Cloud Application Development

SIES (NERUL) COLLEGE OF ARTS SCIENCE AND COMMERCE

NAAC ACCREDITED 'A' GRADE COLLEGE (ISO

9001:2015 CERTIFIED INSTITUTION)

NERUL, NAVI MUMBAI - 400706

DEPARTMENT OF INFORMATION TECHNOLOGY

PRACTICAL BOOK ON

Cloud Application Development

SUBMITTED TO,

UNIVERSITY OF MUMBAI

BY

VISHAL VIJAY VARMA

MASTER OF INFORMATION TECHNOLOGY

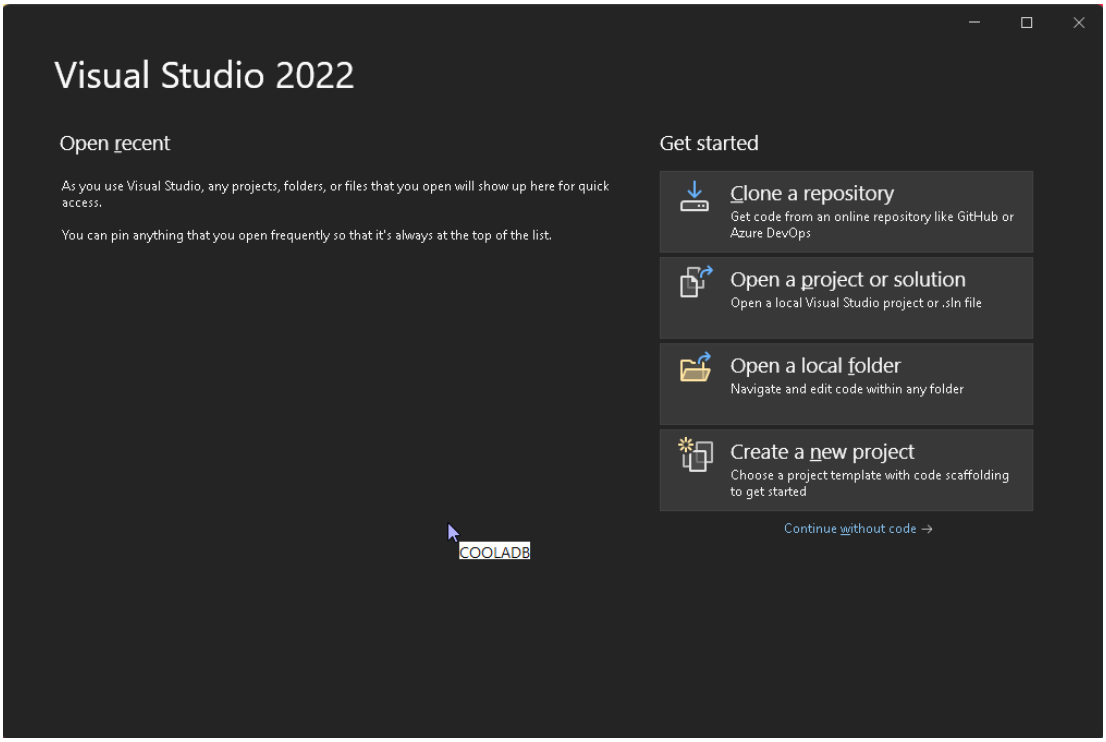
PART - II (SEMESTER - III)

(2022-2023)

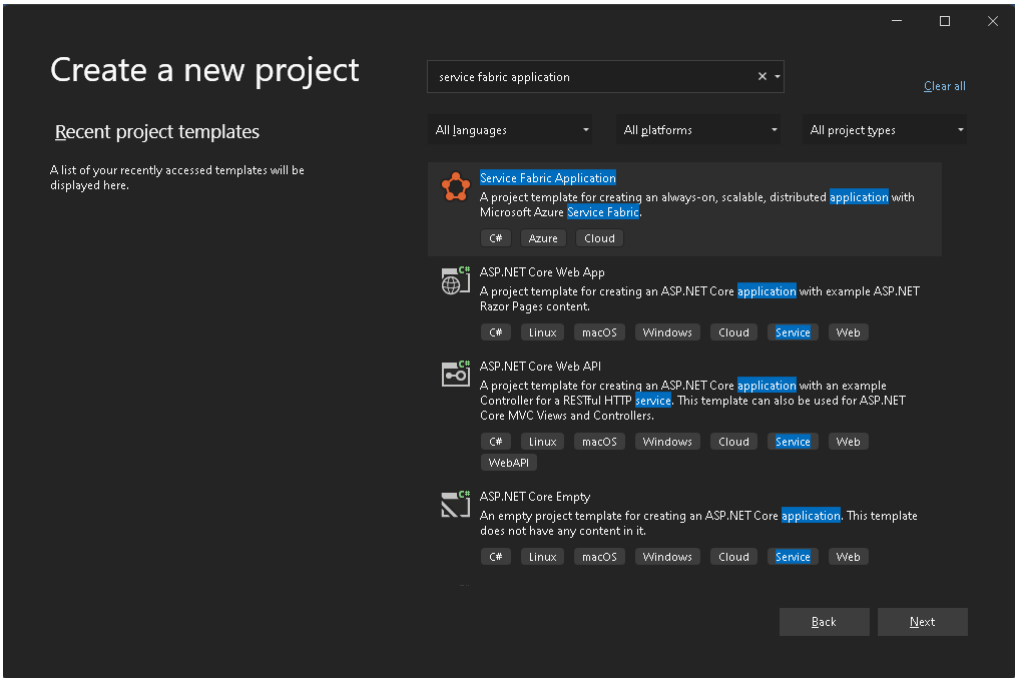
INDEX

Steps :-

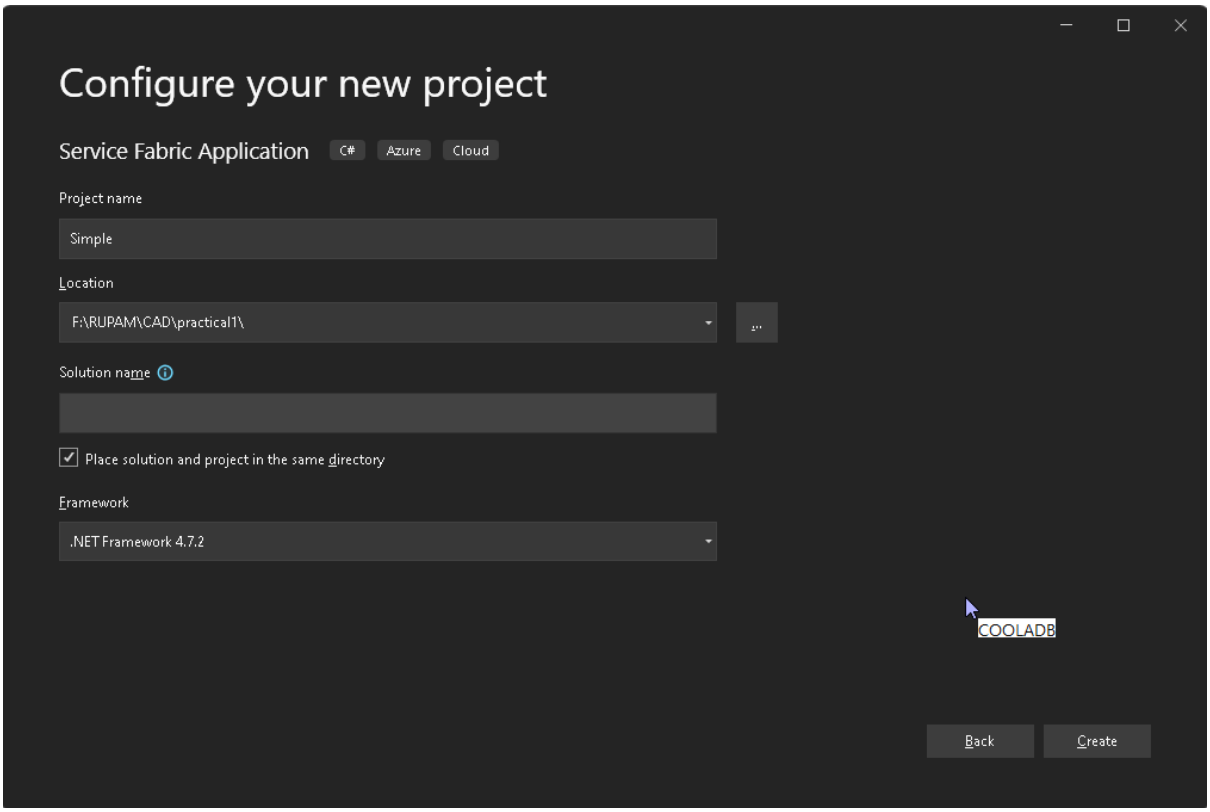
1. Launch Visual Studio 2017/19/22 as an administrator.
2. Create a new project



