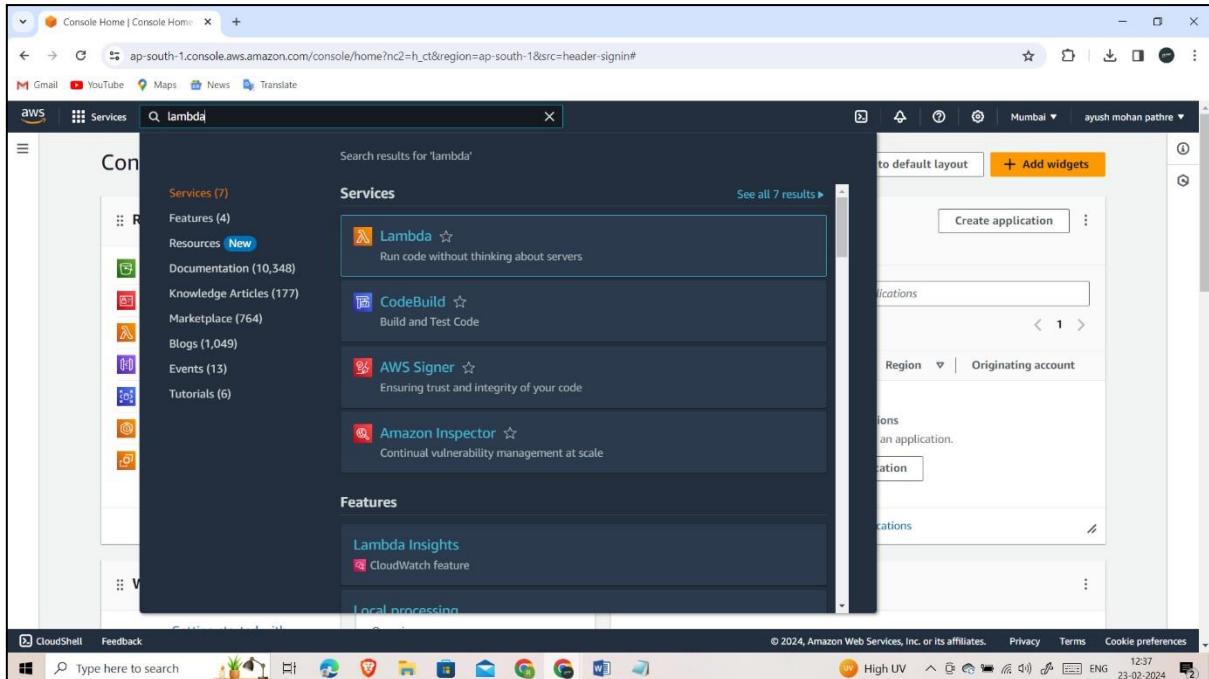
At the bottom of the 'Message' tab, there are 'Send' and 'Trace' buttons, and a checkbox for 'Bypass CORS proxy'. The status bar at the bottom right shows the date and time as 02-03-2024 13:42.

Practical No: 6

AWS API Gateway Authorizer

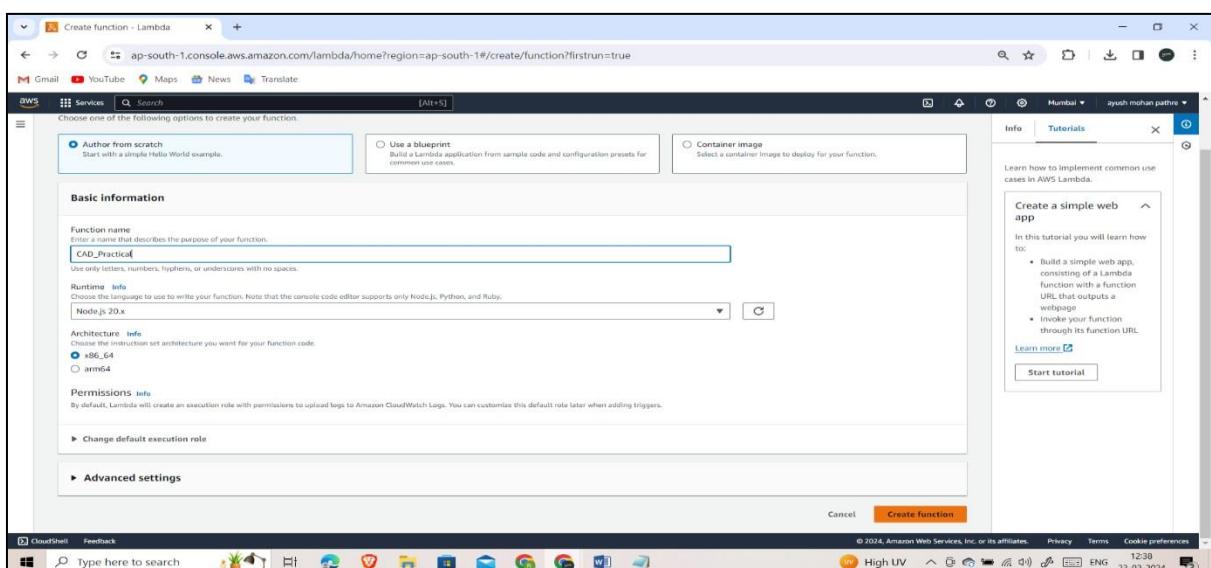
Step 1 – Login to aws.amazon.com and Search for “Lambda”

Step 2 – Click on Create a Function

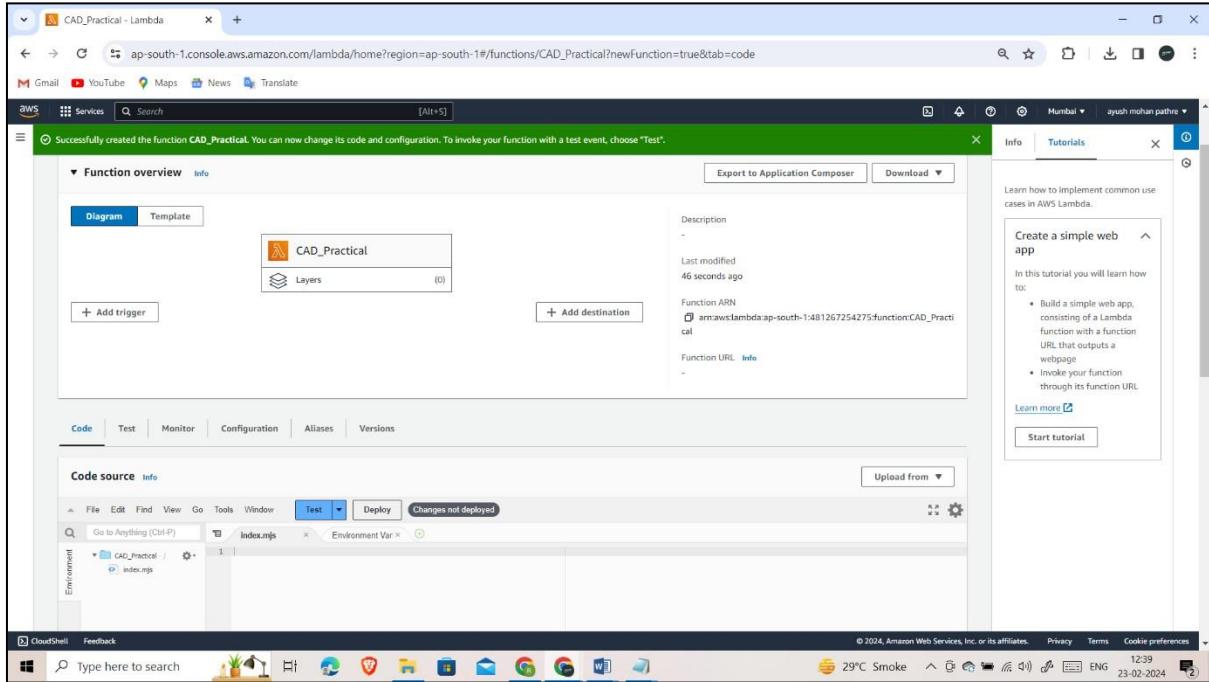


Name – CAD_Practical

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;
}
```

The screenshot shows the AWS Lambda function editor for a function named 'CAD_Practical'. The code source tab is open, displaying the 'index.js' file. The code implements an IAM policy generator based on the provided token. A success message at the top indicates the function has been successfully created. On the right side, there is a sidebar titled 'Create a simple web app' with a 'Start tutorial' button.

