

INDEX

SR.NO	Date	Description	Page Number	Signature
1		Develop an ASP.NET Core MVC based Stateless Web App.	5	
2		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	
3		Create an AKS cluster a. From the portal b. With Azure CLI	37	
4		a. Create an API management service b. Create an API gateway service	44	
5		Demonstrate a. Securing APIs with Azure Active Directory b. AWS API Gateway Authorizer	78	
6		Create a serverless API using Azure functions	95	
7		Create an AWS Lambda function	100	
8		Build AWS Lambda with AWS API gateway	108	

PRACTICAL 1

AIM: Develop an ASP.NET Core MVC based Stateless Web App.

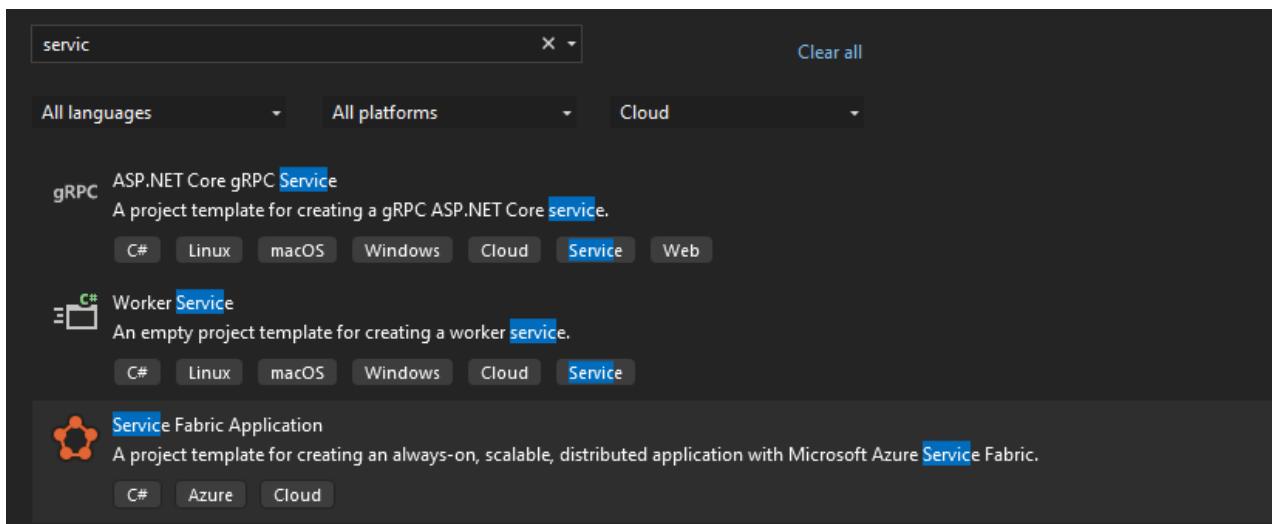
CODE:

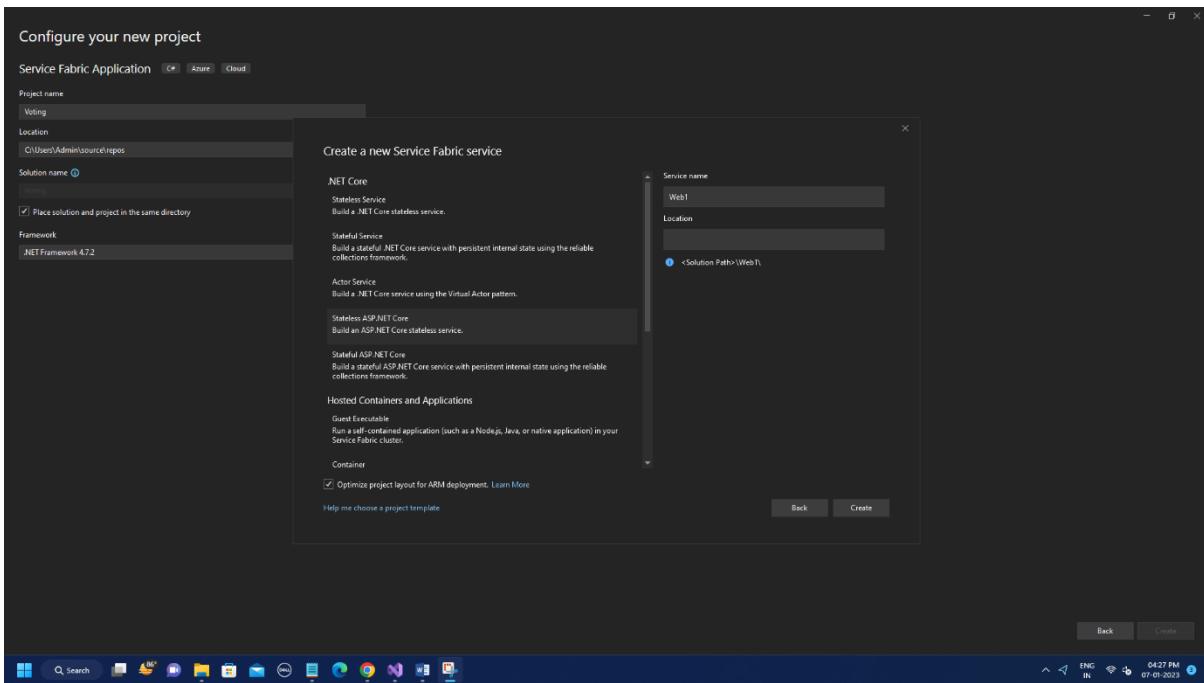
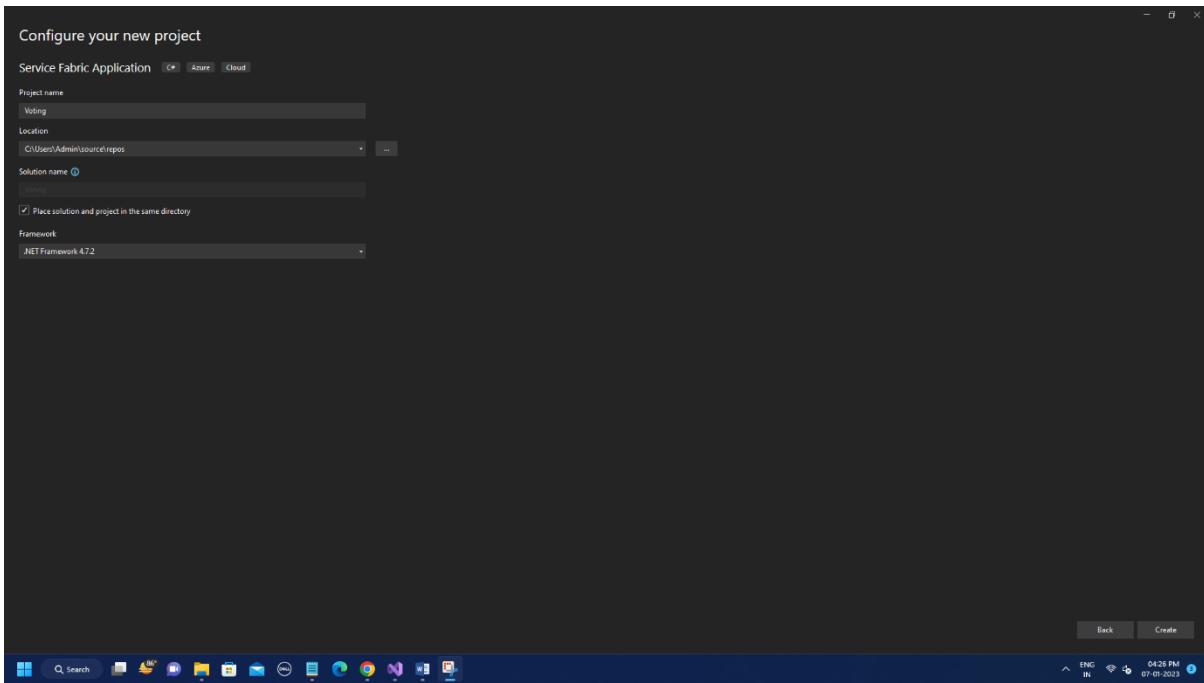
Step 1:

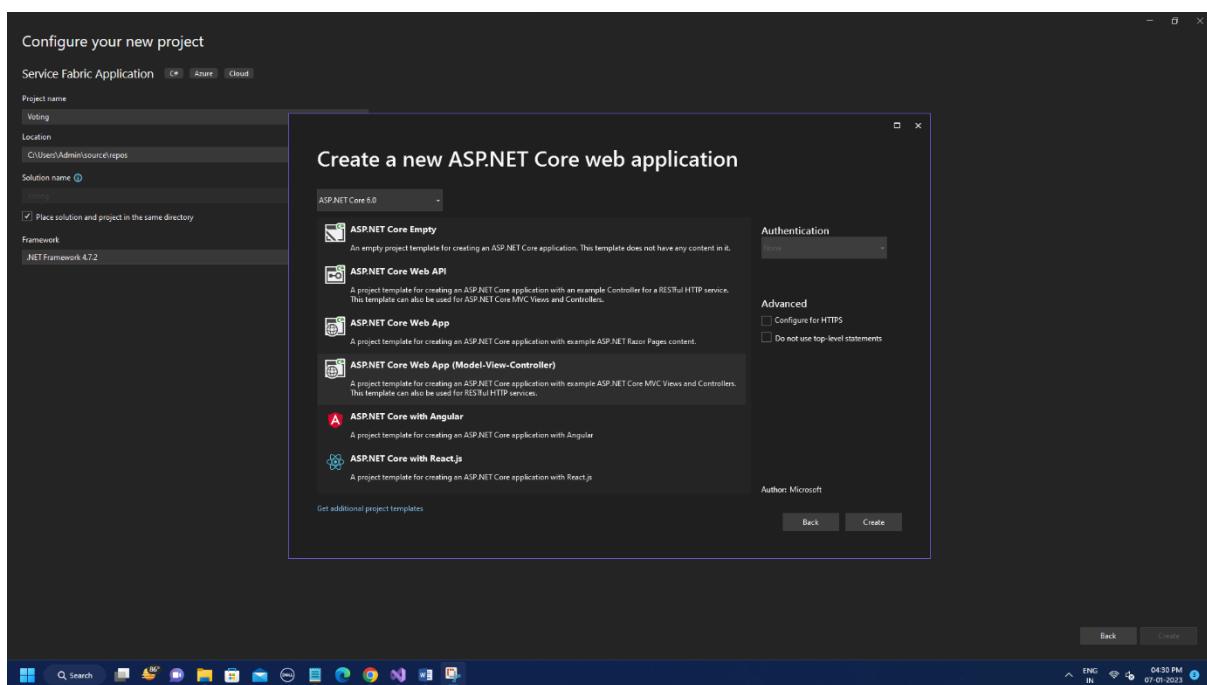
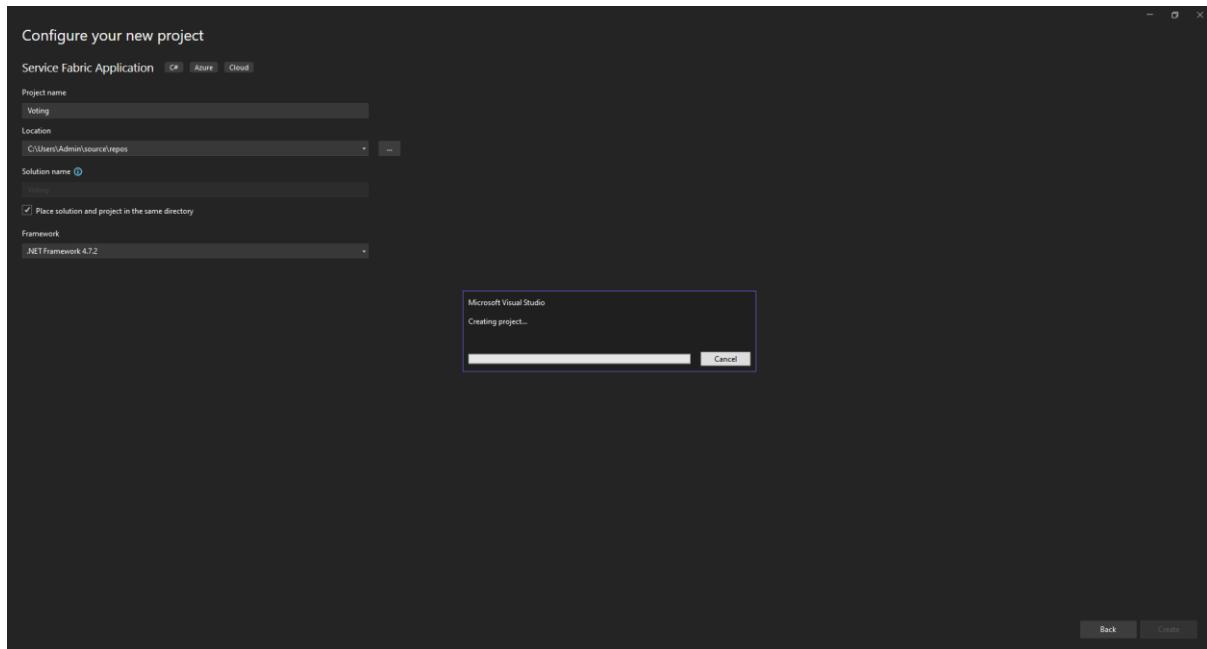
- Create Project Launch Visual Studio as an administrator.
- Create a project with File > New > Project
- In the New Project dialog, choose Cloud > Service Fabric Application.
- Name the application Voting and click OK.
- On the New Service Fabric Service page, choose Stateless ASP.NET Core, name your service VotingWeb, then click OK.

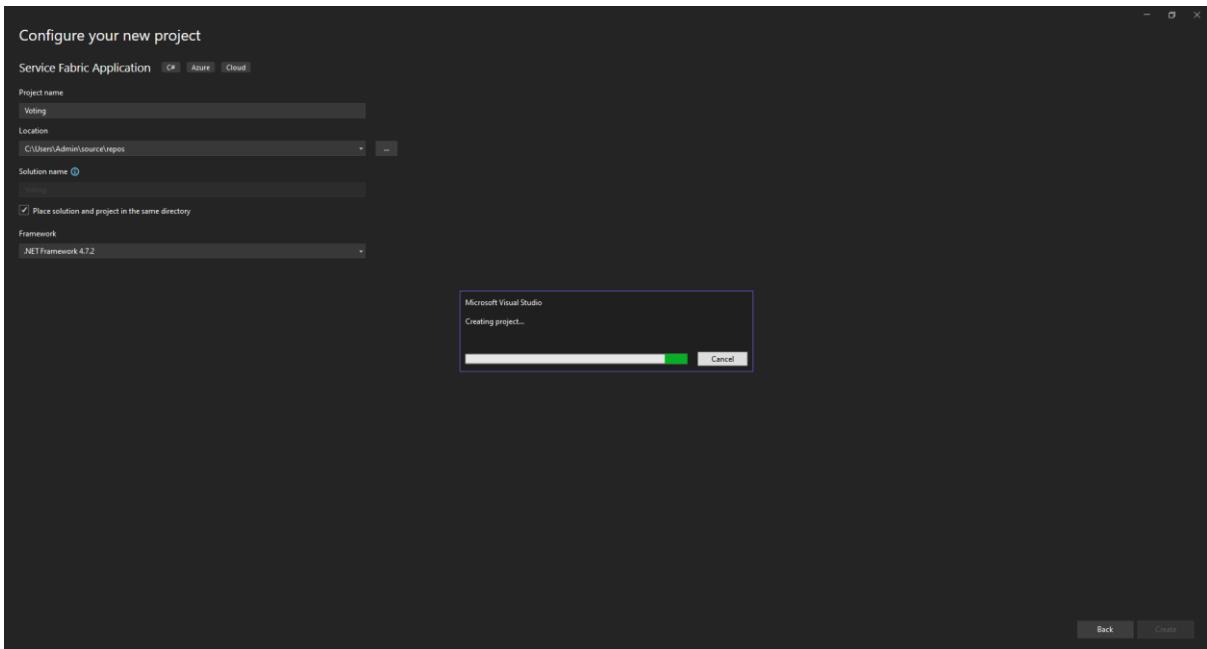
Step 2: The next page provides a set of ASP.NET Core project templates. For this practical, choose Web Application (Model-View-Controller), then click OK.

Step 3: Visual Studio creates an application and a service project and displays them in Solution Explorer.







**Step 4: 2.1 Update the site.js file Open wwwroot/js/site.js.**

Replace its contents with the following JavaScript used by the Home views, then save your changes.

```
var app = angular.module('VotingApp', ['ui.bootstrap']); app.run(function () { });

app.controller('VotingAppController', ['$rootScope', '$scope', '$http', '$timeout', function ($rootScope,
$scope, $http, $timeout) {

    $scope.refresh = function () {
        $http.get('api/Votes?c=' + new Date().getTime())
            .then(function (data, status) {
                $scope.votes = data;
            }, function (data, status) {
                $scope.votes = undefined;
            });
    };

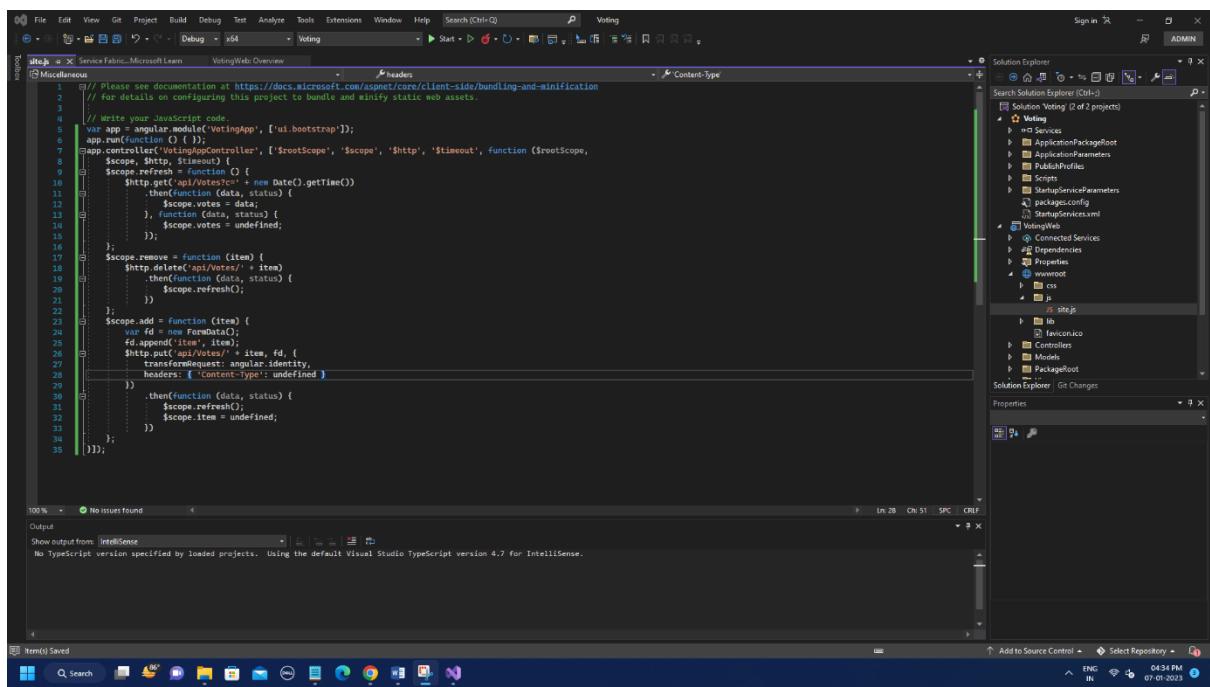
    $scope.remove = function (item) {
```

```

$http.delete('api/Votes/' + item)
    .then(function (data, status) {
        $scope.refresh();
    })
};

$scope.add = function (item) {
    var fd = new FormData(); fd.append('item', item);
    $http.put('api/Votes/' + item, fd, {
        transformRequest: angular.identity, headers: { 'Content-Type': undefined }
    })
    .then(function (data, status) {
        $scope.refresh();
        $scope.item = undefined;
    })
    ;
};

});
```



Step 5: 2.2 Update the Index.cshtml file Open Views/Home/Index.cshtml, the view specific to the Home controller.

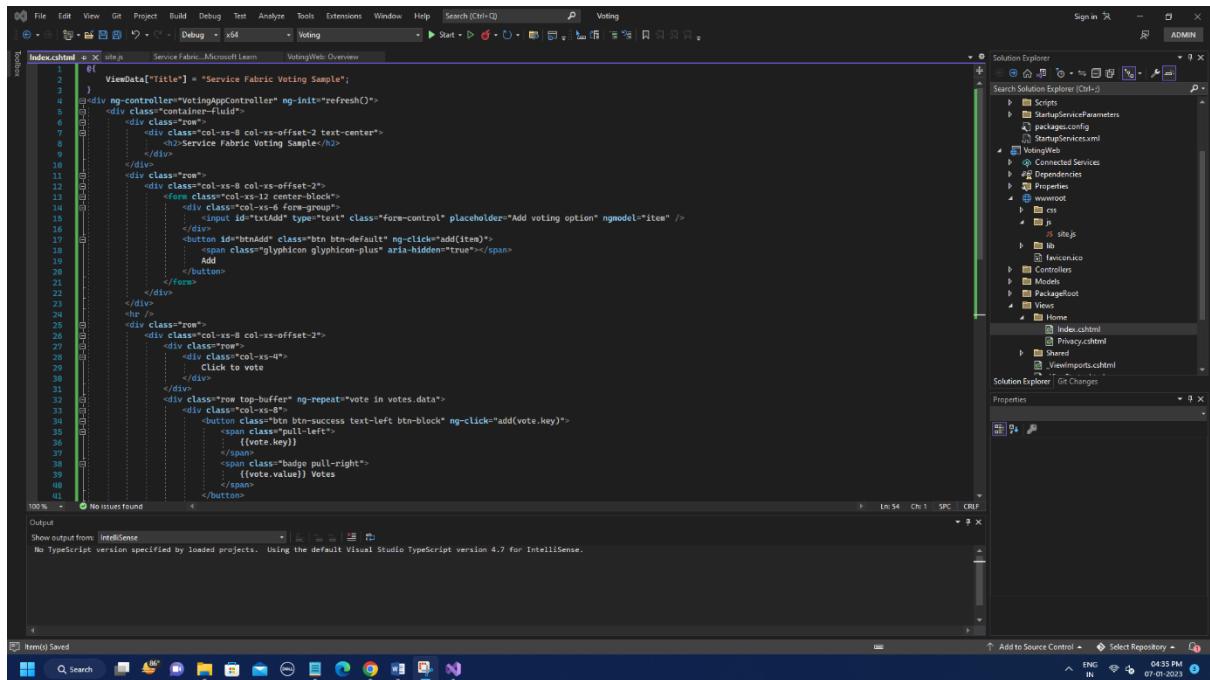
Replace its contents with the following, then save your changes.

```
@{  
    ViewData["Title"] = "Service Fabric Voting Sample";  
}  


<div class="container-fluid">  
        <div class="row">  
            <div class="col-xs-8 col-xs-offset-2 text-center">  
                <h2>Service Fabric Voting Sample</h2>  
            </div>  
        </div>  
        <div class="row">  
            <div class="col-xs-8 col-xs-offset-2">  
                <form class="col-xs-12 center-block">  
                    <div class="col-xs-6 form-group">  
                        <input id="txtAdd" type="text" class="form-control" placeholder="Add voting  
option" ng-model="item" />  
                    </div>  
                    <button id="btnAdd" class="btn btn-default" ng-click="add(item)">  
                        <span class="glyphicon glyphicon-plus" aria-hidden="true"></span> Add  
                    </button>  
                </form>  
            </div>  
        </div>  
        <hr />  
        <div class="row">  
            <div class="col-xs-8 col-xs-offset-2">  
                <div class="row">  
                    <div class="col-xs-4"> Click to vote </div>


```

```
</div>
</div>
<div class="row top-buffer" ng-repeat="vote in votes.data">
  <div class="col-xs-8">
    <button class="btn btn-success text-left btn-block" ng-click="add(vote.key)">
      <span class="pull-left">
        {{ vote.key }}
      </span>
      <span class="badge pull-right">
        {{ vote.value }} Votes
      </span>
    </button>
  </div>
  <div class="col-xs-4">
    <button class="btn btn-danger pull-right btn-block" ng-click="remove(vote.key)">
      <span class="glyphicon glyphicon-remove" aria-hidden="true"></span>
      Remove
    </button>
  </div>
</div>
</div>
</div>
</div>
```



Step 6: Update the _Layout.cshtml file

Open Views/Shared/_Layout.cshtml, the default layout for the ASP.NET app. Replace its contents with the following, then save your changes.

```
<!DOCTYPE html>
<html ng-app="VotingApp" xmlns:ng="http://angularjs.org">
    <head>
        <meta charset="utf-8"/>
        <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
        <title>@ViewData["Title"]</title>

        <link href("~/lib/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet"/>
        <link href("~/css/site.css" rel="stylesheet"/>
    </head>
    <body>
        <div class="container body-content"> @RenderBody()
        </div>
        <script src "~/lib/jquery/dist/jquery.js"></script>
        <script src "~/lib/bootstrap/dist/js/bootstrap.js"></script>
```

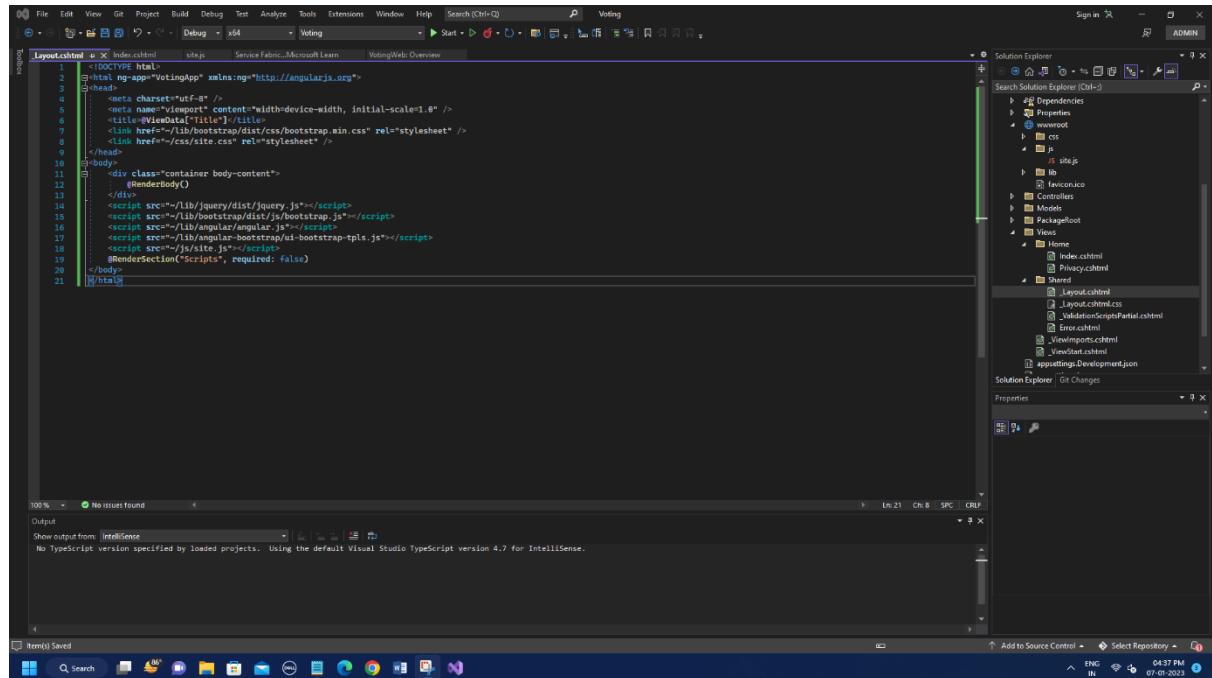
```

<script src="~/lib/angular/angular.js"></script>
<script src="~/lib/angular-bootstrap/ui-bootstrap-tpls.js"></script>
<script src="~/js/site.js"></script>
@RenderSection("Scripts", required: false)

</body>

</html>

```



Step 7: Update the VotingWeb.cs file

Open the VotingWeb.cs file, which creates the ASP.NET Core WebHost inside the stateless service using the WebListener web server. Replace the content with the following code, then save your changes.

```

namespace VotingWeb

{
    using System;
    using System.Collections.Generic;
    using System.Fabric;
    using System.IO;
    using System.Net.Http;

```

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.ServiceFabric.Services.Communication.AspNetCore;
using Microsoft.ServiceFabric.Services.Communication.Runtime;
using Microsoft.ServiceFabric.Services.Runtime;
internal sealed class VotingWeb : StatelessService
{
    public VotingWeb(StatelessServiceContext context)
        : base(context)
    {
    }
    protected override IEnumerable<ServiceInstanceListener> CreateServiceInstanceListeners()
    {
        return new ServiceInstanceListener[]
        {
            new ServiceInstanceListener(serviceContext =>
                new KestrelCommunicationListener(serviceContext, "ServiceEndpoint",
                    (url, listener) => {
                        ServiceEventSource.Current.ServiceMessage(serviceContext, $"Starting
Kestrel on {url}");
                        return new WebHostBuilder()
                            .UseKestrel()
                            .ConfigureServices(services => services
                                .AddSingleton<HttpClient>(new HttpClient())
                                .AddSingleton<FabricClient>(new FabricClient())
                                .AddSingleton<StatelessServiceContext>(serviceContext)
                            )
                            .UseContentRoot(Directory.GetCurrentDirectory())
                            .UseStartup<Startup>()
                            .UseServiceFabricIntegration(listener,
                                ServiceFabricIntegrationOptions.None)
                    })
        };
    }
}
```

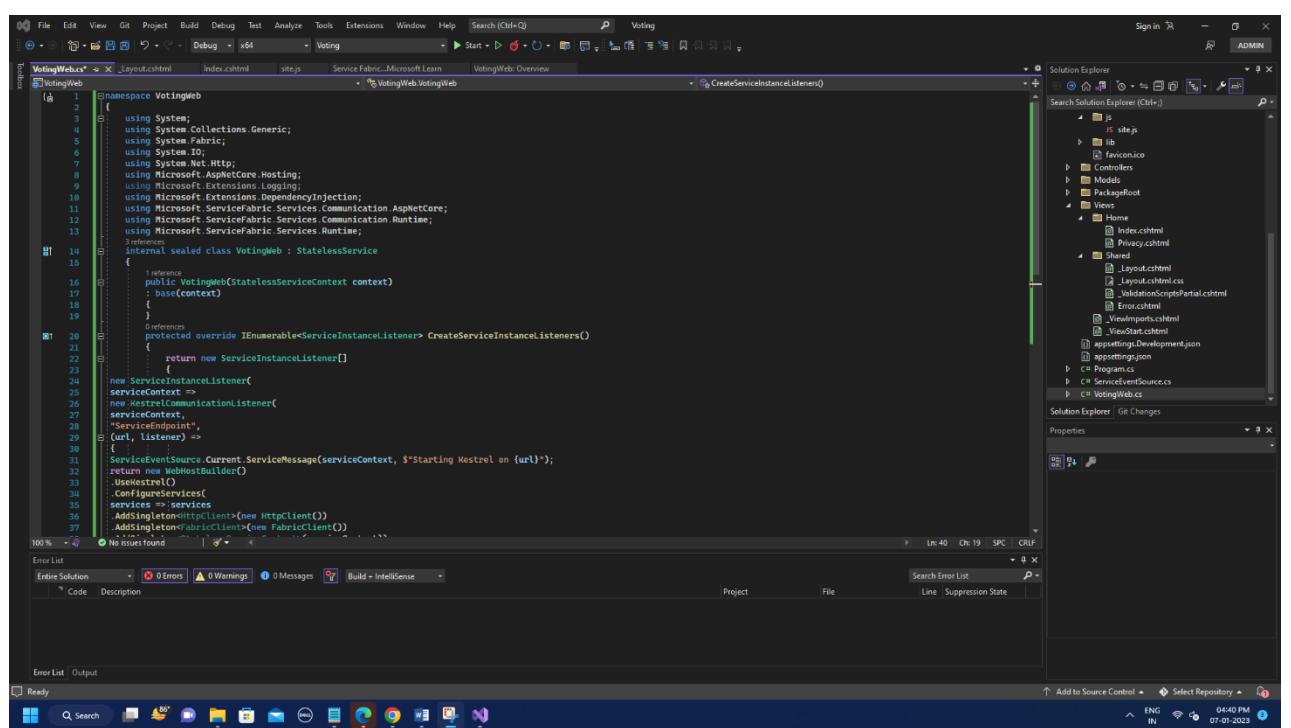
```

        .UseUrls(url)
        .Build();
    })
};

}

internal static Uri GetVotingDataServiceName(ServiceContext context)
{
    return
        new Uri($"{context.CodePackageActivationContext.ApplicationName}/VotingData");
}
}
}

```



Step 8: Add the VotesController.cs file

Add a controller, which defines voting actions. Right-click on the Controllers folder, then select Add->New item->Visual C#->ASP.NET Core->Class. Name the file `VotesController.cs`, then click Add.

Replace the VotesController.cs file contents with the following, then save your changes this file is modified to read and write voting data from the back-end service.