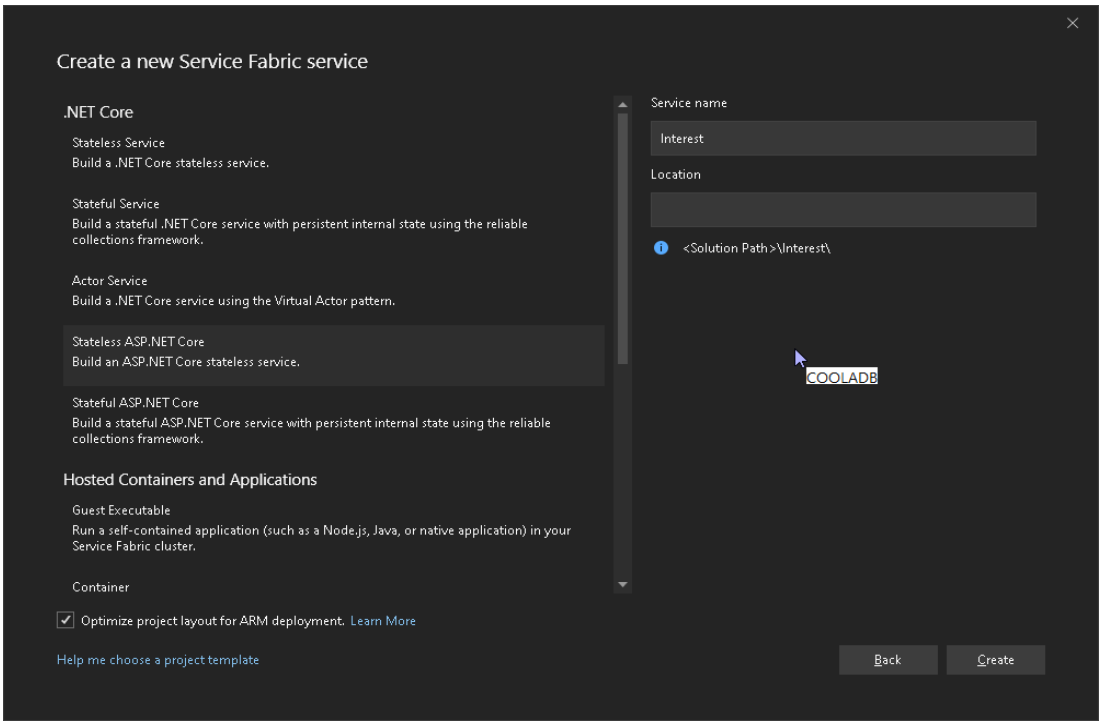
3. In the New Project dialog, choose ➤ Service Fabric Application and Next .



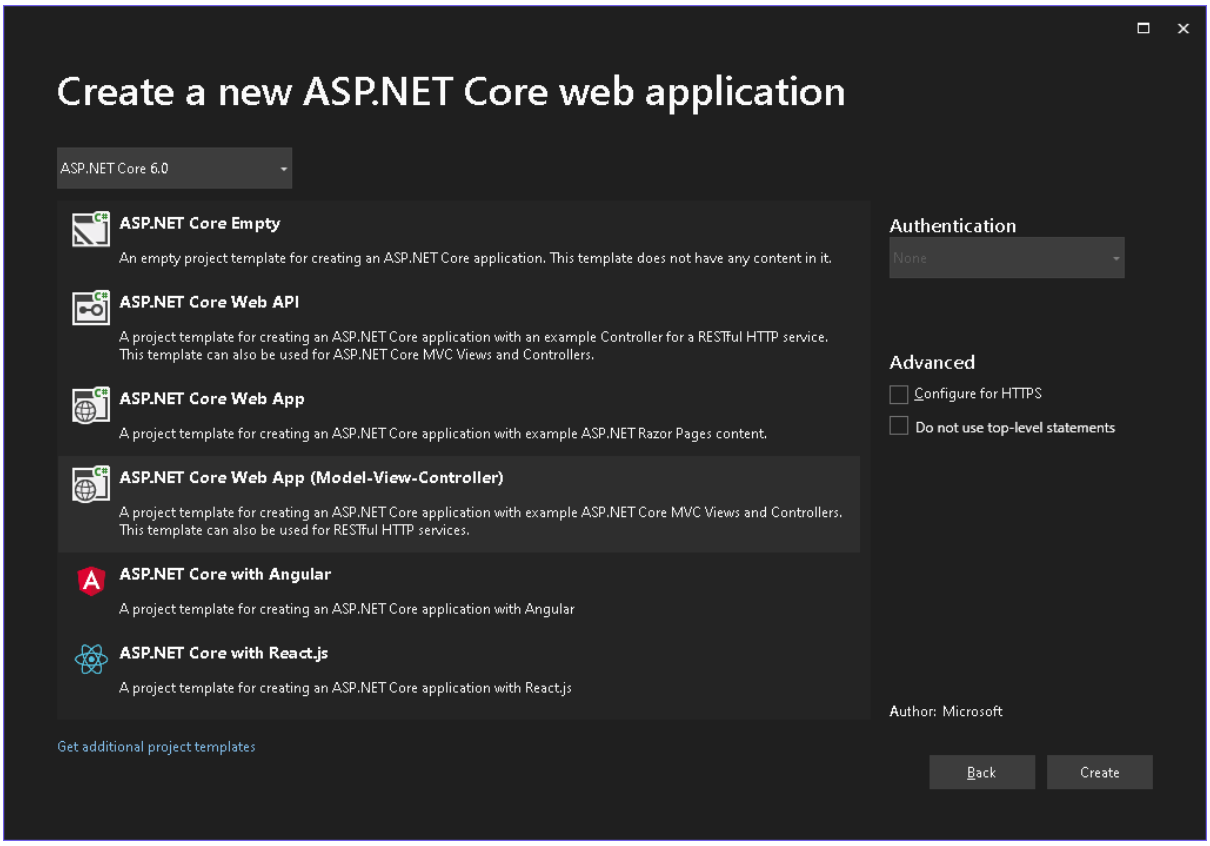
4. Name the Service Fabric application **Simple** and click Create .



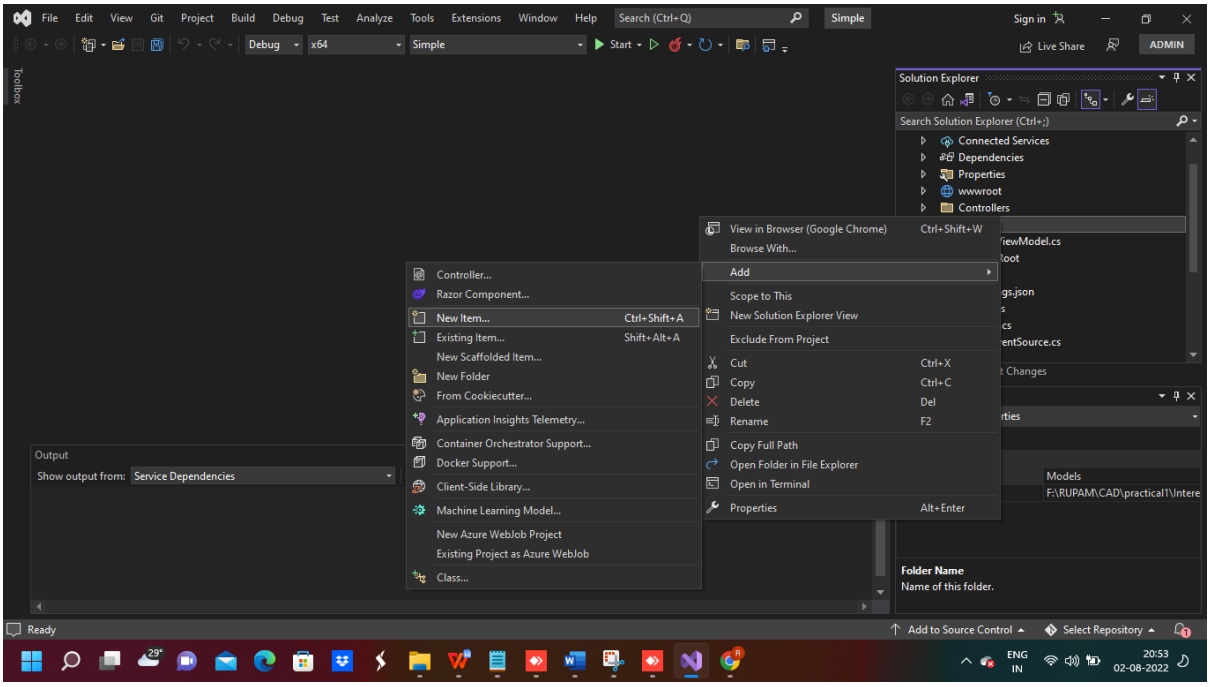
5. Choose Stateful ASP.NET Core, and give the services name **Interest** and click on Create .