```

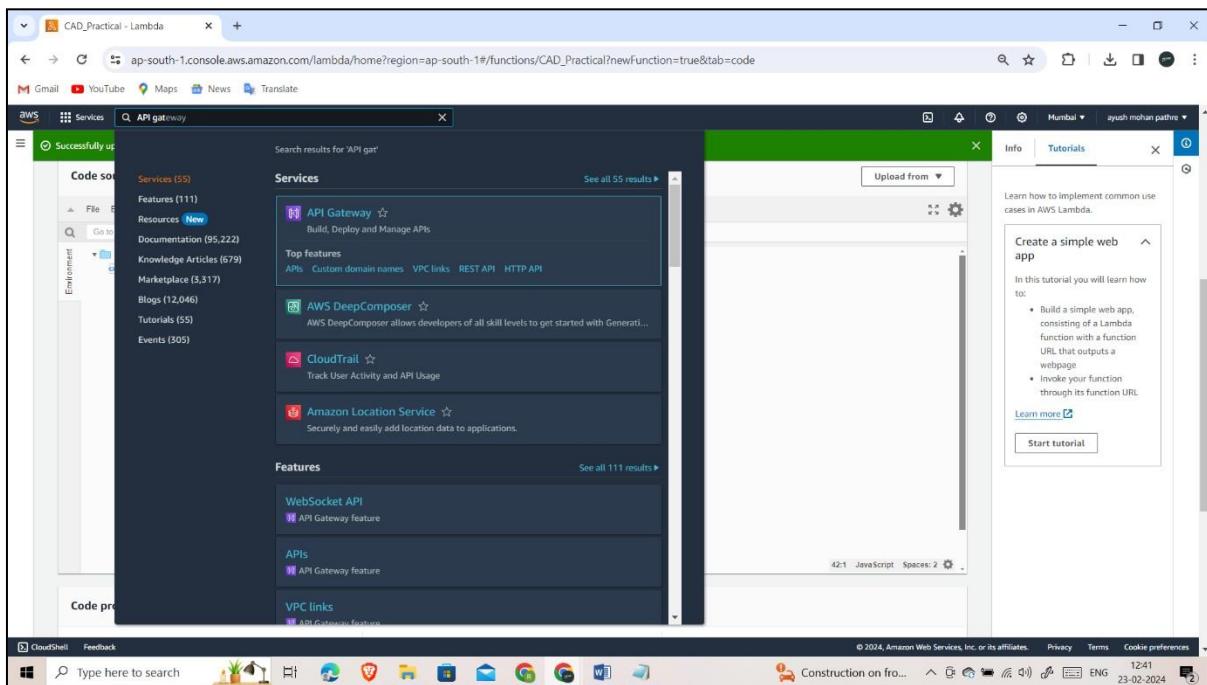
1 export const handler = function(event, context, callback) {
2     var token = event.authorizationToken;
3     switch (token) {
4         case 'Allow':
5             callback(null, generatePolicy('user', 'Allow', event.methodArn));
6             break;
7         case 'Deny':
8             callback(null, generatePolicy('user', 'Deny', event.methodArn));
9             break;
10        case 'Unauthorized':
11            callback("Unauthorized"); // Return a 403 Unauthorized response
12            break;
13        default:
14            callback("Error: Invalid token"); // Return a 500 Invalid token response
15            break;
16    }
17    // Help function to generate an IAM policy
18    var generatePolicy = Function(principalId, effect, resource) {
19        var authResponse = {};
20
21        authResponse.principalId = principalId;
22        if (effect && resource) {
23            authResponse['Effect'] = effect;
24            authResponse['Version'] = '2012-10-17';
25            authResponse['Statement'] = [];
26
27            statementOne.Action = 'execute-api:Invoke';
28            statementOne.Resource = resource;
29            policyDocument.Statement[0] = statementOne;
30            authResponse.policyDocument = policyDocument;
31        }
32    }
33
34    // Optional output with custom properties of the String, Number or Boolean type.
35    authResponse.context = {
36        'stringKey': 'stringVal',
37        'numberKey': 32,
38    };
39

```

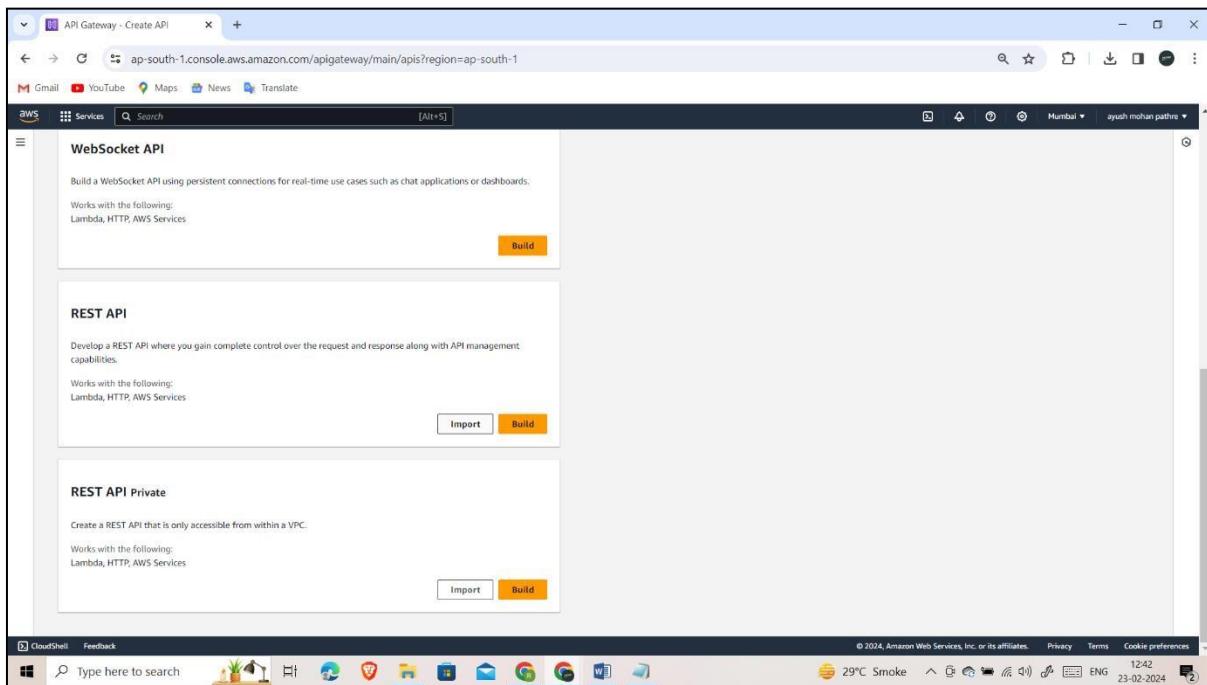
Step 5 – Click on deploy once the code is pasted. The code is successfully deployed

The screenshot shows the AWS Lambda function editor for the same function 'CAD_Practical'. The code source tab is open, displaying the 'index.js' file. The deployment process is shown with a progress bar at the top labeled 'Uploading the function CAD_Practical.' The sidebar on the right remains the same as in the previous screenshot.

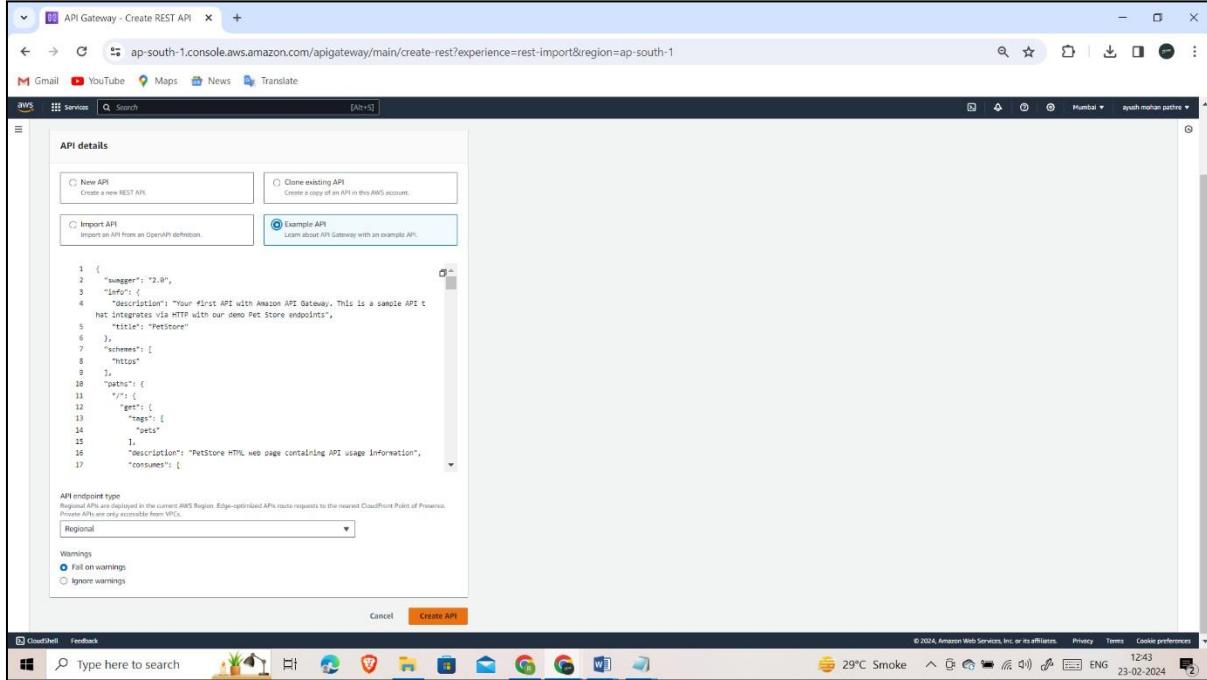
Step 6 – Now search API Gateway in the search bar and open in the new tab



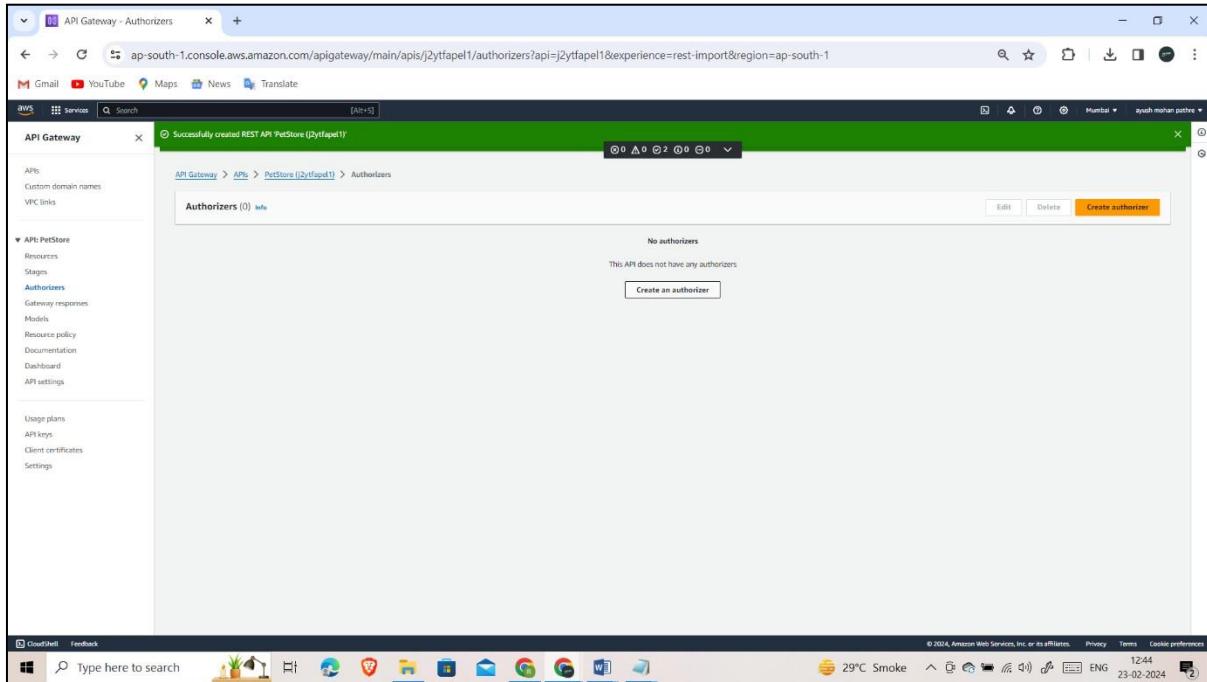
Step 7 – Select Choose an API type - > REST API -> Build



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

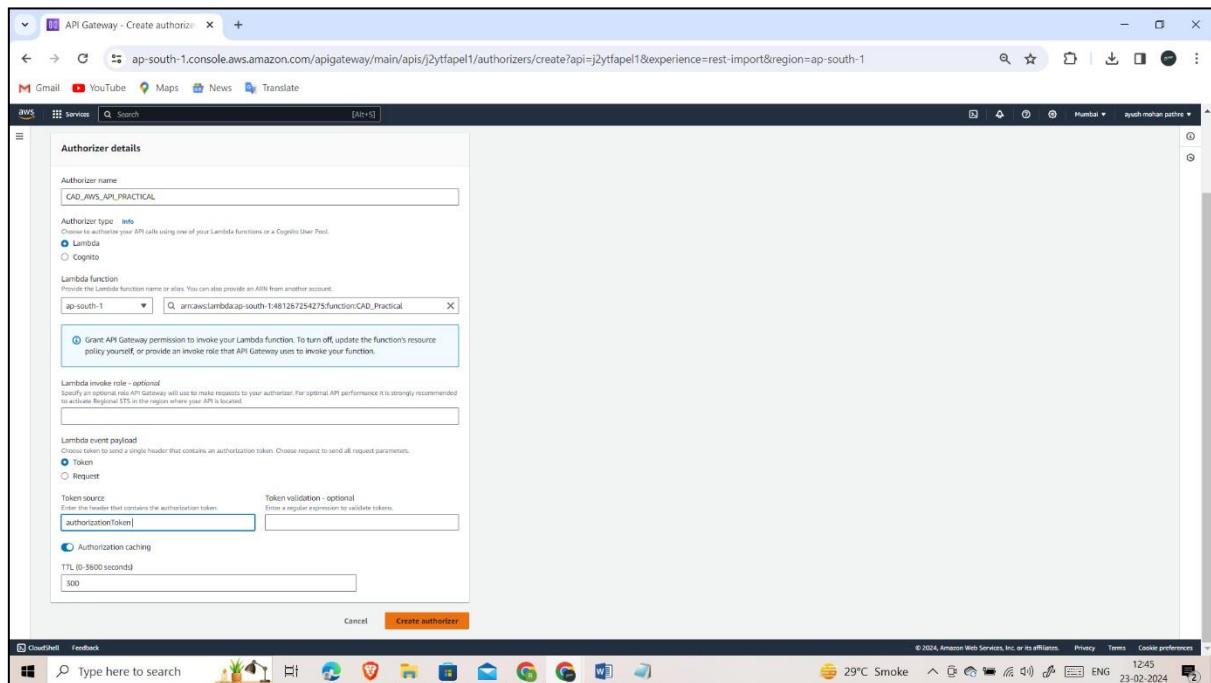
Name – CAD_AWS_API_PRACTICAL

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

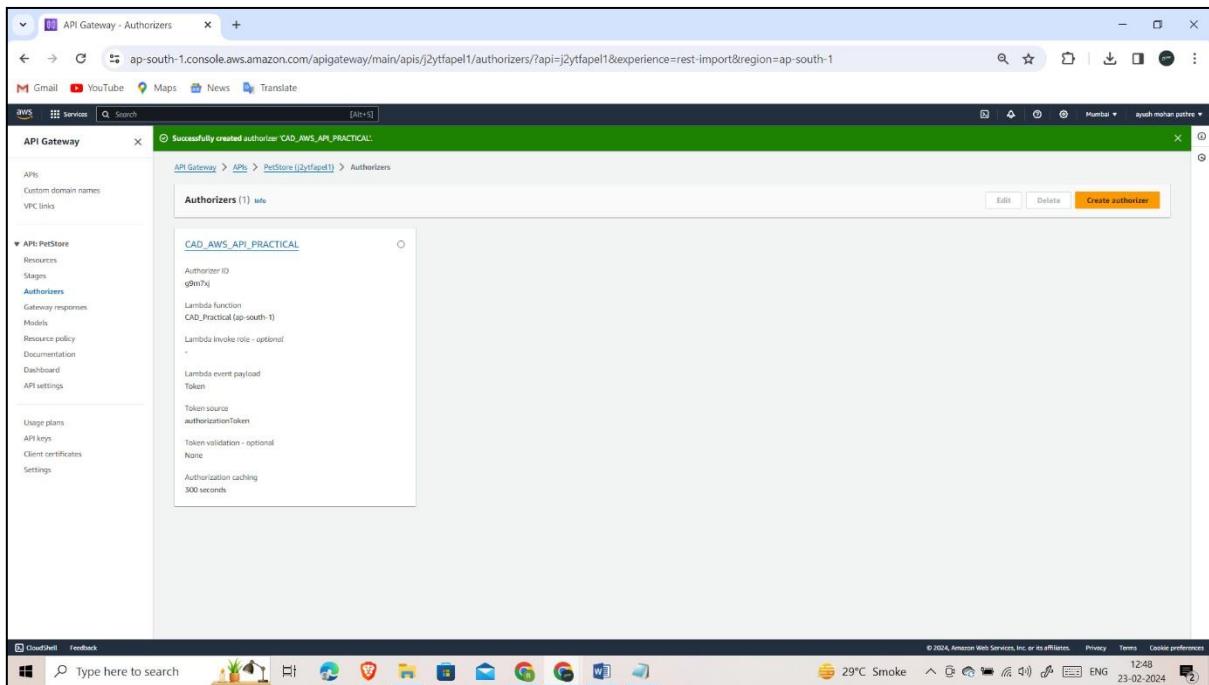
Lambda Invoke Role – Leave it Blank

Token Source – authorizationToken

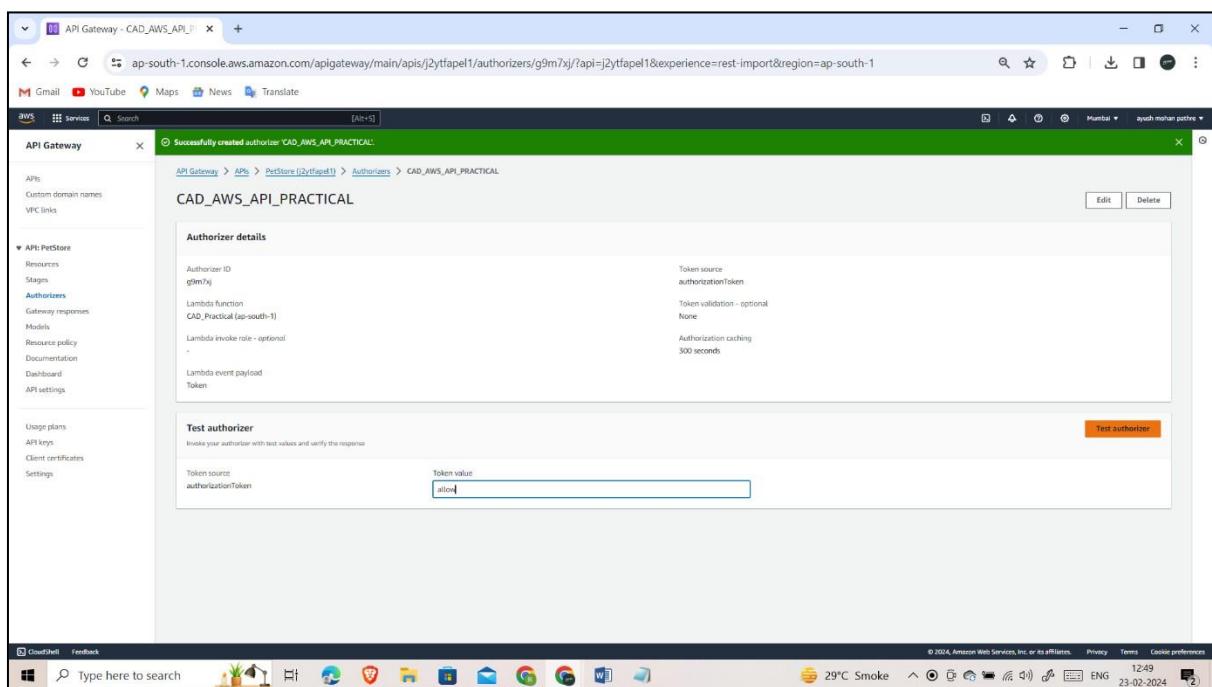
And then click on Create



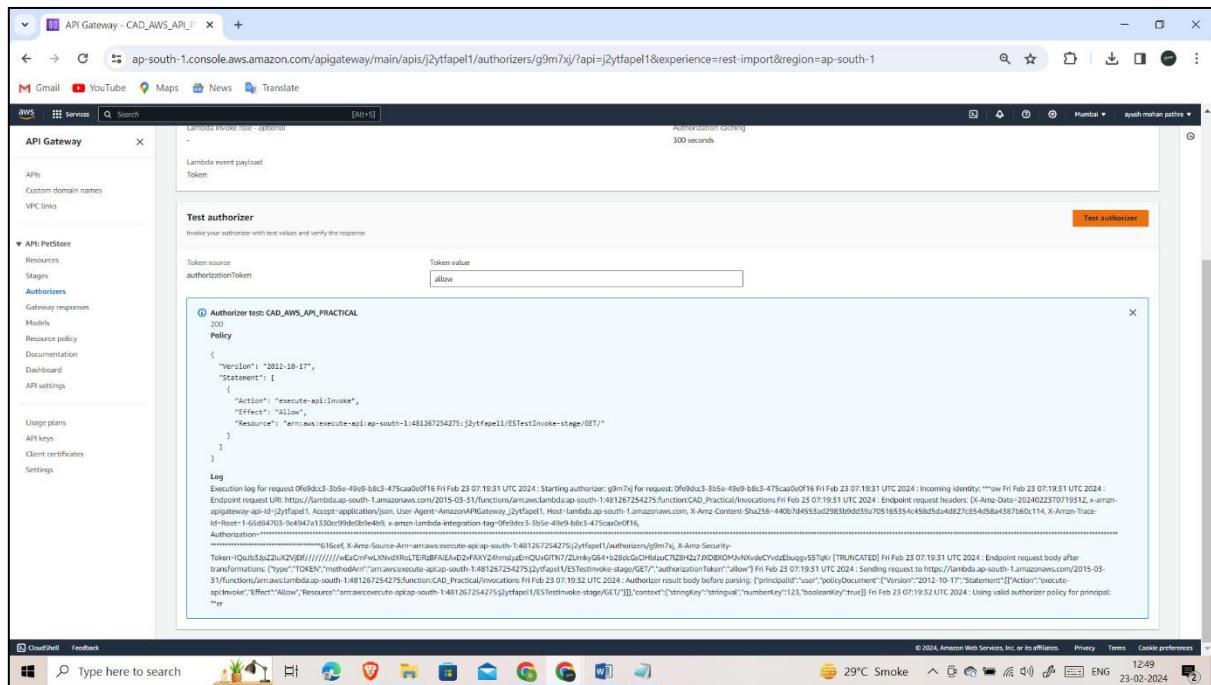
Step 11 – Click on Create



Step 12 – Now click on the test button



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

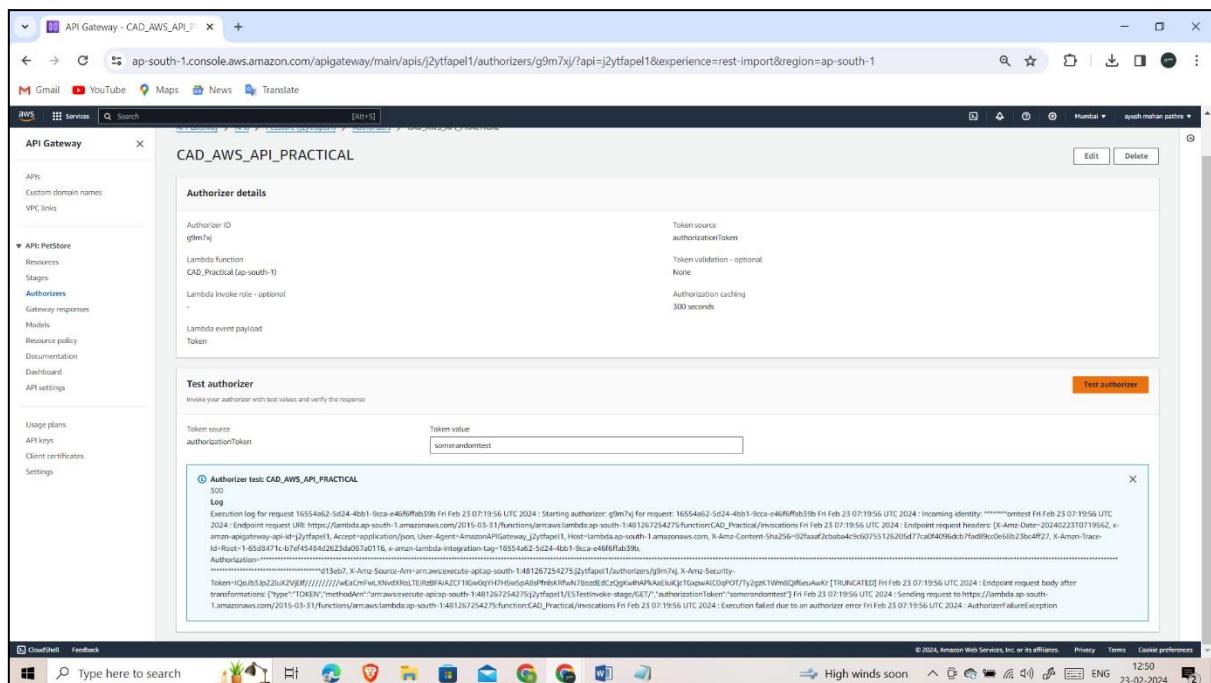


We can see the that function that we created is tokenized. When we input allow, it will check the IAM policy and return policy output of the same.

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':

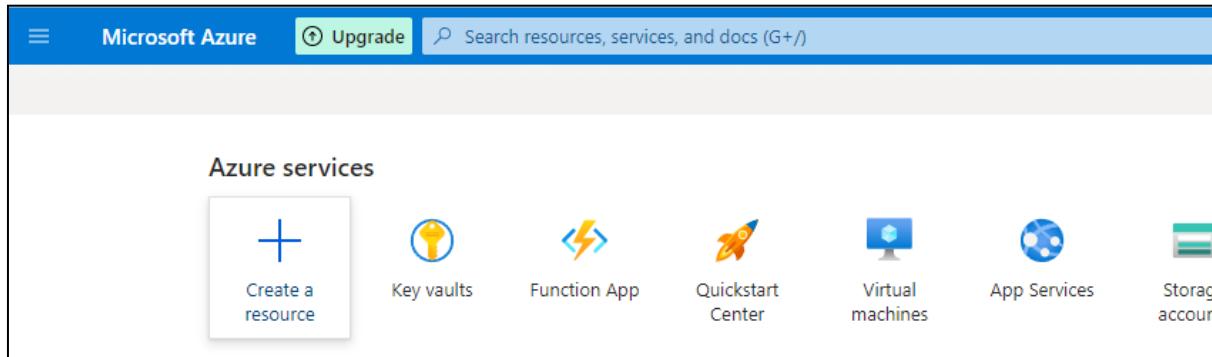
'deny':

'unauthorized':



Practical No: 7

Create a serverless API using Azure functions



The screenshot shows the Microsoft Azure Marketplace page. The left sidebar has a 'Get Started' button highlighted in grey, followed by 'Service Providers', 'Management' (which is underlined), 'Private Marketplace', and 'Private Offer Management'. The 'My Marketplace' section is also underlined. Below that are 'Favorites', 'My solutions', and 'Recently created'. The main area shows a search result for 'function app'. The card for 'Function App' by Microsoft is shown, featuring a lightning bolt icon, the text 'Function App', 'Microsoft', 'Azure Service', and a descriptive paragraph about writing functions. At the bottom of the card are 'Create' and a heart icon.