```
namespace VotingWeb.Controllers {  
    using System;  
    using System.Collections.Generic; using System.Fabric;  
    using System.Fabric.Query; using System.Linq;  
    using System.Net.Http;  
    using System.Net.Http.Headers; using System.Text;  
    using System.Threading.Tasks; using Microsoft.AspNetCore.Mvc; using Newtonsoft.Json;  
  
    [Produces("application/json")][Route("api/[controller]")]  
    public class VotesController : Controller  
    {  
        private readonly HttpClient httpClient; private readonly FabricClient fabricClient; private  
        readonly string reverseProxyBaseUri;  
        private readonly StatelessServiceContext serviceContext;  
  
        public VotesController(HttpClient httpClient, StatelessServiceContext context, FabricClient  
        fabricClient)  
        {  
            this.fabricClient = fabricClient; this.httpClient = httpClient; this.serviceContext =  
            context;  
            this.reverseProxyBaseUri  
            = Environment.GetEnvironmentVariable("ReverseProxyBaseUri");  
        }  
  
        // GET: api/Votes [HttpGet("")]  
        public async Task < IActionResult > Get()  
        {  
            Uri serviceName = VotingWeb.GetVotingDataServiceName(this.serviceContext); Uri  
            proxyAddress = this.GetProxyAddress(serviceName);  
        }  
}
```

```

ServicePartitionList partitions = await
this.fabricClient.QueryManager.GetPartitionListAsync(serviceName);

List < KeyValuePair < string, int >> result = new List < KeyValuePair < string, int >>
(); foreach (Partition partition in partitions)

{

string proxyUrl =

"${proxyAddress}/api/VoteData?PartitionKey={({Int64RangePartitionInformation)
partition.PartitionInformation).LowKey }&PartitionKind=Int64Range";

using (HttpResponseMessage response = await this.httpClient.GetAsync(proxyUrl))
{
    if (response.StatusCode != System.Net.HttpStatusCode.OK) {
        continue;
    }

    result.AddRange(JsonConvert.DeserializeObject < List < KeyValuePair < string,
int >>> (await response.Content.ReadAsStringAsync()));
}
}

return this.Json(result);
}

// PUT: api/Votes/name [HttpPut("{name}")]
public async Task < IActionResult > Put(string name)
{
    Uri serviceName = VotingWeb.GetVotingDataServiceName(this.serviceContext); Uri
proxyAddress = this.GetProxyAddress(serviceName);
    long partitionKey = this.GetPartitionKey(name); string proxyUrl =

"${proxyAddress}/api/VoteData/{name}?PartitionKey={partitionKey}&PartitionKind=Int64R
ange";
}

```

```
StringContent putContent = new StringContent($"{{ 'name' : '{name}' }}", Encoding.UTF8,
"application/json");

putContent.Headers.ContentType = new MediaTypeHeaderValue("application/json");

using HttpResponseMessage response = await this.httpClient.PutAsync(proxyUrl,
putContent))

{

    return new ContentResult()

    {

        StatusCode = (int) response.StatusCode,
        Content = await response.Content.ReadAsStringAsync()
    };
}

}

// DELETE: api/Votes/name [HttpDelete("{name}")]
public async Task < IActionResult > Delete(string name)
{
    Uri serviceName = VotingWeb.GetVotingDataServiceName(this.serviceContext); Uri
proxyAddress = this.GetProxyAddress(serviceName);
long partitionKey = this.GetPartitionKey(name); string proxyUrl =

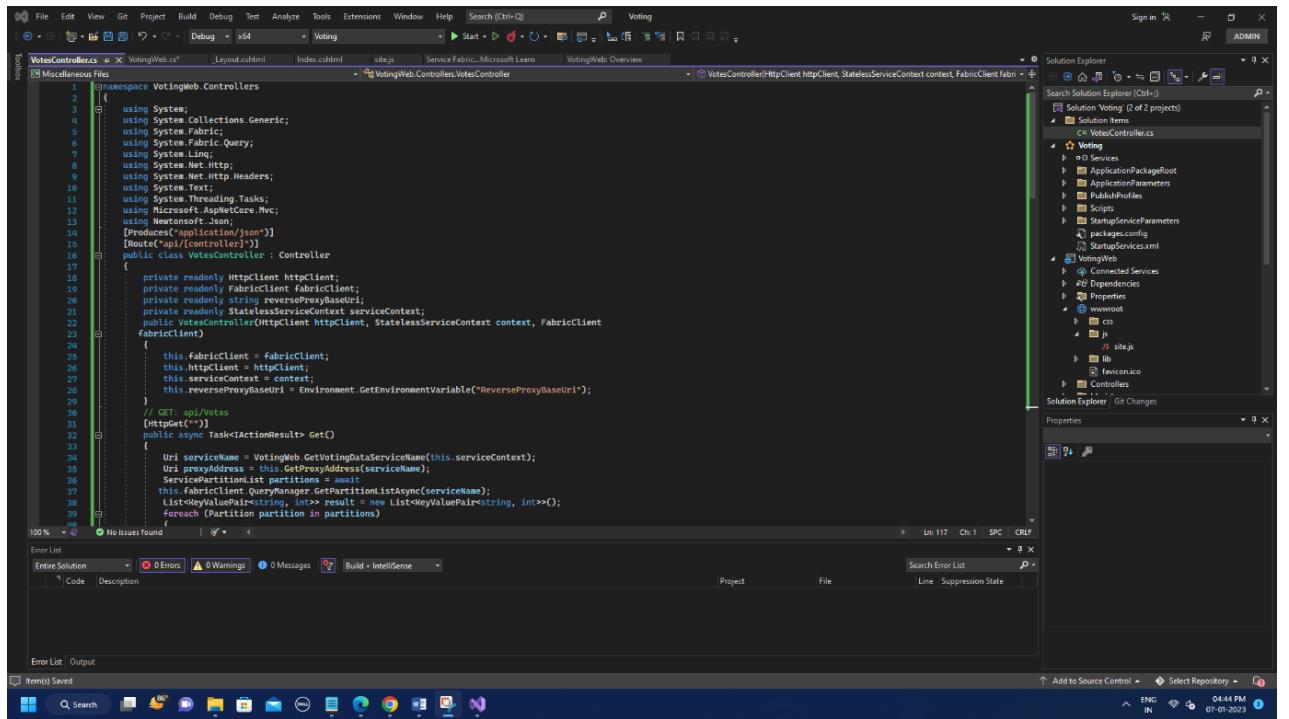
 $"{proxyAddress}/api/VoteData/{name}?PartitionKey={partitionKey}&PartitionKind=Int64R
ange";

using (HttpResponseMessage response = await this.httpClient.DeleteAsync(proxyUrl))
{
    if (response.StatusCode != System.Net.HttpStatusCode.OK) {
        return this.StatusCode((int) response.StatusCode);
    }
}

return new OkResult();
}
```

```
/// <summary>
/// Constructs a reverse proxy URL for a given service.
/// Example: http://localhost:19081/VotingApplication/VotingData/
/// </summary>
/// <param name="serviceName"></param>
/// <returns></returns>
private Uri GetProxyAddress(Uri serviceName)
{
    return new Uri($"'{this.reverseProxyBaseUri}{serviceName.AbsolutePath}'");
}

/// <summary>
/// Creates a partition key from the given name.
/// Uses the zero-based numeric position in the alphabet of the first letter of the name (0-25).
/// </summary>
/// <param name="name"></param>
/// <returns></returns>
private long GetPartitionKey(string name)
{
    return Char.ToUpper(name.First()) - 'A';
}
```



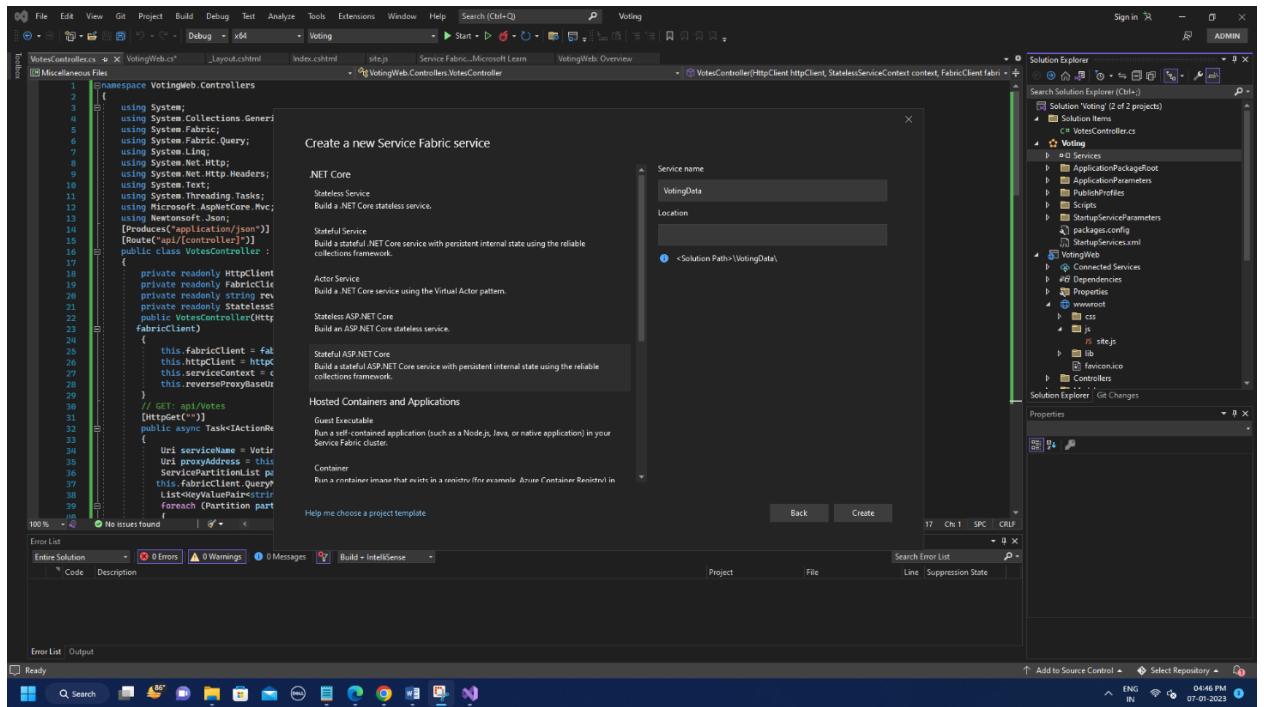
Step 9: Add a stateful back-end service to your application

- In Solution Explorer, right-click Services within the Voting application project and choose Add -> New Service Fabric Service....
- In the New Service Fabric Service dialog, choose Stateful ASP.NET Core, name the service VotingData, then press Create.

Once your service project is created, you have two services in your application. As you continue to build your application, you can add more services in the same way. Each can be independently versioned and upgraded.

The next page provides a set of ASP.NET Core project templates. choose API.

Visual Studio creates the VotingData service project and displays it in Solution Explorer.



Step 10: Add the VoteDataController.cs file

- In the VotingData project, right-click on the Controllers folder, then select Add->New item->Class. Name the file VoteDataController.cs and click Add.
- Replace the file contents with the following, then save your changes.

```

namespace VotingData.Controllers {
    using System.Collections.Generic;
    using System.Threading;
    using System.Threading.Tasks;
    using Microsoft.AspNetCore.Mvc;
    using Microsoft.ServiceFabric.Data;
    using Microsoft.ServiceFabric.Data.Collections;

    [Route("api/[controller]")]
    public class VoteDataController : Controller {
        private readonly IReliableStateManager stateManager;
        public VoteDataController(IReliableStateManager stateManager) {
            this.stateManager = stateManager;
        }
    }
}

```

```

// GET api/VoteData [HttpGet]
public async Task < IActionResult > Get()
{
    CancellationToken ct = new CancellationToken();
    IReliableDictionary < string, int > votesDictionary =
        await this.stateManager.GetOrAddAsync < IReliableDictionary < string, int >>
("counts");

    using(ITransaction tx = this.stateManager.CreateTransaction())
    {
        var list = await votesDictionary.CreateEnumerableAsync(tx); var enumerator =
list.GetAsyncEnumerator();
        List < KeyValuePair < string, int >> result = new List < KeyValuePair < string, int >>
();

        while (await enumerator.MoveNextAsync(ct)) {
            result.Add(enumerator.Current);
        }

        return this.Json(result);
    }
}

// PUT api/VoteData/name [HttpPut("{name}")]
public async Task < IActionResult > Put(string name)
{
    IReliableDictionary < string, int > votesDictionary = await
this.stateManager.GetOrAddAsync < IReliableDictionary < string, int >> ("counts");
    using(ITransaction tx = this.stateManager.CreateTransaction())
    {
        await votesDictionary.AddOrUpdateAsync(tx, name, 1, (key, oldvalue) => oldvalue +
1); await tx.CommitAsync();
    }
}

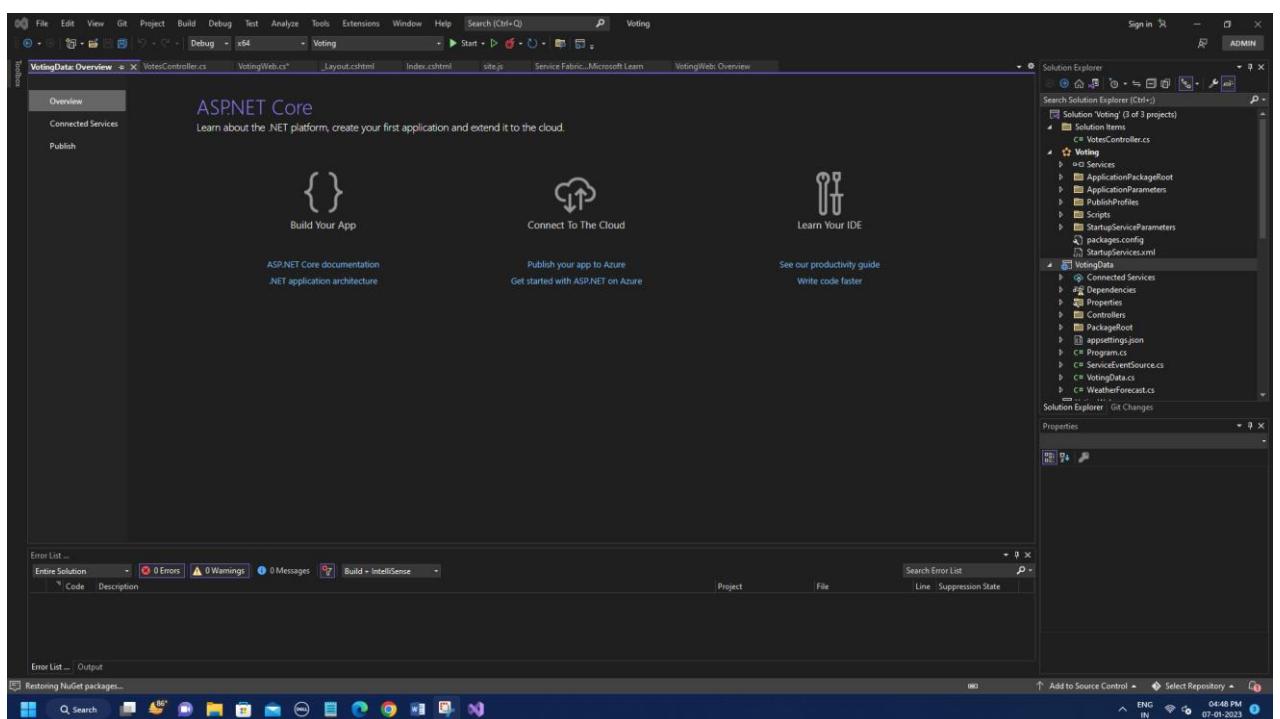
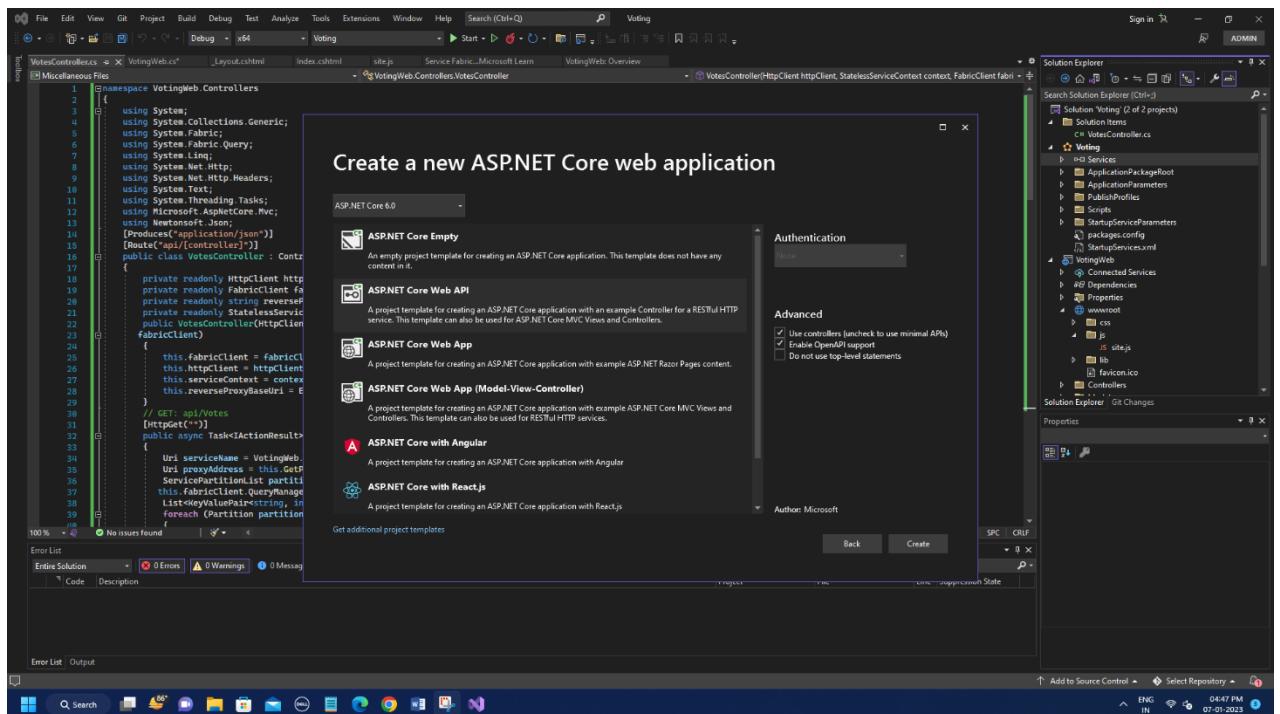
```

```
        return new OkResult();

    }

// DELETE api/VoteData/name [HttpDelete("{name}")]
public async Task < IActionResult > Delete(string name)
{
    IReliableDictionary< string, int > votesDictionary = await
this.stateManager.GetOrAddAsync< IReliableDictionary< string, int >> ("counts");