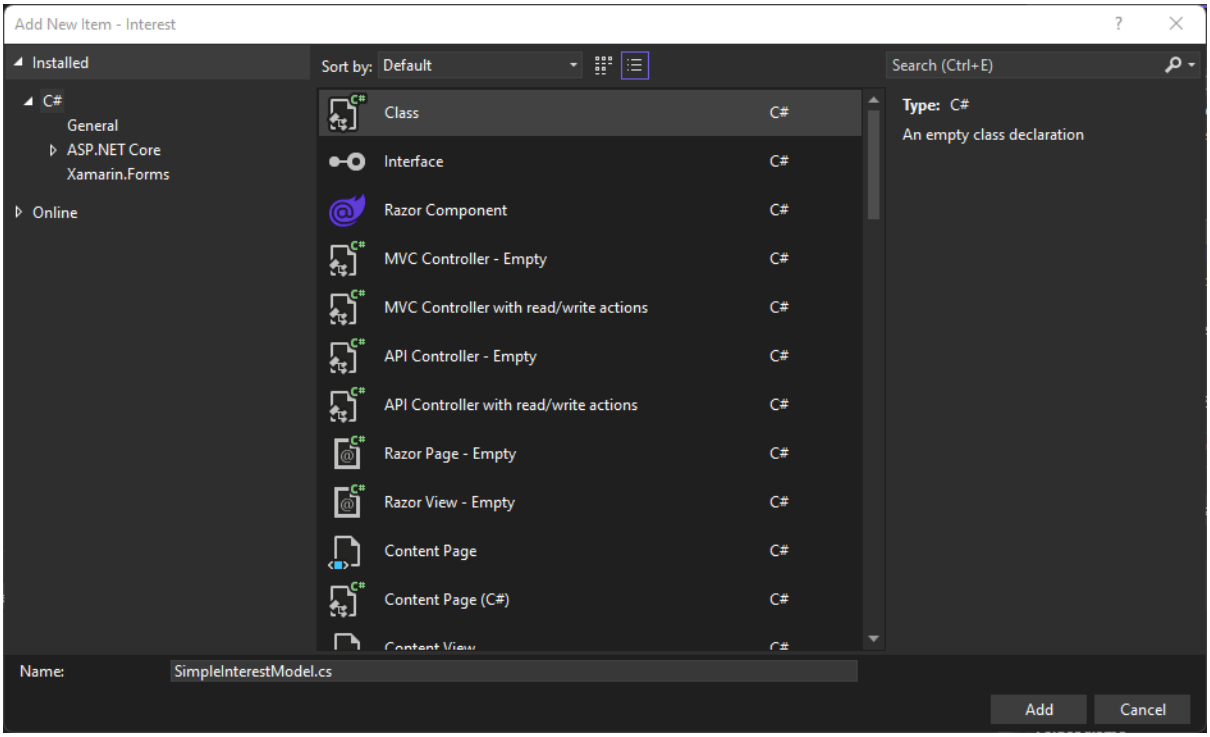


6. Select the ASP.Net Core (MVC) and click on Create .



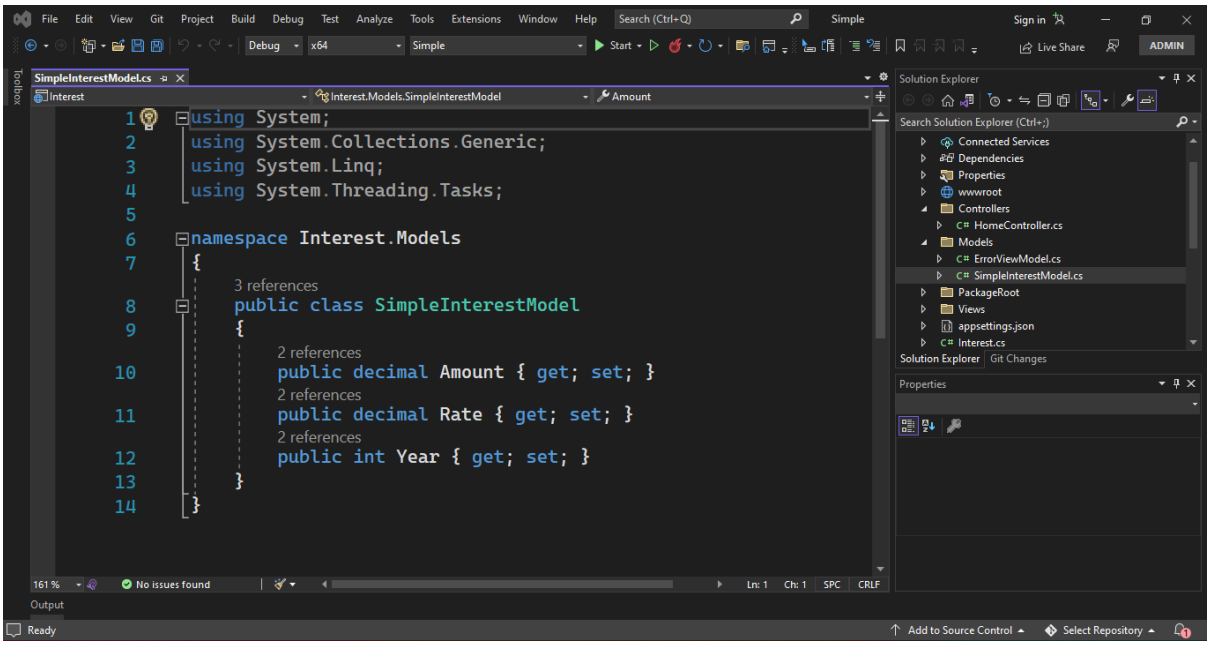
7. Now create a new Folder and name it Models & Right click on Model Folder and click on ➤Add ➤New Item ➤ Class ➤ Name it **"SimpleInterestModel.cs "**





8. Type the Following code In **SimpleInterestModel.cs**

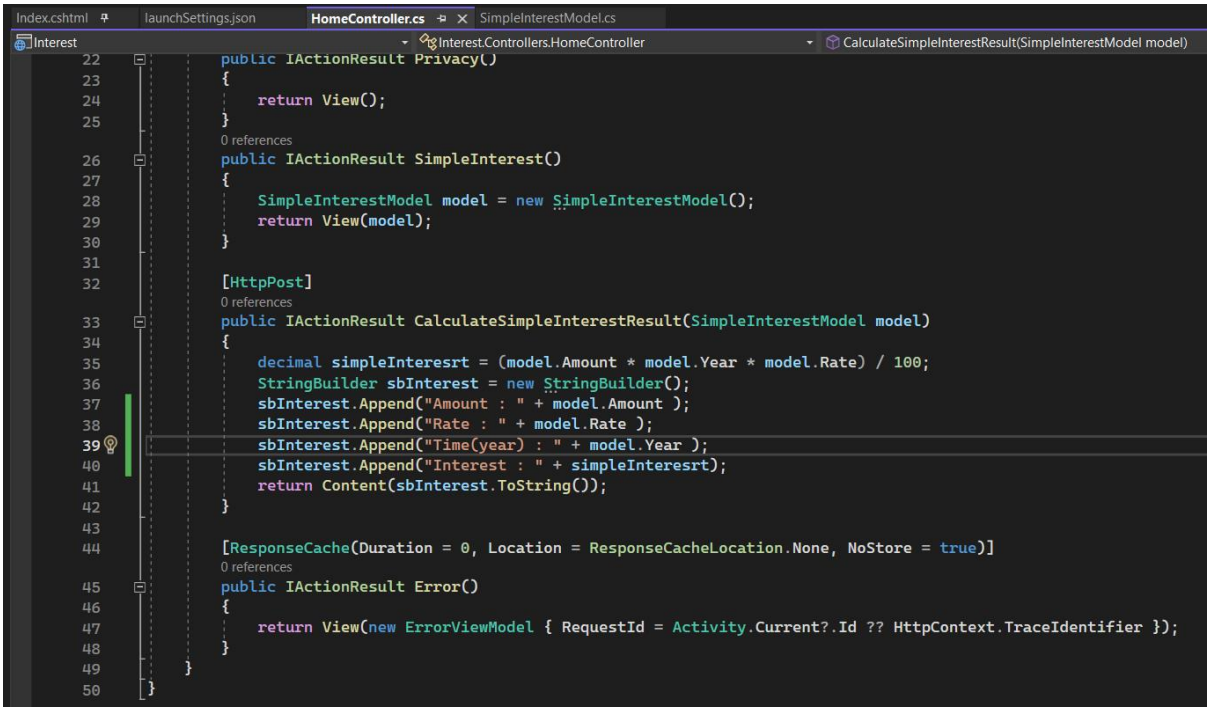
```
{
public decimal Amount { get; set; }
public decimal Rate { get; set; }
public int Year { get; set; }
}
```