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 * Resource Group * Create new

Function App name * .azurewebsites.net

Create Function App

Instance Details

Function App name * .azurewebsites.net

Do you want to deploy code or container image? * Code Container Image

Runtime stack *

Version *

Region *

Operating system

The Operating System has been recommended for you based on your selection of runtime stack.

Operating System * Linux Windows

The screenshot shows the 'Create Function App' wizard on the Microsoft Azure portal. In the 'Operating System' section, it says 'The Operating System has been recommended for you based on your selection of runtime stack.' There are two options: 'Linux' (radio button) and 'Windows' (radio button, selected). In the 'Hosting' section, it says 'The plan you choose dictates how your app scales, what features are enabled, and how it is priced.' There are three options: 'Consumption (Serverless)' (selected), 'Functions Premium' (not supported for selected region and operating system), and 'App service plan' (not supported for selected subscription).

The screenshot shows the 'Networking' tab of the 'Create Function App' wizard. It includes tabs for Basics, Networking (selected), Monitoring, Deployment, Tags, and Review + create. Under the Networking tab, there is a section for 'Enable public access' with 'On' (radio button, selected) and 'Off' (radio button). A warning message states: '⚠ Network injection is only available in Functions Premium and Basic, Standard, Premium, Premium V2, Premium V3 Dedicated App Service plans.' Below this, there is another section for 'Enable network injection' with 'On' (radio button) and 'Off' (radio button selected).

The screenshot shows the 'Create Function App' wizard on the 'Monitoring' tab. At the top, there are tabs for Basics, Networking, Monitoring (which is selected), Deployment, Tags, and Review + create. A search bar at the top right contains the placeholder 'Search resources, services, and docs (G+/-)'. Below the tabs, a descriptive text explains Azure Monitor application insights, mentioning its role in monitoring performance anomalies and providing powerful analytics tools. It also notes that the bill is based on data usage and retention settings, with a link to 'Learn more'. Under the 'Application Insights' section, there is a question 'Enable Application Insights *' with two radio button options: 'No' (unchecked) and 'Yes' (checked). A dropdown menu below shows '(New) serverlessapiawt (Southeast Asia)' with a 'Create new' option. The 'Region' field is set to 'Southeast Asia'. The overall interface is clean with a blue header and white background.

The screenshot shows the 'Create Function App' wizard on the 'Deployment' tab. At the top, there are tabs for Basics, Networking, Monitoring, Deployment (which is selected), Tags, and Review + create. A search bar at the top right contains the placeholder 'Search resources, services, and docs (G+/-)'. Below the tabs, a section titled 'Enable GitHub Actions to continuously deploy your app.' explains that GitHub Actions is an automation framework for building, testing, and deploying apps. It notes that if code is in GitHub, a workflow file will be added to automatically deploy to App Service. There