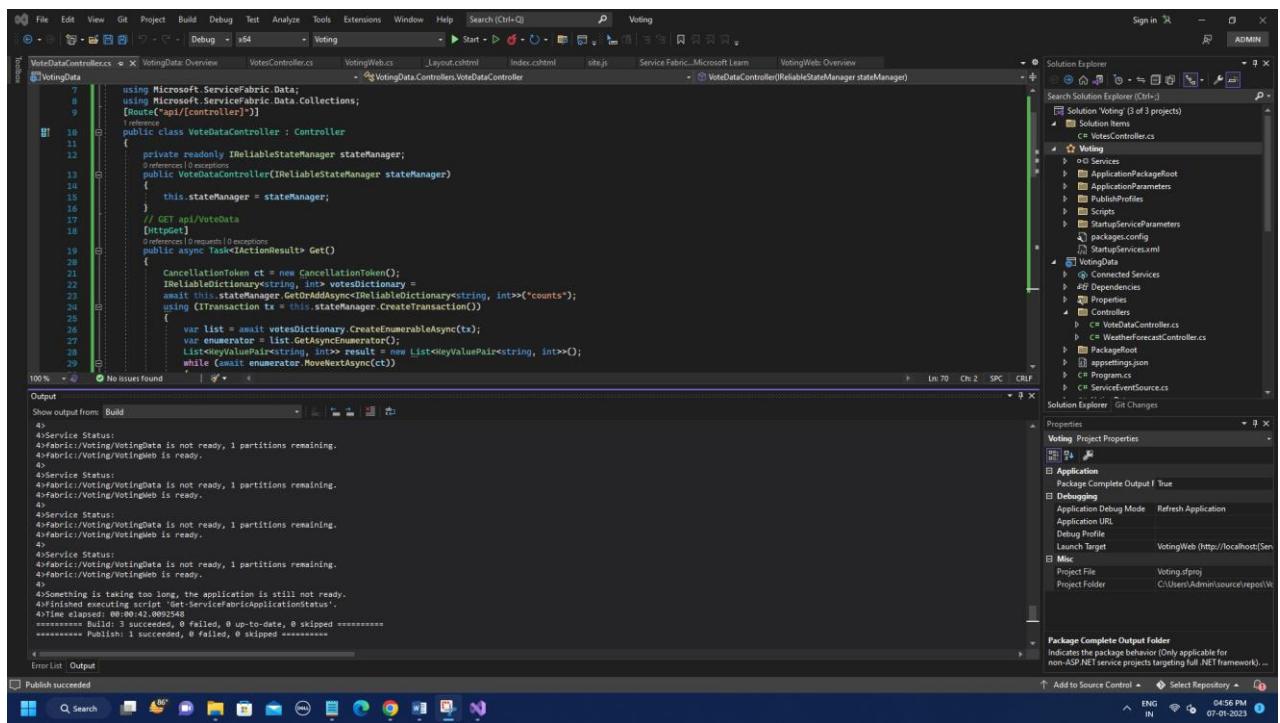
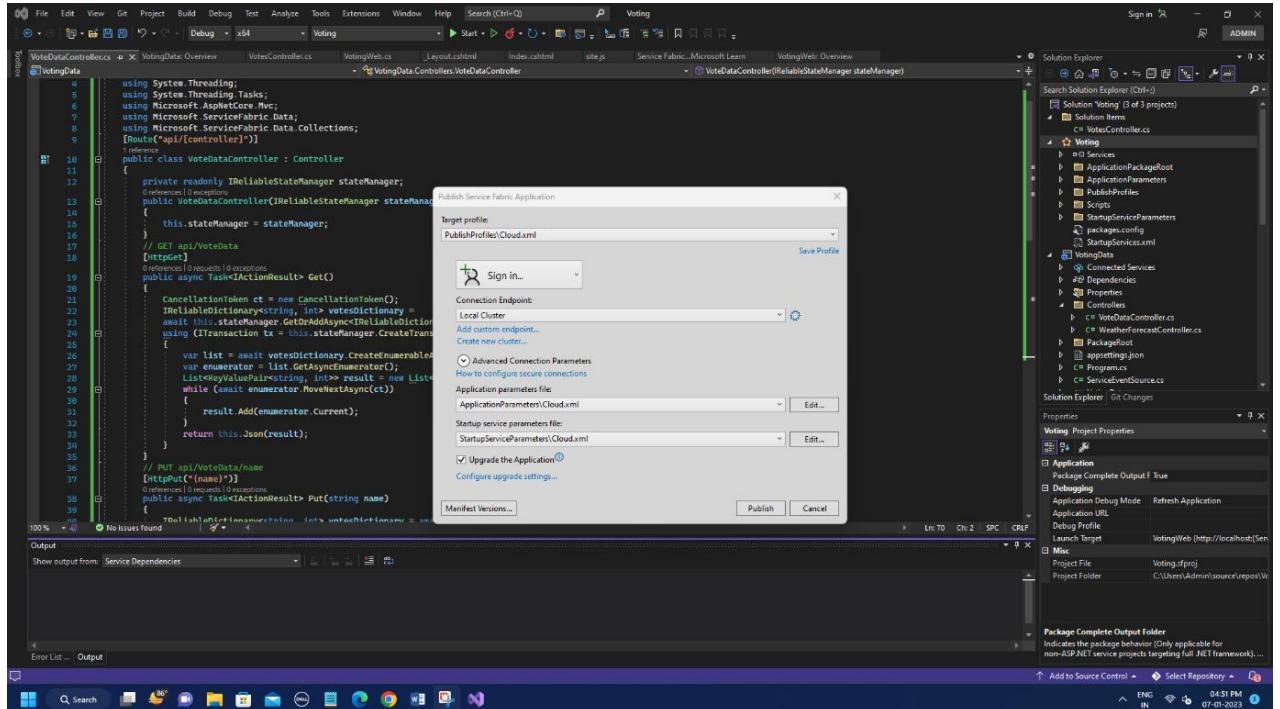
    using(ITransaction tx = this.stateManager.CreateTransaction())
    {
        if (await votesDictionary.ContainsKeyAsync(tx, name)) {
            await votesDictionary.TryRemoveAsync(tx, name); await tx.CommitAsync();
            return new OkResult();
        }
        else {
            return new NotFoundResult();
        }
    }
}
```

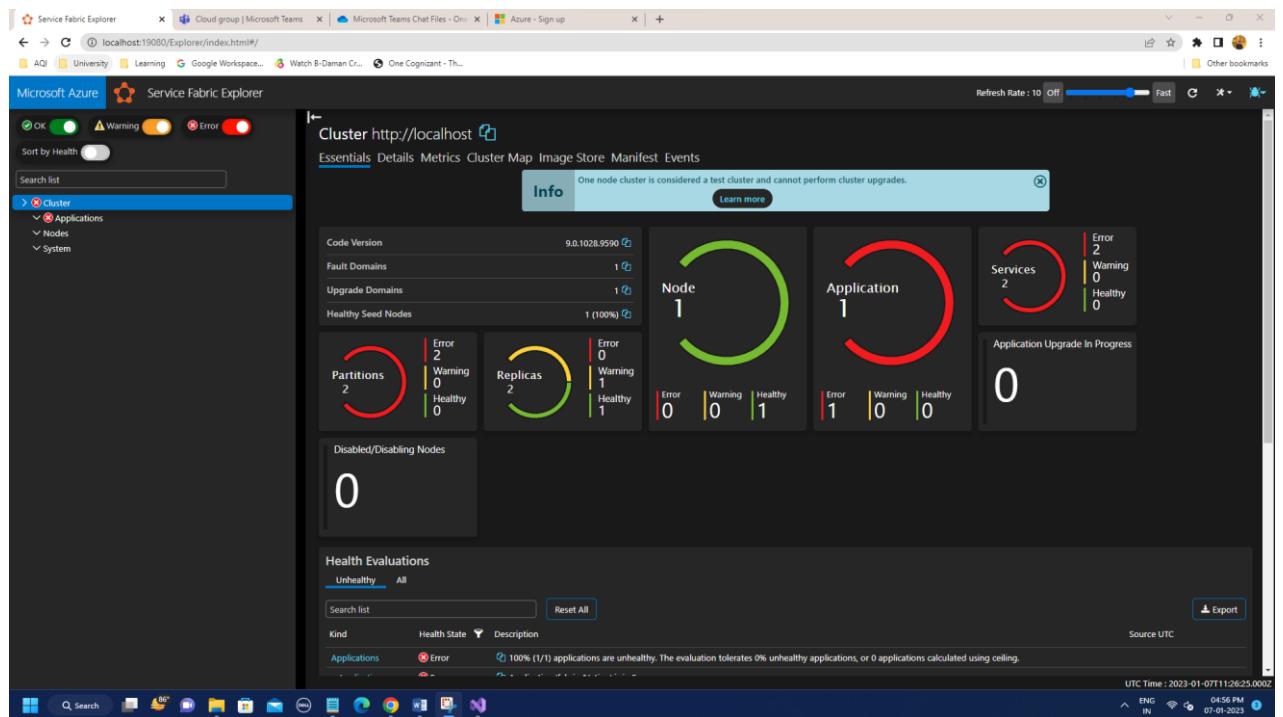


Step 11: Publish Service Fabric application

- In the solution Explorer Right click on the Voting and select Publish. The Publish dialog box appears.
- To open the service fabric explorer, open the Browser and paste the following IP address <http://localhost:19080/>

- Go to cluster > Application > Voting Type > fabric:/Voting > fabric:/VotingWeb > 8482ef73-476d-45a6-aca9-b33d75575855(partition) > _Node_0 Instance
- You will see the endpoint in the address section click on that address to run your app





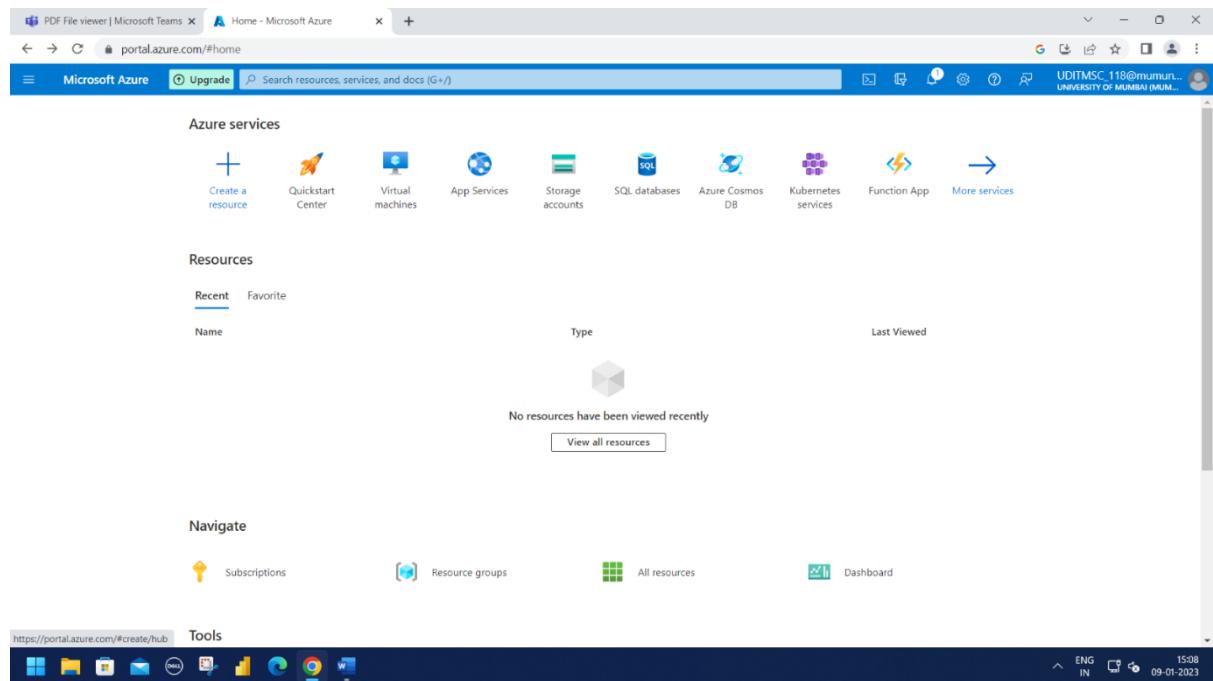
Practical No: 02

Aim: 2A. Create an Azure Kubernetes Service Cluster

Source Code:

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

Showing 1 to 20 of 153 results for 'kubernetes service'. [Clear search](#)

Service	Publisher	Description	Action
Azure Kubernetes Service (AKS)	Microsoft	Azure Service	Create
Kubernetes Fleet Manager	Microsoft	Azure Service	Create
Kubernetes - Azure Arc	Microsoft	Azure Service	Create
TrilioVault for Kubernetes - BYOL	Trilio	Container	Create
Managed Services for Azure Kubernetes Service	Basefarm	Managed Services	Create
Azure Kubernetes Service (AKS) Solution for Sentinel	Oracle	Azure Application	Create
Oracle WebLogic Server on Azure Kubernetes Service	Oracle America, Inc.	Azure Application	Create
Managed Services for Azure Kubernetes Service PCI	Basefarm	Managed Services	Create
Trusted Certificate Service	Intel	Container	Create
D2IQ Kubernetes Platform (D2KP)	D2IQ	SaaS	Create

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

Azure Kubernetes Service (AKS)

Microsoft

★ 4.0 (391 ratings)

Plan

Azure Kubernetes Service (AKS) [Create](#)

[Overview](#) [Plans](#) [Usage Information + Support](#) [Ratings + Reviews](#)

Azure Kubernetes Service is the quickest path from zero to Kubernetes on Azure. This new service features an Azure-hosted control plane, automated upgrades, self-healing, easy scaling, and a simple user experience for both developers and cluster operators. With AKS, customers get the benefits of open source Kubernetes without complexity or operational overhead.

To help you get started with AKS, the managed Kubernetes service is free. You only pay for the VMs in your cluster, and any infrastructure resources consumed like storage and networking. This means on Azure, you will pay nothing for the management of your Kubernetes cluster.

Creating an AKS cluster typically takes a few minutes. Once complete, you can access and manage your cluster through "az aks" and "kubectl".

More products from Microsoft [See All](#)

Firewall	Microsoft Azure Attestation	Service endpoint policy	Azure Managed Grafana
----------	-----------------------------	-------------------------	-----------------------

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 *

Cluster details

Cluster preset configuration To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time. [Learn more and compare presets](#)

Kubernetes cluster name * Region * Availability zones High availability is recommended for standard configuration.

Kubernetes version * API server availability Optimize for availability.

< Previous Next : Node pools >

Primary node pool

The number and size of nodes in the primary node pool in your cluster. For production workloads, at least 3 nodes are recommended for resiliency. For development or test workloads, only one node is required. If you would like to add additional node pools or to see additional configuration options for this node pool, go to the 'Node pools' tab above. You will be able to add additional node pools after creating your cluster. [Learn more about node pools in Azure Kubernetes Service](#)

API server availability Optimize for availability. Optimize for cost. 99.95% API server availability is recommended for standard configuration.

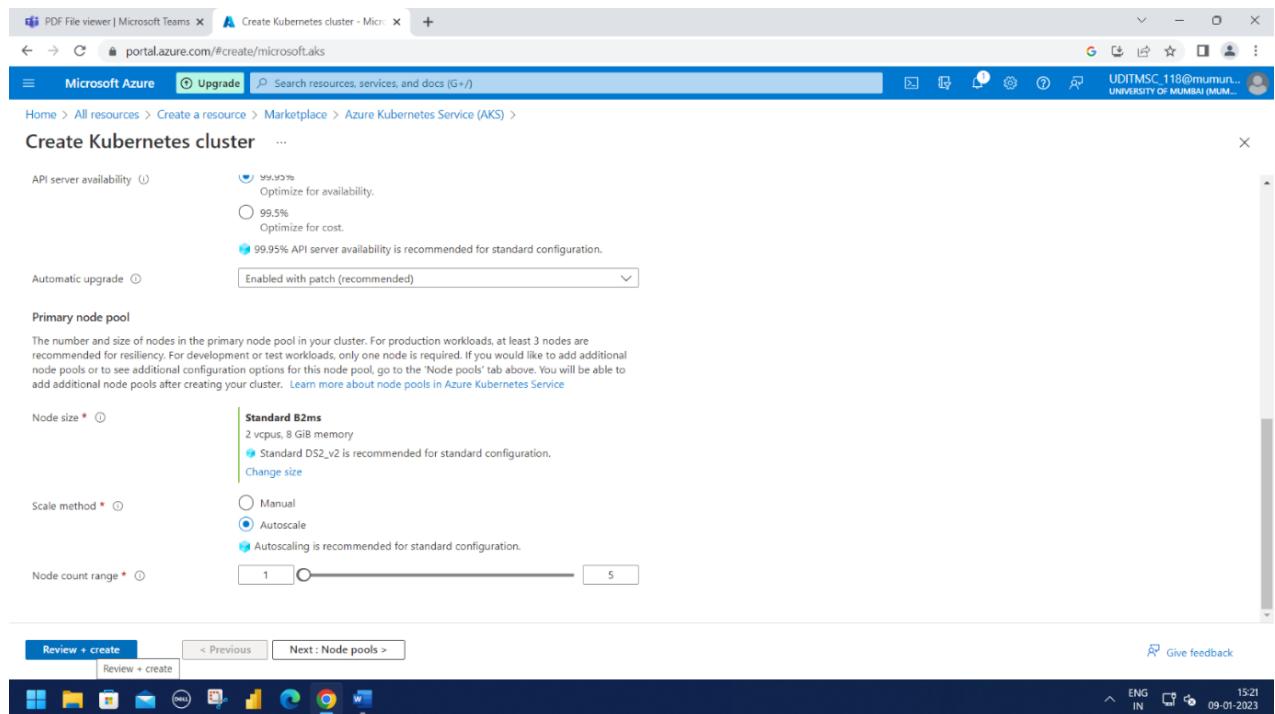
Automatic upgrade

Node size * Standard DS2_v2 is recommended for standard configuration.

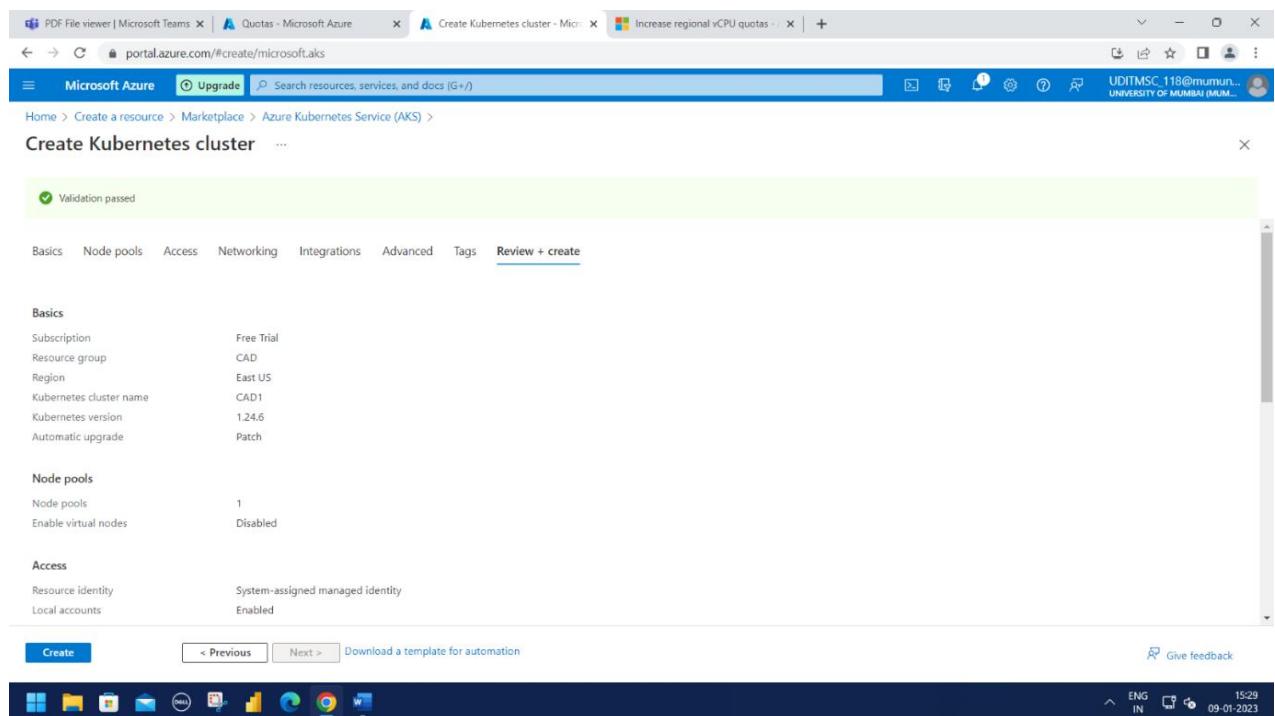
Scale method * Manual Autoscale Autoscaling is recommended for standard configuration.

Node count *

< Previous Next : Node pools >



Step 5: Click on Review + Create Button



Step 6: Click on Create Button

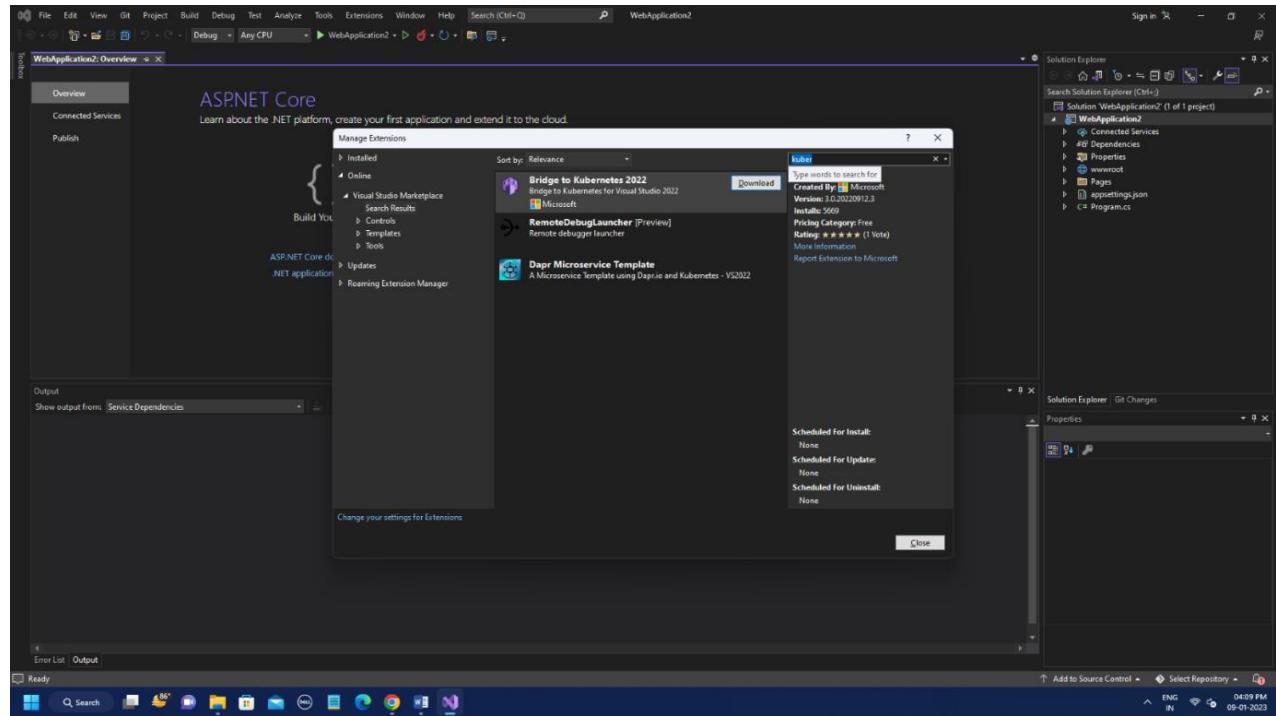
The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes tabs for 'PDF File viewer | Microsoft Teams', 'Quotas - Microsoft Azure', 'microsoft.aks-20230109152806', and 'Increase regional vCPU quotas -'. The main title is 'microsoft.aks-20230109152806 | Overview'. The left sidebar has sections for 'Overview', 'Inputs', 'Outputs', and 'Template'. The main content area displays a green checkmark indicating 'Your deployment is complete'. It shows deployment details: name 'microsoft.aks-20230109152806', subscription 'Free Trial', resource group 'CAD', start time '1/9/2023, 3:30:38 PM', and correlation ID '6c64a160-8e19-49fa-93fa-37c281456cb3'. Below this, there are sections for 'Deployment details', 'Next steps', and links to 'Go to resource' and 'Connect to cluster'. A sidebar on the right provides links to 'Cost Management', 'Container Insights', 'Free Microsoft tutorials', and 'Work with an expert'. The bottom taskbar shows various pinned application icons.

Aim: 2B. Configure Visual Studio to work with an azure kubernetes services Cluster

Source Code:

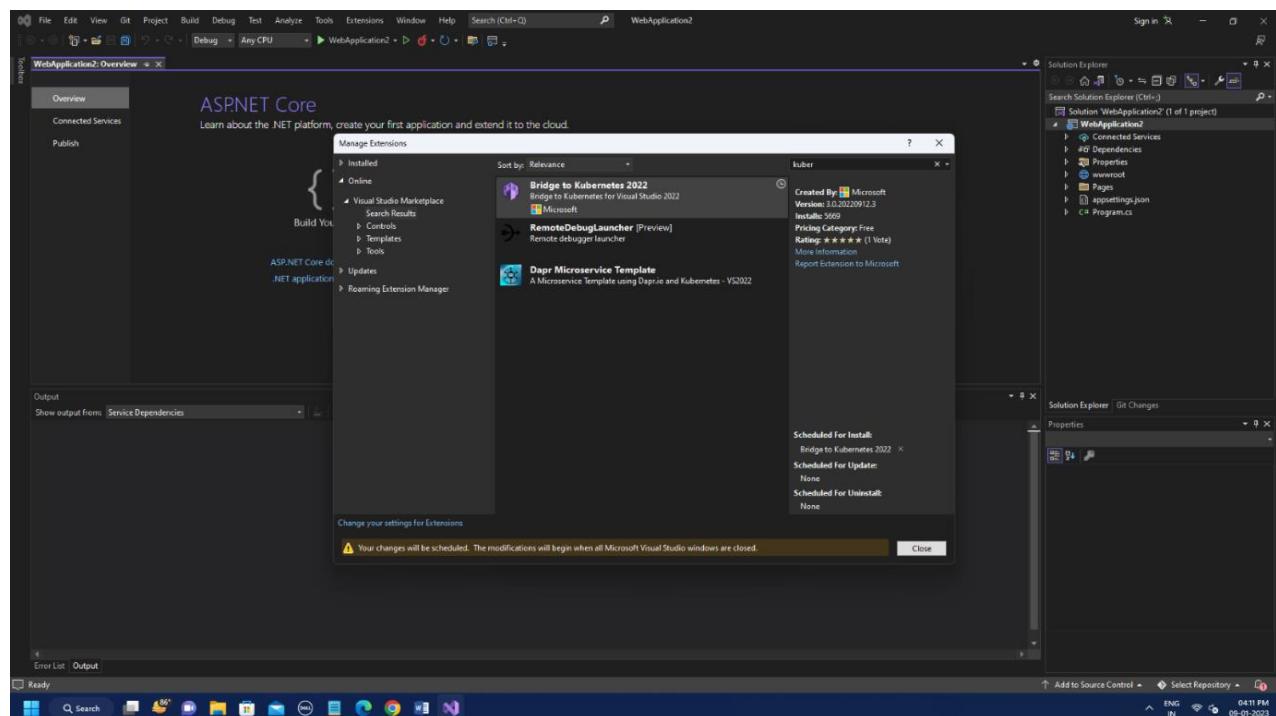
Step 1: Install Visual Studio Enterprise 2022

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



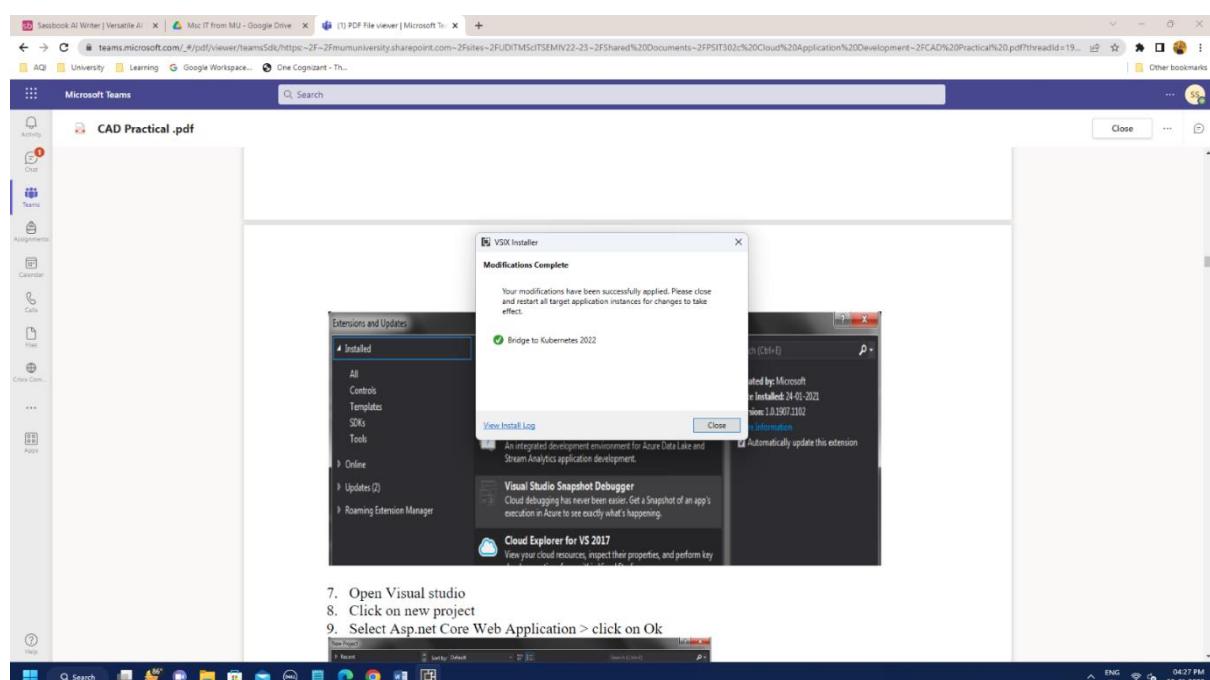
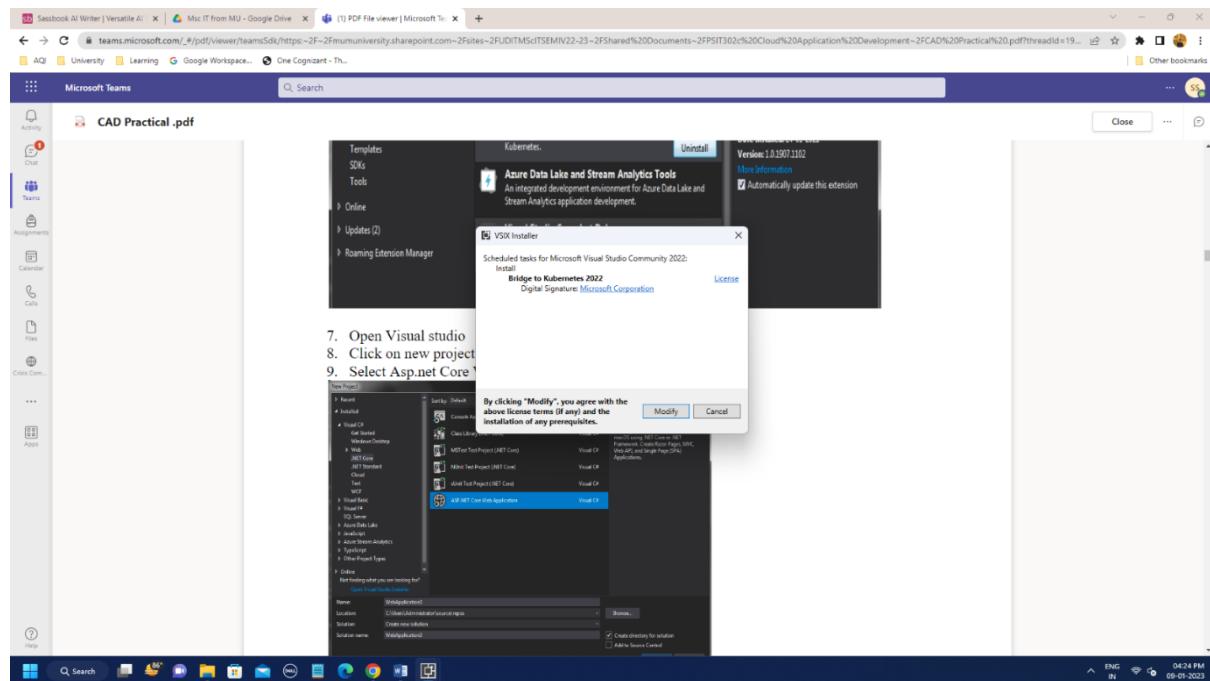
Step 3: Search for Bridge to Kubernetes 2022

Step 4: Click on Download



Step 5: Close visual studio

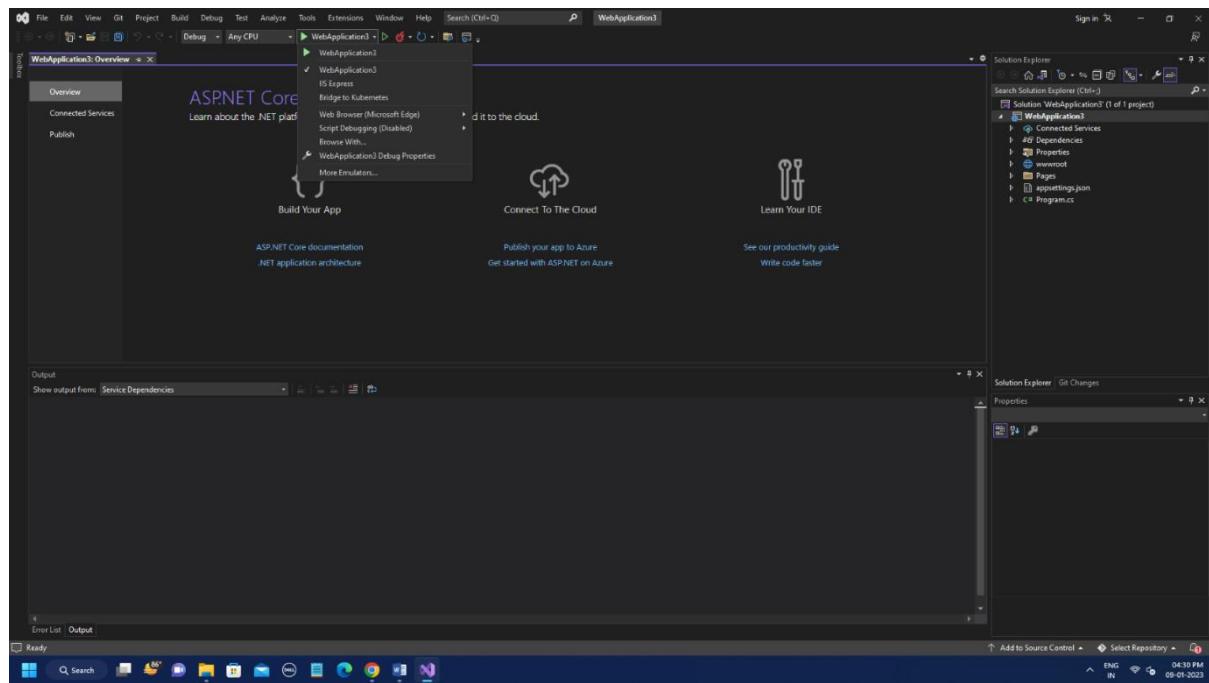
Step 6: click on Modify (install kubernetes tool extension)



7. Open Visual studio
8. Click on new project
9. Select Asp.net Core

Step 7: Open Visual studio

- Click on new project
- Select Asp.net Core Web Application > click on Ok
- Select Model View Control > Click on OK
- Click On Down Arrow of IIS Express
- We can see option of Bridge to Kubernetes



Aim: 2C. Configure Visual Studio Code to work with an azure kubernetes services Cluster

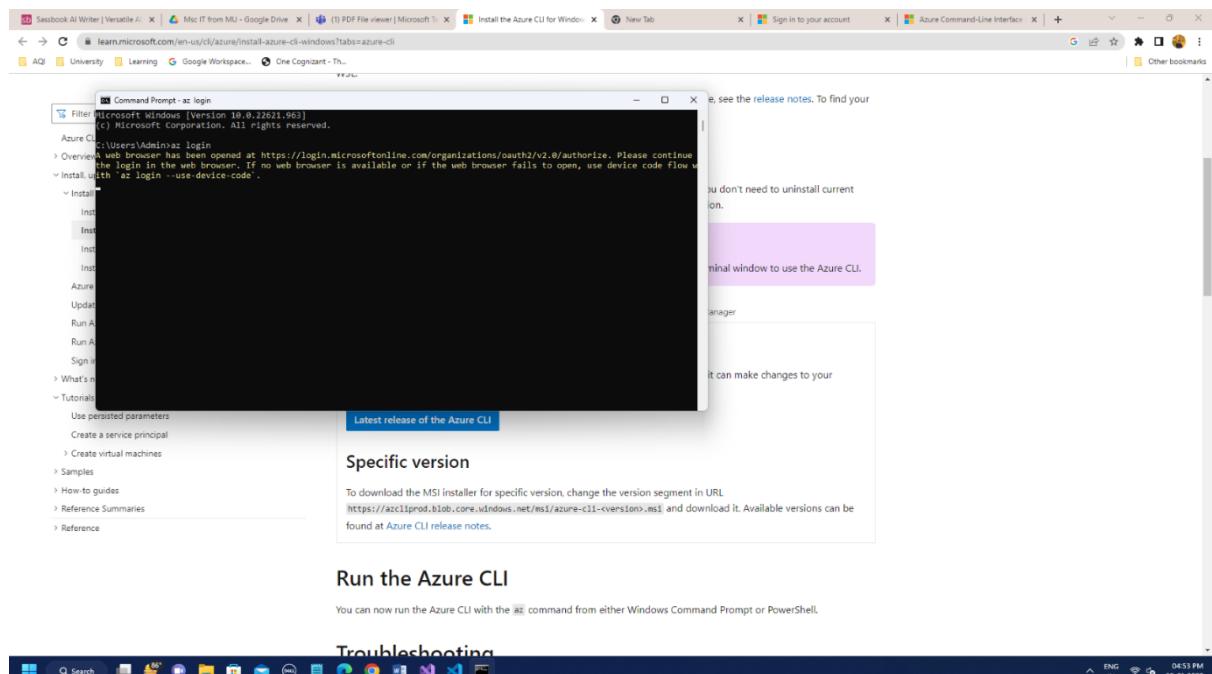
Source Code:

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
- Output on command prompt



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

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

1.

The screenshot shows the Visual Studio Code interface. The left sidebar is titled 'EXPLORER' and says 'NO FOLDER OPENED'. It has buttons for 'Open Folder' and 'Clone Repository'. The main area displays the 'November 2022 (version 1.74)' release notes, which include several updates and highlights. Below the notes is a 'TERMINAL' tab showing a PowerShell session:

```
+ FullyQualifiedErrorId : CommandNotFoundException
PS C:\Users\Admin> az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no web browser is available or if the web browser fails to open, use dev ice code flow with az login --use-device-code .
[{"cloudName": "AzureCloud", "homeTenantId": "4d70a983-523c-4b1a-be42-414e0e0085b17", "isDefault": true, "managedTenants": [], "name": "AzureCloud", "state": "Enabled", "tenantId": "4d70a983-523c-4b1a-be42-414e0e0085b17", "user": {"name": "UDITMSc_119@university.onmicrosoft.com", "type": "user"}}]
```

The bottom of the screen shows the Windows taskbar with various pinned icons and the system tray indicating the date and time as 06:54 PM on 06-01-2023.

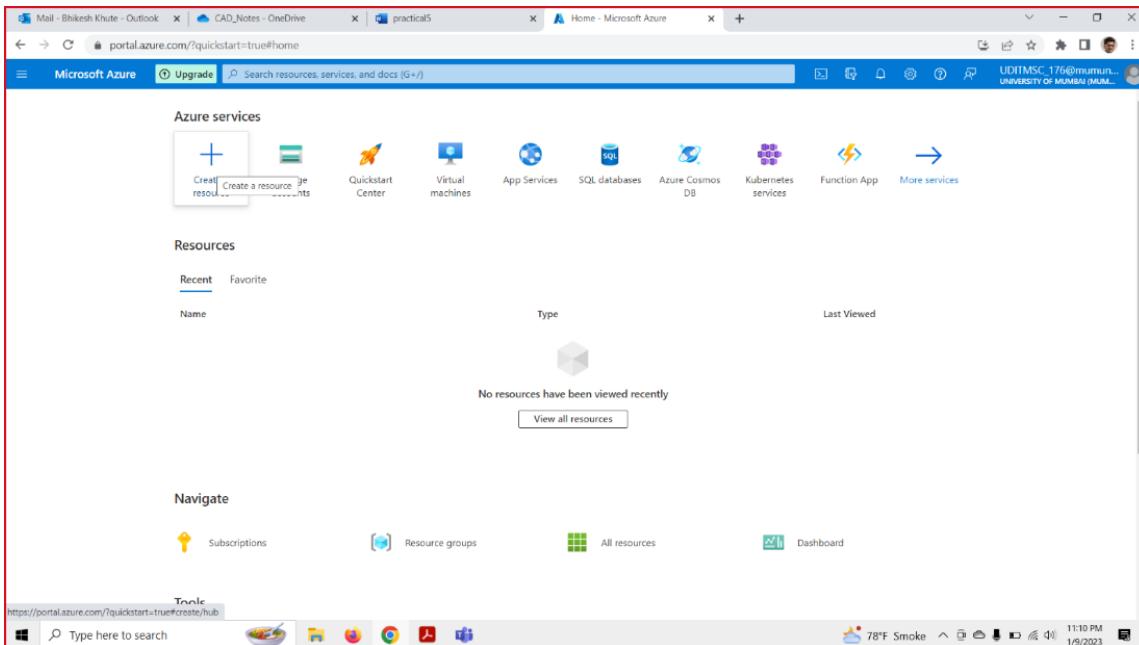
Practical No: 3

Aim: 3a. Create an AKS cluster - From the portal

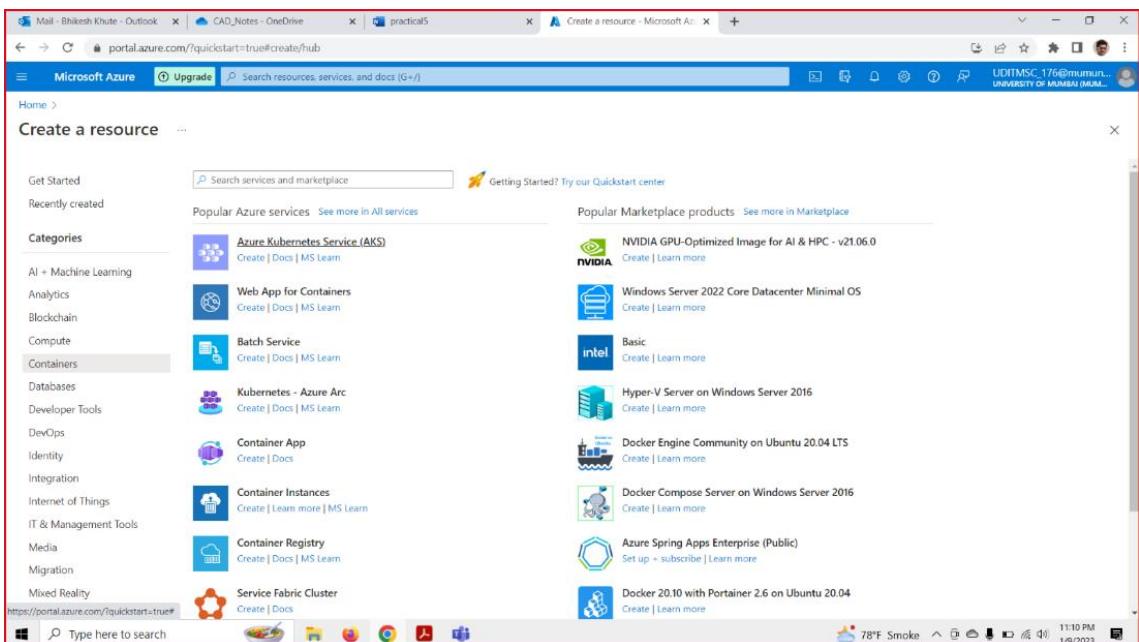
Source Code:

Step 1: Sign in to the Azure portal at <https://portal.azure.com>.

Step 2: On the Azure portal menu or from the home page, select Create a resource.

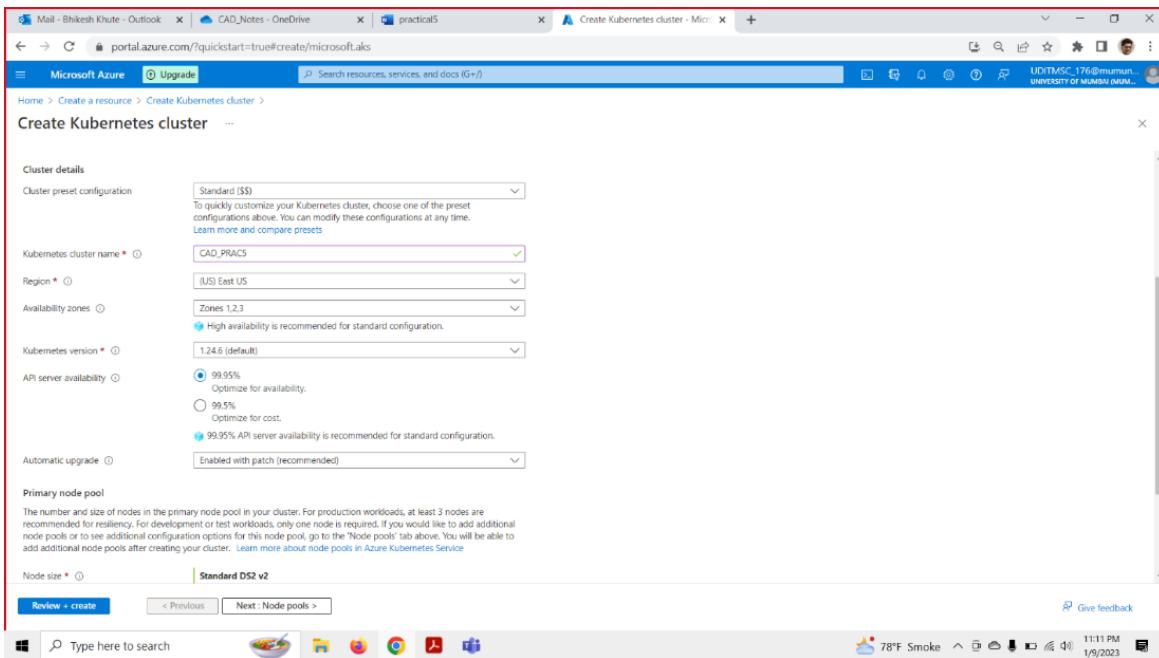


Step 3: Select Containers > Kubernetes Service.



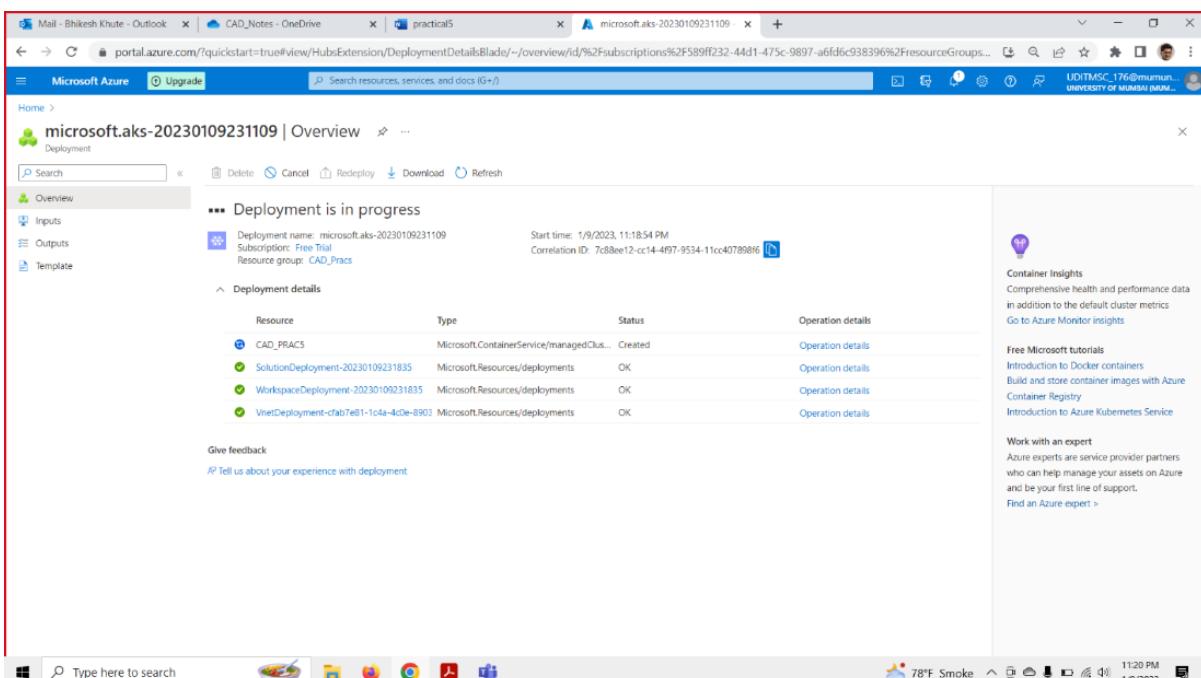
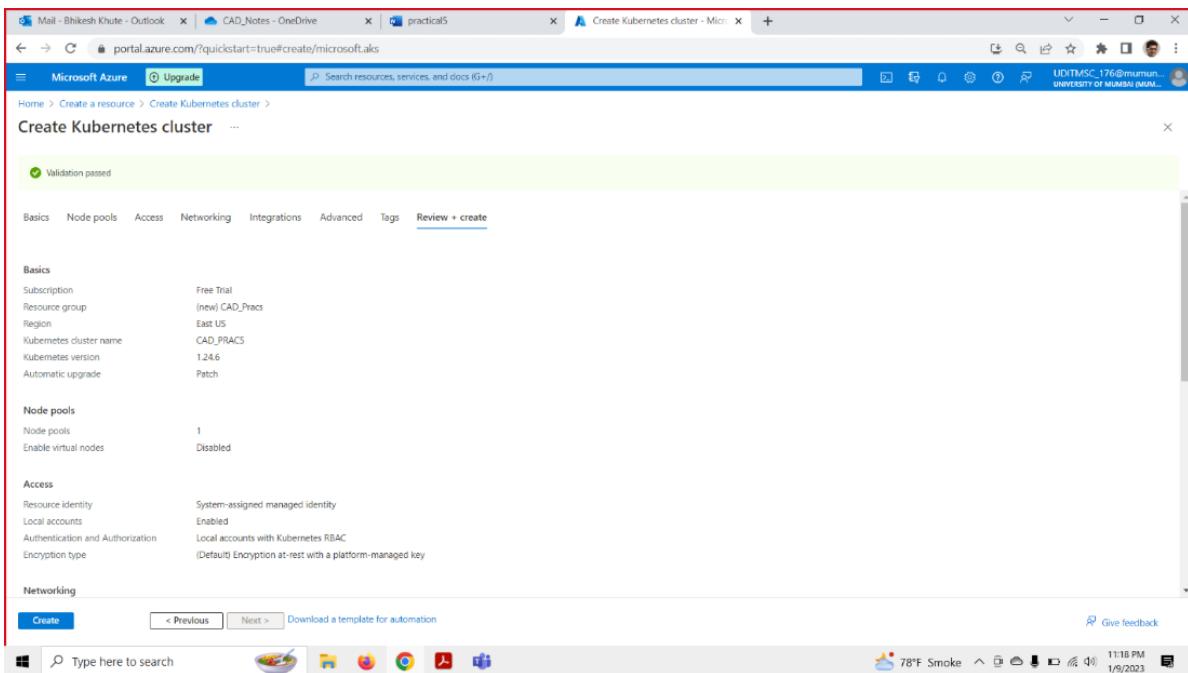
Step 4: On the Basics page, configure the following options:

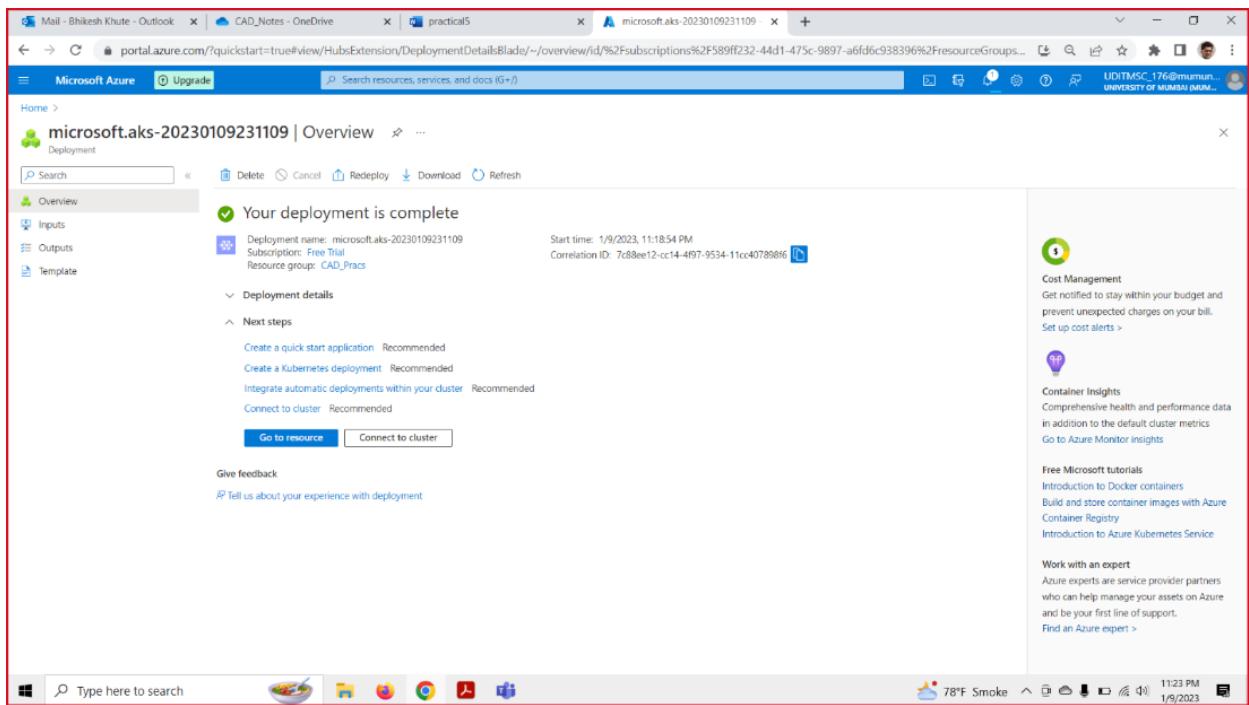
- **Project details:** Select an Azure Subscription, then select or create an Azure Resource group, such as **CAD_PRACS**.
- **Cluster details:** Enter a Kubernetes cluster name, such as **CAD_PRAC5**. Select a Region and Kubernetes version for the AKS cluster.
- **Primary node pool:** Select a VM Node size for the AKS nodes. The VM size can't be changed once an AKS cluster has been deployed. - Select the number of nodes to deploy into the cluster. Set Node count to 1. Node count can be adjusted after the cluster has been deployed.



Step 5: On the Node pools page, **keep the default options**. At the bottom of the screen, click Next: Authentication & keep local access or keep it default what is selected. We can change it later.

Step 6: Click Review + create and then Create when validation completes.





Aim: 3b. Create an AKS cluster – with Azure CLI

Source Code:

Step 1: Install Azure CLI for windows system.

Step 2: Sign in to the Azure CLI by using the **az login** command. To finish the authentication process, follow the steps displayed in your terminal.