9. Now Click on **HomeController.cs** File And Type the Following code .

```
using Interest.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using System.Text;
namespace Interest.Controllers
{

    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Privacy()
        {
            return View();
        }
        public IActionResult SimpleInterest()
        {
            SimpleInterestModel model = new SimpleInterestModel();
            return View(model);
        }
        [HttpPost]

        public IActionResult CalculateSimpleInterestResult(SimpleInterestModel model)
        {
            decimal simpleInteresrt = (model.Amount * model.Year * model.Rate) / 100;
            StringBuilder sbInterest = new StringBuilder();
            sbInterest.Append("<b>Amount :</b> " + model.Amount + "<br/>");
            sbInterest.Append("<b>Rate :</b> " + model.Rate + "<br/>");
            sbInterest.Append("<b>Time(year) :</b> " + model.Year + "<br/>");
            sbInterest.Append("<b>Interest :</b> " + simpleInteresrt);
```

```
            return Content(sbInterest.ToString());
        }
        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```



10. Now Click ➤ view ➤ Home **Index.cshtml** File And Type the Following code .

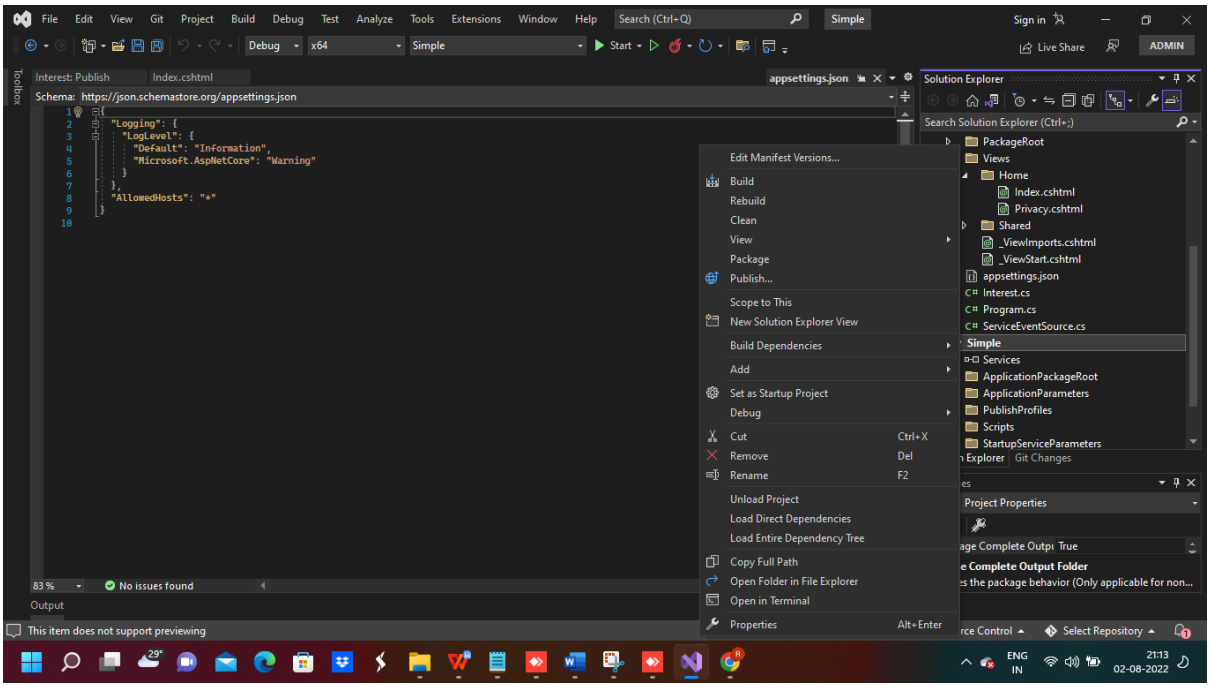
```
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with
ASP.NET Core</a>.</p>
</div>
@model SimpleInterestModel

@{
    ViewBag.Title = "SimpleInterest";
}

<h2>Calculate Simple Interest</h2>

@using (Html.BeginForm("CalculateSimpleInterestResult", "Home", FormMethod.Post))
{

    <fieldset>
        <legend>Calculate Simple Interest</legend>
        <div id="divInterestDetails"></div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Amount)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Amount)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Rate)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Rate)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Year)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Year)
        </div>
        <p>
```

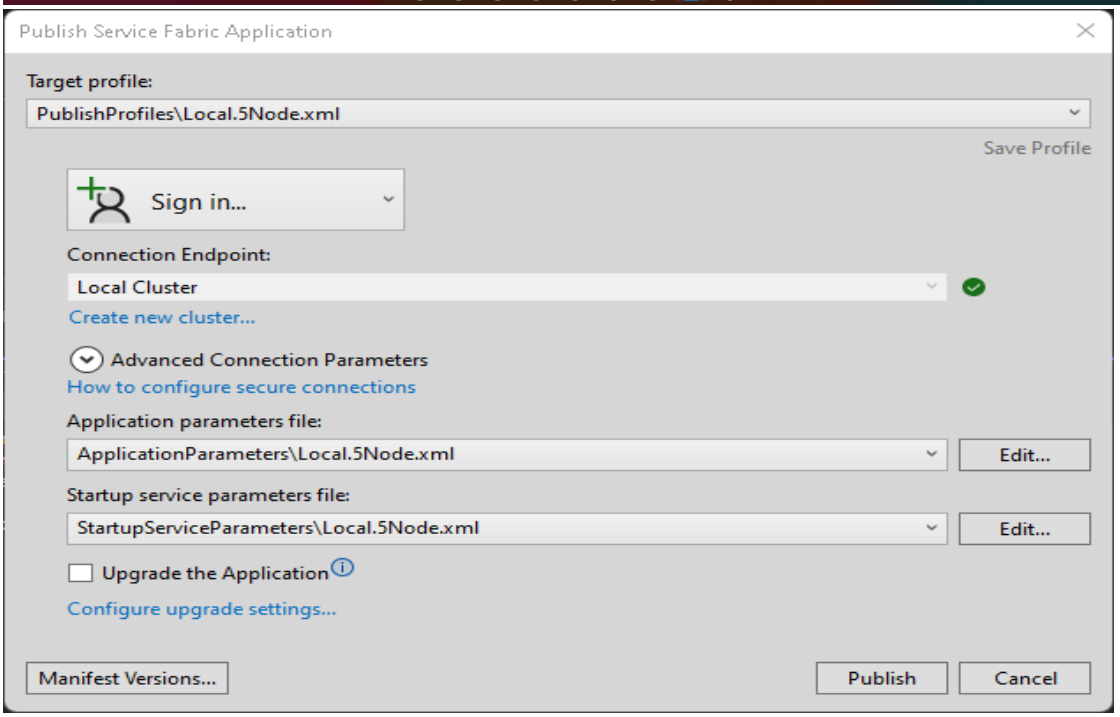
```
            <input type="submit" value="Calculate" />
        </p>
    </fieldset>
}
```

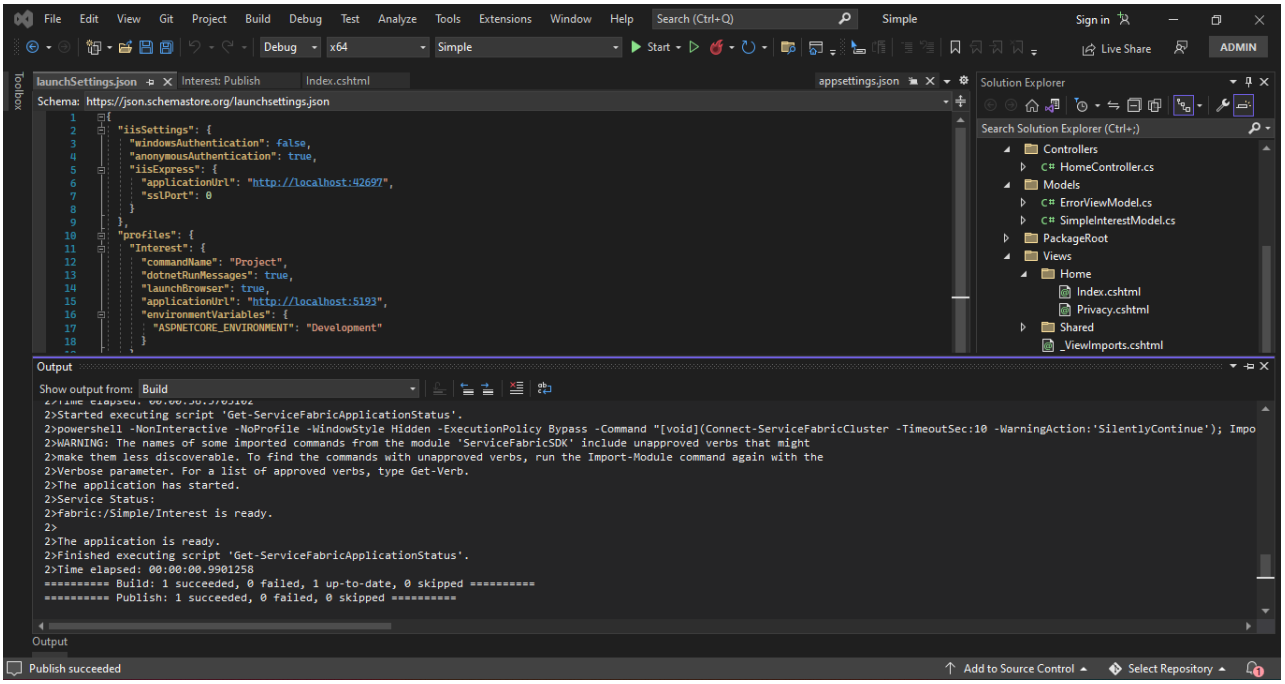
Index.cshtml launchSettings.json HomeController.cs SimpleInterestModel.cs

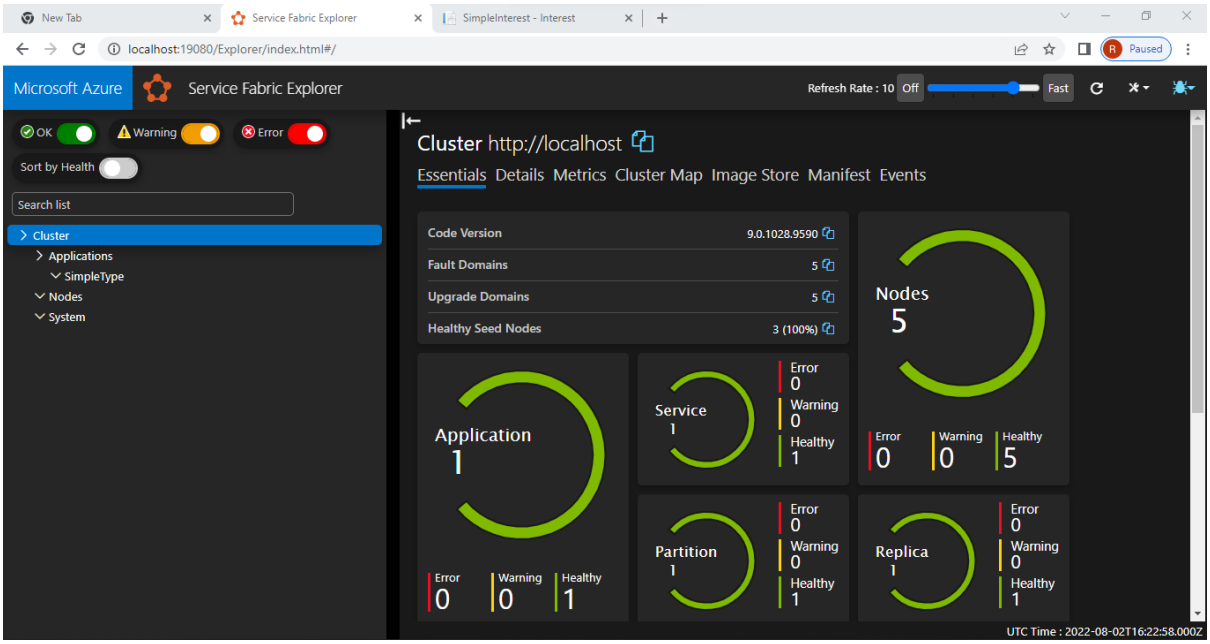
```
@using (Html.BeginForm("CalculateSimpleInterestResult", "Home", FormMethod.Post))
{