is a link to 'Learn more'. Under the 'GitHub Actions settings' section, there is a question 'Continuous deployment' with two radio button options: 'Disable' (checked) and 'Enable'. Below this, the 'GitHub Actions details' section asks to select GitHub details for repository access. It includes fields for 'GitHub account' (with an 'Authorize' button), 'Organization' (with a dropdown menu), and 'Repository' (with a dropdown menu). The interface follows the same clean design with a blue header and white background as the previous screenshot.

The screenshot shows the 'Create Function App' wizard in the Microsoft Azure portal. The current step is 'Review + create'. The app details are as follows:

- Subscription:** 5037bd55-a725-439b-8977-6ce99cb02934
- Resource Group:** pract
- Name:** serverlessapiawt
- Runtime stack:** .NET 6 (LTS)

The 'Hosting' section shows a 'Plan (New)' option. At the bottom, there are buttons for 'Create', '< Previous', 'Next >', and 'Download a template for automation'.

The screenshot shows the overview page for the 'serverlessapiawt' function app. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Functions, App keys, App files, Proxies, Deployment, and Deployment slots. The main area displays the following information:

- Subscription (move):** Free Trial
- Subscription ID:** 5037bd55-a725-439b-8977-6ce99cb02934
- Tags (edit):** :

The 'Functions' tab is selected. A callout box at the top right says 'Create functions in your preferred way'. Below it, two options are shown:

- Create in Azure portal**: Best optimized for:
 - Getting started without local setup
 - Choose from our Function templates
- VS Code Desktop**: Best optimized for:
 - Local development within VS Code
 - Custom development tool

Buttons for 'Create function' and 'Create with VS Code Desktop' are visible at the bottom.

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](#)

Template	Description
HTTP trigger	A function that will be run whenever it receives an HTTP request, responding based on the 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 Storage queue

Template details

We need more information to create the HTTP trigger function. [Learn more](#)

New Function*

Authorization level* [Anonymous](#)

[Create](#) [Cancel](#)

[Activate Windows](#)
Go to Settings to activate Windows.

Microsoft Azure [Upgrade](#) ≡ ☰ 🔍 ⚙️ 🌐 ⓘ

Home > serverlessapiawt >

Customer [Function](#)

✓ Enable [Disable](#) [Delete](#) [Get Function Url](#) [Refresh](#)

Overview ☰

Developer

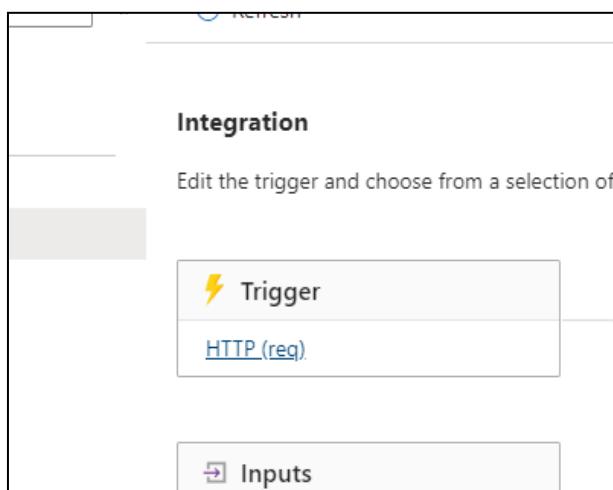
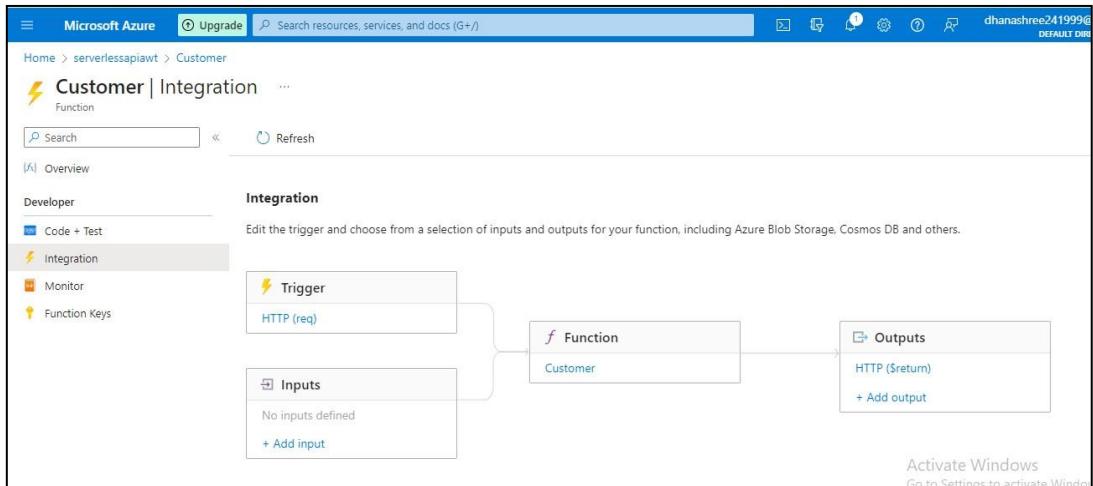
- [Code + Test](#)
- [Integration](#)
- [Monitor](#)
- [Function Keys](#)

Essentials

Resource group (move) :	pract	Application Insights :	serverlessapiawt
Location :	Southeast Asia		
Subscription (move) :	Free Trial		
Subscription ID :	5037bd55-a725-439b-8977-6ce99cb02934		
Function app :	serverlessapiawt		
Tags (edit) :	Add tags		

[See more](#)

Total Execution Count Successful Execution Count



Edit Trigger

Save **Discard** **Delete**

Binding Type
HTTP

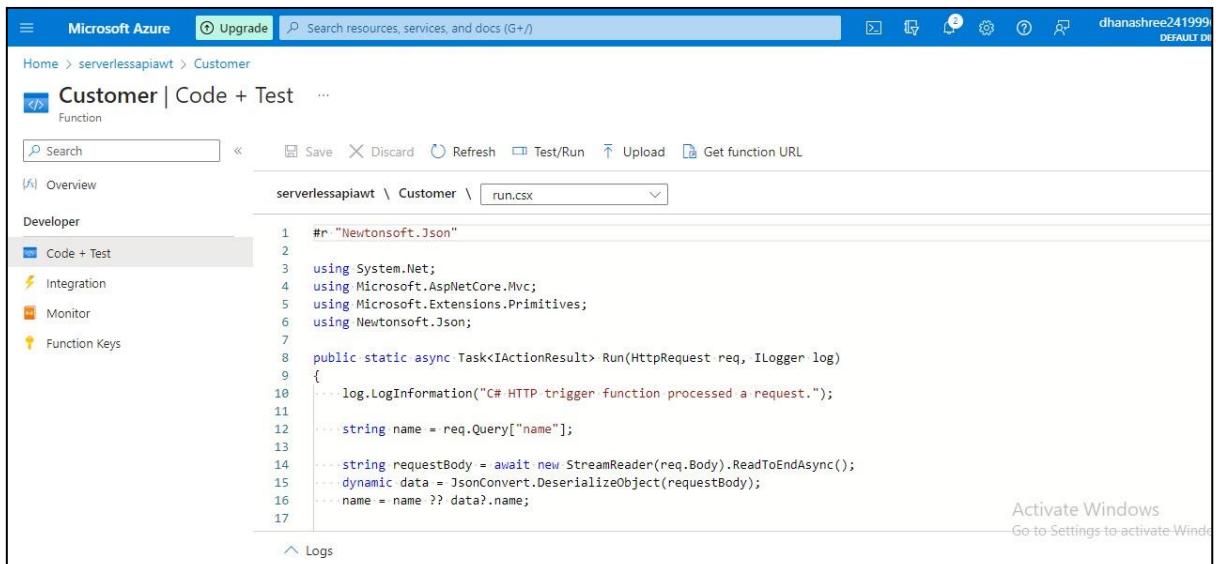
Request parameter name *①
req

Route template ①
customer/{customerName}

Authorization level *①
Anonymous

Go to Settings to activate Windows.
Selected HTTP methods ①

GET POST



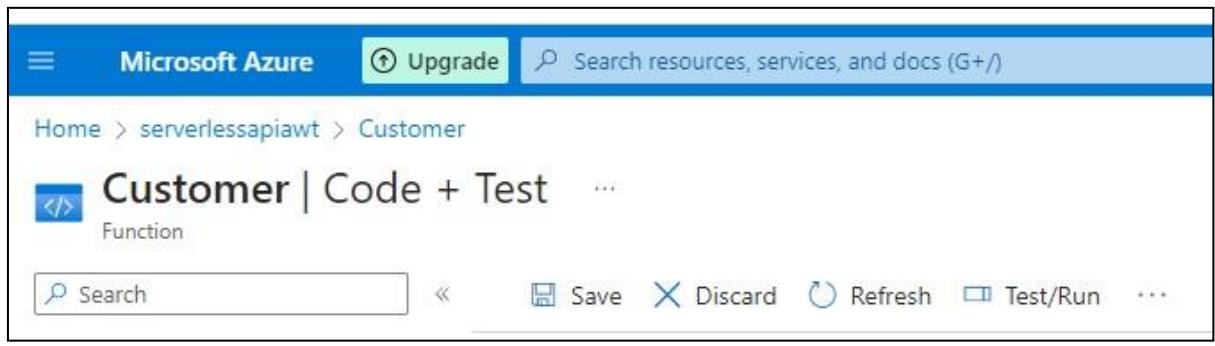
The screenshot shows the Microsoft Azure Functions portal interface. The left sidebar has 'Customer | Code + Test' selected under 'Function'. The main area displays the C# code for the 'run.csx' file:

```

1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7
8 public static async Task<IActionResult> Run(HttpContext req, ILogger log)
9 {
10     log.LogInformation("C# HTTP trigger function processed a request.");
11
12     string name = req.Query["name"];
13
14     string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
15     dynamic data = JsonConvert.DeserializeObject(requestBody);
16     name = name ?? data?.name;
17 }

```

At the bottom right, there is a note: "Activate Windows Go to Settings to activate Windows".



The screenshot shows the Microsoft Azure Functions portal interface. The left sidebar has 'Customer | Code + Test' selected under 'Function'. The main area displays the C# code for the 'run.csx' file:

```

#r "Newtonsoft.Json"

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

public static async Task<IActionResult> Run(HttpContext 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);
}

```

Query

Name	Value
customerName	pooja

+ Add parameter

Headers

+ Add header

Body

```

1  {
2   "body": ""
3 }

```

Activate Windows
Go to Settings to activate Windows.

Run **Close**

Input **Output**

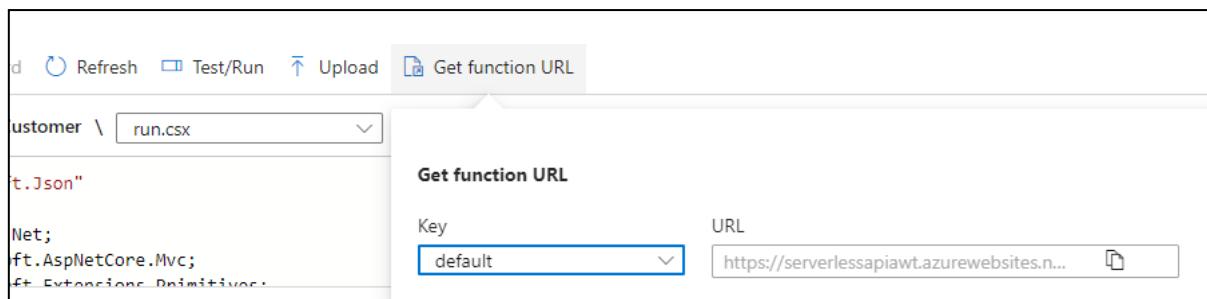
HTTP response code
200 OK

HTTP response content

```
Hello, pooja. You are a customer.
```

Activate Windows
Go to Settings to activate Windows.

Run **Close**



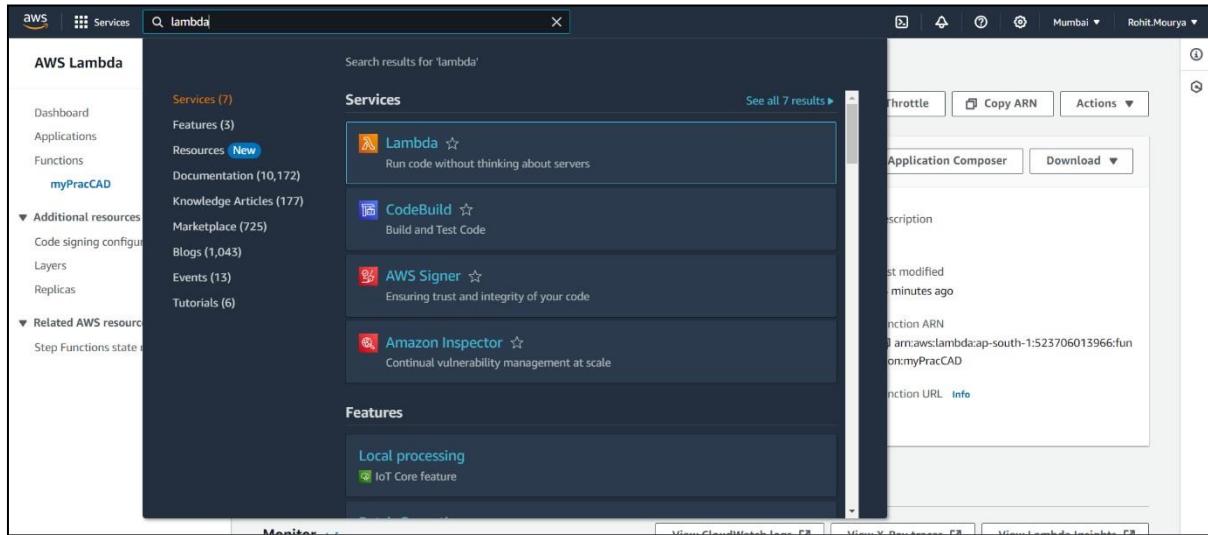
Practical No: 8

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

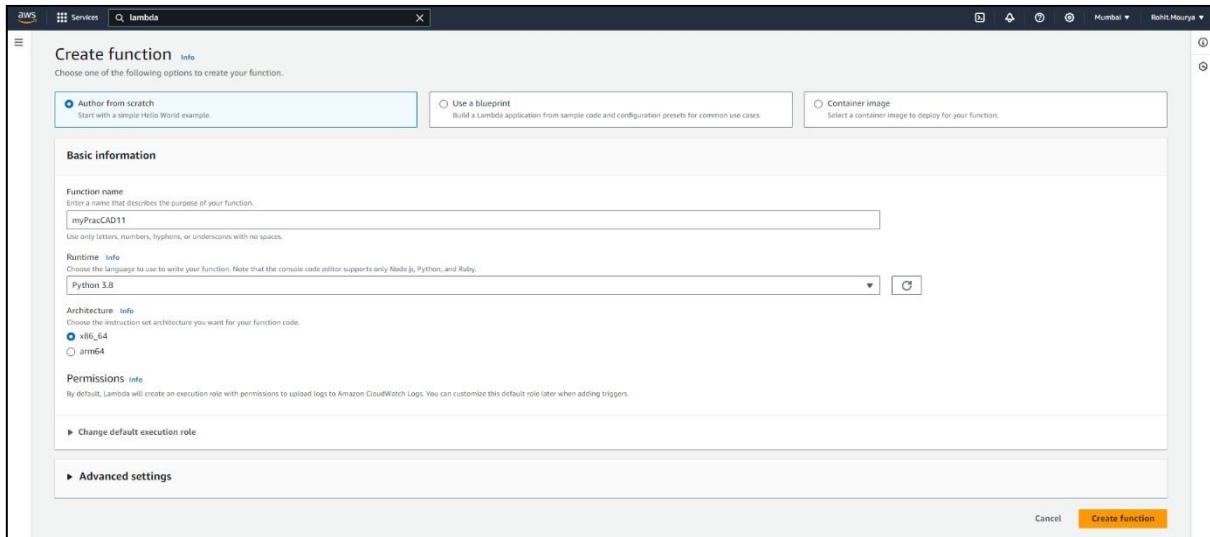
Choose create Function.

Select **Author from scratch**.

Give function name as '**mypracCAD11**'.

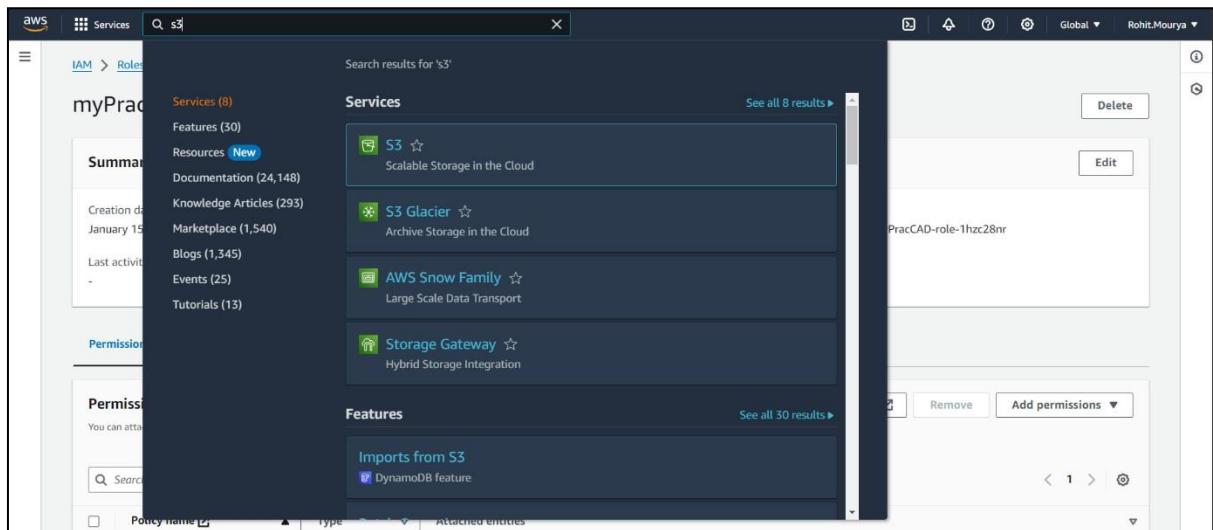
Select **Runtime** as **Python 3.8**.

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**.

The screenshots show the AWS S3 'Create bucket' wizard. The top screenshot is the 'General configuration' step, where you can set the AWS Region (Asia Pacific (Mumbai) ap-south-1), Bucket name (mybucket), and copy settings from existing buckets. The bottom screenshot is the 'Advanced settings' step, which includes options for Tags, Default encryption (Server-side encryption with Amazon S3 managed keys), and Bucket Key (using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS). Both screenshots end with a 'Create bucket' button.

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')
```

The screenshot shows the AWS IAM Roles page. The left sidebar includes options like Dashboard, Access management (selected), Roles, Policies, Identity providers, Account settings, Access reports, and Credential report. The main content area displays a table of roles:

Role name	Trusted entities	Last activity
AWSServiceRoleForSupport	AWS Service: support (Service-Linked)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	Yesterday
myPracCAD-role-1hzc28nr	AWS Service: lambda	31 minutes ago

Below the table, there's a section titled "Roles Anywhere" with three options: "Access AWS from your non AWS workloads" (X.509 Standard), "X.509 Standard", and "Temporary credentials".

Add the "AmazonS3FullAccess" policy to the IAM role.

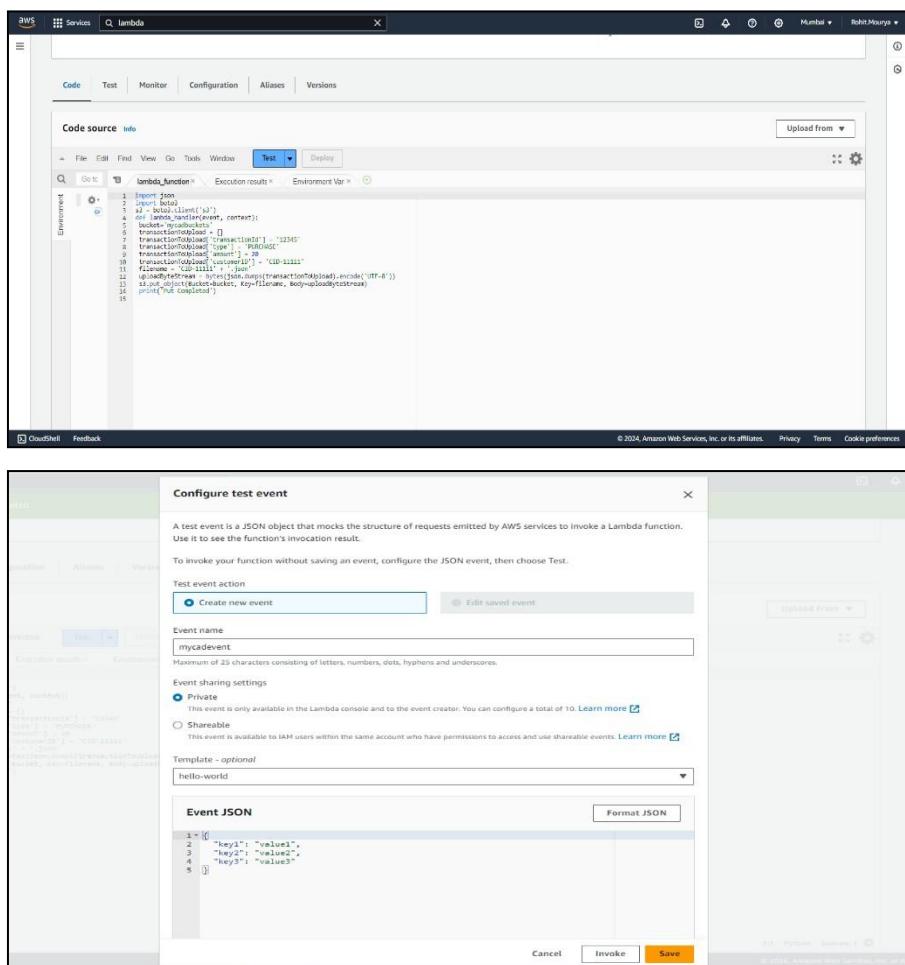
The screenshot shows the details for the "myPracCAD-role-1hzc28nr" role. The left sidebar is identical to the previous screenshot. The main content area has a "Summary" section and a "Permissions" tab. In the "Permissions" tab, under "Permissions policies", there are two entries: "Policy name" (selected) and "AmazonS3FullAccess". A context menu is open over the "AmazonS3FullAccess" entry, with options like "Delete", "Edit", "Simulate", "Remove", "Add permissions", "Attach policies", and "Create inline policy".