```
PS C:\Users\Admin> az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow with az login --use-device-code.

{
  "cloudName": "AzureCloud",
  "homeTenantId": "4d7ba981-523c-4b1a-be42-414e0e085b17",
  "id": "589ff232-44d1-475c-9897-a64d6c938396",
  "isDefault": true,
  "managedByTenants": [],
  "name": "free Trial",
  "state": "Enabled",
  "tenantId": "4d7ba981-523c-4b1a-be42-414e0e085b17",
  "user": {
    "name": "UDITHEC_176@mumuniversity.onmicrosoft.com",
    "type": "user"
  }
}
PS C:\Users\Admin>
```

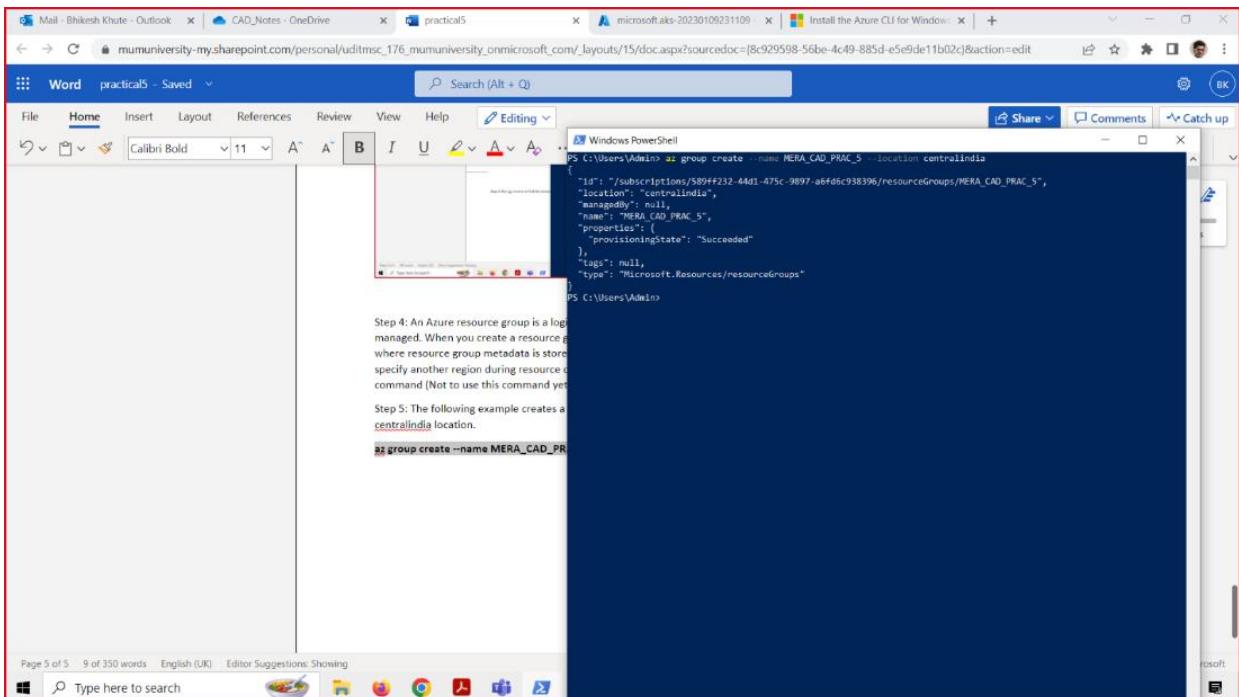
Step 3: Run **az version** to find the version and dependent libraries that are

```
PS C:\Users\Admin> az version
{
  "azure-cli": "2.43.0",
  "azure-cli-core": "2.43.0",
  "azure-cli-telemetry": "1.6.8",
  "extensions": {}
}
PS C:\Users\Admin>
```

Step 4: An Azure resource group is a logical group in which Azure resources are deployed and managed. When you create a resource group, you are asked to specify a location. This location is where resource group metadata is stored, it is also where your resources run in Azure if you don't specify another region during resource creation. Create a resource group using the az group create command (Not to use this command yet. Refer the command in next step)

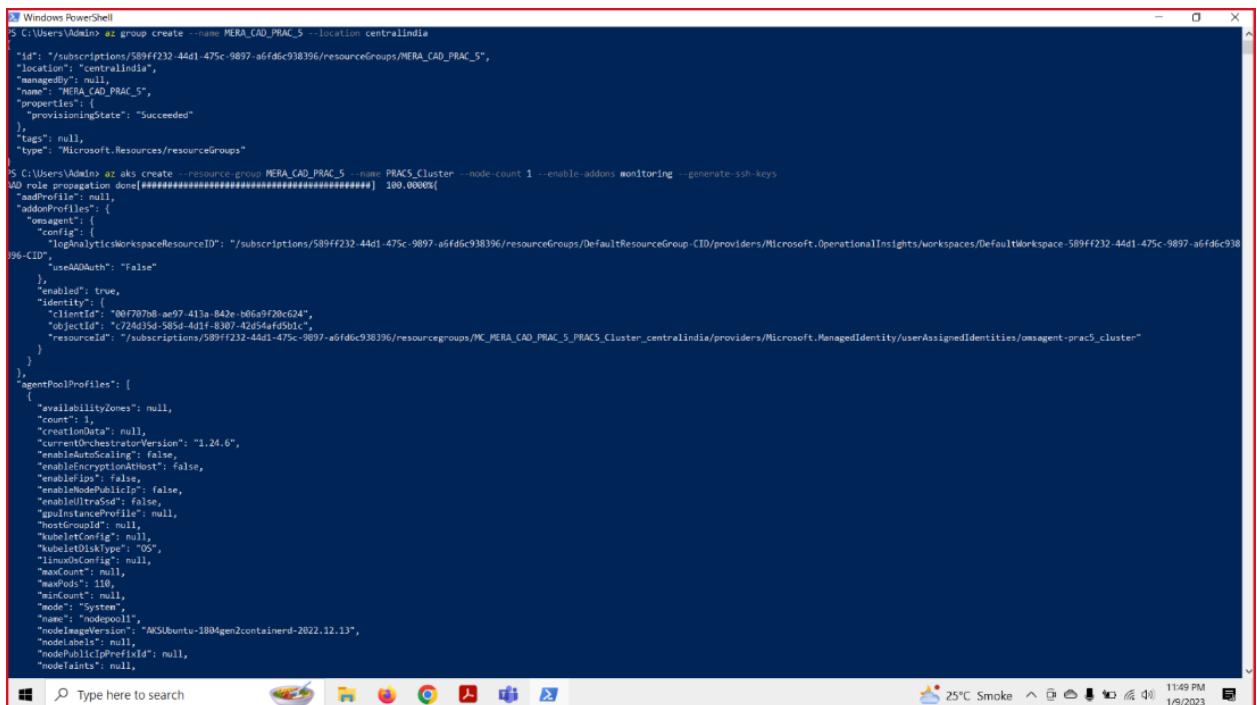
Step 5: The following example creates a resource group named myResourceGroup in the centralindia location.

```
az group create --name MERA_CAD_PRAC_5 --location centralindia
```



Step 6: Use following command to create an AKS cluster. The following example creates a cluster with one node. This will take several minutes to complete.

```
az aks create --resource-group MERA_CAD_PRAC_5 --name PRAC5_Cluster --node-count 1 --enable-addons monitoring --generate-ssh-keys
```

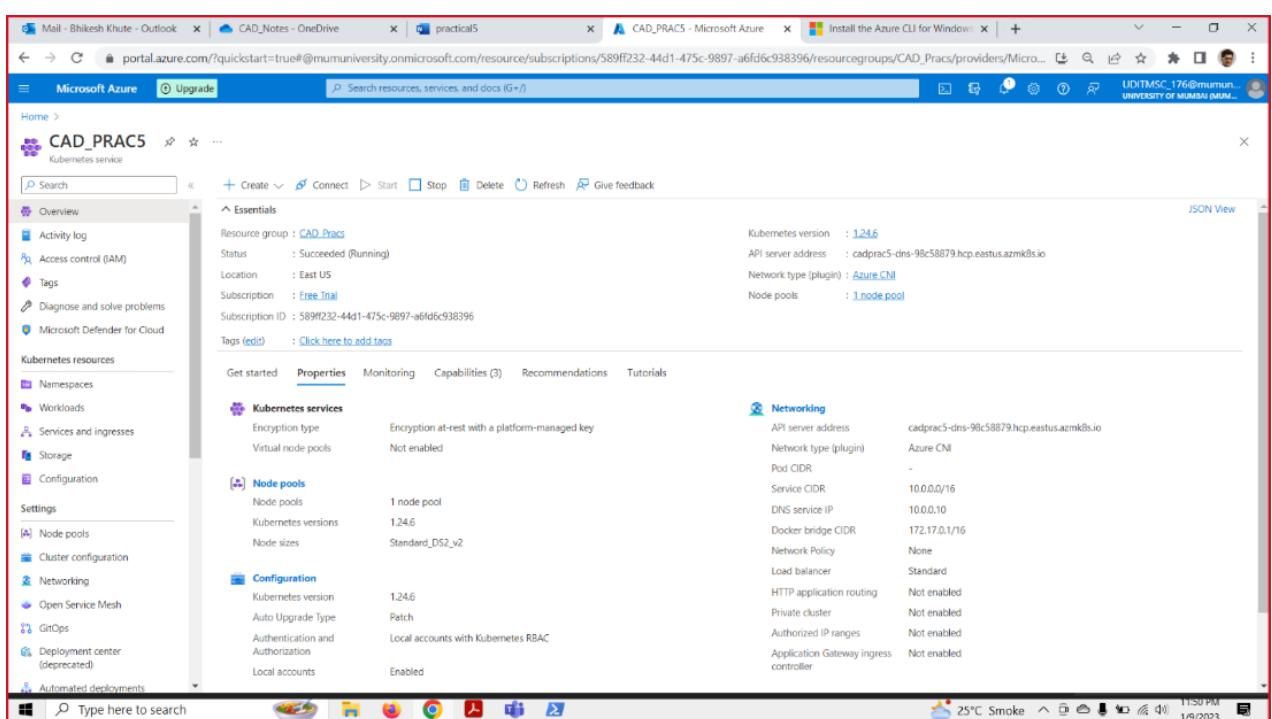


```

Windows PowerShell
PS C:\Users\Admin\> az group create --name MERA_CAD_PRAC_5 --location centralindia
{
  "id": "/subscriptions/589ff232-44d1-475c-9897-a6fd6c938396/resourceGroups/MERA_CAD_PRAC_5",
  "location": "centralindia",
  "managedBy": null,
  "name": "MERA_CAD_PRAC_5",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}

PS C:\Users\Admin\> azaks create --resource-group MERA_CAD_PRAC_5 --name PRAC5_Cluster --node-count 1 --enable-addons monitoring --generate-ssh-keys
#0 role propagation done[########################################] 100.0000%
{
  "addOnProfiles": null,
  "addonsProfile": {
    "omsagent": {
      "config": {
        "logAnalyticsWorkspaceResourceID": "/subscriptions/589ff232-44d1-475c-9897-a6fd6c938396/resourceGroups/DefaultResourceGroup-CID/providers/Microsoft.OperationalInsights/workspaces/DefaultWorkspace-589ff232-44d1-475c-9897-a6fd6c938396-CID"
      }
    }
  },
  "agentPoolProfiles": [
    {
      "availabilityZones": null,
      "count": 1,
      "creationData": null,
      "currentorchestratorVersion": "1.24.6",
      "enableAutoScaling": false,
      "enableEncryptionAtHost": false,
      "enableIps": false,
      "enableKubeletPort": false,
      "enableLoadBalanced": false,
      "enableMetricsProfile": null,
      "hostGroup": null,
      "kubeletConfig": null,
      "kubeletDiskType": "OS",
      "linuxProfile": null,
      "maxCount": null,
      "maxPods": 110,
      "minCount": null,
      "mode": "System",
      "name": "nodepool1",
      "nodeLabelSelector": "NKSUbuntu-1004gen2containerd-2022.12.13",
      "nodeLabels": null,
      "nodePublicIPPrefixId": null,
      "nodeSaftey": null,
      "osType": "Ubuntu"
    }
  ],
  "clusterProfile": {
    "controlPlaneVmSize": "Standard_DS2_v2",
    "nodeVmSize": "Standard_DS2_v2"
  }
}

```



CAD_PRAC5 - Microsoft Azure

Kubernetes service

Essentials

- Resource group : CAD_Prac5
- Status : Succeeded (Running)
- Location : East US
- Subscription : Free Trial
- Subscription ID : 589ff232-44d1-475c-9897-a6fd6c938396
- Tags (edit) : Click here to add tags

Kubernetes resources

- Namespaces
- Workloads
- Services and ingresses
- Storage
- Configuration

Properties

Property	Value
Kubernetes version	1.24.6
API server address	cadprac5-dns-98c58879.hcp.eastus.azmk8s.io
Network type (plugin)	Azure CNI
Node pools	1.node_pool

Networking

Setting	Value
API server address	cadprac5-dns-98c58879.hcp.eastus.azmk8s.io
Network type (plugin)	Azure CNI
Pod CIDR	-
Service CIDR	10.0.0.0/16
DNS service IP	10.0.0.10
Docker bridge CIDR	172.17.0.1/16
Network Policy	None
Load balancer	Standard
HTTP application routing	Not enabled
Private cluster	Not enabled
Authorized IP ranges	Not enabled
Application Gateway ingress controller	Not enabled

Get started

Properties

Monitoring

Capabilities (3)

Recommendations

Tutorials

Practical No: 04

Create an API gateway service

Aim: 4A. Create an API Management Service.

Source Code:

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, select “Create API management service” button.

Step 6: Provide a name. This name sets the URL of the API gateway and portal [Name-AzureManagementService]

Step 7: Select the subscription and resource group (or create a new resource group), and select the location. [Resource NameAzureManagement, Location- SouthEast Asia]

Step 8: Specify the organization name. This name will appear in the developer portal as the organization that publishes the API. [Name-CAD Prac 8]

Step 9: Specify the email address of the administrator. The user who creates the service instance will be the default administrator, so it's best to provide the email address of this user until you want someone else to serve as administrator.

Step 10: Select the pricing tier. The Developer tier is the most comprehensive offering, with sufficient request/response limitations in dev/test scenarios.

Step 11: After Completing the form click on review + create button to create “Azure API Management service instance”.

Microsoft Azure Upgrade Search resources, services, and documentation

Home > API Management services >

Install API Management gateway

API Management service

Basics Monitoring Scale Managed identity Virtual network Protocol settings Tags Review + install

Project details

Select the 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) AzureManagementService Create new

Instance details

Region * (Asia Pacific) Southeast Asia

Resource name * AzureManagement

Organization name * CAD Prac 8

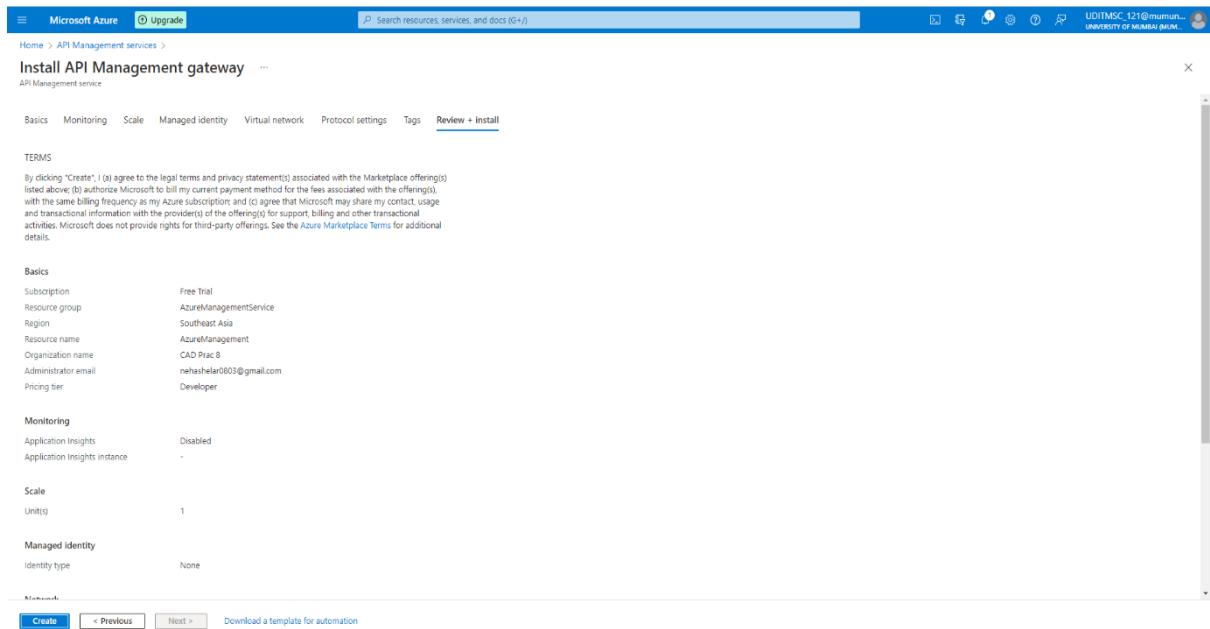
Administrator email * nehashelar0803@gmail.com

Pricing tier

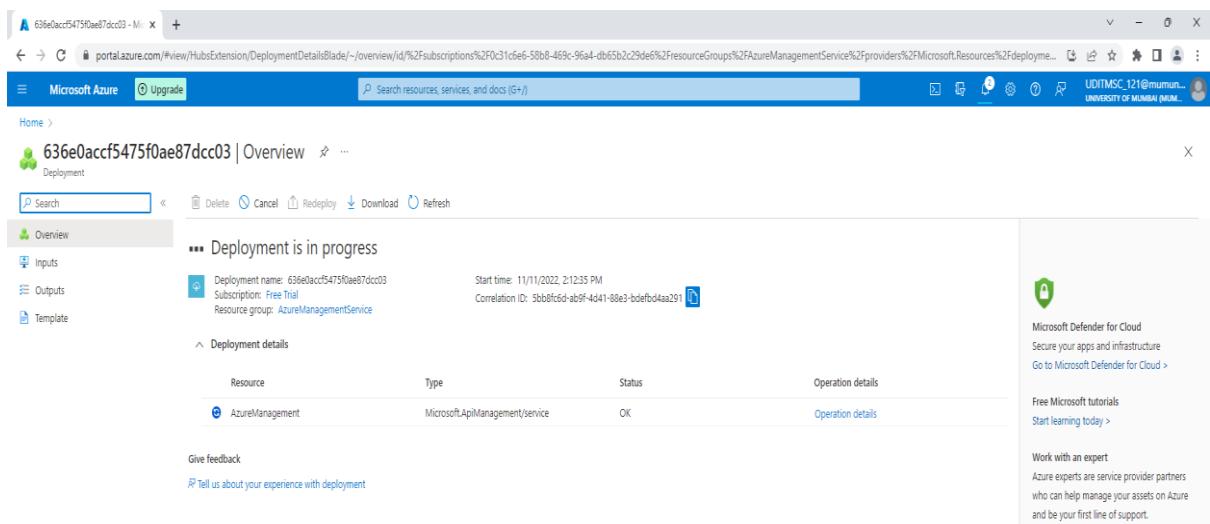
The Developer tier of API Management does not include SLA and should not be used for production purposes. Your service may experience intermittent outages, for example during upgrades. [Learn more](#)

Pricing tier Developer (no SLA)

Review + create < Previous Next: Monitoring >



Step 14: After clicking on the create button the API management service instance will go to deployment stage. After the deployment is complete the instance will be shown



The screenshot shows the Microsoft Azure Deployment Overview page. At the top, it displays the deployment name: 636e0accf5475f0ae87dcc03 | Overview. Below this, a message says "Your deployment is complete". It provides deployment details: Deployment name: 636e0accf5475f0ae87dcc03, Subscription: Free Trial, Resource group: AzureManagementService. It also shows the start time: 11/11/2022, 2:12:35 PM and Correlation ID: 5bb8fc6d-ab9f-4d41-88e3-bdefbd4aa291. There are tabs for Overview, Inputs, Outputs, and Template. A "Go to resource" button is present.

The screenshot shows the Microsoft Azure API Management services list view. It lists one service named "AzureManagement" which is Activating, developed by Developer, and located in Southeast Asia. The service belongs to the AzureManagementService resource group and is associated with a Free Trial subscription. The interface includes filters for Name, Status, Tier, Type, Location, Resource group, and Subscription, along with buttons for Create, Manage view, Refresh, Export to CSV, Open query, and Assign tags.

The screenshot shows the Microsoft Azure API Management service configuration page for "AzureManagement". On the left, there's a sidebar with options like Properties, Locks, APIs, Products, Subscriptions, Named values, Backends, Policy fragments, API Tags, Schemas, Authorizations (preview), Power Platform, Developer portal, Portal overview, Portal settings, and Users. The main area is titled "Define a new API" and shows four options: HTTP (Manually define an HTTP API), WebSocket (Streaming full-duplex communication with a WebSocket server), GraphQL (Access the full capability of your data from a single endpoint), and Synthetic GraphQL (Build a GraphQL service using existing HTTP APIs). Below this, there are sections for "Create from definition" (OpenAPI, WADL, WSDL) and "Create from Azure resource".

Create from OpenAPI specification

Basic Full

* OpenAPI specification	<input type="text" value="https://conferenceapi.azurewebsites.net/?format=json"/>	or	<input type="file"/> Select a file (maximum size 4 MiB)
* Display name	<input type="text" value="Demo Conference API"/>		
* Name	<input type="text" value="demo-conference-api"/>		
API URL suffix	<input type="text" value="conference"/>		
Base URL	<input type="text" value="http(s)://azuremanagement.azure-api.net/conference"/>		

The screenshot shows the Azure Management portal interface. On the left, there's a sidebar with navigation links like Home, API Management services, AzureManagement, Create, Manage view, etc. The main area is titled "AzureManagement | APIs" and shows a list of APIs under "All APIs". One API, "Demo Conference API", is selected. To the right of the API list, there's a detailed view of the API's operations. The "Design" tab is selected, showing a table for parameters and a section for "HTTP request". The "HTTP request" section contains the following details:

```

GET https://azuremanagement.azure-api.net/conference/speakers
Host: azuremanagement.azure-api.net
  
```

Below the "HTTP request" section are buttons for "Send", "Trace", and "Bypass CORS proxy".

HTTP response

Message Trace [Generate definition](#)

```
HTTP/1.1 200 OK
cache-control: no-cache
content-length: 40606
content-type: application/vnd.collection+json
date: Fri, 11 Nov 2022 10:13:17 GMT
expires: -1
pragma: no-cache
vary: Origin
x-aspnet-version: 4.0.30319
x-powered-by: ASP.NET

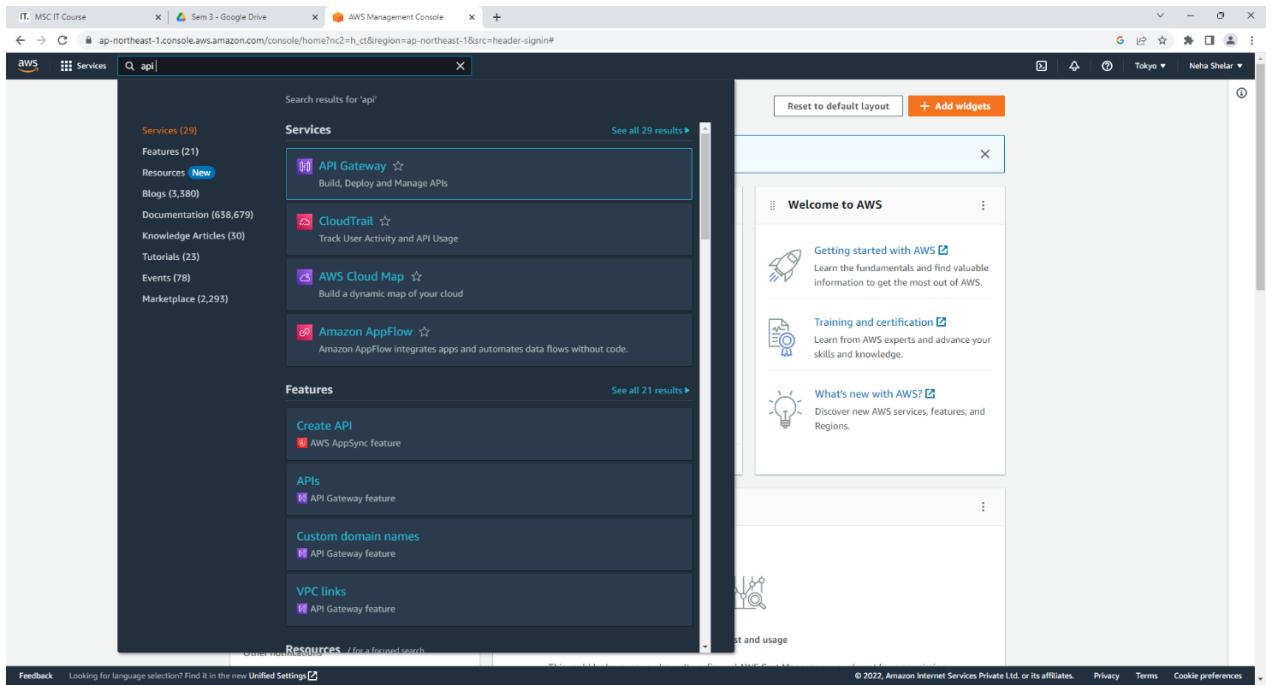
{
  "collection": {
    "version": "1.0",
    "href": "https://conferenceapi.azurewebsites.net:443/speakers",
    "links": [],
    "items": [
      {
        "href": "https://conferenceapi.azurewebsites.net/speaker/1",
        "d---", rr
```

Aim: 4B. Create an API Gateway Service.

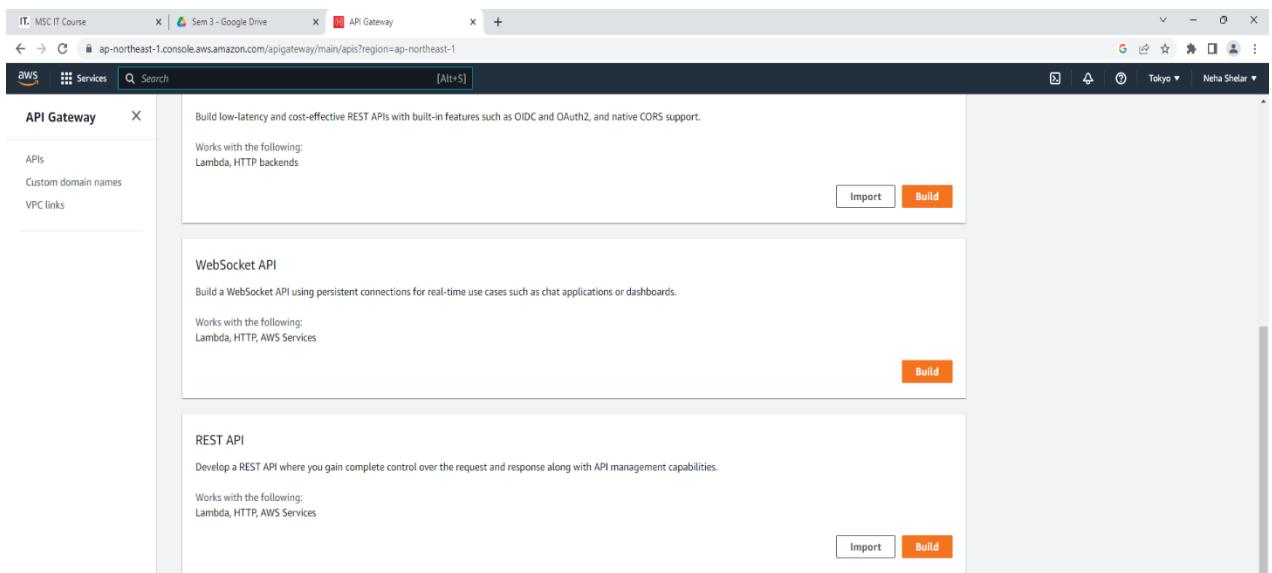
Source Code:

Step 1: Sign in to your AWS account.

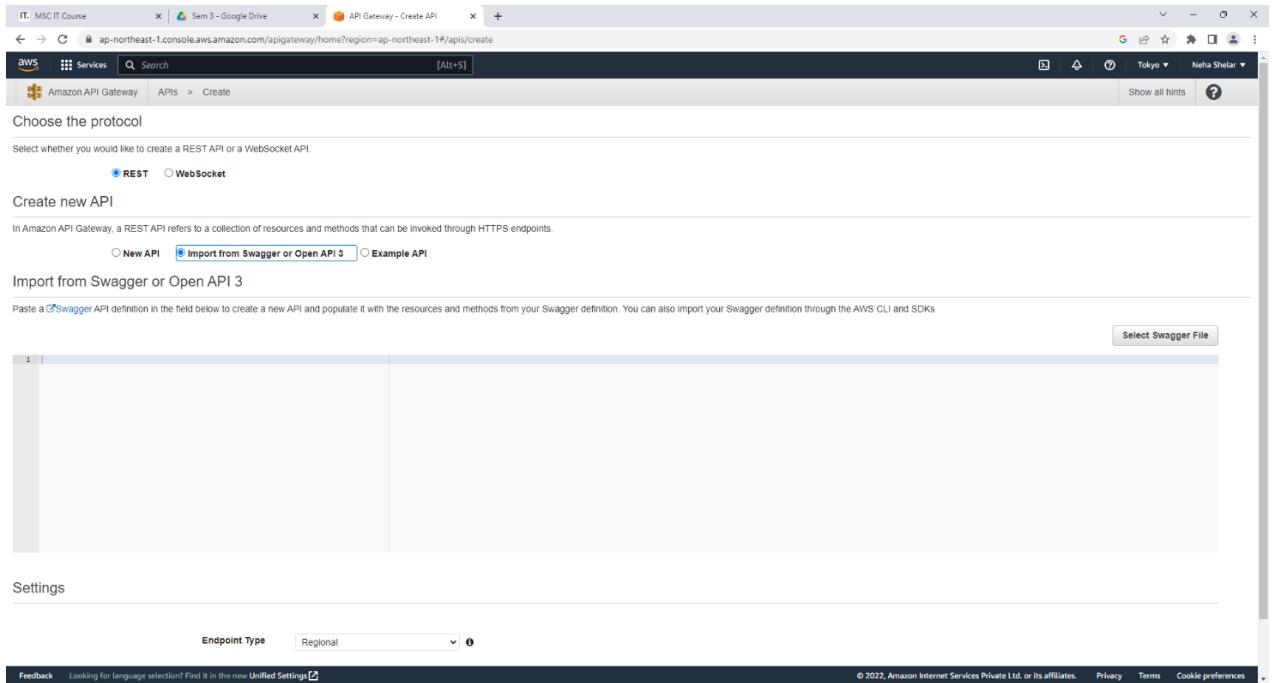
Step 2: search for API Gateway, and create an AWS Gateway instance as shown in Image 1.



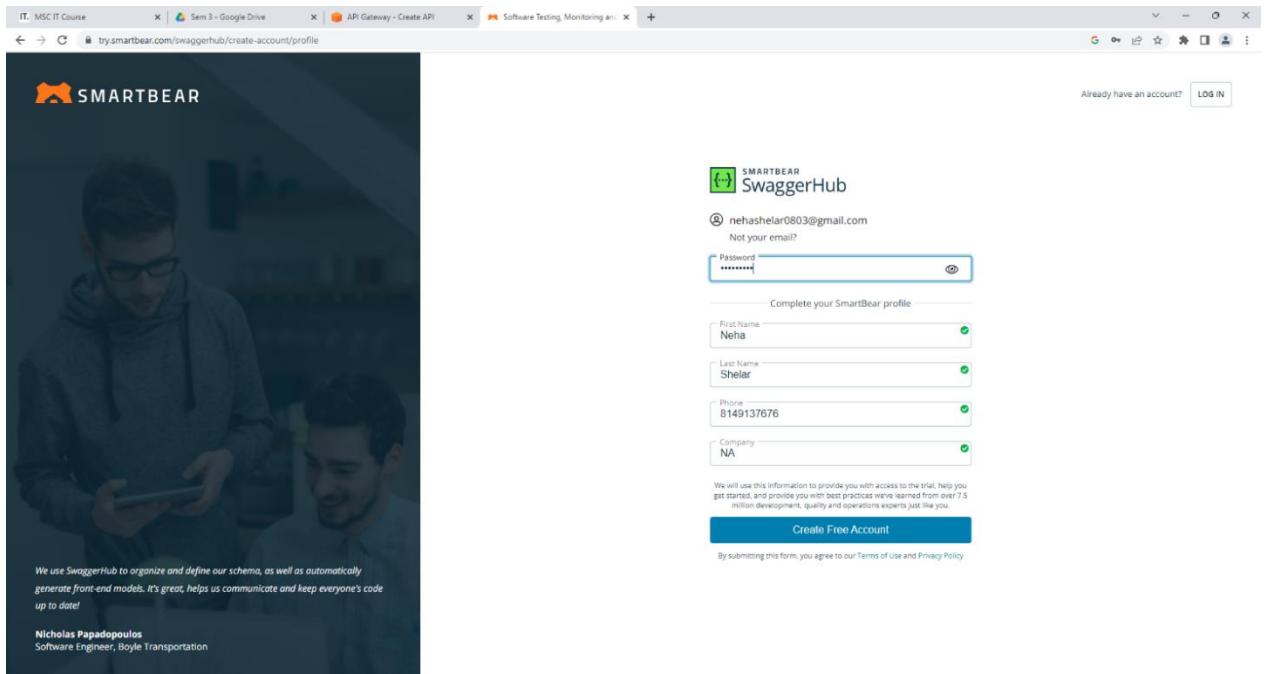
Step 3: Select Import from the REST API selection as shown in image 2.

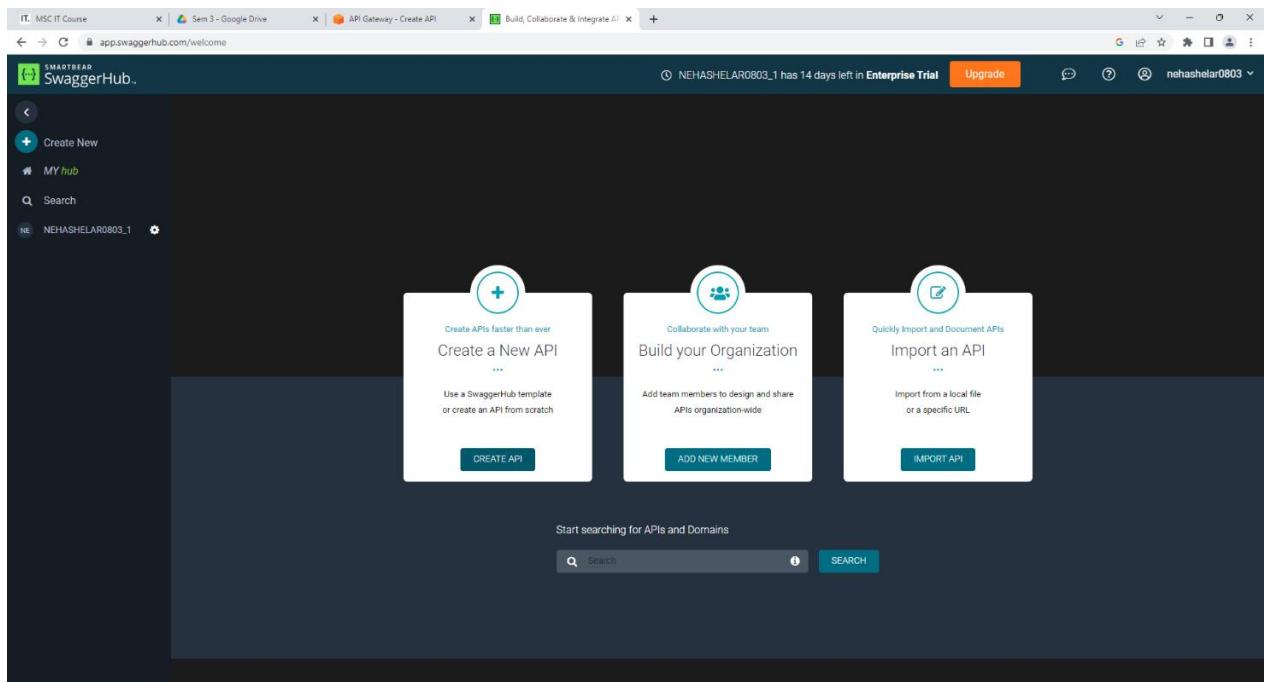


Step 4: Select Import from swagger and copy or select the swagger file as shown in image 3

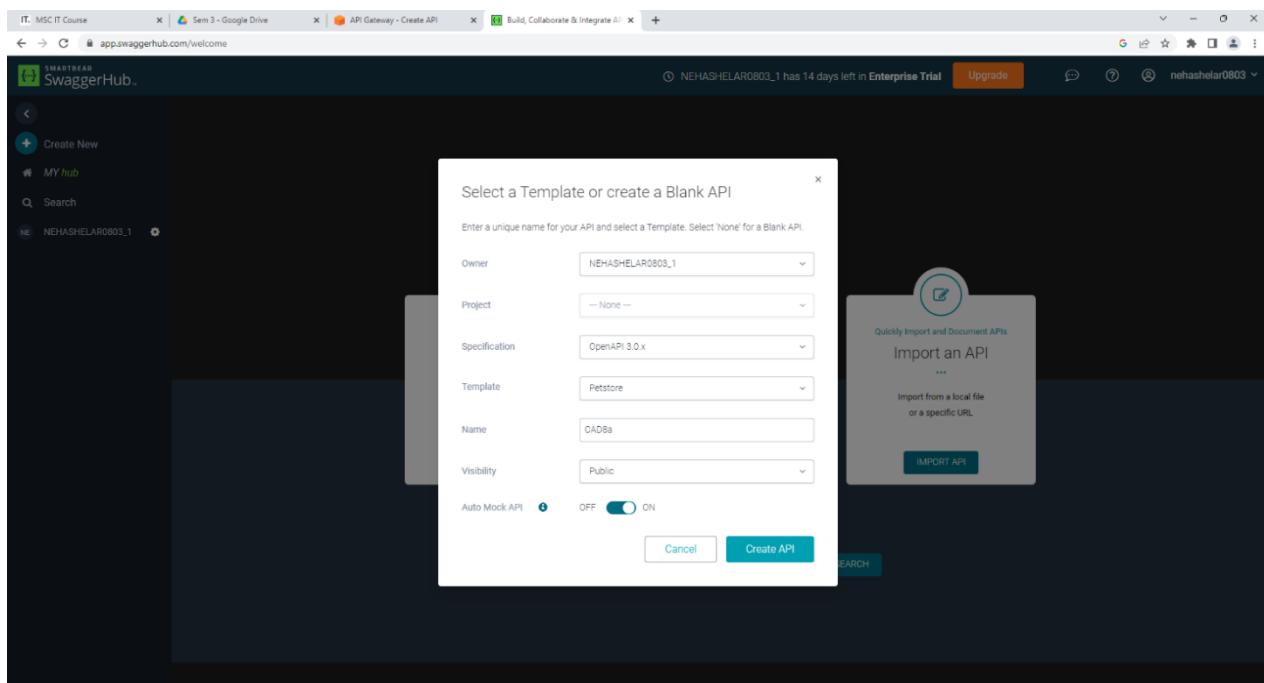


Step 5: To Create an API definition in Swagger. First login to the swagger account. As depicted in image 4.





Step 6: Then fill the Form which includes the version, name and other details click on Create API button. As depicted in image 5.



Step 7: Copy the API and definition from swagger as shown in image 6

The screenshot shows the Swagger Petstore API definition on the SwaggerHub interface. The left sidebar lists various endpoints and their methods (e.g., GET /store/inventory, POST /store/order). The main content area displays the detailed JSON schema for the /user endpoint, specifically focusing on the 'User' schema and its properties like 'username' and 'password'. On the right, there's a preview of the API documentation with examples for 'pet' operations like POST /pet, PUT /pet, and GET /pet. The top navigation bar shows the URL as app.swaggerhub.com/apis/NEHASHELAR0803_1/CAD8a/1.0.0.

Step 8: Paste the definition and in In the Settings section, select the endpoint type. Select “Edge Optimized” option and then click on import button

The screenshot shows the AWS API Gateway 'Create API' page. In the top navigation bar, the URL is ap-northeast-1.console.aws.amazon.com/apigateway/home?region=ap-northeast-1#/apis/create. The main area is titled 'Create new API' and shows the 'Import from Swagger or Open API 3' option selected. A large text input field contains the Swagger JSON definition copied from the previous step. Below the input field, the 'Settings' section is open, showing the 'Endpoint Type' dropdown set to 'Edge optimized'. At the bottom right, there are 'Fail on warnings' and 'Ignore warnings' checkboxes, and a prominent blue 'Import' button.

openapi: 3.0.0

info:

description: |

This is a sample Petstore server. You can find out more about Swagger at

<http://swagger.io> or on

[irc.freenode.net, #swagger](<http://swagger.io/irc/>).

version: "1.0.0"

title: Swagger Petstore

termsOfService: '<http://swagger.io/terms/>'

contact:

email: apiteam@swagger.io

license:

name: Apache 2.0

url: '<http://www.apache.org/licenses/LICENSE-2.0.html>'

servers:

Added by API Auto Mocking Plugin

- description: SwaggerHub API Auto Mocking

url: https://virtserver.swaggerhub.com/NEHASHELAR0803_1/CAD8a/1.0.0

- url: '<https://petstore.swagger.io/v2>'

tags:

- name: pet

description: Everything about your Pets

externalDocs:

description: Find out more

url: '<http://swagger.io>'

- name: store

description: Access to Petstore orders

- name: user

description: Operations about user