    <fieldset>
        <legend>Calculate Simple Interest</legend>
        <div id="divInterestDetails"></div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Amount)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Amount)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Rate)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Rate)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Year)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Year)
        </div>
        <p>
            <input type="submit" value="Calculate" />
        </p>
    </fieldset>
}
```

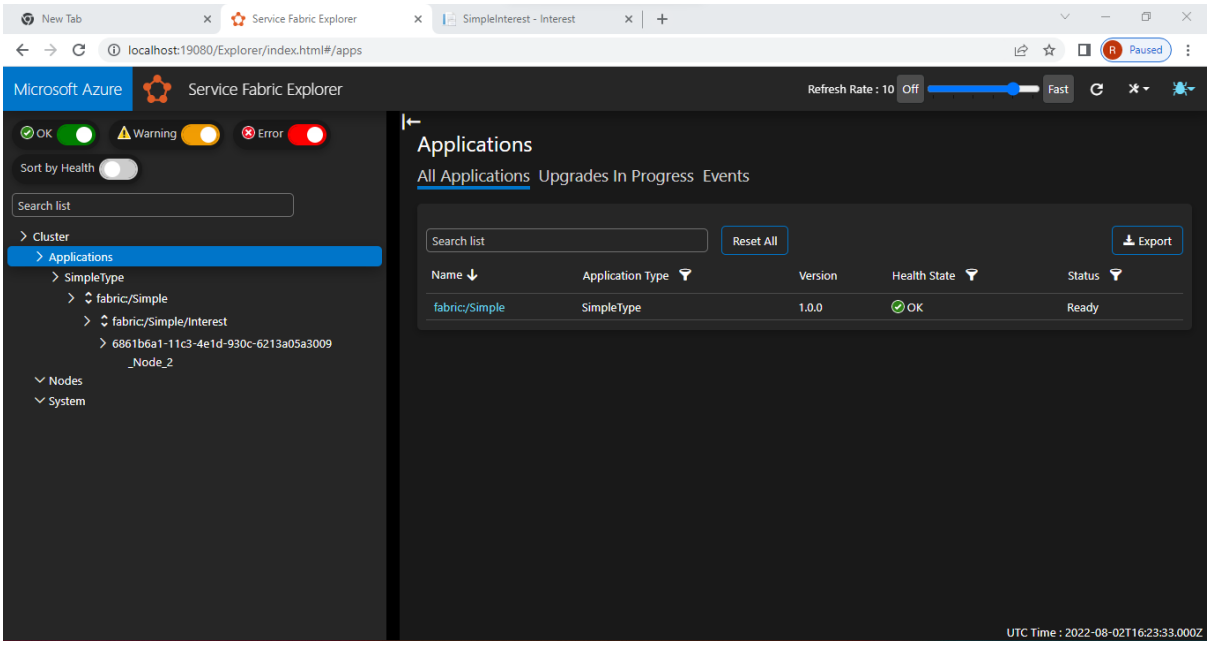
11. Now Right click on Project **“Simple”** ➤ Publish ➤ Select the Node 5 and click on Publish and see the **output** .

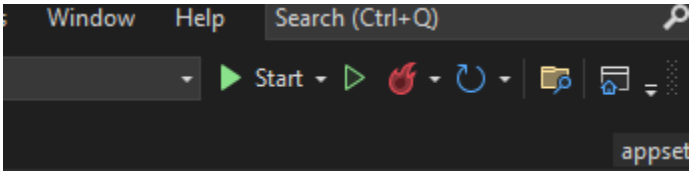


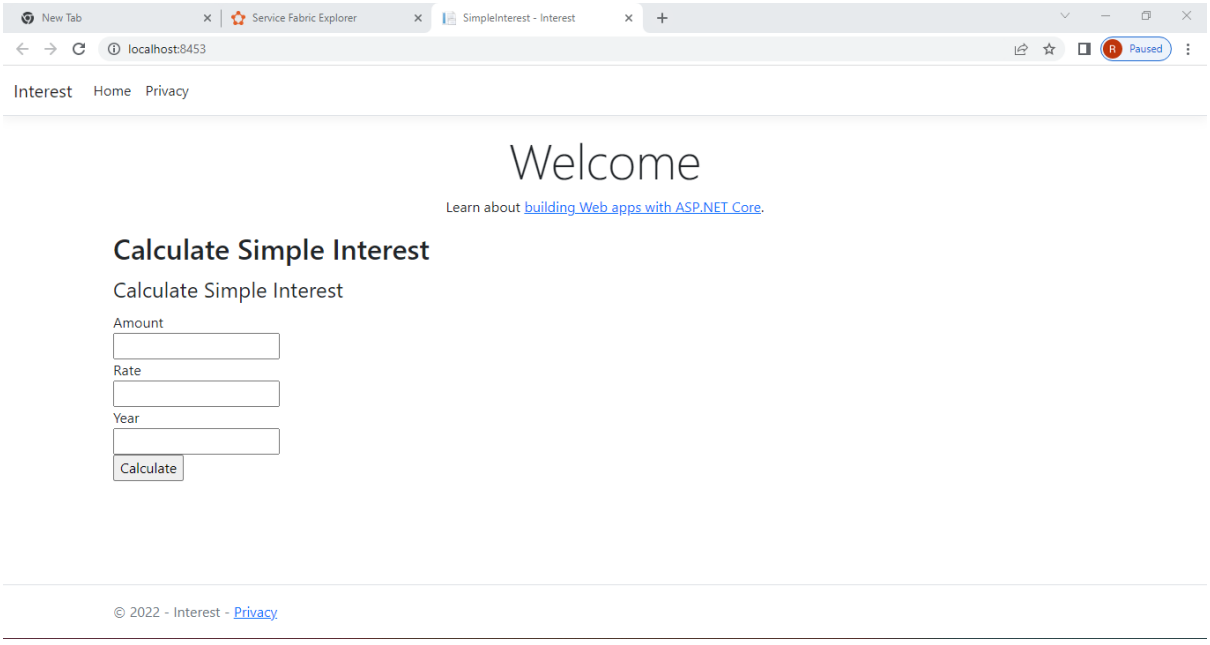


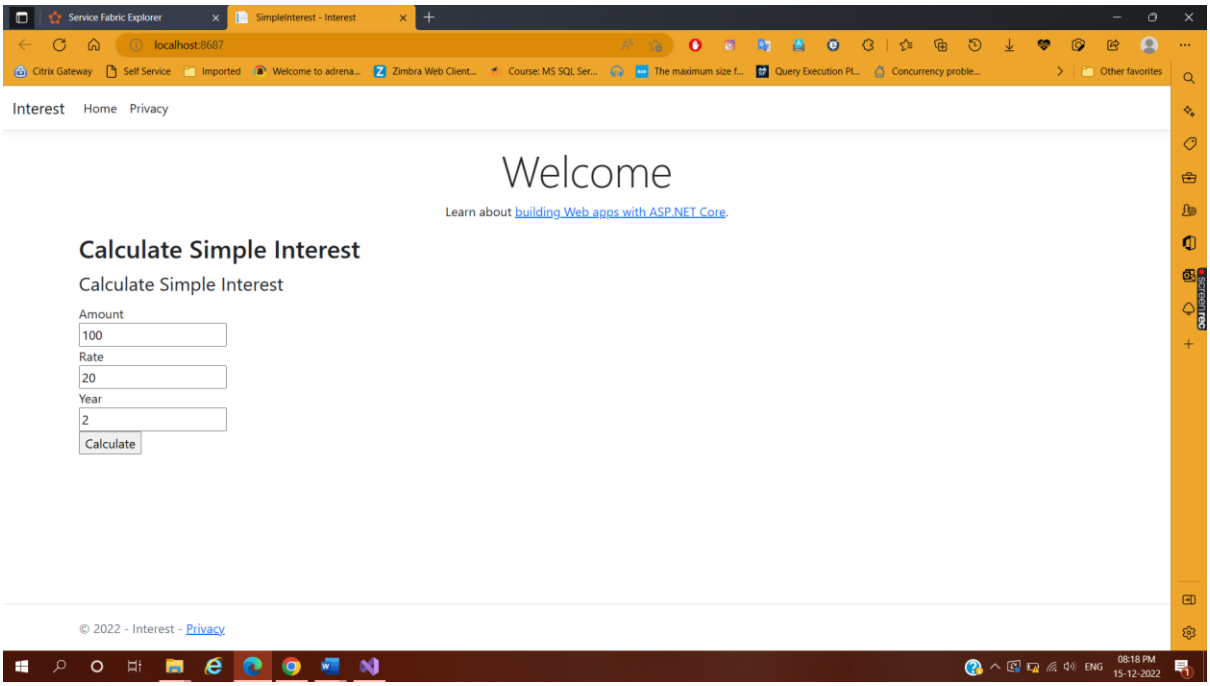

launchSettings.json
Interest: Publish
Index.cshtml
appsettings.json
Schema: https://json.schemastore.org/launchsettings.json
{
"iisSettings": {
"windowsAuthentication": false,
"anonymousAuthentication": true,
"iisExpress": {
"applicationUrl": "http://localhost:42697",
"sslPort": 0
}
},
"profiles": {
"Interest": {
"commandName": "Project",
"dotnetRunMessages": true,
"launchBrowser": true,
"applicationUrl": "http://localhost:5193",
"environmentVariables": {
"ASPNETCORE_ENVIRONMENT": "Development"
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Controllers
HomeController.cs
Models
ErrorViewModel.cs
SimpleInterestModel.cs
PackageRoot
Views
Home
Index.cshtml
Privacy.cshtml
Shared
_ViewImports.cshtml
Output
Show output from: Build
2>Started executing script 'Get-ServiceFabricApplicationStatus'.
2>powershell -NonInteractive -NoProfile -WindowStyle Hidden -ExecutionPolicy Bypass -Command "[void](Connect-ServiceFabricCluster -TimeoutSec:10 -WarningAction:'SilentlyContinue'); Impo
2>WARNING: The names of some imported commands from the module 'ServiceFabricSDK' include unapproved verbs that might
2>make them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the
2>Verbose parameter. For a list of approved verbs, type Get-Verb.