Step: 05

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

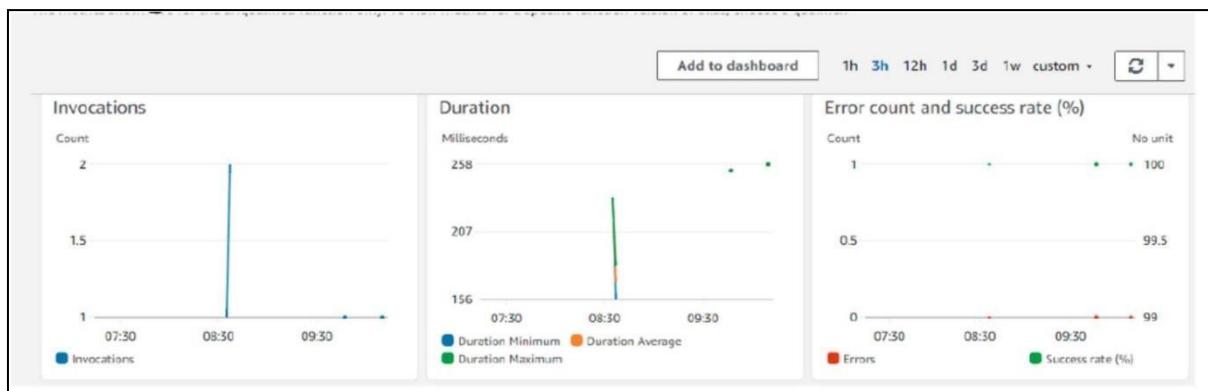
Then, create a **Test case** and **deploy** the code.

Code:



OUTPUT:-

Monitoring Result:



Execution Result:

The interface shows the execution results for the lambda_function:

- Environment:** mycadevent
- Test Event Name:** mycadevent
- Response:** null
- Function Logs:**

```
START RequestId: 72dc62c6-62ba-4b75-8908-262381dca66b Version: $LATEST
Put Completed
END RequestId: 72dc62c6-62ba-4b75-8908-262381dca66b
REPORT RequestId: 72dc62c6-62ba-4b75-8908-262381dca66b Duration: 248.15 ms Billed Duration: 249 ms Memory Size: 128 MB Max Memory Used: 74 MB Init Duration: 422.45 ms
RequestID
72dc62c6-62ba-4b75-8908-262381dca66b
```

Practical No: 9

Build AWS Lambda with AWS API Gateway

Steps to be followed:-

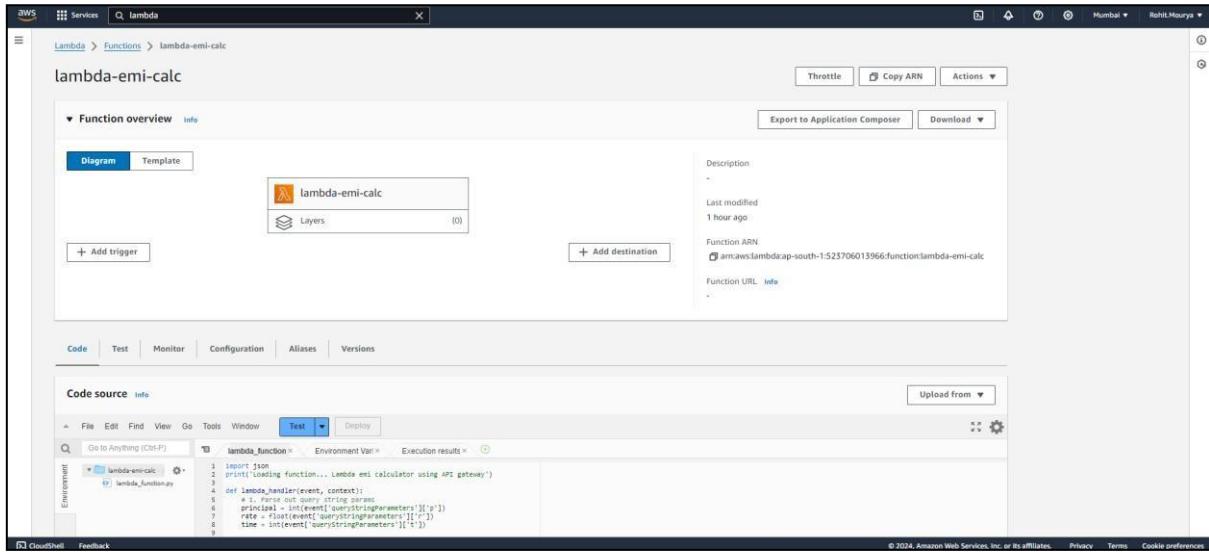
Step: 01

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
    return {
        'statusCode': 200,
        'body': json.dumps(transactionResponse)
    }
```

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

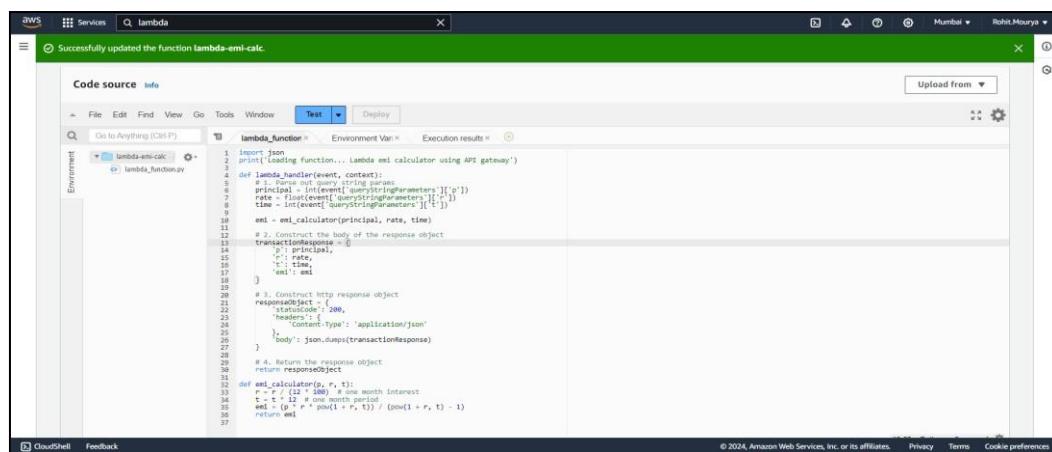
t = t * 12 # one month period

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

return emi

```

Deploy the lambda function



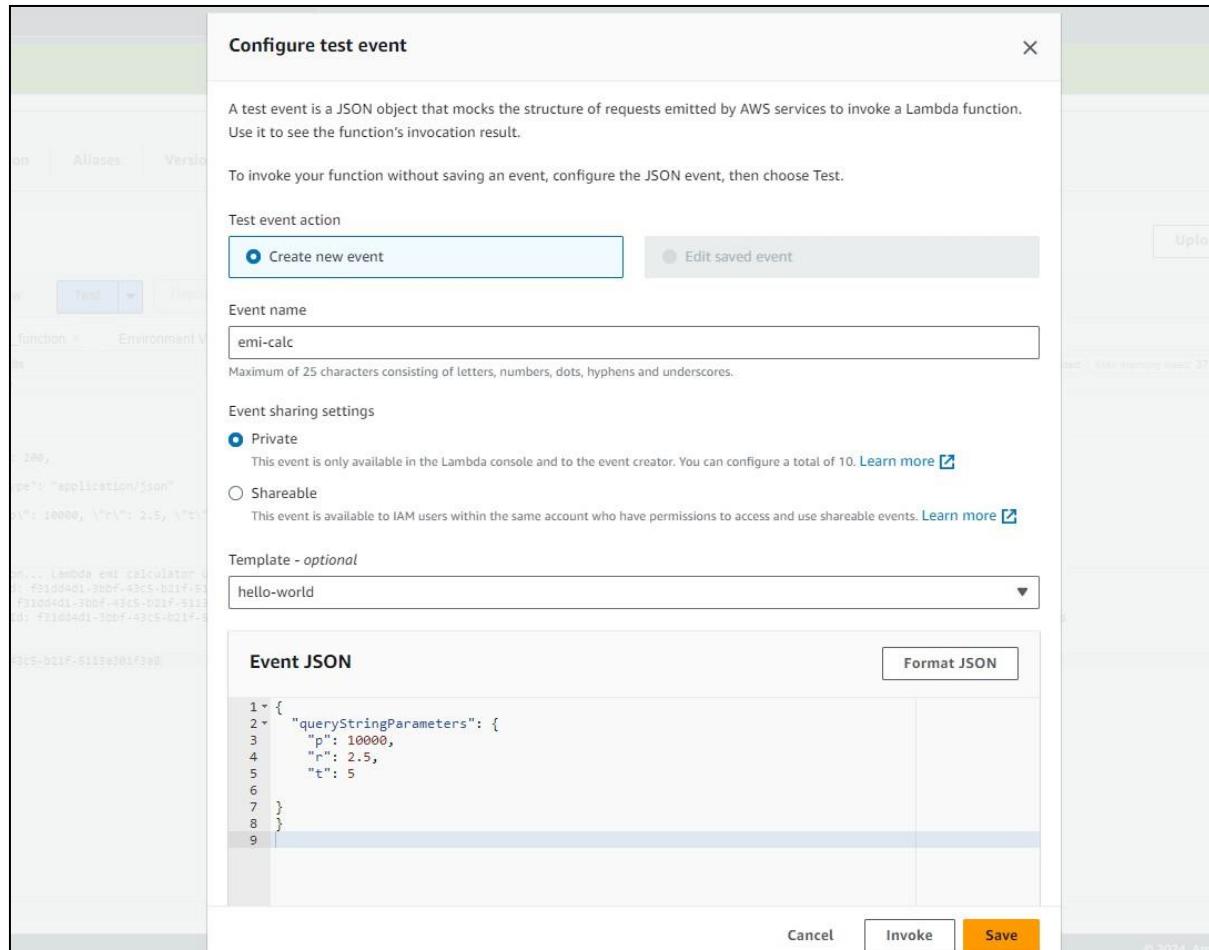
Step: 03

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

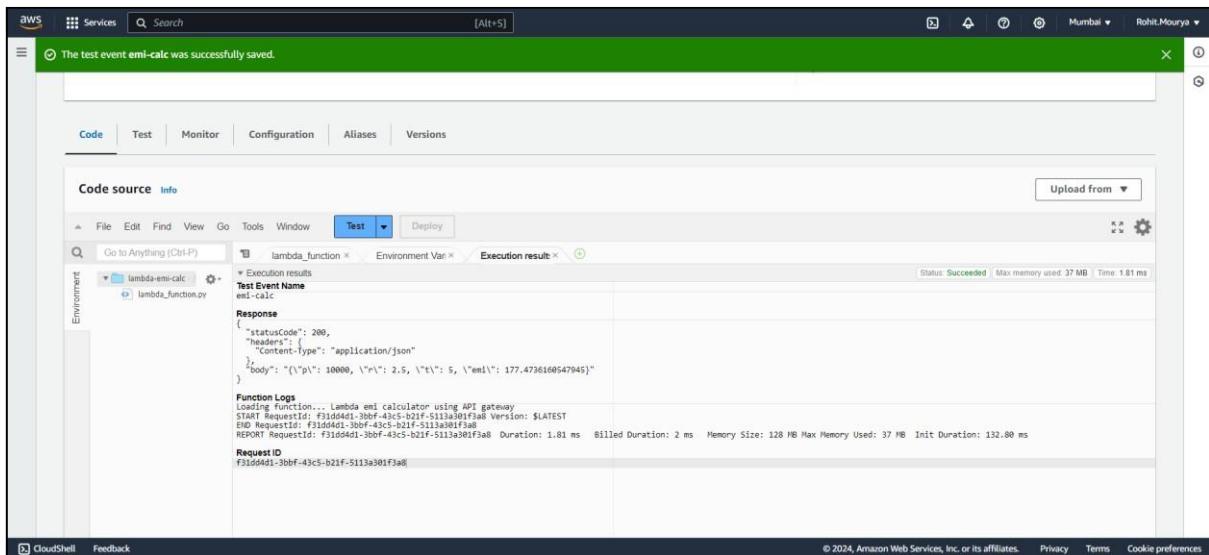
Code:

```
{
"queryStringParameters": {
"p": 10000,
"r": 2.5,
}
```