```
externalDocs:  
  description: Find out more about our store  
  url: 'http://swagger.io'  
  
paths:  
  /pet:  
    post:  
      tags:  
        - pet  
      summary: Add a new pet to the store  
      operationId: addPet  
      responses:  
        '405':  
          description: Invalid input  
      security:  
        - petstore_auth:  
          - 'write:pets'  
          - 'read:pets'  
      requestBody:  
        $ref: '#/components/requestBodies/Pet'  
    put:  
      tags:  
        - pet  
      summary: Update an existing pet  
      operationId: updatePet  
      responses:  
        '400':  
          description: Invalid ID supplied  
        '404':  
          description: Pet not found  
        '405':
```

```
description: Validation exception
security:
  - petstore_auth:
    - 'write:pets'
    - 'read:pets'
requestBody:
  $ref: '#/components/requestBodies/Pet'
/pet/findByStatus:
  get:
    tags:
      - pet
summary: Finds Pets by status
description: Multiple status values can be provided with comma separated strings
operationId: findPetsByStatus
parameters:
  - name: status
    in: query
    description: Status values that need to be considered for filter
    required: true
    explode: true
    schema:
      type: array
      items:
        type: string
      enum:
        - available
        - pending
        - sold
      default: available
responses:
```

```
'200':  
    description: successful operation  
    content:  
        application/json:  
            schema:  
                type: array  
                items:  
                    $ref: '#/components/schemas/Pet'  
        application/xml:  
            schema:  
                type: array  
                items:  
                    $ref: '#/components/schemas/Pet'  
'400':  
    description: Invalid status value  
    security:  
        - petstore_auth:  
            - 'write:pets'  
            - 'read:pets'  
/pet/findByTags:  
  
get:  
    tags:  
        - pet  
    summary: Finds Pets by tags  
    description: >-  
        Muliple tags can be provided with comma separated strings. Use\\ tag1,  
        tag2, tag3 for testing.  
    operationId: findPetsByTags  
    parameters:
```

```
- name: tags
  in: query
  description: Tags to filter by
  required: true
  explode: true
  schema:
    type: array
    items:
      type: string

  responses:
    '200':
      description: successful operation
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/Pet'

        application/xml:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/Pet'

    '400':
      description: Invalid tag value
      security:
        - petstore_auth:
            - 'write:pets'
            - 'read:pets'
```

```
deprecated: true
'/pet/{petId}':
  get:
    tags:
      - pet
    summary: Find pet by ID
    description: Returns a single pet
    operationId: getPetById
    parameters:
      - name: petId
        in: path
        description: ID of pet to return
        required: true
        schema:
          type: integer
          format: int64
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Pet'
          application/xml:
            schema:
              $ref: '#/components/schemas/Pet'
      '400':
        description: Invalid ID supplied
      '404':
```

description: Pet not found

security:

- api_key: []

post:

tags:

- pet

summary: Updates a pet in the store with form data

operationId: updatePetWithForm

parameters:

- name: petId
in: path
description: ID of pet that needs to be updated
required: true
schema:
 - type: integer
format: int64

responses:

'405':

description: Invalid input

security:

- petstore_auth:
 - 'write:pets'
 - 'read:pets'

requestBody:

content:

application/x-www-form-urlencoded:

schema:

- type: object

properties:

```
name:  
  description: Updated name of the pet  
  type: string  
  
status:  
  description: Updated status of the pet  
  type: string  
  
delete:  
  tags:  
    - pet  
  
  summary: Deletes a pet  
  operationId: deletePet  
  
parameters:  
  - name: api_key  
    in: header  
    required: false  
    schema:  
      type: string  
  - name: petId  
    in: path  
    description: Pet id to delete  
    required: true  
    schema:  
      type: integer  
      format: int64  
  
responses:  
  
  '400':  
    description: Invalid ID supplied  
  '404':  
    description: Pet not found
```

```
  security:  
    - petstore_auth:  
        - 'write:pets'  
        - 'read:pets'  
  
  '/pet/{petId}/uploadImage':  
    post:  
      tags:  
        - pet  
      summary: uploads an image  
      operationId: uploadFile  
      parameters:  
        - name: petId  
          in: path  
          description: ID of pet to update  
          required: true  
          schema:  
            type: integer  
            format: int64  
  
      responses:  
        '200':  
          description: successful operation  
          content:  
            application/json:  
              schema:  
                $ref: '#/components/schemas/ApiResponse'  
  
  security:  
    - petstore_auth:  
        - 'write:pets'  
        - 'read:pets'  
  
  requestBody:
```

```
content:  
application/octet-stream:  
schema:  
  type: string  
  format: binary  
  
/store/inventory:  
get:  
tags:  
- store  
summary: Returns pet inventories by status  
description: Returns a map of status codes to quantities  
operationId: getInventory  
responses:  
'200':  
description: successful operation  
content:  
application/json:  
schema:  
  type: object  
additionalProperties:  
  type: integer  
  format: int32  
security:  
- api_key: []  
  
/store/order:  
post:  
tags:  
- store  
summary: Place an order for a pet  
operationId: placeOrder
```

```
responses:  
  '200':  
    description: successful operation  
    content:  
      application/json:  
        schema:  
          $ref: '#/components/schemas/Order'  
      application/xml:  
        schema:  
          $ref: '#/components/schemas/Order'  
  '400':  
    description: Invalid Order  
requestBody:  
  content:  
    application/json:  
      schema:  
        $ref: '#/components/schemas/Order'  
    description: order placed for purchasing the pet  
    required: true  
/store/order/{orderId}':  
  get:  
    tags:  
      - store  
    summary: Find purchase order by ID  
    description: >-  
      For valid response try integer IDs with value >= 1 and <= 10.\ \ Other  
      values will generated exceptions  
  operationId: getOrderBy Id  
  parameters:  
    - name: orderId
```

```
in: path
description: ID of pet that needs to be fetched
required: true
schema:
  type: integer
  format: int64
  minimum: 1
  maximum: 10
responses:
  '200':
    description: successful operation
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Order'
      application/xml:
        schema:
          $ref: '#/components/schemas/Order'
  '400':
    description: Invalid ID supplied
  '404':
    description: Order not found
delete:
tags:
- store
summary: Delete purchase order by ID
description: >
  For valid response try integer IDs with positive integer value.\\
  Negative or non-integer values will generate API errors
operationId: deleteOrder
```

```
parameters:  
  - name: orderId  
    in: path  
    description: ID of the order that needs to be deleted  
    required: true  
    schema:  
      type: integer  
      format: int64  
      minimum: 1  
  
responses:  
  '400':  
    description: Invalid ID supplied  
  '404':  
    description: Order not found  
  
/user:  
post:  
tags:  
  - user  
summary: Create user  
description: This can only be done by the logged in user.  
operationId: createUser  
responses:  
  default:  
    description: successful operation  
requestBody:  
content:  
  application/json:  
    schema:  
  
$ref: '#/components/schemas/User'
```

```
    description: Created user object
    required: true

/user/createWithArray:

post:
tags:
- user

summary: Creates list of users with given input array
operationId: createUsersWithArrayInput

responses:
default:
description: successful operation

requestBody:
$ref: '#/components/requestBodies/UserArray'

/user/createWithList:

post:
tags:
- user

summary: Creates list of users with given input array
operationId: createUsersWithListInput

responses:
default:
description: successful operation

requestBody:
$ref: '#/components/requestBodies/UserArray'

/user/login:

get:
tags:
- user

summary: Logs user into the system
operationId: loginUser
```

parameters:

- name: username

- in: query

- description: The user name for login

- required: true

- schema:

- type: string

- name: password

- in: query

- description: The password for login in clear text

- required: true

- schema:

- type: string

responses:

'200':

- description: successful operation

- headers:

- X-Rate-Limit:

- description: calls per hour allowed by the user

- schema:

- type: integer

- format: int32

- X-Expires-After:

- description: date in UTC when token expires

- schema:

- type: string

- format: date-time

- content:

- application/json:

- schema:

```
    type: string
  application/xml:
    schema:
      type: string
  '400':
    description: Invalid username/password supplied
/user/logout:
  get:
    tags:
      - user
    summary: Logs out current logged in user session
    operationId: logoutUser
  responses:
    default:
      description: successful operation
'/user/{username}':
  get:
    tags:
      - user
    summary: Get user by user name
    operationId: getUserByName
  parameters:
    - name: username
      in: path
      description: The name that needs to be fetched. Use user1 for testing.
      required: true
    schema:
      type: string
  responses:
```

```
'200':  
    description: successful operation  
    content:  
        application/json:  
            schema:  
                $ref: '#/components/schemas/User'  
        application/xml:  
            schema:  
                $ref: '#/components/schemas/User'  
'400':  
    description: Invalid username supplied  
'404':  
    description: User not found  
  
put:  
tags:  
- user  
summary: Updated user  
description: This can only be done by the logged in user.  
operationId: updateUser  
parameters:  
- name: username  
  in: path  
  description: name that need to be updated  
  required: true  
  schema:  
    type: string  
responses:  
'400':  
    description: Invalid user supplied  
'404':
```

```
description: User not found
requestBody:
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/User'
  description: Updated user object
  required: true
delete:
  tags:
    - user
  summary: Delete user
  description: This can only be done by the logged in user.
  operationId: deleteUser
  parameters:
    - name: username
      in: path
      description: The name that needs to be deleted
      required: true
      schema:
        type: string
  responses:
    '400':
      description: Invalid username supplied
    '404':
      description: User not found
externalDocs:
  description: Find out more about Swagger
  url: 'http://swagger.io'
```

components:

schemas:

Order:

type: object

properties:

id:

type: integer

format: int64

petId:

type: integer

format: int64

quantity:

type: integer

format: int32

shipDate:

type: string

format: date-time

status:

type: string

description: Order Status

enum:

- placed

- approved

- delivered

complete:

type: boolean

default: false

xml:

name: Order

Category:

type: object

properties:

id:

type: integer

format: int64

name:

type: string

xml:

name: Category

User:

type: object

properties:

id:

type: integer

format: int64

username:

type: string

firstName:

type: string

lastName:

type: string

email:

type: string

password:

type: string

phone:

type: string

userStatus:

```
type: integer
format: int32
description: User Status

xml:
  name: User

Tag:
type: object
properties:
  id:
    type: integer
    format: int64
  name:
    type: string

xml:
  name: Tag

Pet:
type: object
required:
  - name
  - photoUrls
properties:
  id:
    type: integer
    format: int64
  category:
    $ref: '#/components/schemas/Category'
  name:
    type: string
    example: doggie
  photoUrls:
```

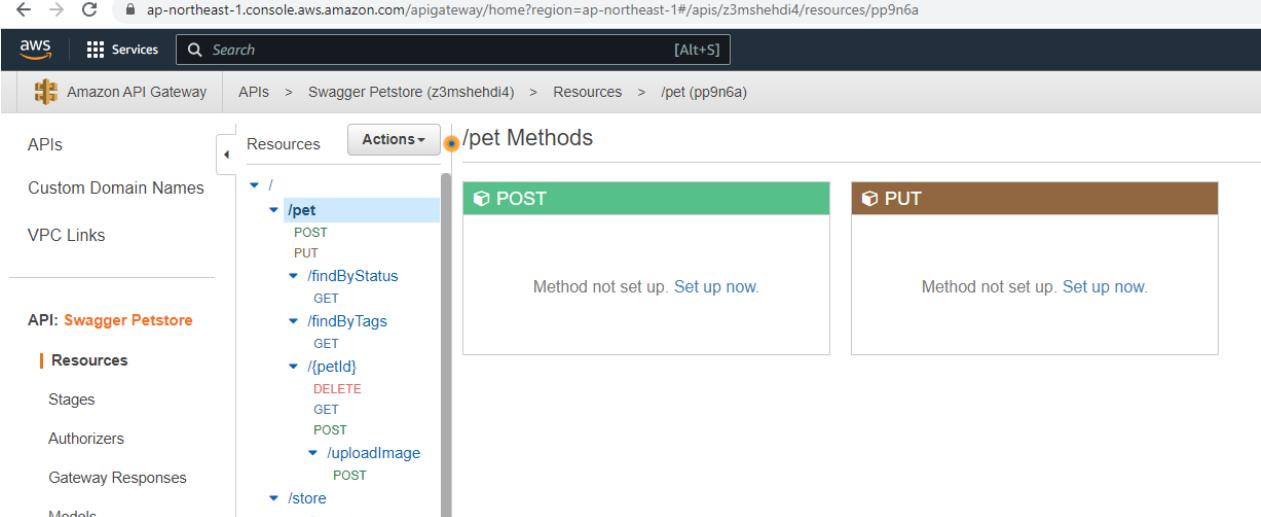
```
type: array
xml:
  name: photoUrl
  wrapped: true
items:
  type: string
tags:
  type: array
  xml:
    name: tag
    wrapped: true
items:
$ref: '#/components/schemas/Tag'
status:
  type: string
  description: pet status in the store
enum:
  - available
  - pending
  - sold
xml:
  name: Pet
ApiResponse:
  type: object
  properties:
    code:
      type: integer
      format: int32
    type:
      type: string
```

```
message:  
  type: string  
  
requestBodies:  
  Pet:  
    content:  
      application/json:  
        schema:  
          $ref: '#/components/schemas/Pet'  
      application/xml:  
        schema:  
          $ref: '#/components/schemas/Pet'  
  
    description: Pet object that needs to be added to the store  
    required: true  
  
  UserArray:  
    content:  
      application/json:  
        schema:  
          type: array  
          items:  
            $ref: '#/components/schemas/User'  
  
    description: List of user object  
    required: true  
  
  securitySchemes:  
    petstore_auth:  
      type: oauth2  
  
  
flows:  
  implicit:  
    authorizationUrl: 'http://petstore.swagger.io/oauth/dialog'  
    scopes:
```

```
'write:pets': modify pets in your account  
'read:pets': read your pets  
  
api_key:  
  type: apiKey  
  name: api_key  
  in: header
```

Output:

If the import is successful, we can view the “API Structure in AWS API Gateway” where we can see methods of the API. As shown in image 8



The screenshot shows the AWS API Gateway console with the following details:

- Region:** ap-northeast-1
- API:** Swagger Petstore
- Resource Path:** /pet
- Actions:** POST and PUT
- Sub-resources:** /findByStatus, /findByTags, /{petId}, /uploadImage
- Sub-methods:** GET, DELETE, POST
- Notes:** Method not set up. Set up now.

Practical No: 05

Aim: 5a Demonstrate - Securing APIs with Azure Active Directory

Step 1 – Login to Azure Portal and select/search Azure Active Directory service.

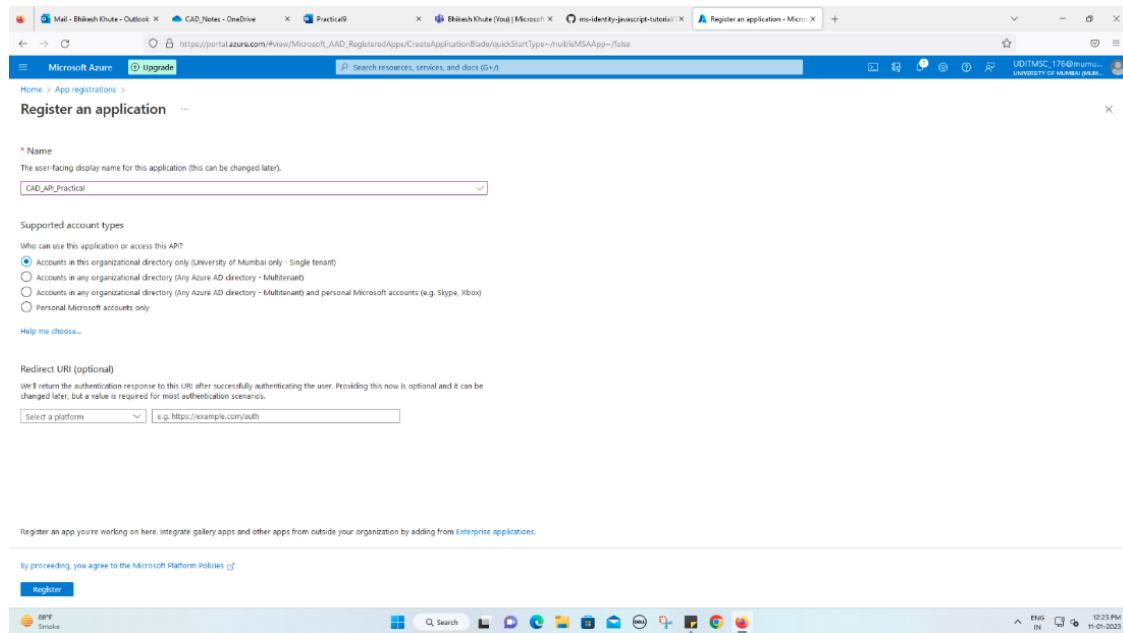
Step 2 - Select the App Registrations blade on the left, then select new registration.

Application (client) ID	Created on	Certificates & secrets
ea575024-551c-4534-8a7a-e1e46a9e341c	1/10/2023	-
b6d20f76-20aa-47a7-bd59-0ddabc0d42a29	1/10/2023	-
f72023db-c715-42b1-92af-1c8007be0bbf	1/11/2023	-

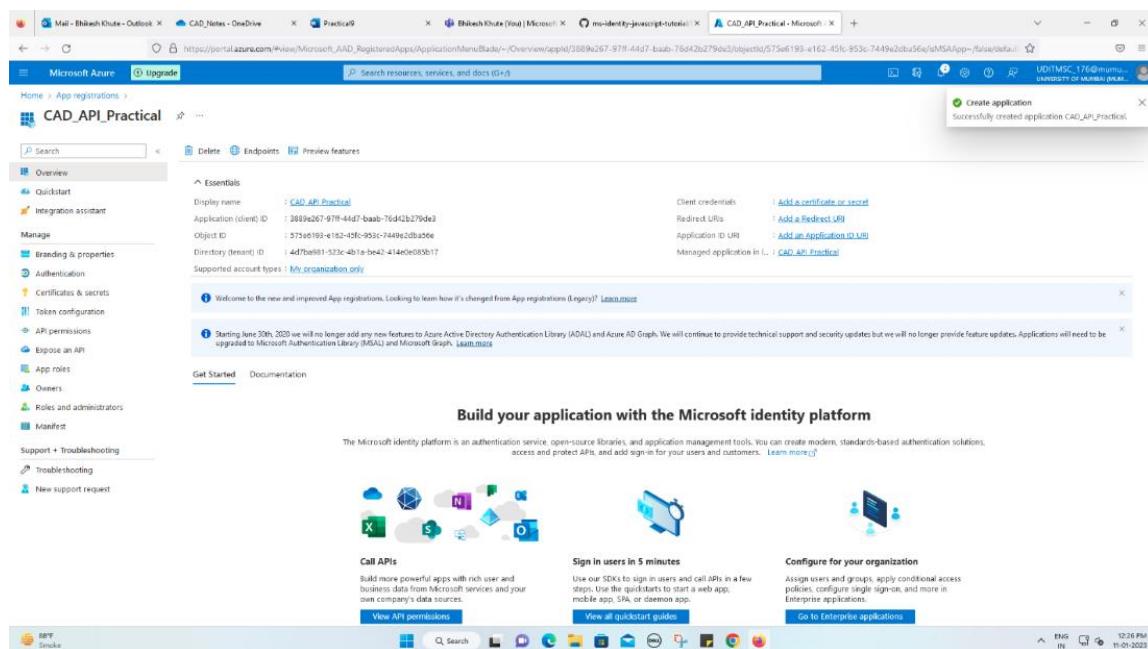
Step 3 – Click on New Registration and

- I. add the name of the app as “CAD_API_Practical”.
- II. Accounts in this organizational directory only (University of Mumbai only - Single tenant)
- III. Ignore the Redirect URI(We'll see it in the next step)

And click on Register



Step 4 – The App is created.



Note down the **Application & Tenant ID** for further use for authentication.

Step 5 – Now click **Add a Redirect URI** & click on **+Add Platform**. Select **Single-page Application**

The screenshot shows the Azure portal interface for managing app registrations. On the left, a sidebar lists various configuration sections like Overview, Quickstart, Integration assistant, Manage, Branding & properties, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, App roles, Owners, Roles and administrators, and Manifest. The 'Authentication' section is currently selected.

In the main content area, the title is 'CAD_API_Practical | Authentication'. Under 'Platform configurations', it says 'Depending on the platform or device this application is targeting, additional configuration may be required such as redirect URIs, specific authentication settings, or fields specific to the platform.' A link '+ Add a platform' is present.

The 'Supported account types' section shows two options: 'Accounts in this organizational directory only (University of Mumbai only - Single tenant)' (selected) and 'Accounts in any organizational directory (Any Azure AD directory - Multitenant)'. A note states: 'Due to temporary differences in supported functionality, we don't recommend enabling personal Microsoft accounts for an existing registration. If you need to enable personal accounts, you can do so using the manifest editor. Learn more about these restrictions.'

The 'Advanced settings' section includes 'Allow public client flows' (checkbox checked), 'Enable the following mobile and desktop flows' (checkbox checked), and a note: 'App collects plaintext password (Resource Owner Password Credential flow) Learn more', 'No keyboard (Device Code Flow) Learn more', and 'SSO for domain-joined Windows (Windows Integrated Auth Flow) Learn more'.

The 'App instance property lock' section has a note: 'Configure the application instance modification lock. Learn more'.

At the bottom right of the main content area, there are 'Save' and 'Discard' buttons. The status bar at the bottom indicates '12:29 PM 11-01-2023'.

Step 6 – Add Redirect URI as <http://localhost:3000> & check Access & ID tokens mentioned in the screenshot

This screenshot shows the 'Configure single-page application' section of the Azure portal. The left sidebar is identical to the previous screenshot, with 'Authentication' selected.

The main content area title is 'CAD_API_Practical | Authentication'. The 'Configure single-page application' section has a note: 'The latest version of MSAL.js uses the authorization code flow with PKCE and CORS. Learn more'.

The 'Redirect URIs' section contains a single entry: 'http://localhost:3000'. A note explains: 'The URL must be a domain that receives authentication responses (tokens) after a user successfully authenticates or signs out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs. Learn more about Redirect URIs and their restrictions.'

The 'Front-channel logout URL' section has a note: 'This is where we send a request to have the application clear the user's session data. This is required for single-sign-out to work correctly.' An input field shows 'e.g. https://example.com/logout'.

The 'Grant types' section notes: 'MSAL.js 2.0 does not support implicit grant. Enable implicit grant settings only if your app is using MSAL.js 1.0. Learn more about auth code flow.' A note says: 'Your Redirect URI is eligible for the Authorization Code Flow with PKCE.'

The 'Implicit grant and hybrid flows' section has a note: 'Request a token directly from the authorization endpoint. If the application has a single-page architecture (SPA) and doesn't use the authorization code flow, or if it makes a web API call via JavaScript, select both access tokens and ID tokens. For ASP.NET Core web apps and other web apps that use hybrid authentication, select only ID tokens. Learn more about tokens.'

Under 'Select the tokens you would like to be issued by the authorization endpoint:', two checkboxes are checked: 'Access tokens (used for implicit flows)' and 'ID tokens (used for implicit and hybrid flows)'.

At the bottom right, there are 'Configure' and 'Cancel' buttons. The status bar at the bottom indicates '12:29 PM 11-01-2023'.

Step 7 – Click on Configure and output is as follow

The screenshot shows the Microsoft Azure portal's 'App registrations' section for the 'CAD_API_Practical' application. Under the 'Authentication' tab, the 'Single-page application' configuration is displayed. The 'Add URI' field contains 'http://localhost:3000'. A success message 'Successfully updated CAD_API_Practical' is shown in the top right corner.

Step 8 – Now download the application from Github from the below link for secure authentication to the API we created.

git clone <https://github.com/Azure-Samples/ms-identity-javascript-tutorial.git>

```
MINGW64:/c/Users/Admin/Desktop/CADPRAC
$ pwd
/c/Users/Admin/Desktop/CADPRAC
Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC
$ git clone https://github.com/Azure-Samples/ms-identity-javascript-tutorial.git
Cloning into 'ms-identity-javascript-tutorial'...
remote: Enumerating objects: 2206, done.
remote: Counting objects: 100% (433/433), done.
remote: Compressing objects: 100% (193/193), done.
remote: Total 2206 (delta 343), reused 240 (delta 240), pack-reused 1773
Receiving objects: 100% (2206/2206), 2.01 MiB | 4.48 MiB/s, done.
Resolving deltas: 100% (1391/1391), done.

Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC
$ ls ms-identity-javascript-tutorial/1-Authentication/
1-sign-in/ 2-sign-in/b2c/
Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC
$ ls ms-identity-javascript-tutorial/1-Authentication/1-sign-in/
App/ AppCreationScripts/ README-Incremental.md README.md READMEFiles/ package-lock.json package.json sample.test.js server.js
Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC
$ |
```

Step 9 – Install NodeJs on the system using the following link -

<https://nodejs.org/en/download/>

Step 10 – Install the project dependencies using the following command in the mentioned directory below

<folder>/ms-identity-javascript-tutorial/1-Authentication/1-sign-in

npm install

```

MINGW64:/c/Users/Admin/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in
$ ls -ltr
total 404
drwxr-xr-x 1 Admin 197121 0 Jan 11 12:35 App/
-rw-r--r-- 1 Admin 197121 18392 Jan 11 12:35 README.md
-rw-r--r-- 1 Admin 197121 17383 Jan 11 12:35 README-incremental.md
drwxr-xr-x 1 Admin 197121 0 Jan 11 12:35 AppCreationScripts/
-rw-r--r-- 1 Admin 197121 350545 Jan 11 12:35 package-lock.json
drwxr-xr-x 1 Admin 197121 0 Jan 11 12:35 ReadmeFiles/
-rw-r--r-- 1 Admin 197121 1867 Jan 11 12:35 server.js
-rw-r--r-- 1 Admin 197121 2120 Jan 11 12:35 sample.test.js
-rw-r--r-- 1 Admin 197121 1129 Jan 11 12:35 package.json

Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in (main)
$ npm install
npm WARN deprecated w3c-hr-time@1.0.2: Use your platform's native performance.now() and performance.timeOrigin.
npm WARN deprecated formidable@1.2.6: Please upgrade to latest, formidable@v2 or formidable@v3! Check these notes: https://bit.ly/2ZEQIau
npm WARN deprecated superagent@6.1.0: Please upgrade to v7.0.2+ of superagent. We have fixed numerous issues with streams, form-data, attach(), filesystem errors not bubbling up (ENDENT on attach()), and all tests are now passing. See the releases tab for more information at <https://github.com/visionmedia/superagent/releases>.

added 430 packages, and audited 431 packages in 1m

34 packages are looking for funding
  run 'npm fund' for details

8 low severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.
npm notice
npm notice New major version of npm available! 8.19.3 -> 9.2.0
npm notice Changelog: <https://github.com/npm/cli/releases/tag/v9.2.0>
npm notice Run 'npm install -g npm@9.2.0' to update!
npm notice

Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in (main)
$ 
```

Step 11 – Now Open the App\authConfig.js file to add the App ID & Directory ID for authentication.

Go the path - /ms-identity-javascript-tutorial/1-Authentication/1-sign-in/App
 Filname - authConfig.js

```

MINGW64:/c/Users/Admin/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in/App
$ ls
App/ AppCreationScripts/ README-incremental.md README.md ReadmeFiles/ node_modules/ package-lock.json package.json sample.test.js server.js

Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in (main)
$ cd App

Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in/App (main)
$ ls
authConfig.js authPopup.js authRedirect.js favicon.svg index.html redirect.html signout.html styles.css ui.js

Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in/App (main)
$ pwd
/c/Users/Admin/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in/App

Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in/App (main)
$ ls
authConfig.js authPopup.js authRedirect.js favicon.svg index.html redirect.html signout.html styles.css ui.js

Admin@DESKTOP-NVDIEB4 MINGW64 ~/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in/App (main)
$ 
```

Step 12 – Copy the Application ID & Directory ID from the Overview Section

The screenshot shows the Microsoft Azure portal's App registrations section. The application 'CAD_API_Practical' is selected. In the 'Overview' tab, the following details are visible:

- Display name:** CAD API Practical
- Application (client) ID:** 3889e267-97ff-44d7-baab-76d42b279de3
- Object ID:** 575e6193-e162-45fc-953c-7449e2dba56e
- Directory (tenant) ID:** 4d7ba981-523c-4b1a-be42-414e0e085b17
- Supported account types:** My organization only
- Client credentials:** Add a certificate or secret
- Redirect URLs:** 0 web, 1 spa, 0 public client
- Application ID URI:** Add an Application ID URI
- Managed application in L:** CAD API Practical

Step 13 – Paste the ID in the authConfig.js file mentioned in the screenshot

```
/* Configuration object to be passed to MSAL instance on creation.
 * For a full list of MSAL.js configuration parameters, visit:
 * https://github.com/AzureAD/microsoft-authentication-library-for-js/blob/dev/lib/msal-browser/docs/configuration.md
 */
const msalConfig = {
  auth: [
    clientId: '3889e267-97ff-44d7-baab-76d42b279de3', // This is the ONLY mandatory field that you need to supply.
    authority: 'https://login.microsoftonline.com/4d7ba981-523c-4b1a-be42-414e0e085b17', // Defaults to "https://login.microsoftonline.com/common"
    redirectUri: '/', // You must register this URI on Azure Portal/App Registration. Defaults to window.location.href e.g. http://localhost:3000/
    navigateToLoginRequestUrl: true, // If "true", will navigate back to the original request location before processing the auth code response.
  ],
  cache: [
    cacheLocation: 'sessionStorage', // Configures cache location. "sessionStorage" is more secure, but "localStorage" gives you SSO.
    storeAuthStateInCookie: false, // set this to true if you have to support IE
  ],
  system: [
    loggerOptions: {
      loggerCallback: (level, message, containsPii) => {
        if (containsPii) {
          return;
        }
        switch (level) {
          case msal.LogLevel.Error:
            console.error(message);
            return;
          case msal.LogLevel.Info:
            console.info(message);
            return;
          case msal.LogLevel.Verbose:
            console.debug(message);
            return;
          case msal.LogLevel.Warning:
            console.warn(message);
            return;
        }
      },
    },
  ],
};
```

Step 14 – Save the file and run the following command

Npm start

```
MINGW64:/c/Users/Admin/Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in/App
Administrator@DESKTOP-NVDIEB4 MINGW64 ~ /Desktop/CADPRAC/ms-identity-javascript-tutorial/1-Authentication/1-sign-in/App (main)
$ npm start
> ms-identity-javascript-clis@1.0.0 start
> node server.js
Sample app listening on port 3000!
```

Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of Node.js JavaScript Runtime on all public and private networks.

Name: Node.js JavaScript Runtime
Publisher: Node.js
Path: C:\program files\nodejs\node.exe

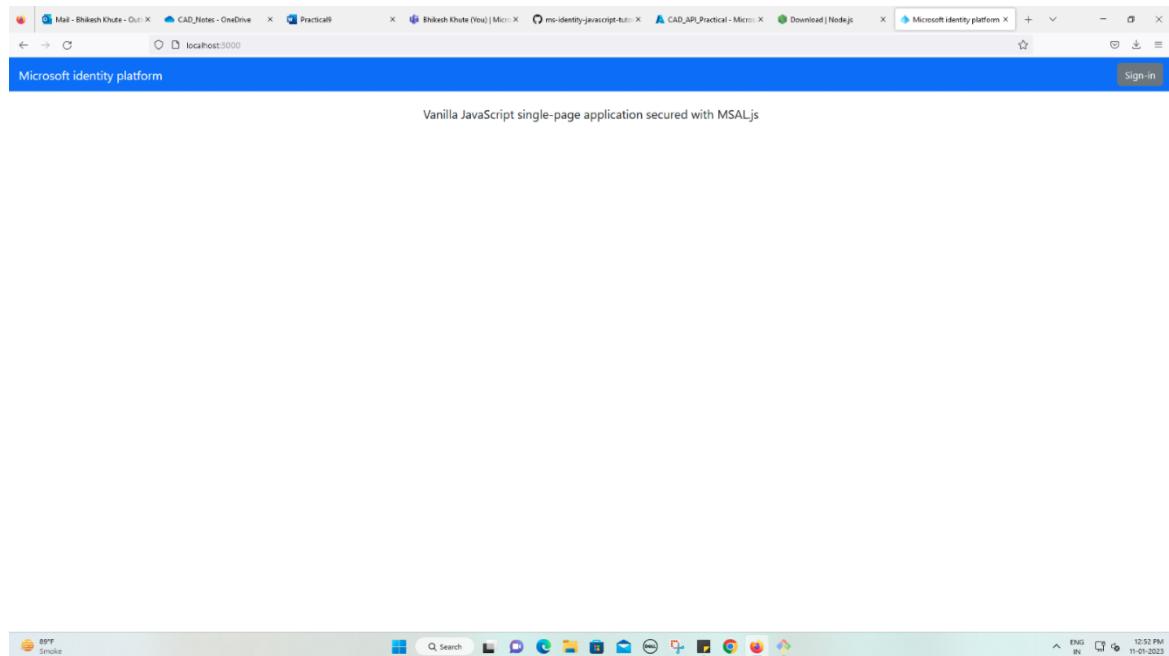
Allow Node.js JavaScript Runtime to communicate on these networks:

Private networks, such as my home or work network
 Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

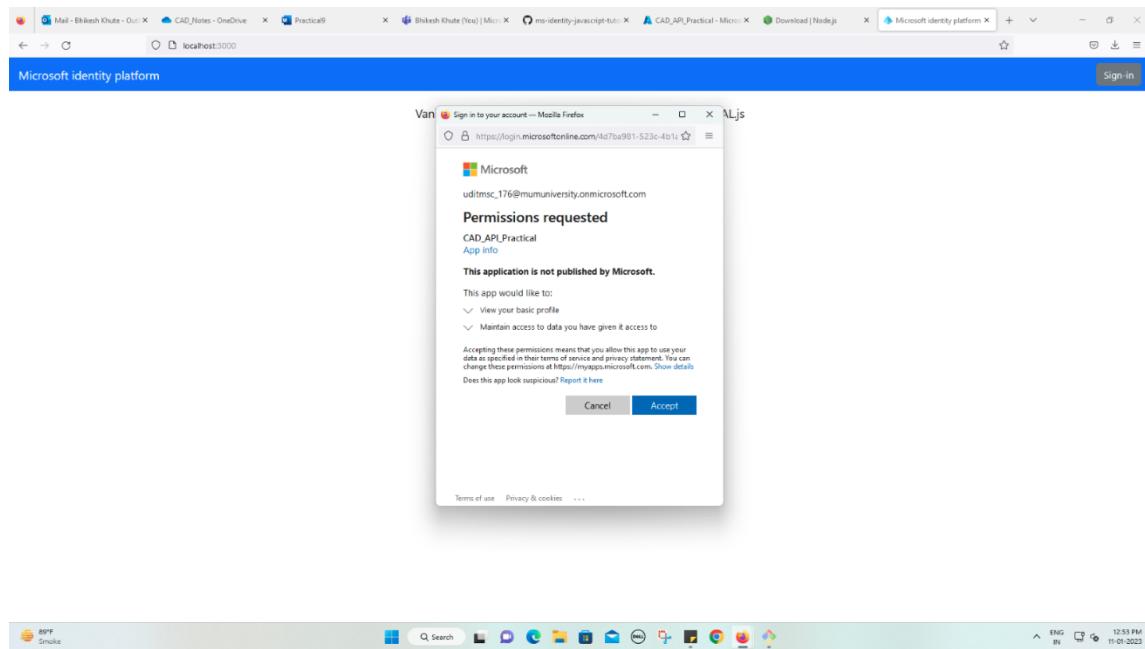
What are the risks of allowing an app through a firewall?

Allow access Cancel

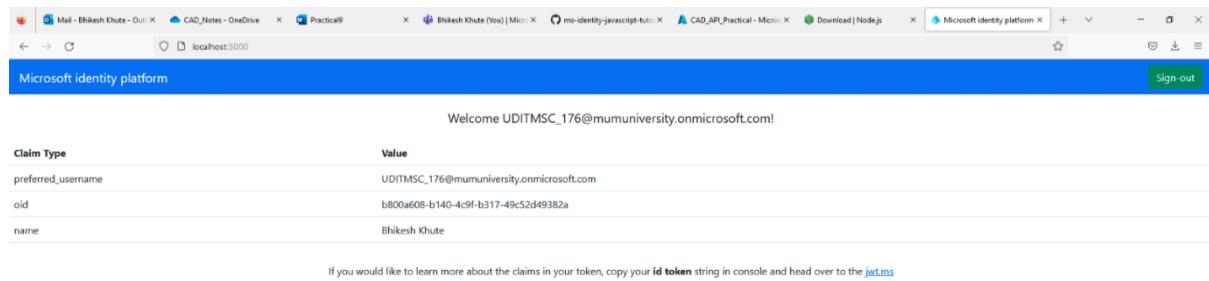
Step 15 – Now go the following URL on the browser – <http://localhost:3000>



Step 16 – Click on Sign In. A popup will appear



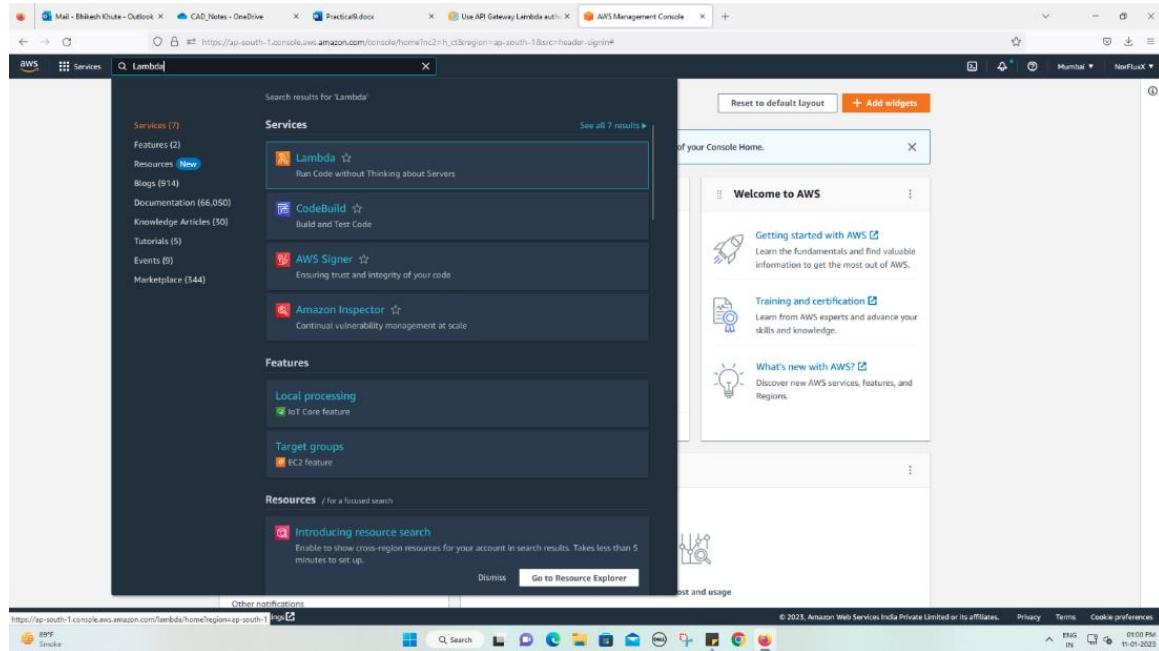
Step 17 – We have successfully signed in using AD securely.



The above following are details about my AD details.

Aim:5B - AWS API Gateway Authorizer

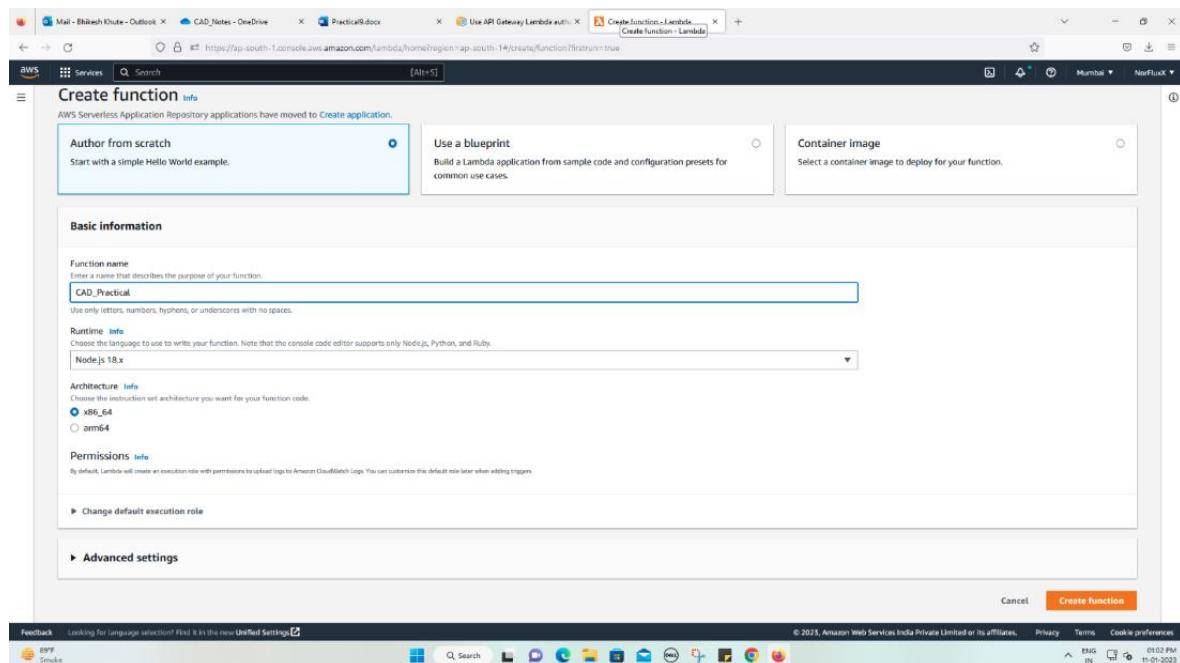
Step 1 – Login to aws.amazon.com and Search for “Lambda”



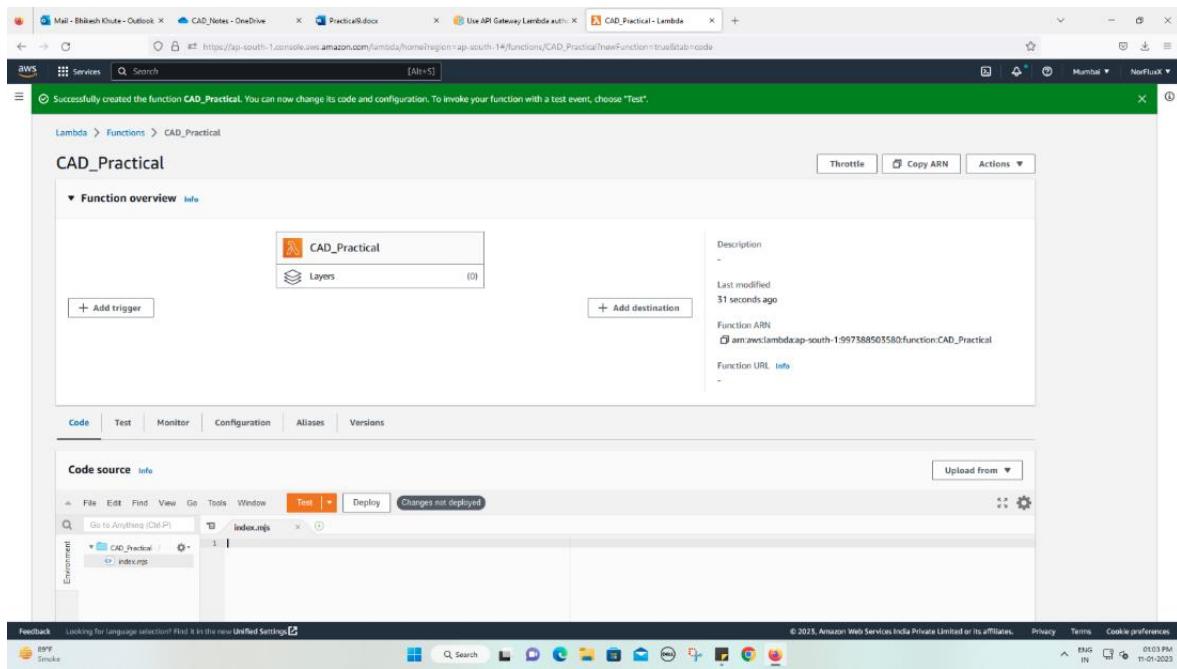
Step 2 – Click on Create a Function

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",

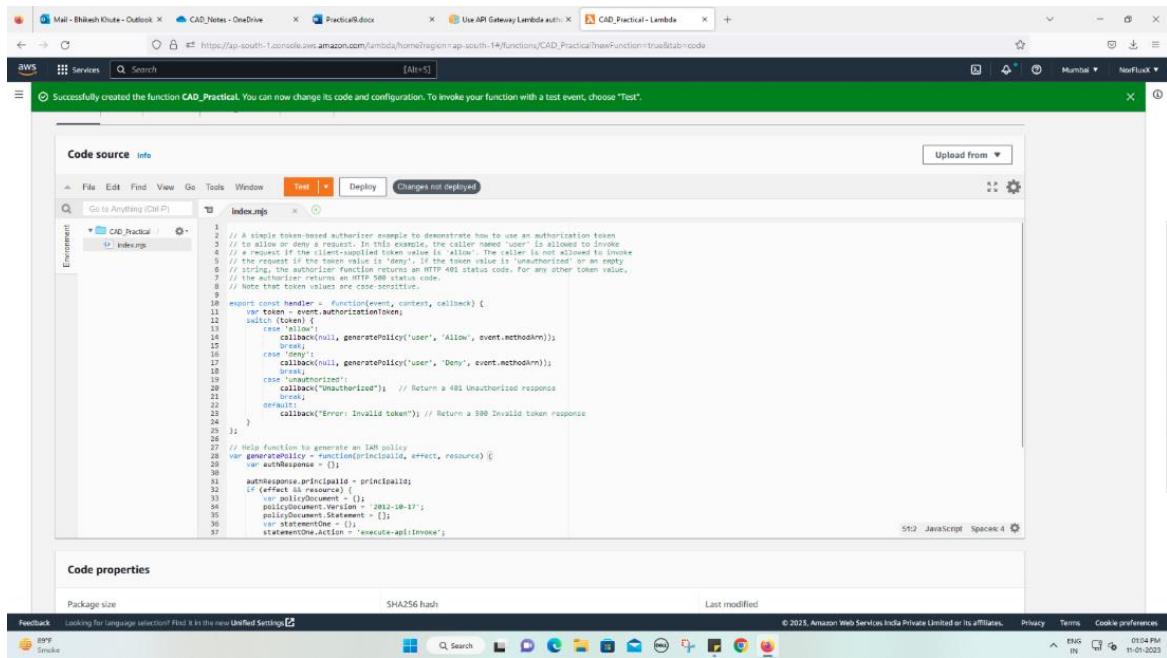
        "numberKey": 123,

        "booleanKey": true

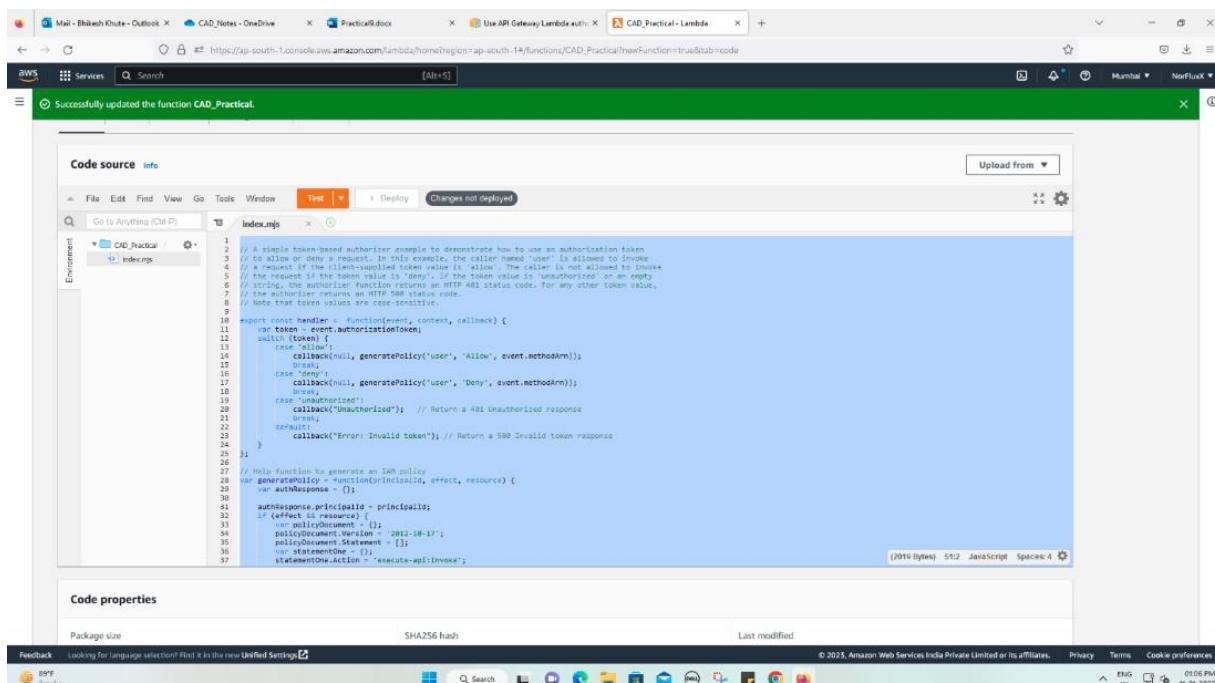
    };

    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

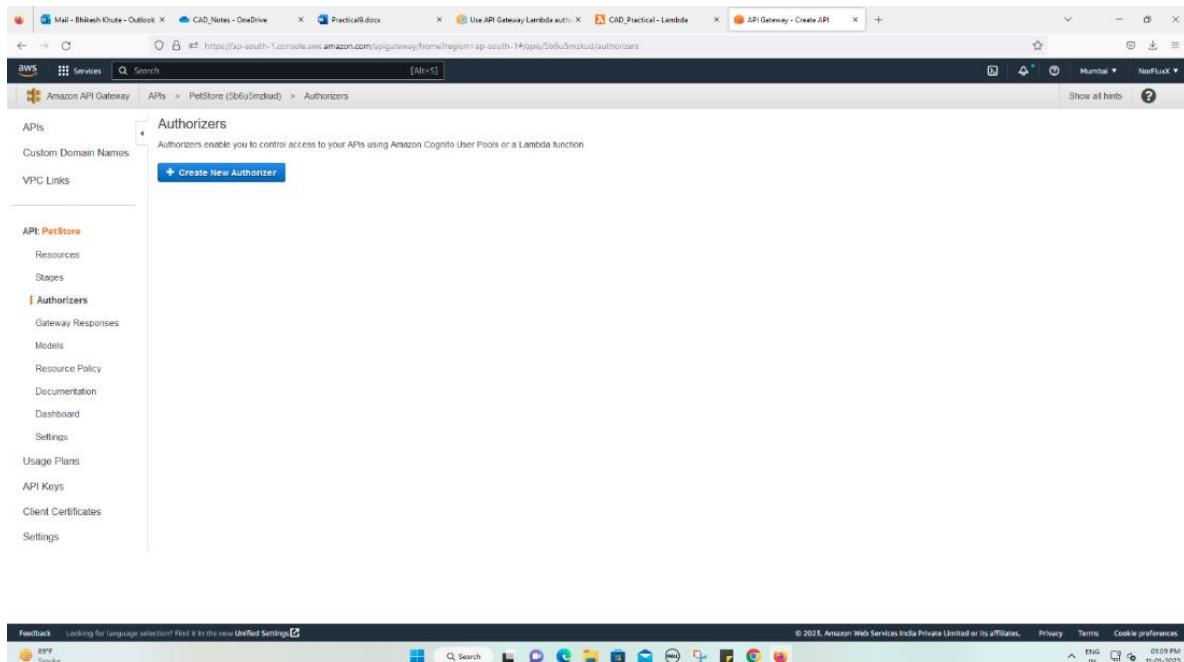
The screenshot shows the AWS API Gateway 'Create API' interface. At the top, there are several tabs: 'Mail - Bhilash Khute - Outlook', 'CAD_Notes - OneDrive', 'Practical9.docx', 'Use API Gateway Lambda auth', 'CAD_Practical - Lambda', and 'API Gateway - Create API'. The 'API Gateway - Create API' tab is active. Below the tabs, the URL is https://ap-south-1.console.aws.amazon.com/api-gateway/home?region=ap-south-1#/apis/create. The main area contains a large code block representing a Swagger definition:

```

149     "responses": {
150       "default": {
151         "statusCode": "200",
152         "responseParameters": {
153           "method.response.header.Access-Control-Allow-Origin": "*"
154         }
155       }
156     },
157     "url": "http://petstore.execute-api.ap-south-1.amazonaws.com/petstore/pets",
158     "passThroughBehavior": "when_no_match",
159     "httpMethod": "POST",
160     "type": "http"
161   },
162   "options": {
163     "consumes": [
164       "application/json"
165     ],
166     "produces": [
167       "application/json"
168     ],
169     "responses": {
170       "200": {
171         "description": "Successful operation",
172         "schema": {
173           "$ref": "#/definitions/Empty"
174         },
175         "headers": {
176           "Access-Control-Allow-Origin": [
177             "www.sample.com"
178           ]
179         }
180       }
181     }
182   }
183 }
```

Below the code, there is a 'Settings' section with an 'Endpoint Type' dropdown set to 'Regional'. At the bottom right of the settings section are two buttons: 'Fail on warnings' (checked) and 'Ignore warnings', followed by a blue 'Import' button.

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 Grant & Create

Step 12 – Now click on the test button

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 Authorizers (which is selected). Under APIs, it shows PetStore. The main content area is titled "Authorizers" and contains details for "CAD_AWS_API_PRACTICAL". It includes sections for "Lambda Function" (set to CAD_Practical), "Lambda Event Payload" (Token), "Token Source" (authorizationToken), "Token Validation" (none), and "Authorization Caching" (Authorization cached for 5 minutes). At the bottom right of this card are "Edit" and "Test" buttons. The URL in the browser is https://ap-south-1.console.aws.amazon.com/apigateway/home?region=ap-south-1#/apis/5b6u5mzkud/authorizers.

Step 13 – Type **allow** in the box as mentioned in the screenshot

This screenshot shows the "Test" dialog for the CAD_AWS_API_PRACTICAL authorizer. In the "Authorization Token" field, the value "allow" is entered. Below the dialog, the "Response" section displays the IAM policy output. The policy is as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      "Resource": "arn:aws:execute-api:ap-south-1:997388503540:5b6u5mzkud"
    }
  ]
}
```

The "Execution Log" at the bottom of the dialog shows the request ID and the start of the authorization process. The URL in the browser is https://ap-south-1.console.aws.amazon.com/apigateway/home?region=ap-south-1#/apis/5b6u5mzkud/authorizers.

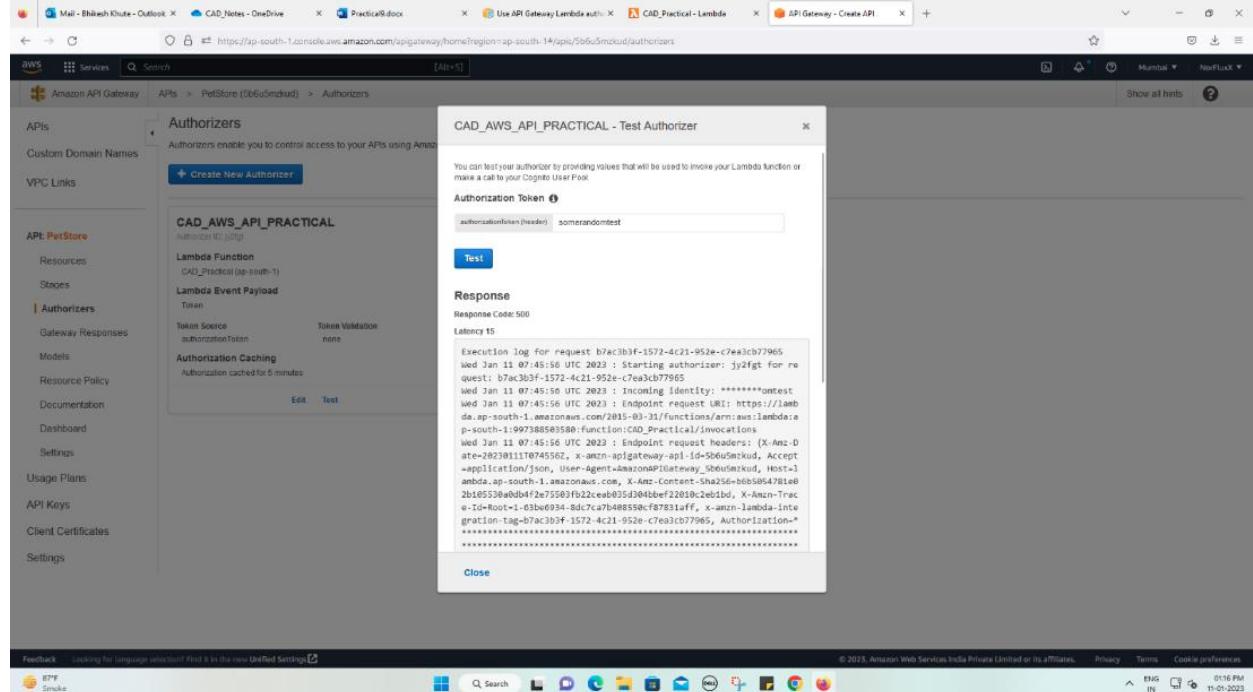
We can see that the function we created is tokenized. When we input allow, it will check the IAM policy and return policy output of the same.

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: 06

Aim: Create a serverless API using Azure functions

Step 1 - Sign in with your Microsoft account to access various Azure functionalities

Step 2 - After Successful sign in, Click on Function app→Click on Create Function App

Step 3 - Click on Resource Group →Create New

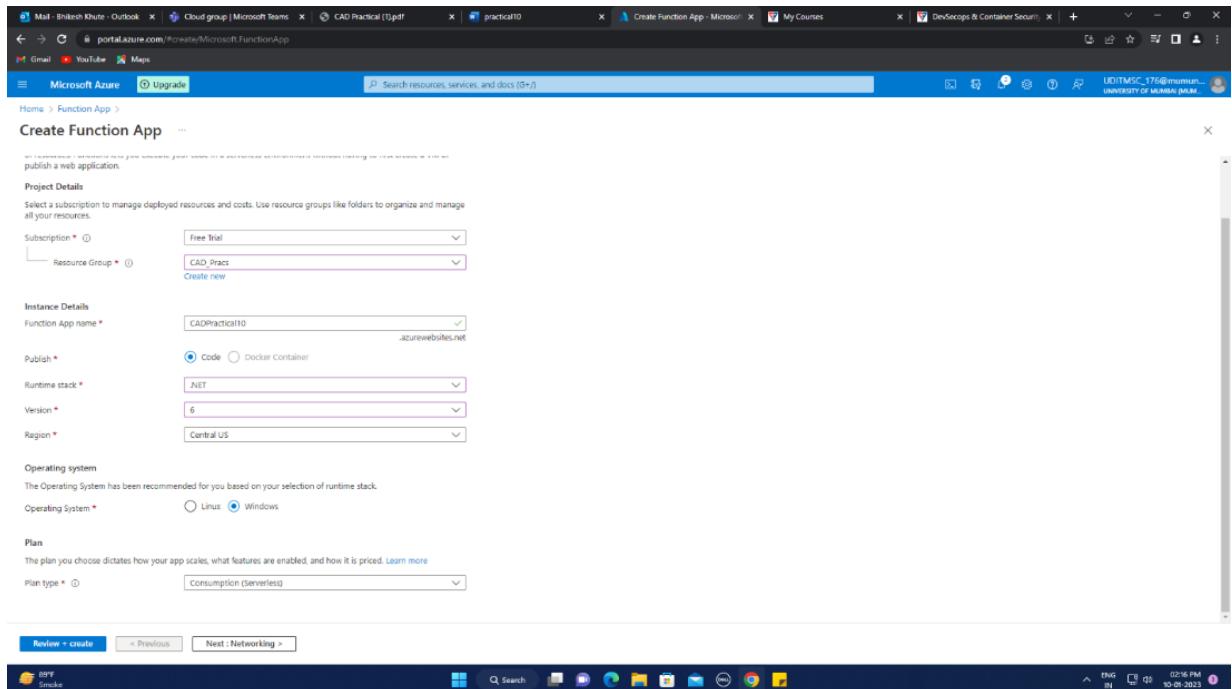
Resource Group Name: CAD10_Practical

Function App Name: CAD10_Practical

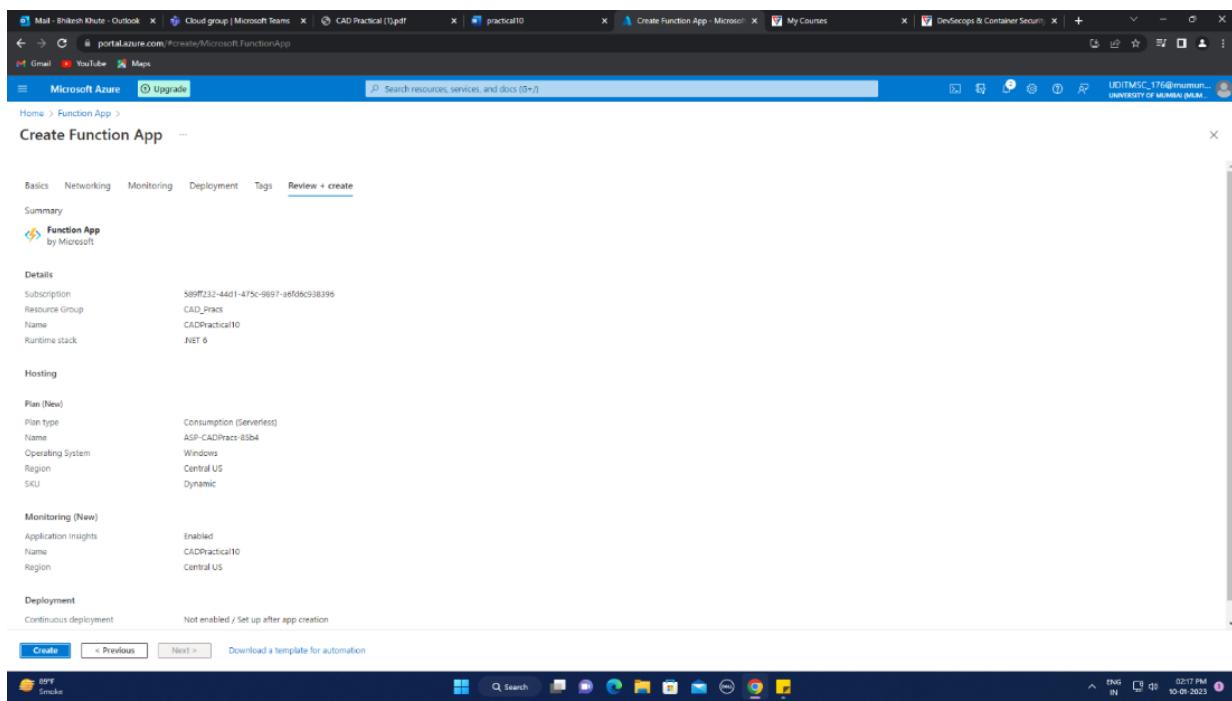
Runtime stack: .NET Core

Version 6

Region: Central US



Step 4 - Click on Review and Create:



Deployment will automatically start, please wait for it.