2>The application has started.
2>Service Status:
2>fabric:/Simple/Interest is ready.
2>
2>The application is ready.
2>Finished executing script 'Get-ServiceFabricApplicationStatus'.
2>Time elapsed: 00:00:00.9901258
========== Build: 1 succeeded, 0 failed, 1 up-to-date, 0 skipped ==========
========== Publish: 1 succeeded, 0 failed, 0 skipped ==========
Output
Publish succeeded
Add to Source Control
Select Repository

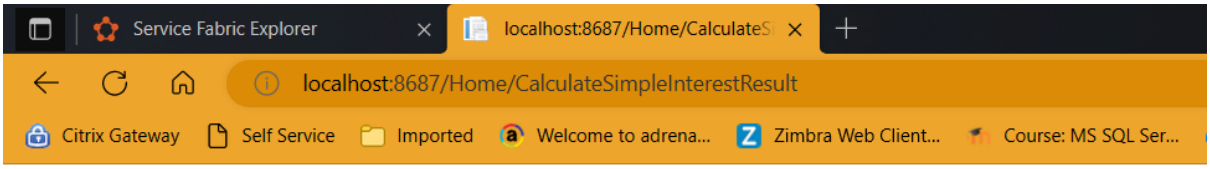

New Tab
Service Fabric Explorer
SimpleInterest - Interest
localhost:19080/Explorer/index.html#/
Paused
Microsoft Azure
Service Fabric Explorer
Refresh Rate : 10
Off
Fast
OK
Warning
Error
Sort by Health
Search list
Cluster
Applications
SimpleType
Nodes
System
Cluster http://localhost
Essentials Details Metrics Cluster Map Image Store Manifest Events
Code Version 9.0.1028.9590
Fault Domains 5
Upgrade Domains 5
Healthy Seed Nodes 3 (100%)
Nodes
5
Error 0
Warning 0
Healthy 5
Application
1
Error 0
Warning 0
Healthy 1
Service
1
Error 0
Warning 0
Healthy 1
Partition
1
Error 0
Warning 0
Healthy 1
Replica
1
Error 0
Warning 0
Healthy 1
UTC Time : 2022-08-02T16:22:58.000Z


New Tab
Service Fabric Explorer
SimpleInterest - Interest
localhost:19080/Explorer/index.html#/apps
Paused
Microsoft Azure
Service Fabric Explorer
Refresh Rate : 10
Off
Fast
OK
Warning
Error
Sort by Health
Search list
Cluster
Applications
SimpleType
fabric/Simple
fabric/Simple/Interest
6861b6a1-11c3-4e1d-930c-6213a05a3009
_Node_2
Nodes
System
Applications
All Applications
Upgrades In Progress
Events
Search list
Reset All
Export
Name
Application Type
Version
Health State
Status
fabric:/Simple
SimpleType
1.0.0
OK
Ready
UTC Time : 2022-08-02T16:23:33.000Z


Window
Help
Search (Ctrl+Q)
Start
appset


New Tab
Service Fabric Explorer
SimpleInterest - Interest
localhost:8453
Paused
Interest
Home
Privacy
Welcome
Learn about building Web apps with ASP.NET Core.
Calculate Simple Interest
Calculate Simple Interest
Amount
Rate
Year
Calculate
© 2022 - Interest - Privacy

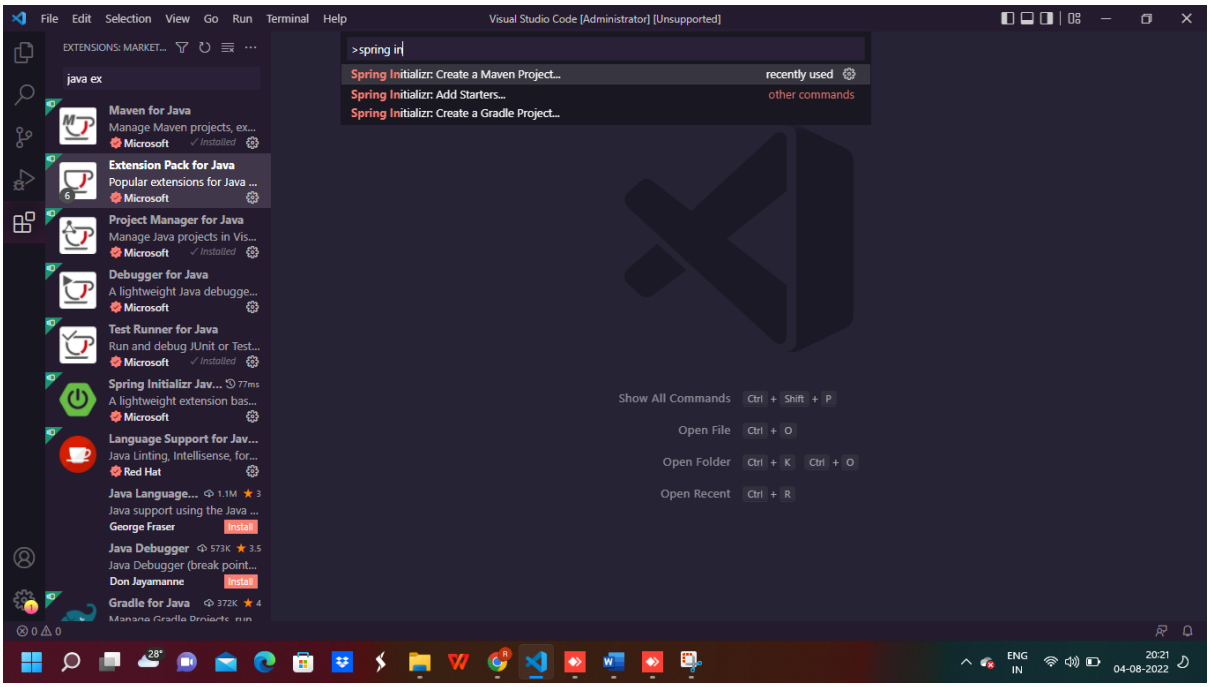




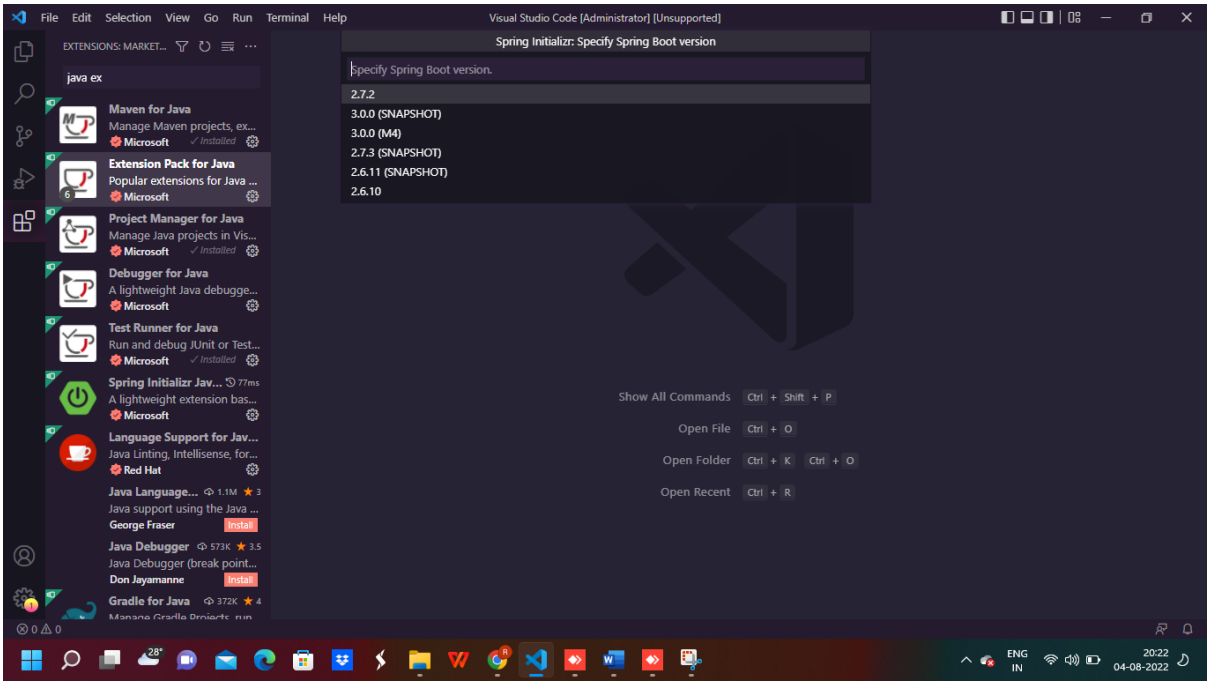