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

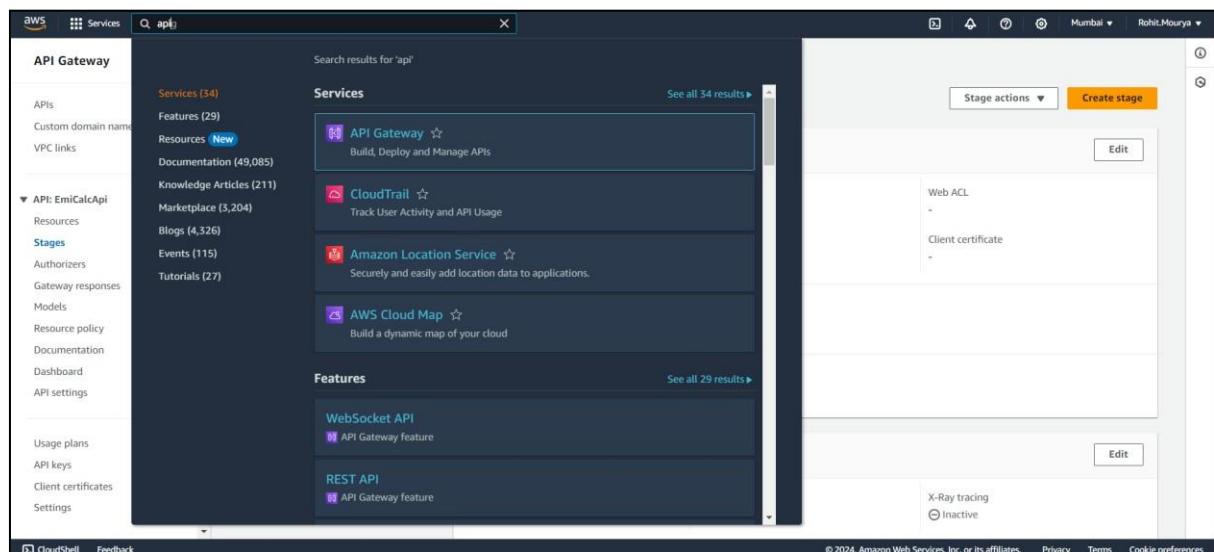


Test the code and get output without any errors.



Step: 04

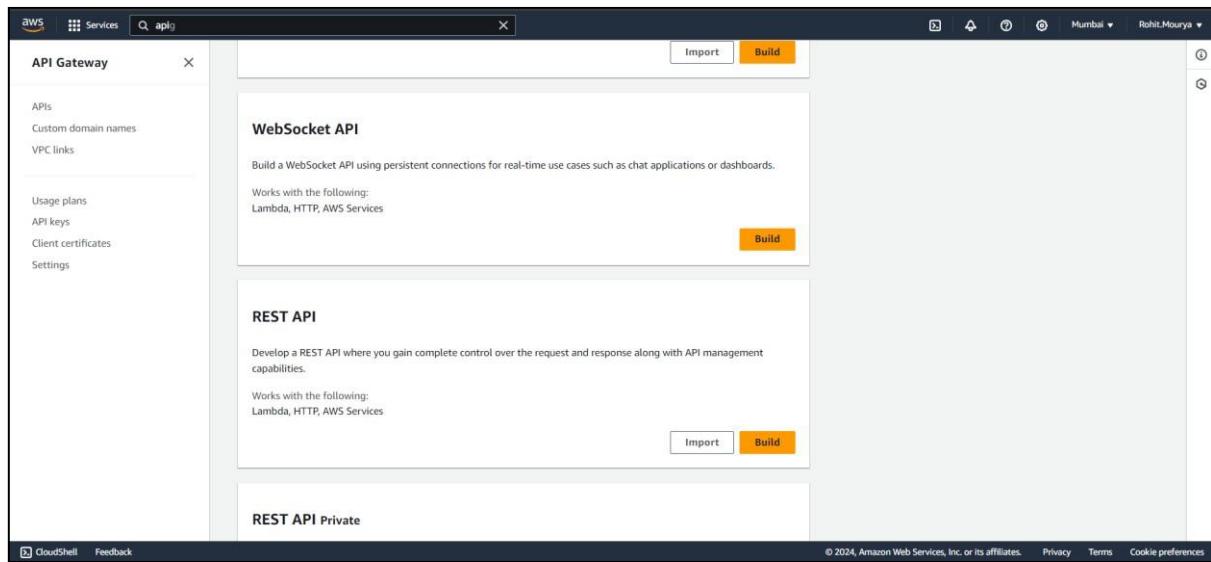
Go above and search for API gateway on the service bar and open it.



Click on create API gateway

Choose Rest API

Click on import



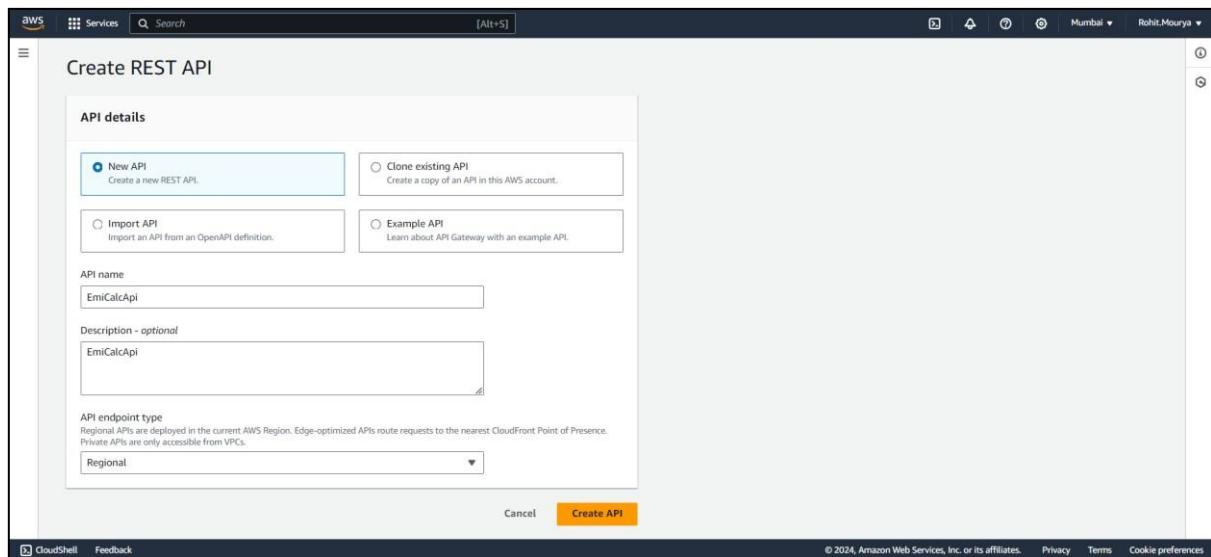
Create new API

API name = EmiCalcApi

Description = EmiCalcApi

Endpoint type = Regional

Click on create tab



Step: 05

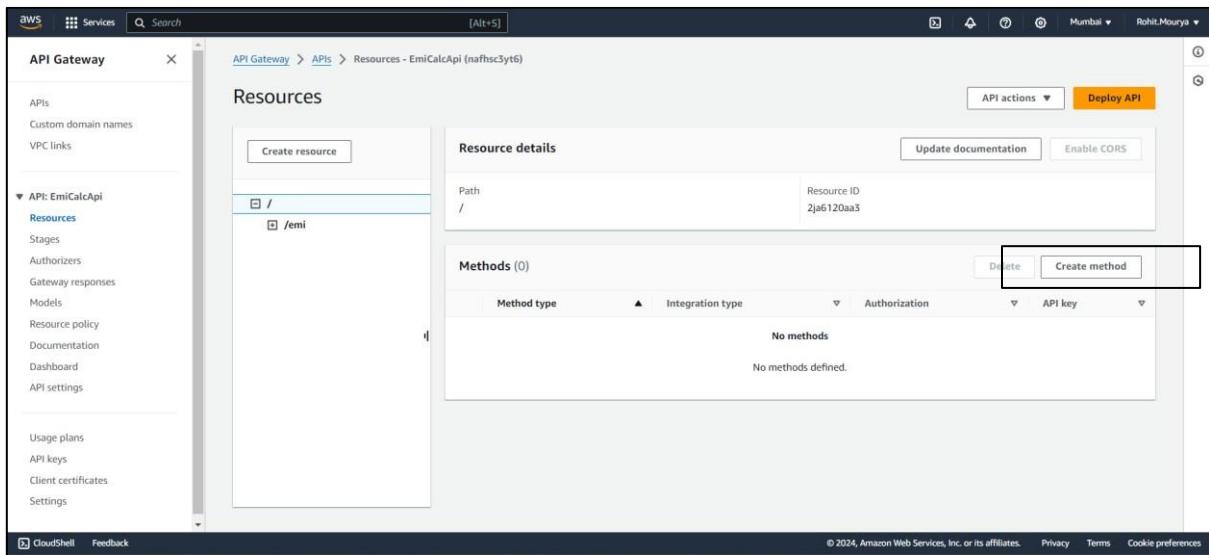
Next step clicks on actions then create resource

The screenshot shows the AWS API Gateway interface. On the left, a sidebar for the 'EmiCalcApi' API is open, showing various settings like Stages, Authorizers, and Models. The main area is titled 'Resources' and shows a hierarchical tree structure. At the top of the tree, there is a blue-bordered box containing a white 'Create resource' button. To the right of the tree, there is a 'Resource details' panel with the path '/'. Below it is a 'Methods' section with a table header for Method type, Integration type, Authorization, and API key. The table body contains the message 'No methods' and 'No methods defined.' At the bottom right of the screen, there are footer links for 'CloudShell', 'Feedback', and copyright information.

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

This screenshot shows the 'Create resource' dialog box. It has a title 'Create resource' and a 'Resource details' section. In the 'Resource path' field, the value '/' is entered. In the 'Resource name' field, the value 'emi' is entered. Below these fields are two info sections: 'Proxy resource info' (which is checked) and 'CORS (Cross Origin Resource Sharing) info' (which is unchecked). At the bottom of the dialog are 'Cancel' and 'Create resource' buttons. The background of the dialog is light gray, and the overall interface follows the AWS color scheme.

Now click on create method.