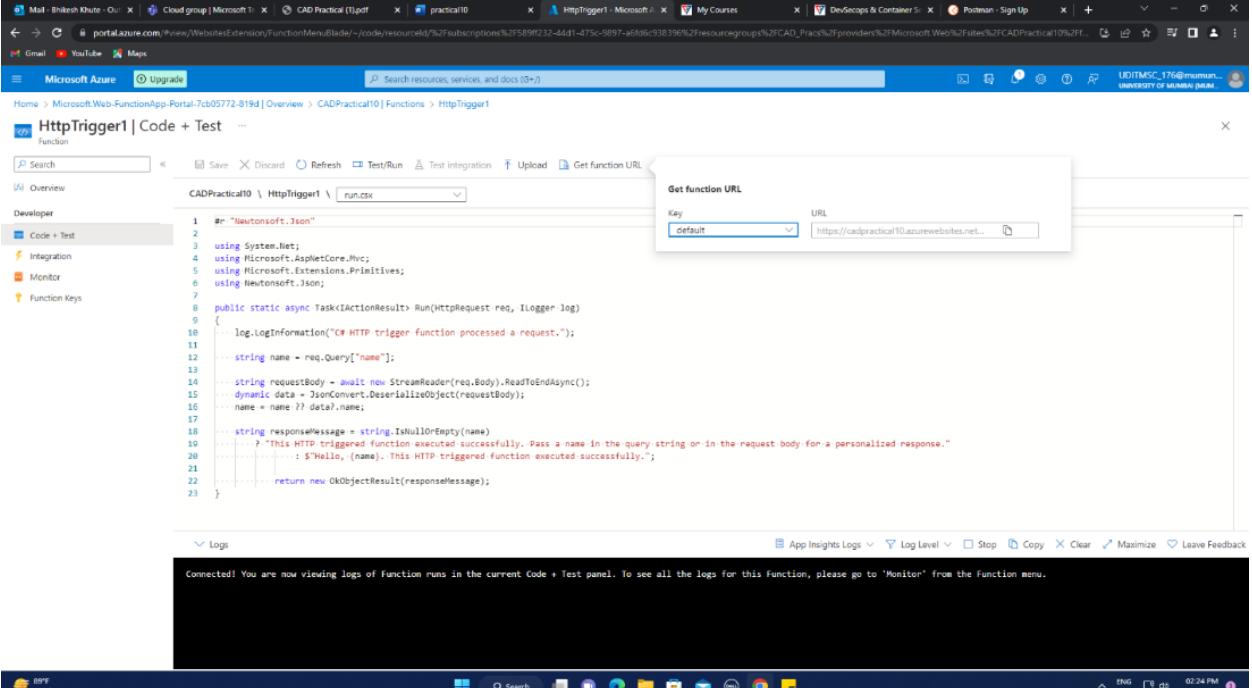
Step 5 – Function is now created. Now, Click on Functions → Add

The screenshot shows the Microsoft Azure portal interface for the 'CADPractical10' function app. The left sidebar navigation includes 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Microsoft Defender for Cloud', 'Events (preview)', 'Functions' (selected), 'App keys', 'App files', 'Protocols', 'Deployment' (with 'Deployment slots' and 'Deployment Center'), 'Settings' (with 'Configuration', 'Authentication', 'Application insights', 'Identity', 'Backups', 'Custom domains', and 'Custom domain (classic)'), and 'Logs'. The main content area shows the 'Essentials' section with details: Resource group (CAD_Prac), Status (Running), Location (Central US), Subscription (Free Trial), Subscription ID (589ff232-44d1-475c-9897-a6fdfe938396), Tags (Click here to add tags), URL (https://cadpractical10.azurewebsites.net), Operating System (Windows), App Service Plan (ASP-CADPractical10), Properties (See More), and Runtime version (4.14.0.19631). Below this are two charts: 'Memory working set (Size)' and 'Function Execution Count (Avg)'. The 'Memory working set (Size)' chart shows values from 0B to 10GB over time from 1:30 PM to 2 PM UTC+05:30. The 'Function Execution Count (Avg)' chart shows values from 0 to 100 over the same time period. The bottom status bar indicates TNG IN, 09:20 PM, 10-01-2023.

Step 6 - Click on HTTP Trigger and then click on Add & create the same.

The screenshot shows the 'Create function' dialog box in the Microsoft Azure portal. The left sidebar navigation is identical to the previous screenshot. The main content area has a title 'Create function' and a 'Select development environment' section with a note: 'Instructions will vary based on your development environment. Learn more'. A dropdown menu 'Development environ...' is set to 'Develop in portal'. Below this is a 'Select a template' section with a note: 'Use a template to create a function. Triggers describe the type of events that invoke your functions. Learn more'. A 'Filter' input field is present. A table lists available templates: 'HTTP trigger' (A function that will be run whenever it receives an HTTP request, responding based on data in 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), 'Azure Service Bus Queue trigger' (A function that will be run whenever a message is added to a specified Service Bus queue), 'Azure Service Bus Topic trigger' (A function that will be run whenever a message is added to the specified Service Bus topic), 'Azure Blob Storage trigger' (A function that will be run whenever a blob is added to a specified container), and 'Azure Event Hub trigger' (A function that will be run whenever an event hub receives a new event). The 'Template details' section contains a note: 'We need more information to create the HTTP trigger function. Learn more'. It includes fields for 'New Function' (set to 'HttpTrigger1') and 'Authorization level' (set to 'Function'). At the bottom are 'Create' and 'Cancel' buttons. The bottom status bar indicates TNG IN, 09:21 PM, 10-01-2023.

Step 7 – The Http Trigger is now created & the URL too.



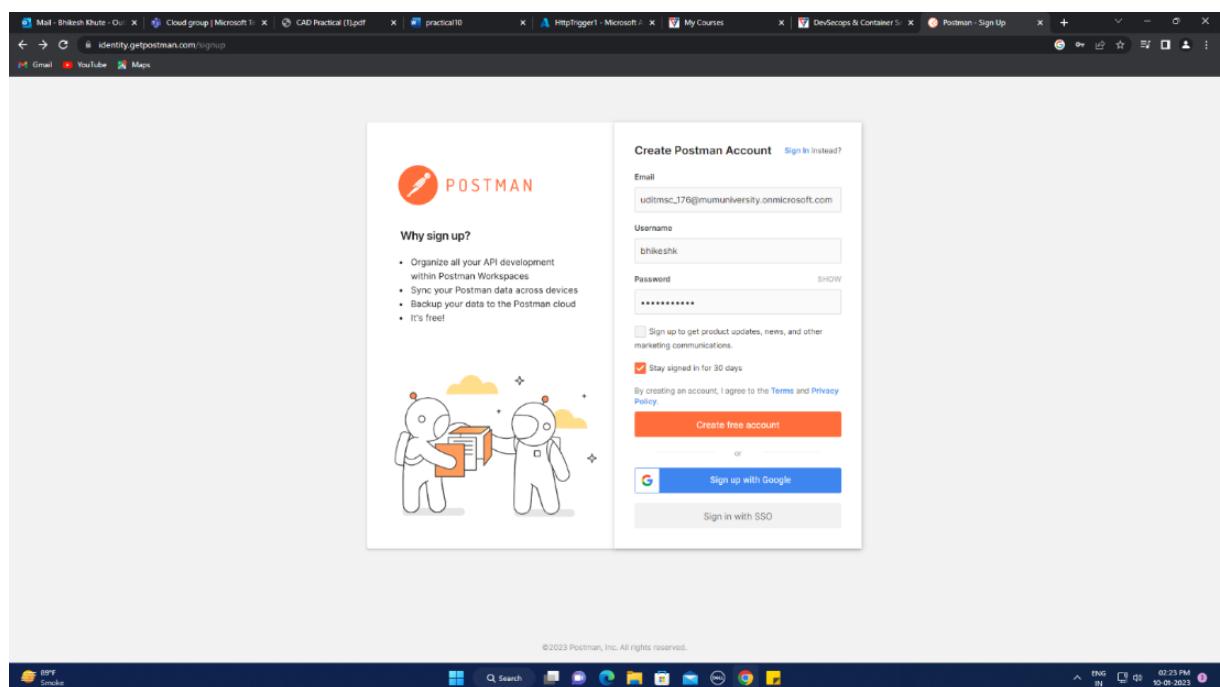
The screenshot shows the Microsoft Azure Functions portal. In the center, there's a code editor for a C# function named `HttpTrigger1`. The code defines a single HTTP trigger function that logs the request, reads the name from the query string or body, and returns a personalized response. Above the code editor, a modal window titled "Get function URL" is open, showing the key "default" and the URL `https://cadpractical10.azurewebsites.net/api/HttpTrigger1`. Below the code editor, a log viewer shows a single log entry indicating the function has been triggered successfully. The bottom of the screen shows a Windows taskbar with various icons.

```

1 # "Newtonsoft.Json"
2
3 using System;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7
8 public static async Task<ActionResult> 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
18     string responseMessage = string.IsNullOrWhiteSpace(name)
19     ? "This HTTP triggered function executed successfully. Pass a name in the query string or in the request body for a personalized response."
20     : $"Hello, {name}. This HTTP triggered function executed successfully.";
21
22     return new OkObjectResult(responseMessage);
23 }

```

Step 8 – Signup on the postman using any SSO to test the http trigger.



Step 9 – Select HTTP Trigger & Click on GET function. Once the GET function is visible, paste the function URL to test the same. Now, we can see in the output of the successful hit request.

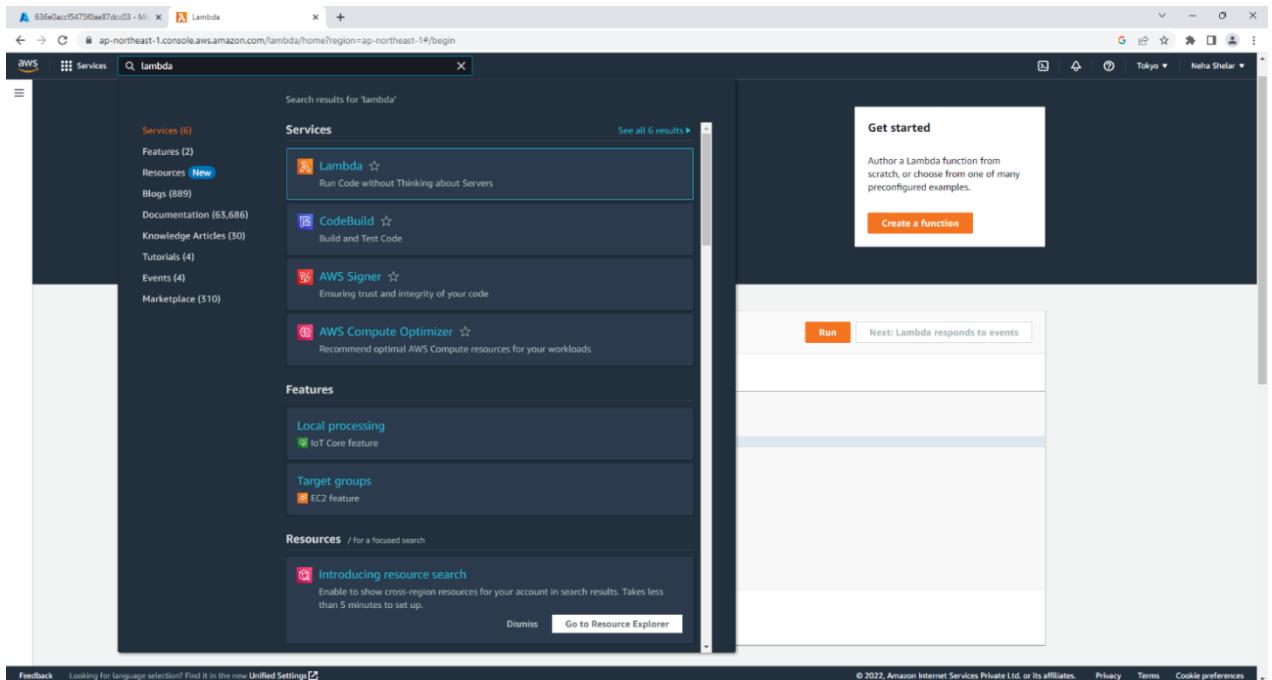
The screenshot shows the Postman application interface. In the top navigation bar, there are several tabs including 'Mail - Bhiksh Khute - Out', '(1) Cloud group | Microsoft', 'CAD Practical (1).pdf', 'practical10', 'HttpTrigger1 - Microsoft', 'My Courses', 'DevSecOps & Container S...', and a browser tab for 'https://cadpractical10.azure...'. The main workspace is titled 'My Workspace' and contains sections for 'Collections', 'APIs', 'Environments', 'Mock Servers', 'Monitors', 'Power', and 'History'. A 'Create a collection for your requests' section is present. On the right, a request is being configured for 'https://cadpractical10.azurewebsites.net/api/HttpTrigger1?code=b_N_6v6X3-mDJSM4h7SgMB4m9KD3gXN_nDrIcHSLkAxYfukdOtoQ=='. The method is set to 'GET'. Under 'Params', there is a single parameter 'code' with the value 'b_N_6v6X3-mDJSM4h7SgMB4m9KD3gXN_nDrIcHSLkAxYfukdOtoQ=='. The 'Body' tab shows the response body: 'This HTTP triggered function executed successfully. Pass a name in the query string or in the request body for a personalized response.' The status bar at the bottom indicates 'Status 200 OK Time: 1159 ms Size: 375 B Save Response'.

Practical No: 07

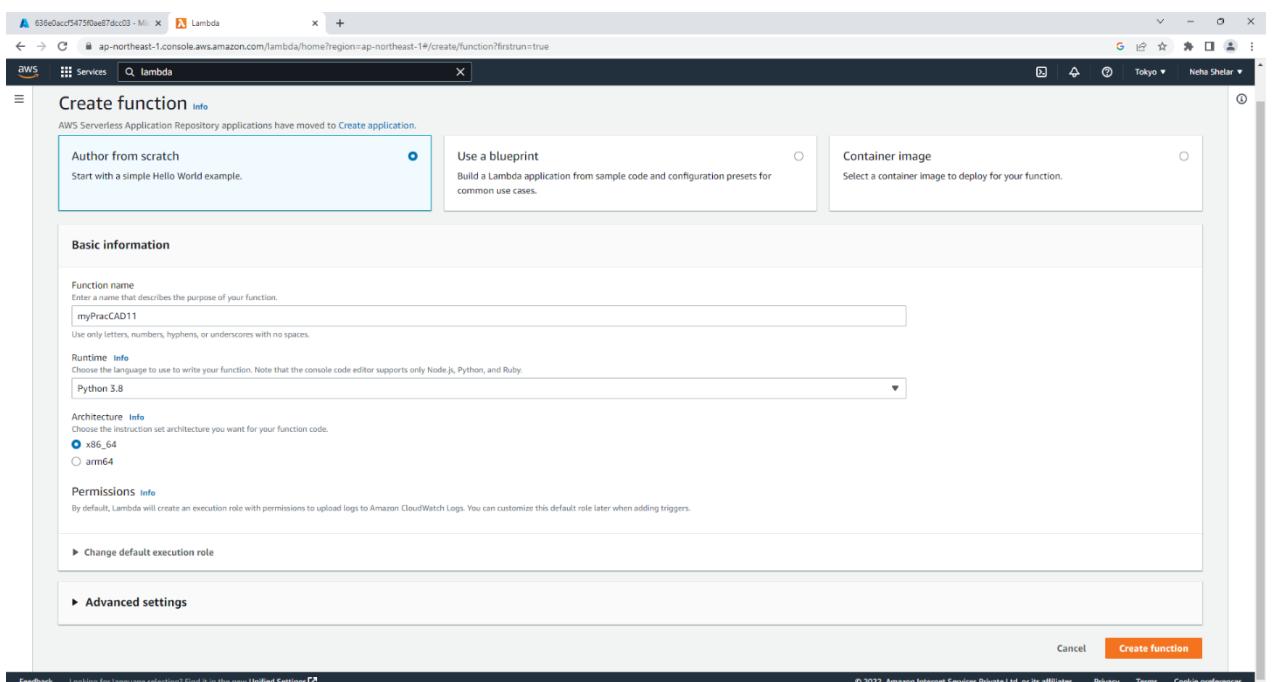
Aim: Create AWS Lambda Function.

To create a Lambda function with the console

Step 1: Open the Functions page on the Lambda console.



Step 2: Choose Create function.



Step 3 - Now search "S3" in the search bar.

The screenshot shows the AWS Lambda search interface. A search bar at the top contains the query 's3'. The left sidebar lists various AWS services and features, including Lambda, Features, Resources, Blogs, Documentation, Tutorials, Events, and Marketplace. The main search results are displayed under two categories: 'Services' and 'Features'.

- Services:**
 - S3** (Scalable Storage in the Cloud): Includes sub-links for Buckets, Access points, and Batch Operations.
 - S3 Glacier** (Archive Storage in the Cloud)
 - Athena** (Query Data in S3 using SQL)
 - AWS Snow Family** (Large Scale Data Transport)
- Features:**
 - Amazon S3 File Gateway** (Storage Gateway feature)
 - Batch Operations** (S3 feature)
 - Buckets** (S3 feature)
 - Access points** (S3 feature)

The right panel displays detailed information for the selected 'S3' service, including its ARN, last modified time (30 seconds ago), and function URL. There is also an 'Upload from' button and a settings gear icon.

The screenshot shows the AWS S3 Management Console. The left sidebar includes options like Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and Access analyzer for S3. It also features links for Block Public Access settings, Storage Lens, Feature spotlight, and AWS Marketplace for S3.

The main content area has a blue header bar with the text: "We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback." Below this, a message about replicating data between regions is shown.

The central part of the screen is titled "Amazon S3 > Buckets". It features an "Account snapshot" section with a link to "View Storage Lens dashboard". Below this is a table titled "Buckets" with a "Info" link. The table has columns for Name, AWS Region, Access, and Creation date. A search bar at the top of the table allows users to "Find buckets by name". The table currently shows "No buckets" and "No buckets". A prominent orange "Create bucket" button is located at the bottom right of the table area.

Step 4 – Create a bucket with appropriate bucket name, aws region and encryption.

The screenshot shows the 'Create bucket' configuration page in the AWS S3 console. The 'Bucket name' field is set to 'mycadbucket'. The 'AWS Region' dropdown is set to 'US East (N. Virginia) us-east-1'. Under 'Object Ownership', 'ACLs disabled (recommended)' is selected. Under 'Block Public Access settings for this bucket', both 'Block public access' and 'Block public ACLs' are checked. At the bottom right, there is a 'Create bucket' button.

The screenshot shows the 'Create bucket' configuration page in the AWS S3 console for the 'ap-northeast-1' region. Under 'Bucket Versioning', 'Disable' is selected. Under 'Tags (0) - optional', there are no tags associated with this bucket. Under 'Default encryption', 'Server-side encryption' is set to 'Disable'. At the bottom right, there is a 'Create bucket' button.

Step 5 - Now go the roles under IAM and select the role and attach policies for s3 bucket access.

The screenshot shows the AWS IAM Management Console. On the left, there's a sidebar with options like 'Identity and Access Management (IAM)', 'Access management', 'Access reports', and 'Related consoles'. The main area is titled 'Roles' and shows a table with three rows. The first two rows are collapsed, while the third row, 'myPracCAD11-role-o9nm6l7d', is expanded, showing its details. Below the table, there are sections for 'Roles Anywhere' and 'Temporary credentials'.

This screenshot shows the detailed view of the 'myPracCAD11-role-o9nm6l7d' role. The sidebar on the left is identical to the previous screenshot. The main area has a header 'Introducing the new IAM roles experience' with a note about the redesign. Below it, the role name 'myPracCAD11-role-o9nm6l7d' is displayed. The 'Summary' section shows creation date (November 11, 2022), last activity (None), ARN, and maximum session duration (1 hour). The 'Permissions' tab is selected, showing a table with one row: 'AWSLambdaBasicExecutionRole'. There are buttons for 'Simulate', 'Remove', and 'Add permissions'.

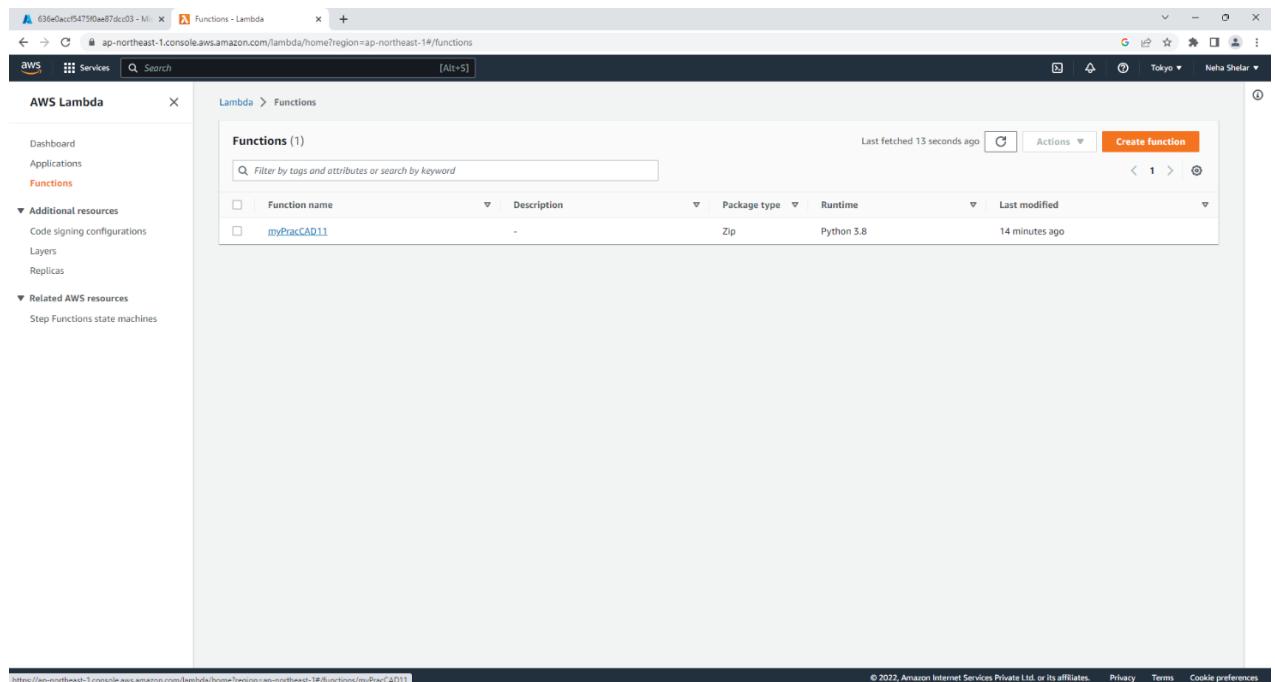
The screenshot shows the AWS IAM Management Console with the URL <https://us-east-1.console.aws.amazon.com/iamv2/home?region=ap-northeast-1#/roles/details/myPracCAD11-role-o9nm6l7d/attach-policies>. The page displays a list of AWS managed policies that can be attached to the selected role. One policy, 'AmazonS3FullAccess', is currently selected.

Policy name	Type	Description
AmazonDMSRedshiftS3Role	AWS managed	Provides access to manage S3 settings for Redshift endpoints for DMS.
AmazonS3FullAccess	AWS managed	Provides full access to all buckets via the AWS Management Console.
QuickSightAccessForS3StorageManagem...	AWS managed	Policy used by QuickSight team to access customer data produced by S3 Storage Management Analytics.
AmazonS3ReadOnlyAccess	AWS managed	Provides read only access to all buckets via the AWS Management Console.
AmazonS3OutpostsFullAccess	AWS managed	Provides full access to Amazon S3 on Outposts via the AWS Management Console.
AWSBackupServiceRolePolicyForS3Backup	AWS managed	Policy containing permissions necessary for AWS Backup to backup data in any S3 bucket. This includes read access to all S3 objects and any decrypt access for all K...
AWSBackupServiceRolePolicyForS3Rest...	AWS managed	Policy containing permissions necessary for AWS Backup to restore a S3 backup to a bucket. This includes read/write permissions to all S3 buckets, and permissions to ...
AmazonS3ObjectLambdaExecutionRoleP...	AWS managed	Provides AWS Lambda functions permissions to interact with Amazon S3 Object Lambda. Also grants Lambda permissions to write to CloudWatch Logs.
AmazonS3OutpostsReadOnlyAccess	AWS managed	Provides read only access to Amazon S3 on Outposts via the AWS Management Console.

The screenshot shows the AWS IAM Management Console with the URL <https://us-east-1.console.aws.amazon.com/iamv2/home?region=ap-northeast-1#/roles/details/myPracCAD11-role-o9nm6l7d?section=permissions>. The 'Permissions' tab is selected, displaying the attached policies and the 'Permissions boundary - (not set)' section.

Policy name	Type	Description
AWSLambdaBasicExecutionRole-b6f5a21d-ad76-40c9-82dd-f696fc2e39ac	Customer managed	
AmazonS3FullAccess	AWS managed	Provides full access to all buckets via the AWS Man...

Step 6 - Now go search function and create a function now and paste the code below and deploy the same.



<https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/functions/myPracCAD11>

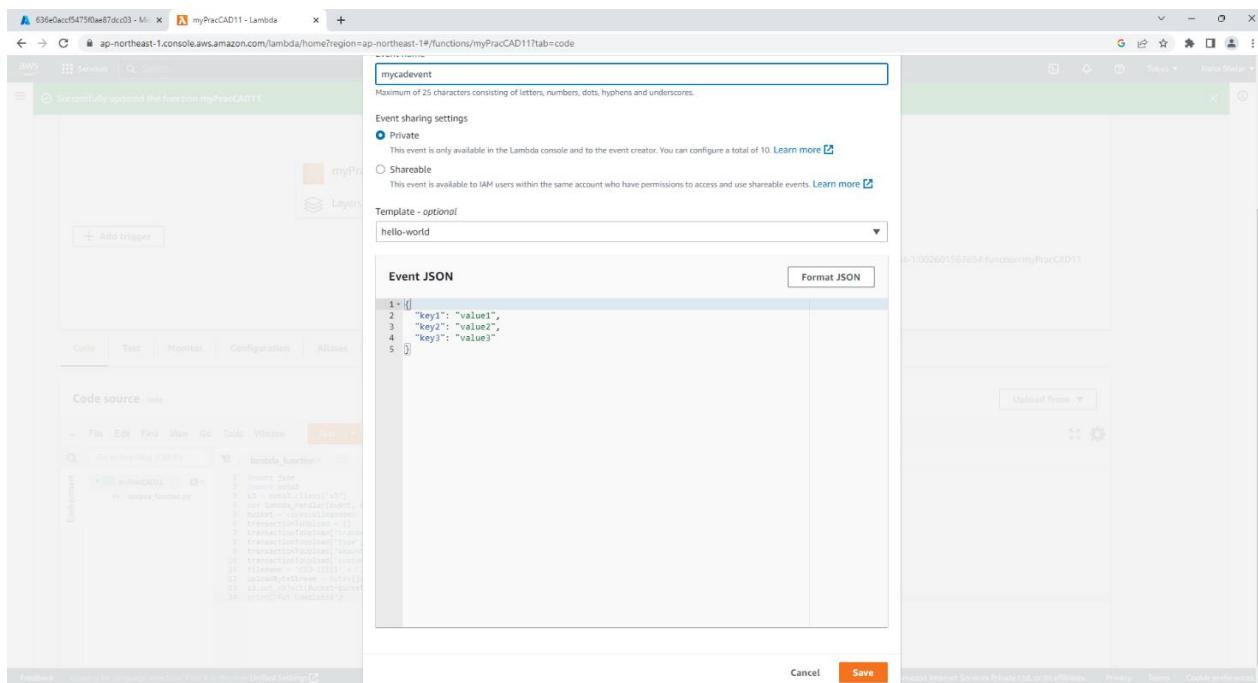
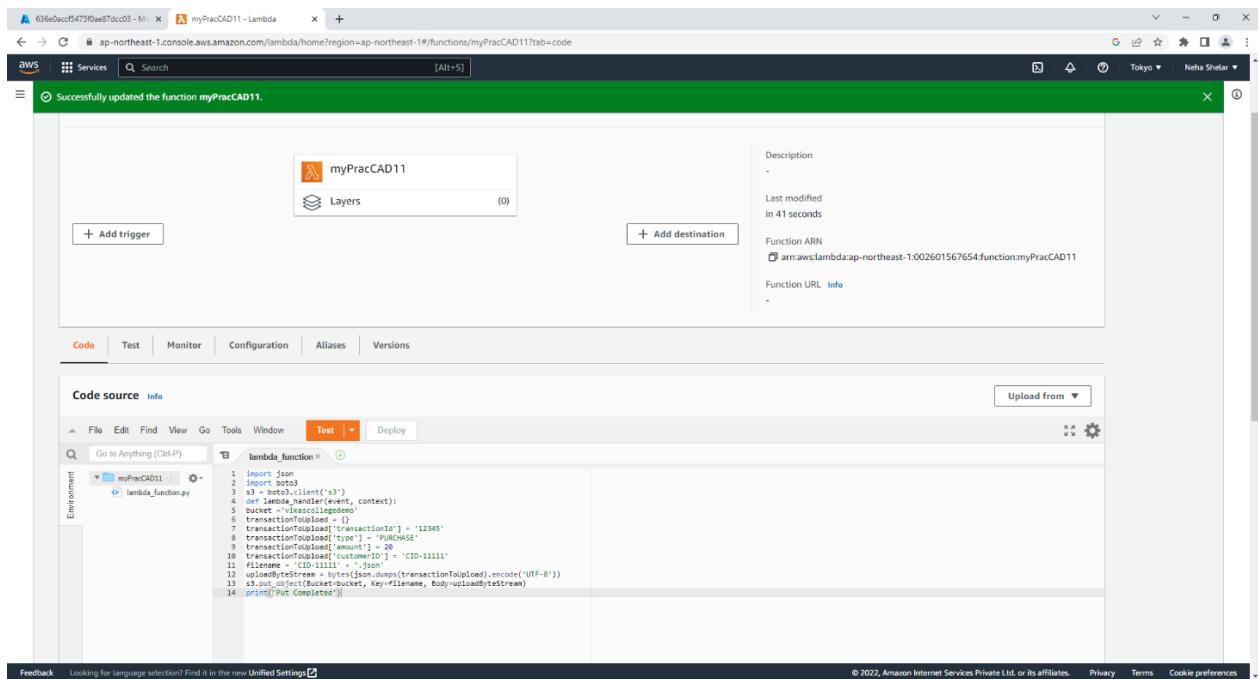
© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

```
import json
import boto3

s3 = boto3.client('s3')

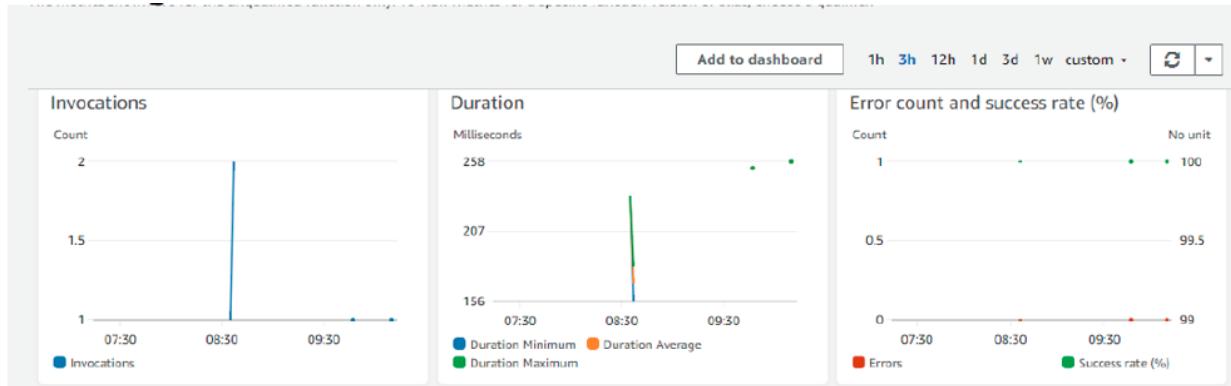
def lambda_handler(event, context):
    bucket='mycadbucket'

    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')
```



OUTPUT:

Monitoring Result:



Execution Result:

The Lambda test interface shows the following details:

- Code source:** myPracCAD11 / lambda_function.py
- Test Event Name:** mycadevent
- Response:** null
- Function Logs:**
 - START RequestId: 610a1fa7-2c9d-41e3-ae51-d26b2a2bc0fb Version: \$LATEST
 - Put Completed
 - END RequestId: 610a1fa7-2c9d-41e3-ae51-d26b2a2bc0fb
 - REPORT RequestId: 610a1fa7-2c9d-41e3-ae51-d26b2a2bc0fb Duration: 1140.79 ms Billed Duration: 1141 ms Memory Size: 128 MB Max Memory Used: 69 MB Init Duration: 457.57 ms
 - Request ID: 610a1fa7-2c9d-41e3-ae51-d26b2a2bc0fb

Practical no. 08

Aim: Build AWS Lambda with AWS API Gateway

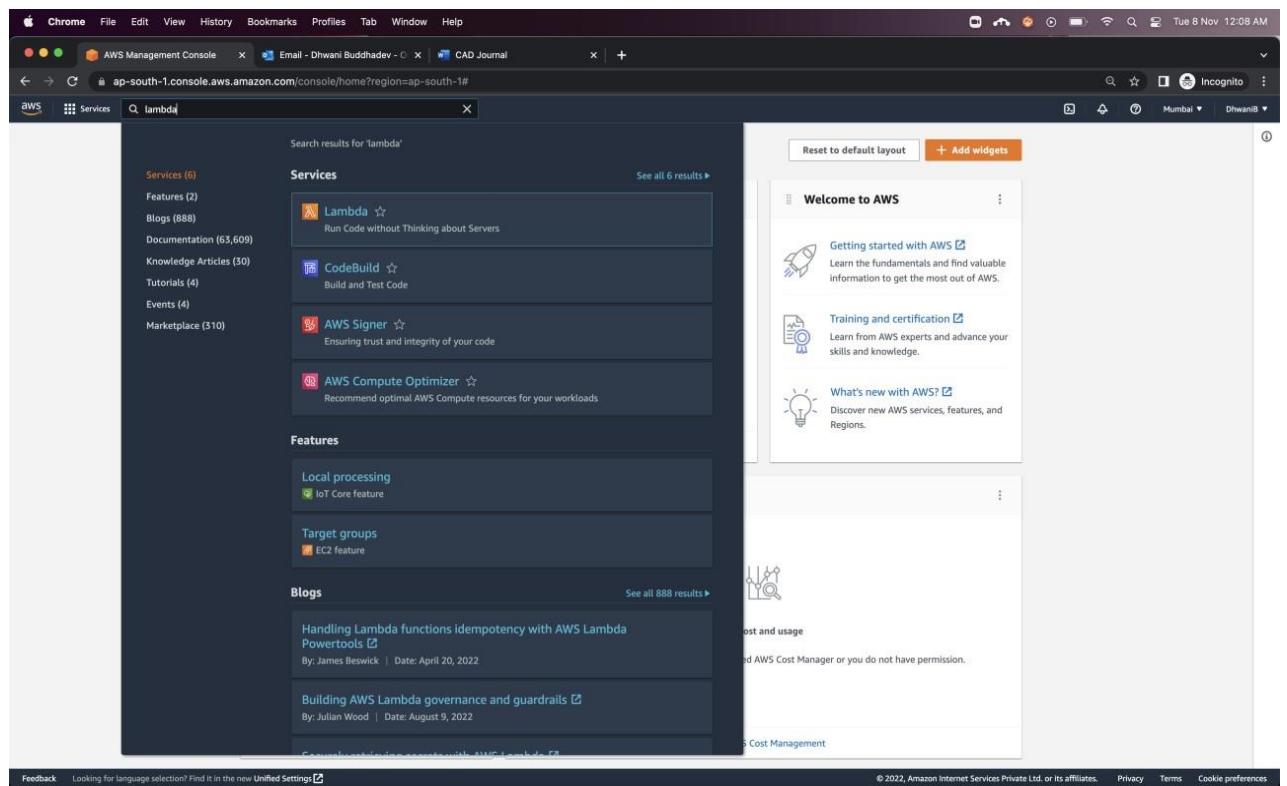
Description of Practical: -

- In this practical we will learn to build AWS Lambda function using API Gateway.
- In order to perform this practical we will take an example.
- The example is EMI Calculator.