```
Amount : 100Rate : 20Time(year) : 2Interest : 40
```

Steps :-

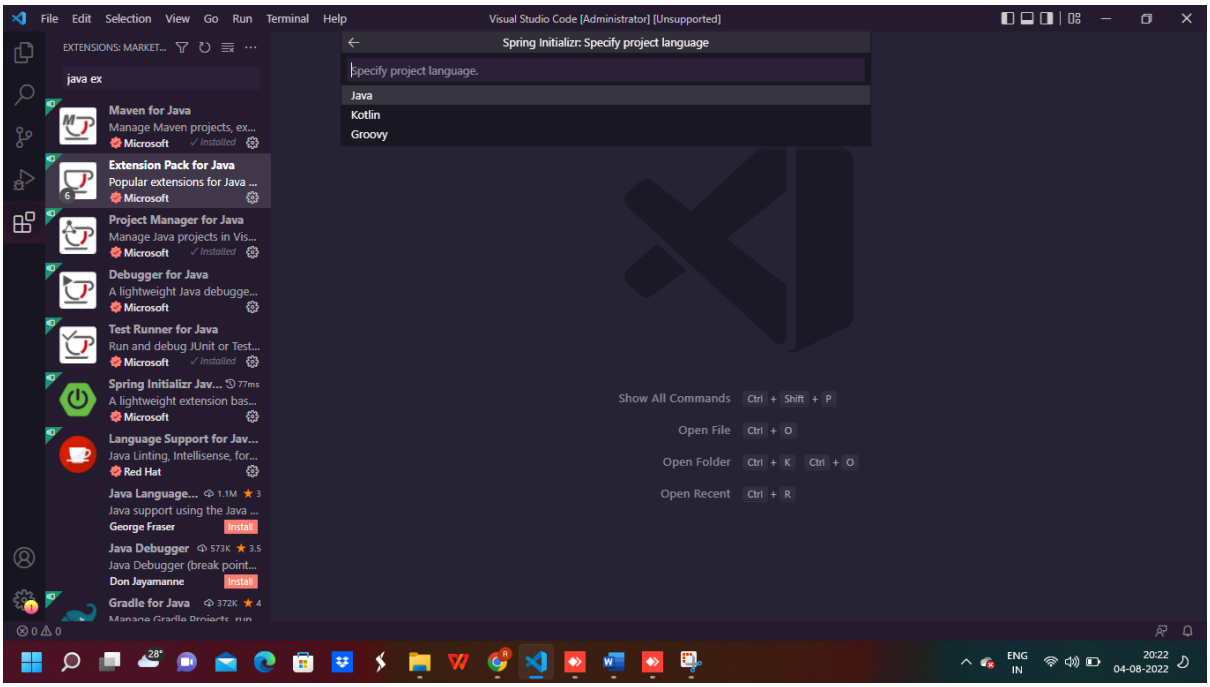
1. Open the Visual Stdio Code
2. Click **ctrl+Shift+P** and search for Spring Initializer:Create Maven Project and hit enter button



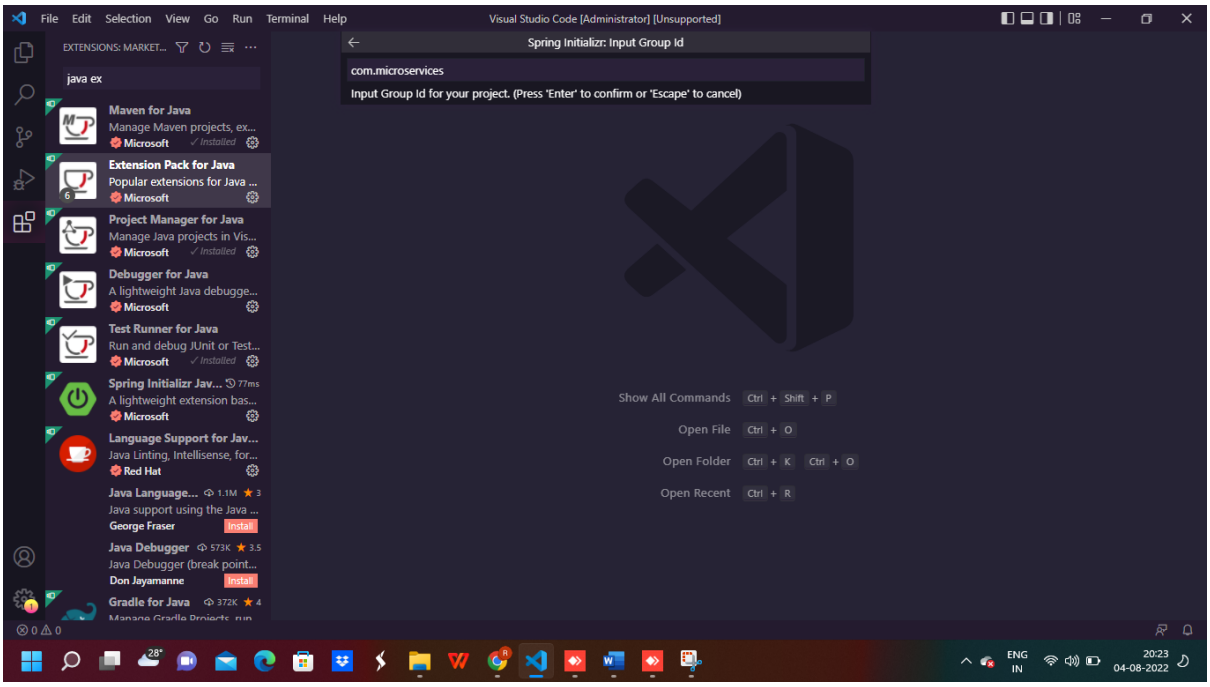