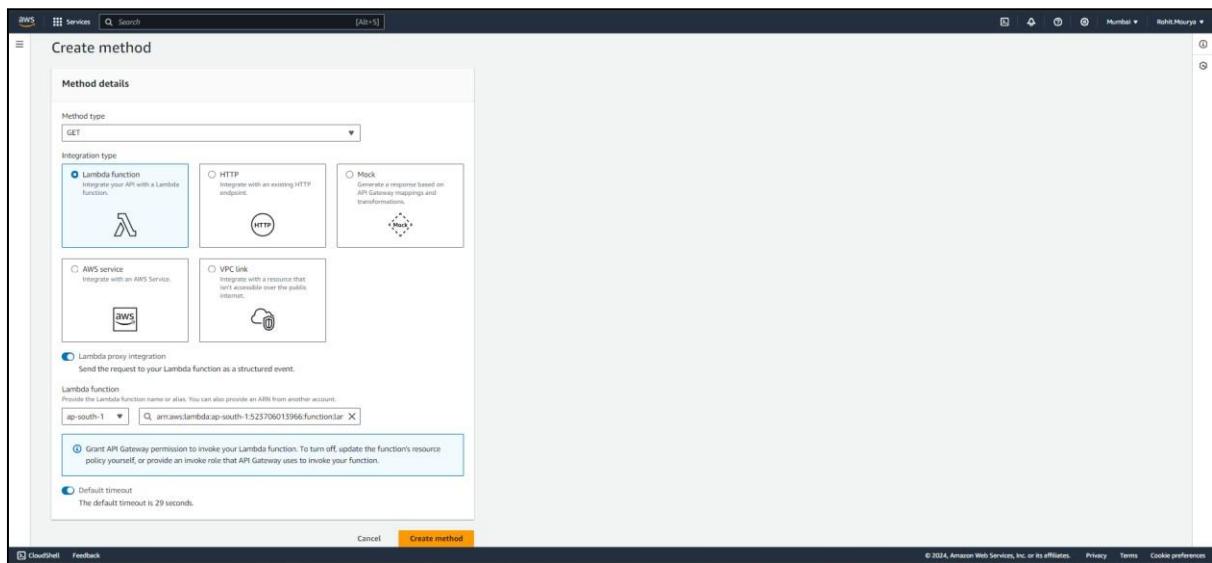
Select Method as GET.

Integration type as Lambda Function.

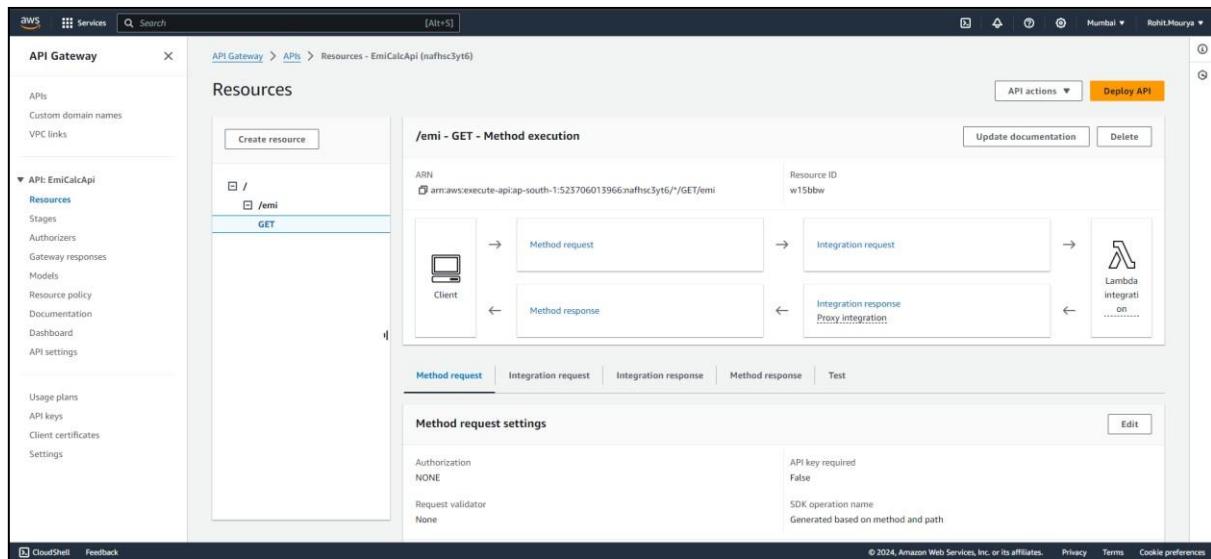
Enable the Lambda proxy integration.

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

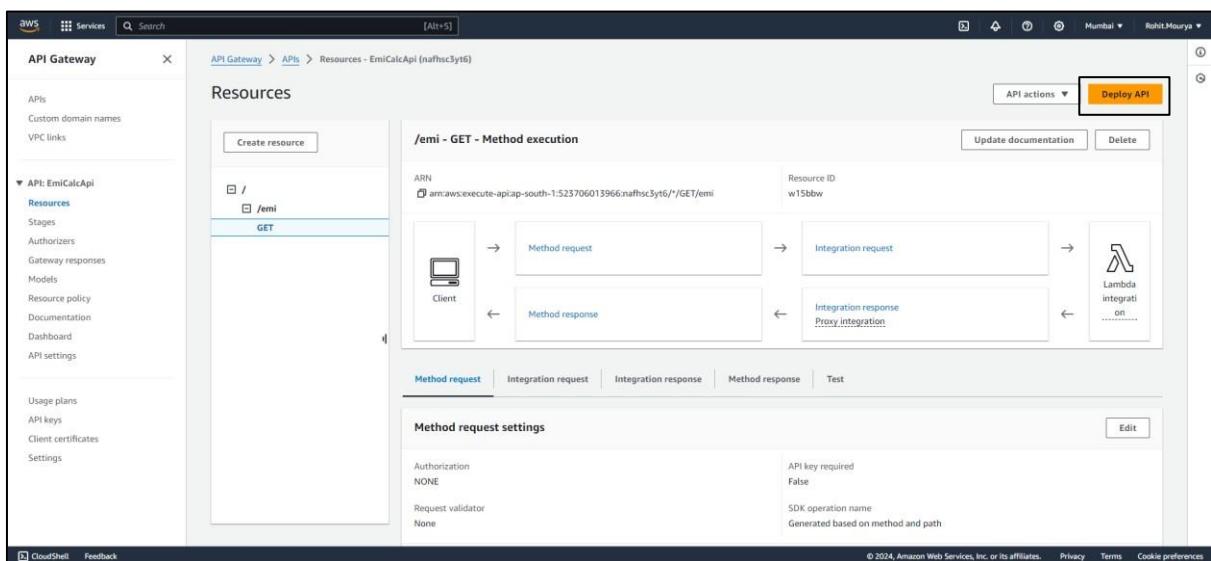
And click on create method.



The GET method will look below.

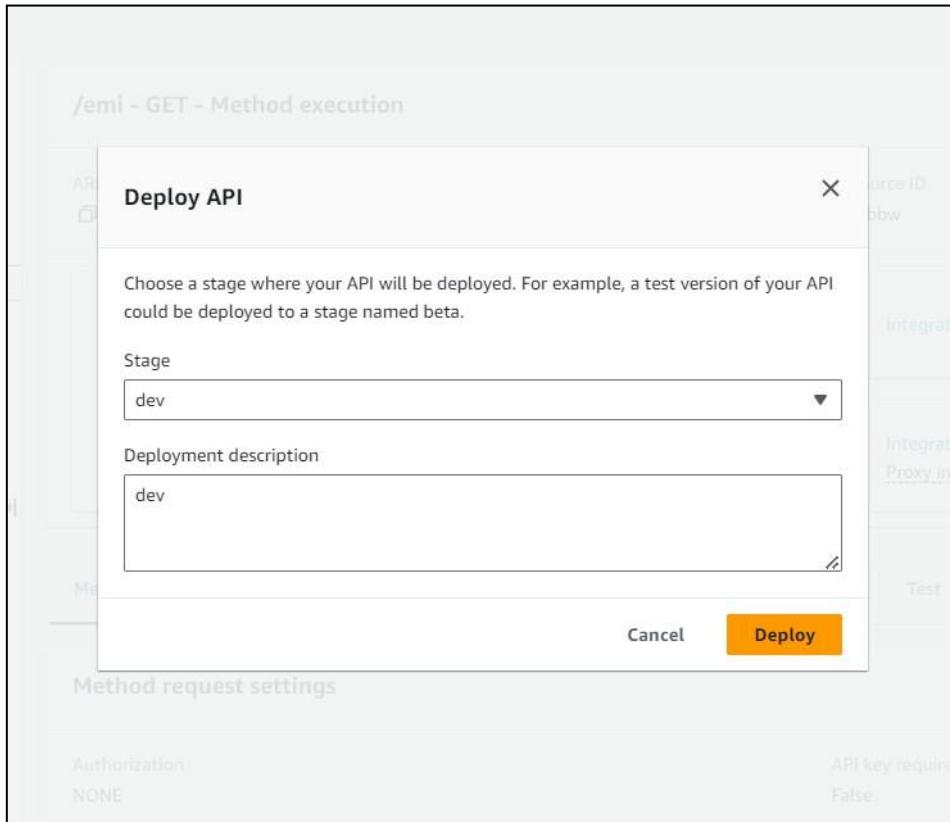


Now deploy the method by click on **Deploy API** button



Enter stage name as **dev**.

Deployment description as **dev**.



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

Stage name	Rate info	Web ACL
dev	5.8	-
API cache	Burst info	Client certificate
Inactive	10	-
Invoke URL https://nafhsc3yt6.execute-api.ap-south-1.amazonaws.com/dev		
Active deployment tz5d03 on January 17, 2024, 11:02 (UTC+05:30)		

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

The screenshot shows the AWS API Gateway interface. On the left, there's a sidebar with options like APIs, Custom domain names, VPC links, and a expanded section for 'API: EmiCalcApi' which includes Resources, Stages, Authorizers, Gateway responses, Models, Resource policy, Documentation, Dashboard, and API settings. Below that are Usage plans and API keys. At the bottom of the sidebar are CloudShell and Feedback buttons. The main area is titled 'Stages' and shows a table with one row for 'dev'. The 'Stage details' section on the right shows the stage name 'dev', rate of 5.8, burst of 10, and an invoke URL: <https://nafhsc3yt6.execute-api.ap-south-1.amazonaws.com/dev>. It also indicates an active deployment. There are 'Edit' buttons in both the stage list and the details panel.

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

The screenshot shows the 'Edit Stage' dialog box. It has fields for 'Stage description - optional' (containing 'dev'), 'Cache settings' (with 'API cache' checked), 'Throttling settings' (with 'Throttling' checked and a note about limiting the rate of calls per second), and 'Firewall and certificate settings' (with 'Web application Firewall (AWS WAF)' set to 'None' and 'Client certificate' set to 'None'). At the bottom are 'Cancel' and 'Save' buttons.

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

The screenshot shows the AWS API Gateway interface. On the left, a sidebar lists 'APIs', 'Custom domain names', 'VPC links', and 'API: EmiCalcApi' (which is expanded). Under 'API: EmiCalcApi', there are 'Resources', 'Stages' (which is selected), 'Authorizers', 'Gateway responses', 'Models', 'Resource policy', 'Documentation', 'Dashboard', and 'API settings'. Below these are 'Usage plans' and 'API keys'. At the bottom of the sidebar are 'CloudShell' and 'Feedback' buttons. The main content area is titled 'Stages' and shows a table with one row for 'dev'. The 'Stage details' section includes fields for 'Stage name' (dev), 'Rate Info' (5.8), 'Web ACL' (-), 'API cache' (Inactive), 'Burst Info' (10), and 'Client certificate' (-). It also displays the 'Invoke URL' as <https://nafhsc3yt6.execute-api.ap-south-1.amazonaws.com/dev>. A note below says 'Active deployment: tz5d03 on January 17, 2024, 11:02 (UTC+05:30)'. At the bottom right of the main area are 'Logs and tracing' and 'Edit' buttons. The footer of the page includes the text '© 2024, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

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

OUTPUT:-

The screenshot shows a web browser window with the address bar containing the URL <https://nafhsc3yt6.execute-api.ap-south-1.amazonaws.com/dev/emi?p=10000&r=5.8&t=10>. The browser toolbar includes icons for back, forward, search, and various AWS services like AWS Skill Builder, AWS re/Start canvas, and AWS management console. The main content area displays the JSON response: {"p": 10000, "r": 5.8, "t": 10, "emi": 110.0188101041876}.