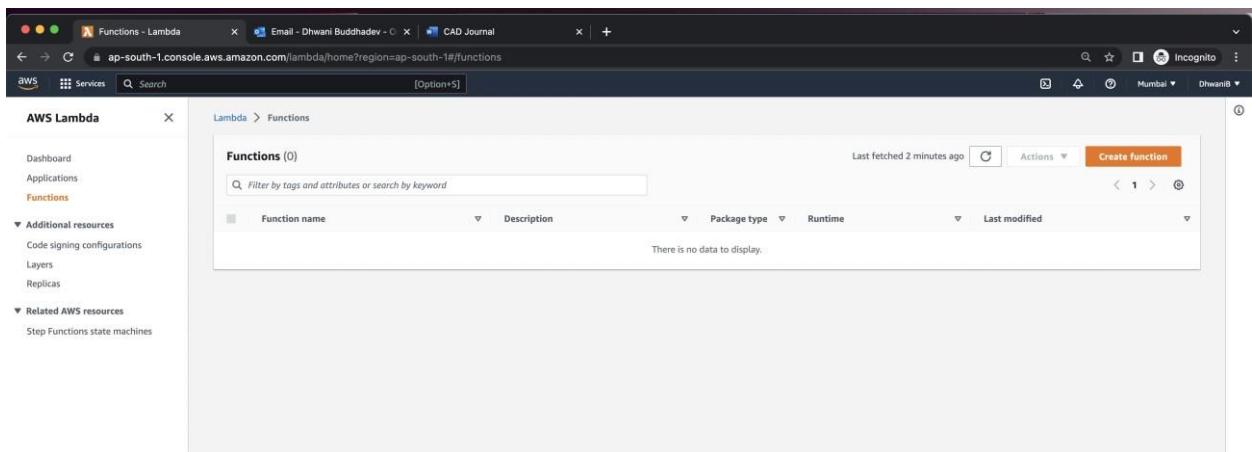
EMI Calculator

Steps

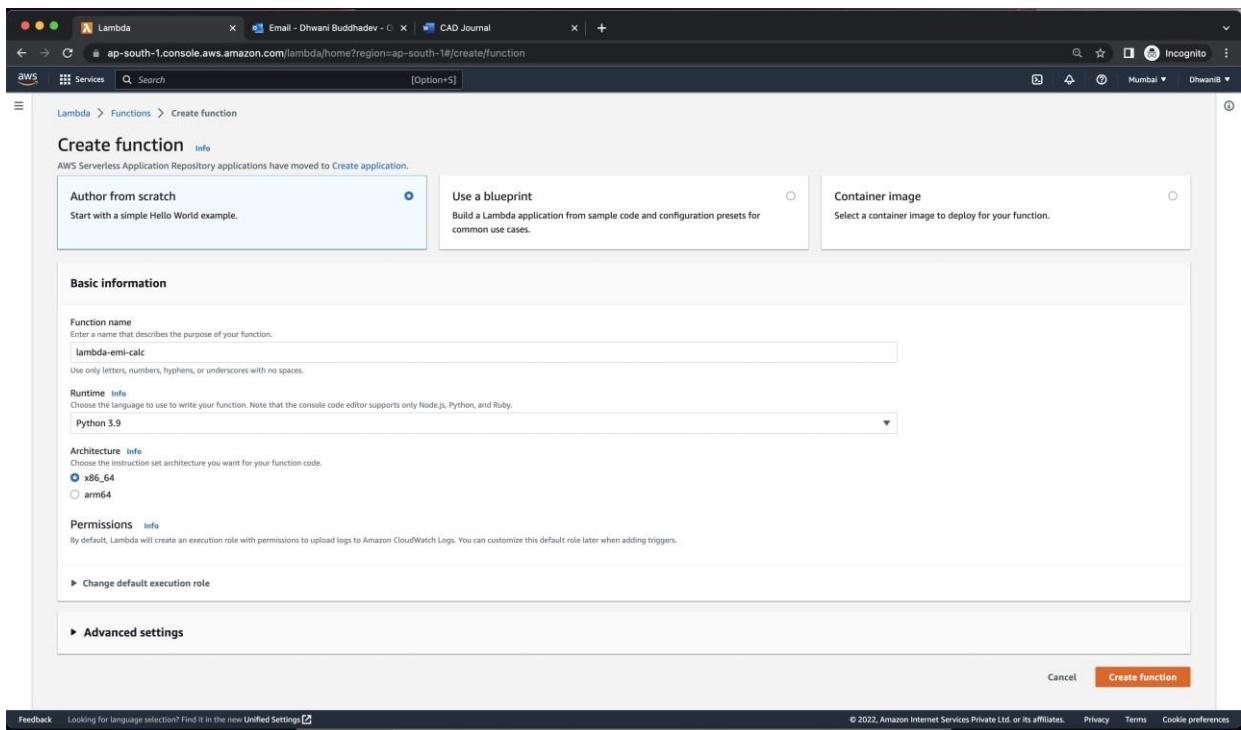
- Login into AWS Console.
- Search for Lambda on the service bar.



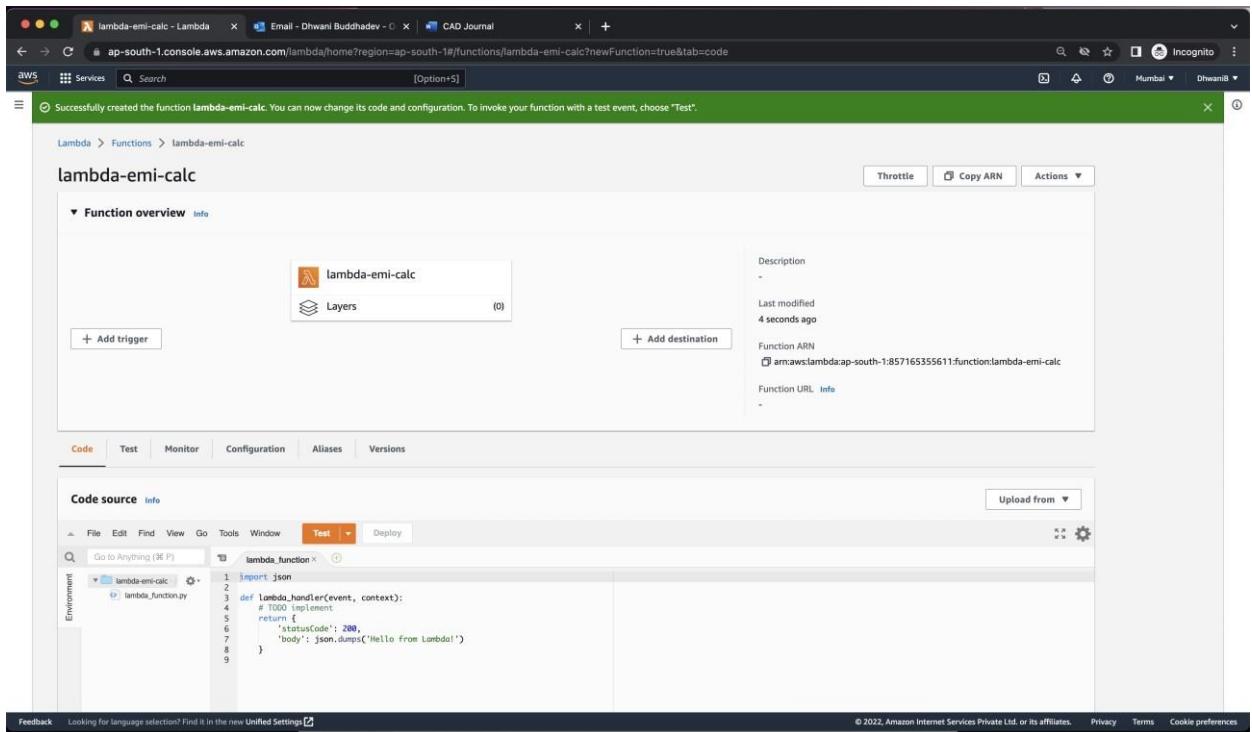
- Here we will create a lambda function for the practical.



- Click on Create Function on the Lambda Tab.



- Here we will choose Author from Scratch.
- After selecting that write the function name as lambda-emi-calc
- We will be running the code on python 3.9 • So, we will choose runtime as python 3.9
- After that click on create function.
- We can see that it has created lambda function with basic default code



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

```
import json print('Loading function... Lambda emi calculator using
API gateway') def lambda_handler(event, context):    #1. Parse
out query string params    principal =
event['queryStringParameters']['p']    rate =
event['queryStringParameters']['r']    time =
event['queryStringParameters']['t']

principal =
int(principal)    rate =
float(rate)    time =
int(time)

emi = emi_calculator(principal, rate, time);
```

#2. Construct the body of the response object

```
transactionResponse = {}
transactionResponse['p'] = principal
transactionResponse['r'] = rate
transactionResponse['t'] = time
transactionResponse['emi'] = emi
```

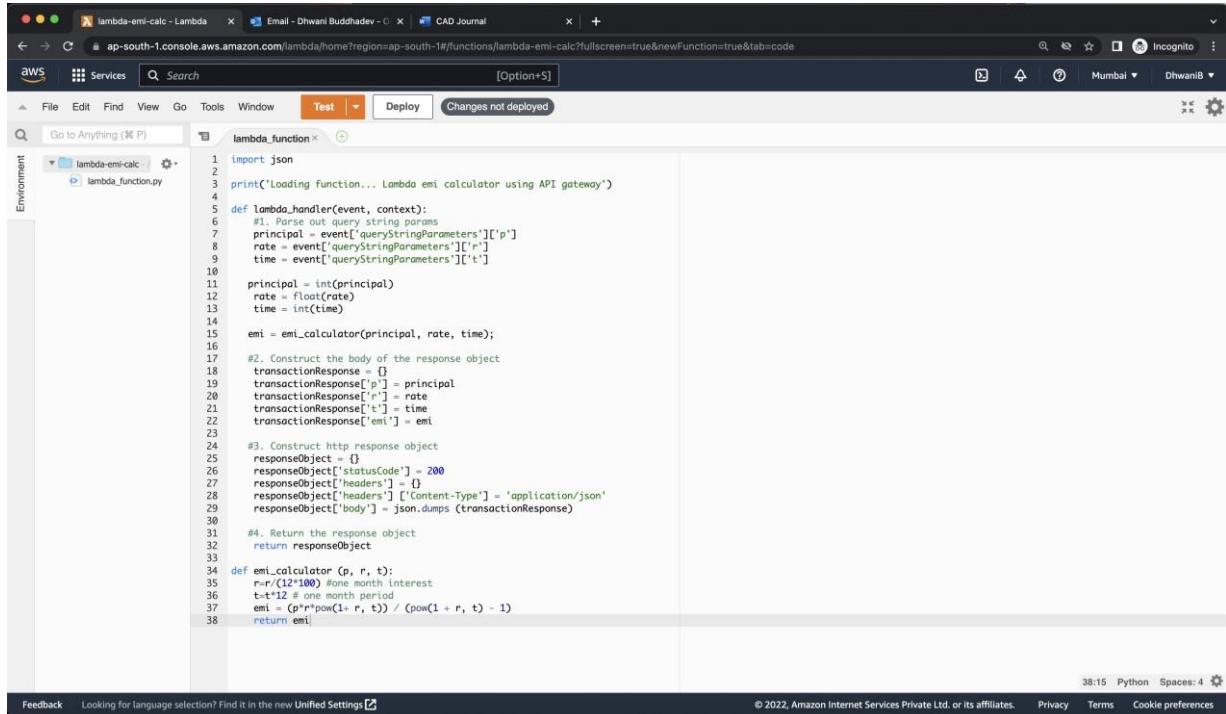
#3. Construct http response object

```
responseObject = {}
responseObject['statusCode'] = 200
responseObject['headers'] = {}
```

```

responseObject['headers']['Content-Type'] = 'application/json'
responseObject['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

```



The screenshot shows the AWS Lambda function editor interface. The code editor displays Python code for calculating EMI. The code imports json, defines a lambda handler, and implements an emi_calculator function. It handles query parameters for principal, rate, and time, constructs a response object with headers and body, and returns the response object.

```

import json
print('Loading function... Lambda emi calculator using API gateway')
def lambda_handler(event, context):
    principal = event['queryStringParameters']['p']
    rate = event['queryStringParameters']['r']
    time = event['queryStringParameters']['t']
    principal = int(principal)
    rate = float(rate)
    time = int(time)
    emi = emi_calculator(principal, rate, time);
    transactionResponse = {}
    transactionResponse['r'] = principal
    transactionResponse['t'] = rate
    transactionResponse['em'] = emi
    responseObject = {
        'statusCode': 200,
        'headers': {
            'Content-Type': 'application/json'
        },
        'body': json.dumps(transactionResponse)
    }
    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
- Then go to dropdown besides test and configure test event with sample input values for testing lambda function code

The screenshot shows the AWS Lambda console interface. At the top, a green banner says "Successfully updated the function lambda-emi-calc." Below it, the "Code" tab is selected, showing the Python code for the function. The configuration tab is open, displaying a "Configure test event" dialog. Inside the dialog, a "Test event action" section has "Create new event" selected. The "Event name" field contains "emi-calc". Under "Event sharing settings", "Private" is chosen. The "Template - optional" dropdown is set to "hello-world". The "Event JSON" field contains the following JSON:

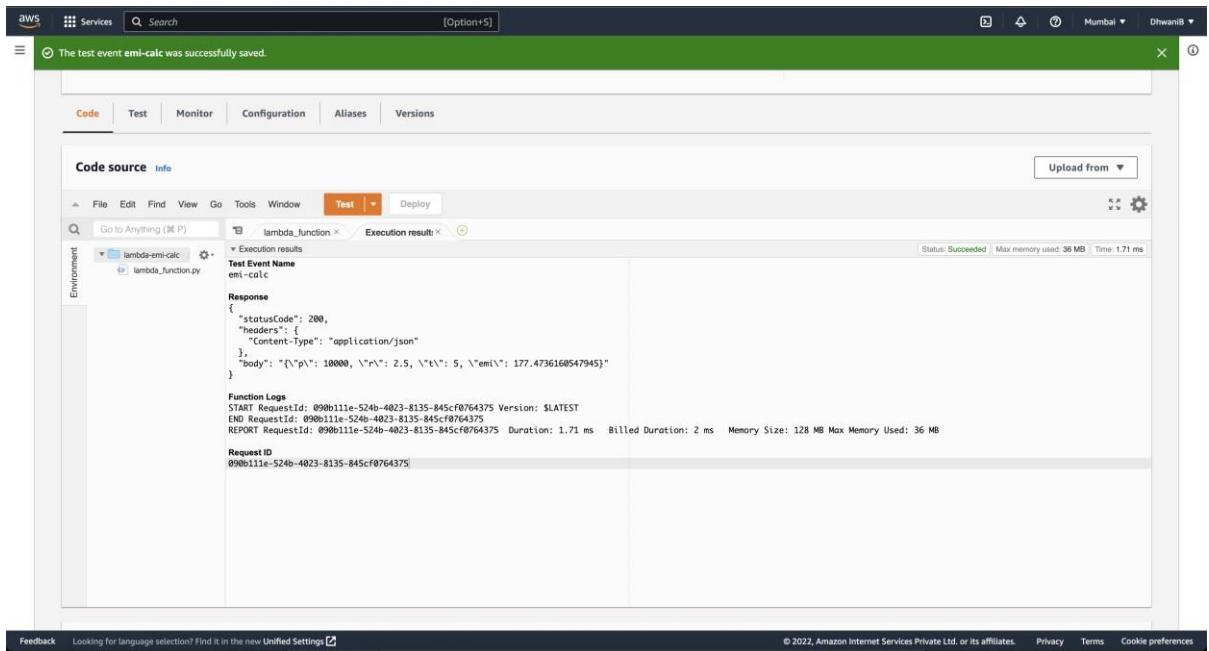
```

{
  "queryStringParameters": {
    "p": 10000,
    "r": 2.5,
    "t": 5
  }
}

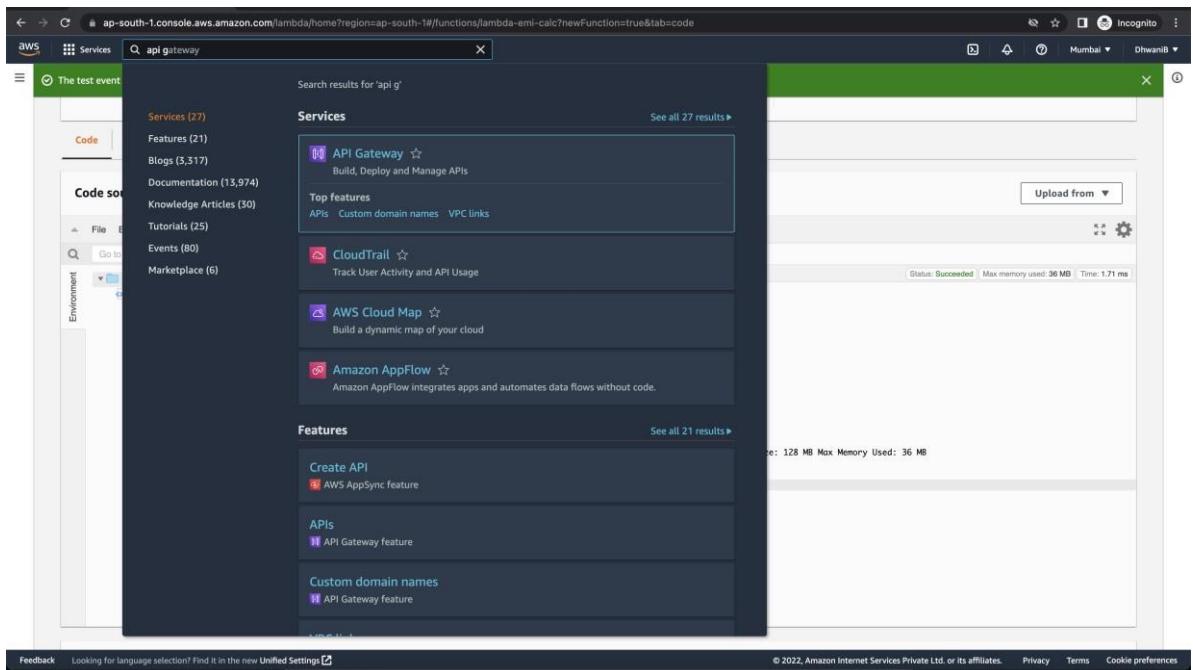
```

```
{
  "queryStringParameters": {
    "p": 10000,
    "r": 2.5,
    "t": 5
  }
}
```

- Test the code and get output without any errors



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



- Then create API gateway
- Choose Rest API and click import

The screenshot shows the AWS API Gateway console under the 'APIs' tab. It displays four API types:

- WebSocket API**: Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards. Works with Lambda, HTTP backends.
- REST API**: Develop a REST API where you gain complete control over the request and response along with API management capabilities. Works with Lambda, HTTP, AWS Services.
- REST API (Private)**: Create a REST API that is only accessible from within a VPC. Works with Lambda, HTTP, AWS Services.
- An additional REST API entry, which is likely a duplicate or a placeholder.

Each API entry has 'Import' and 'Build' buttons. The bottom of the screen shows standard AWS navigation links and copyright information.

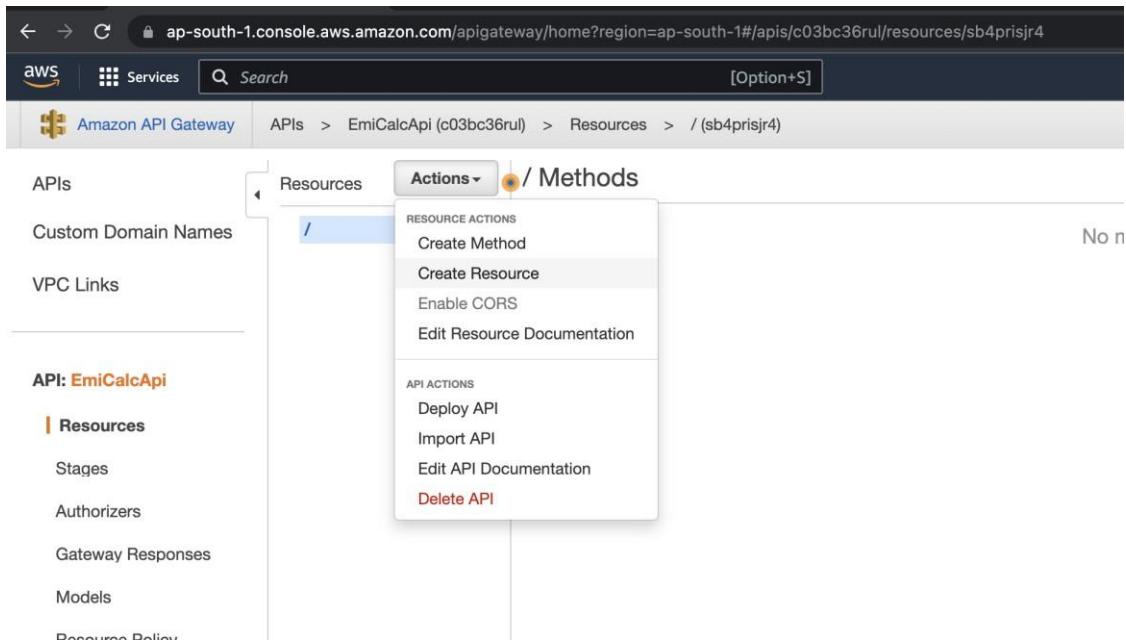
- Create new API
- API name = EmiCalcApi
- Description = EmiCalcApi
- Endpoint type = Regional
- Click on create tab

The screenshot shows the 'Create new API' wizard:

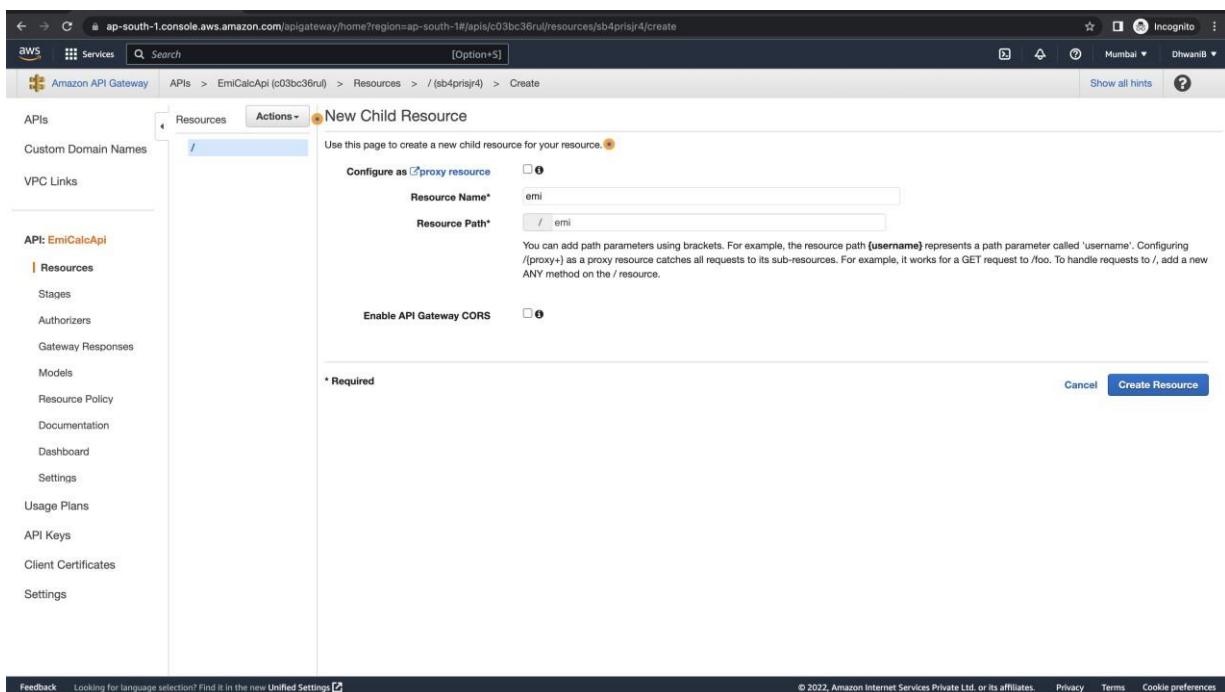
- Choose the protocol**: REST is selected.
- Create new API**: New API is selected.
- Settings**: API name is set to EmiCalcApi, Description is EmiCalcApi, and Endpoint Type is set to Regional.

A 'Create API' button is located at the bottom right of the form.

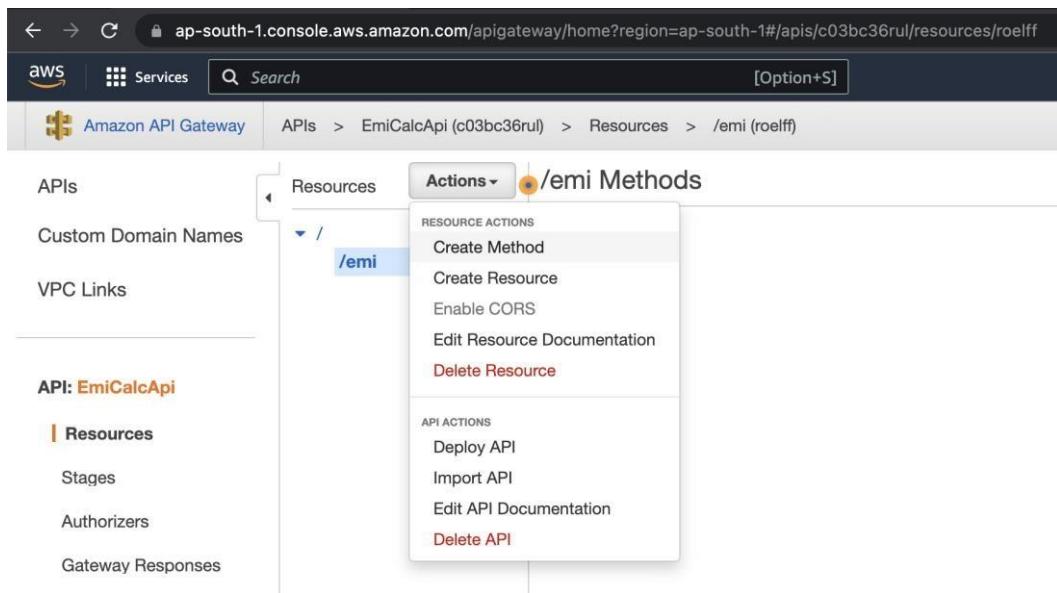
- Next step click on actions then create resource



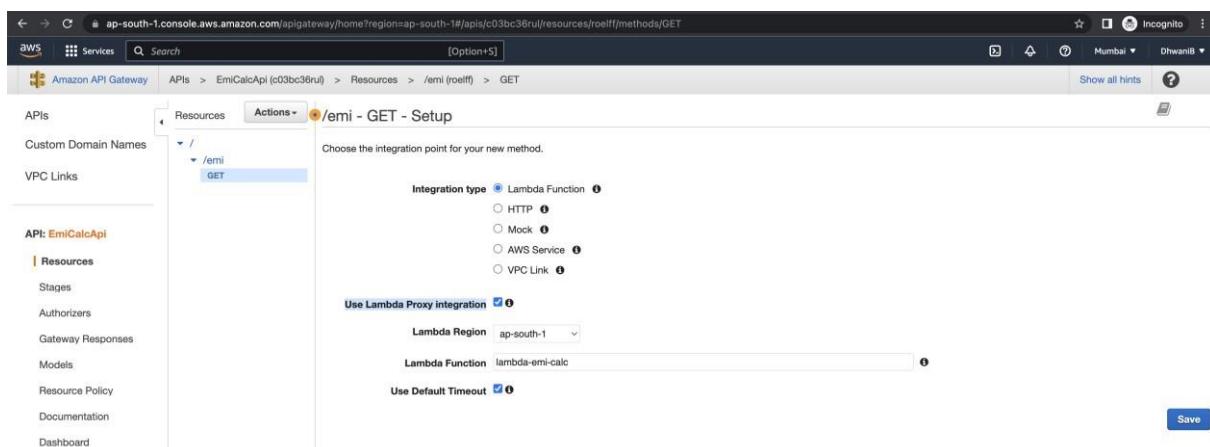
- Enter resource name as emi and click create resource

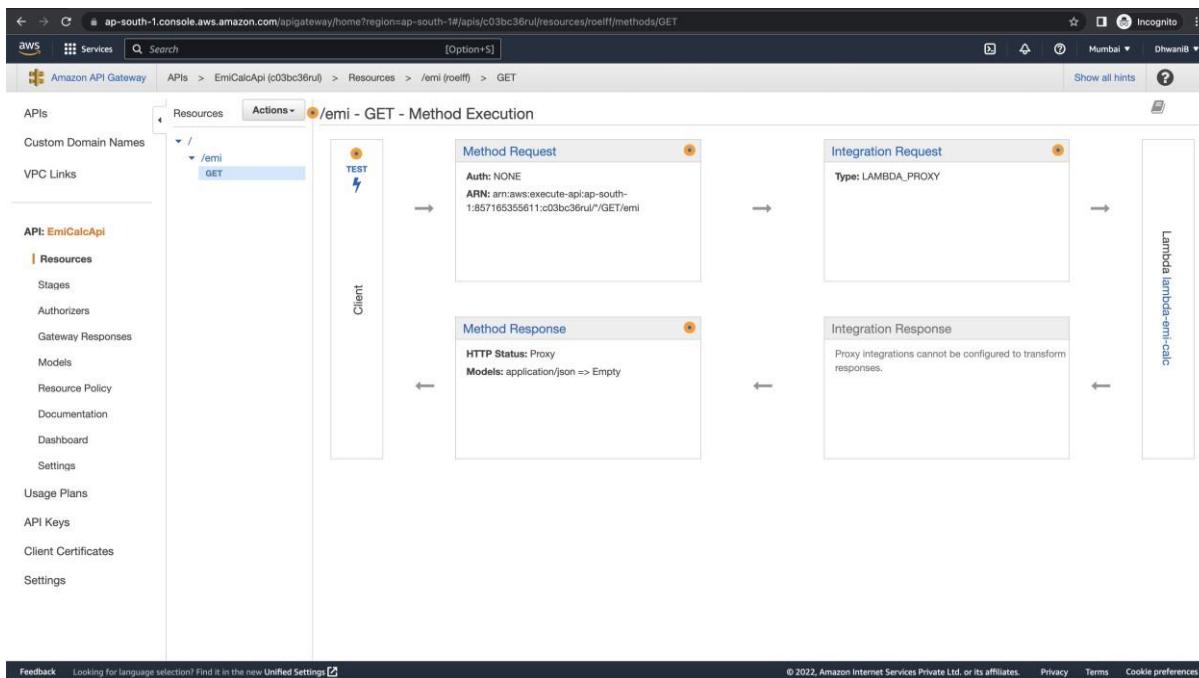


- Now select on actions tab and create method

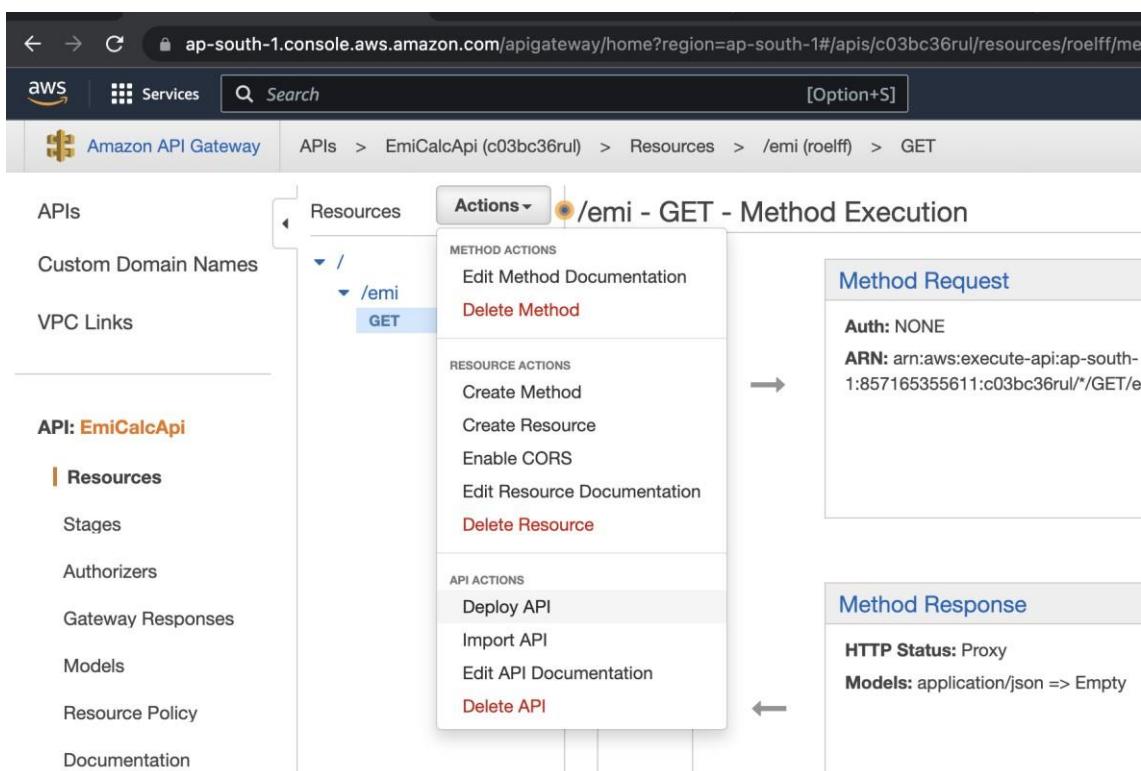


- Use Get parameter
- We are using integration type as Lambda Function
- Enter lambda function same as created above **lambda-emi-calc**
- Check **Use Lambda Proxy integration**
- Save the changes, ok





- Next step is to click on actions and then deploy api



- Enter deployment stage as New Stage • Enter stage name as dev
- Stage description as dev
- Deployment description as dev
- After that click on GET method
- Copy the URL and open in browser in new tab

The screenshot shows two main views of the AWS API Gateway console:

- Top View (Deploy API):** A modal window titled "Deploy API" is open, prompting the user to choose a stage for deployment. The "Deployment stage" dropdown is set to "[New Stage]". Other fields include "Stage name" (dev), "Stage description" (dev), and "Deployment description" (dev). Buttons for "Cancel" and "Deploy" are at the bottom.
- Bottom View (Stage Editor):** The main interface shows the "Stages" section with a "Create" button. A "dev Stage Editor" is open, displaying settings for the "dev" stage. It includes sections for "Cache Settings" (with "Enable API cache" checked), "Default Method Throttling" (with "Enable throttling" checked, Rate: 10000 requests per second, Burst: 5000 requests), and "Web Application Firewall (WAF)" (with "Web ACL" set to "None"). There's also a "Client Certificate" section where "Certificate" is set to "None". A "Save Changes" button is at the bottom right.

- 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
- Need to add after url - **/emi?p=10000&r=5.8&t=10**
- Click on enter and you will have your required emi

```
curl -X GET https://c03bc36ru.execute-api.ap-south-1.amazonaws.com/dev/emi?p=10000&r=5.8&t=10
{"p": 10000, "r": 5.8, "t": 10, "emi": 110.0188101041876}
```