3. Search for **2.7.2** version and press enter .



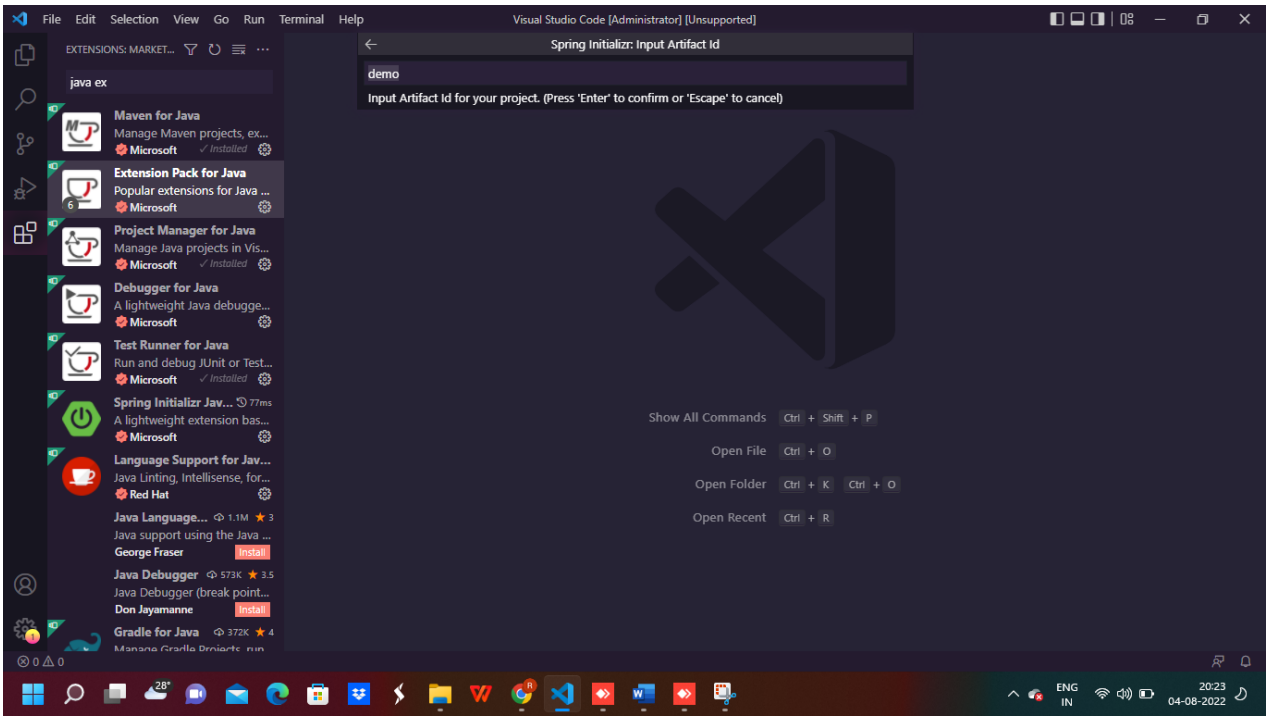
4. Search For **Java** and press enter .



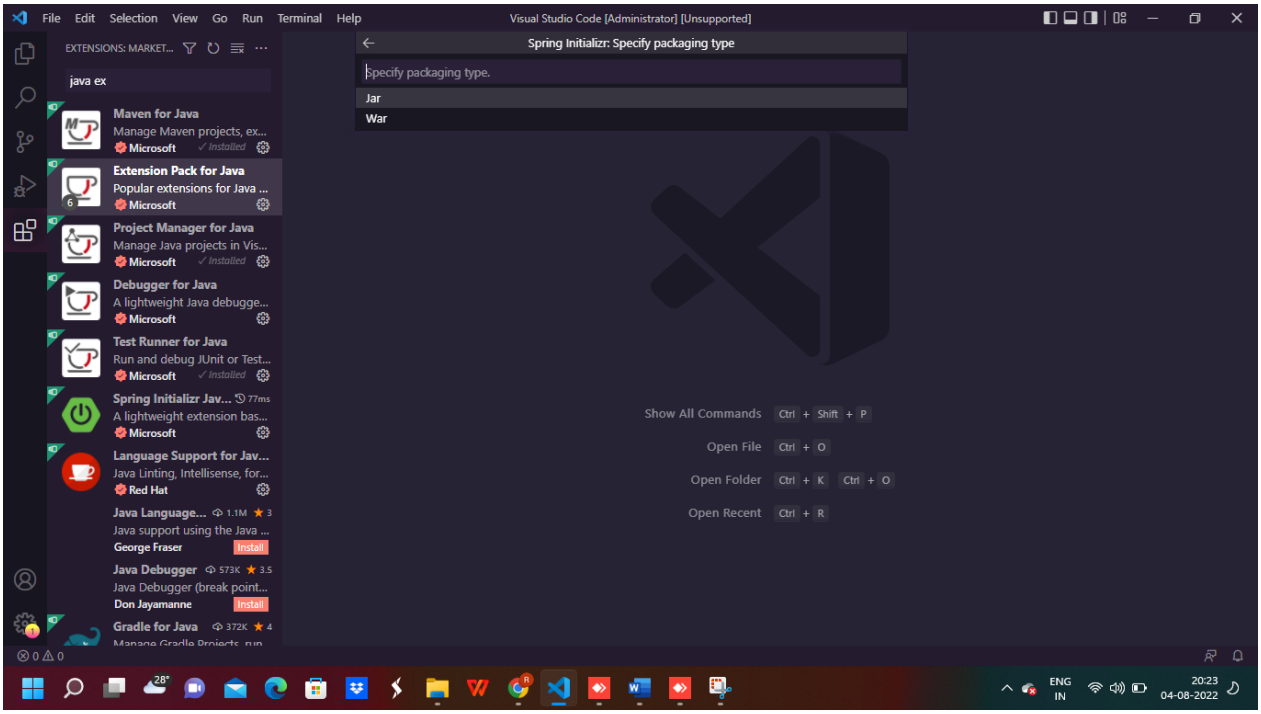
5. Now Type com.microservices and press enter .

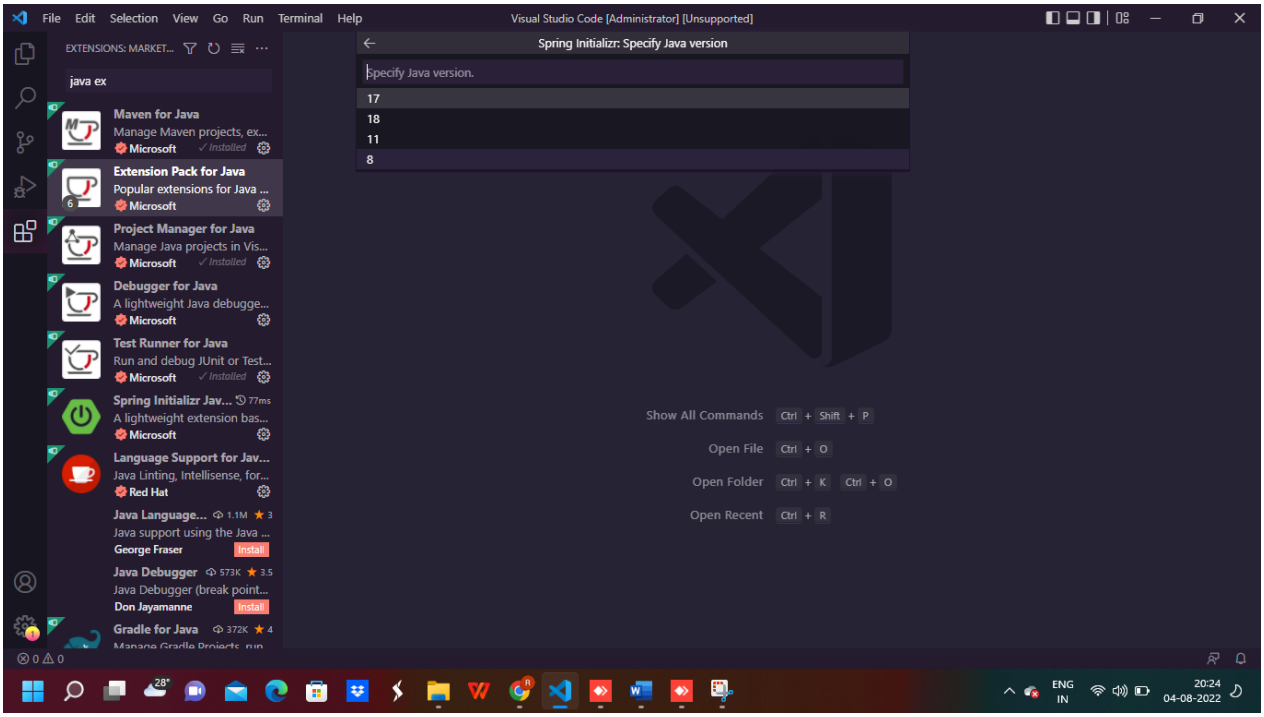


6. Type **Demo** and press enter .

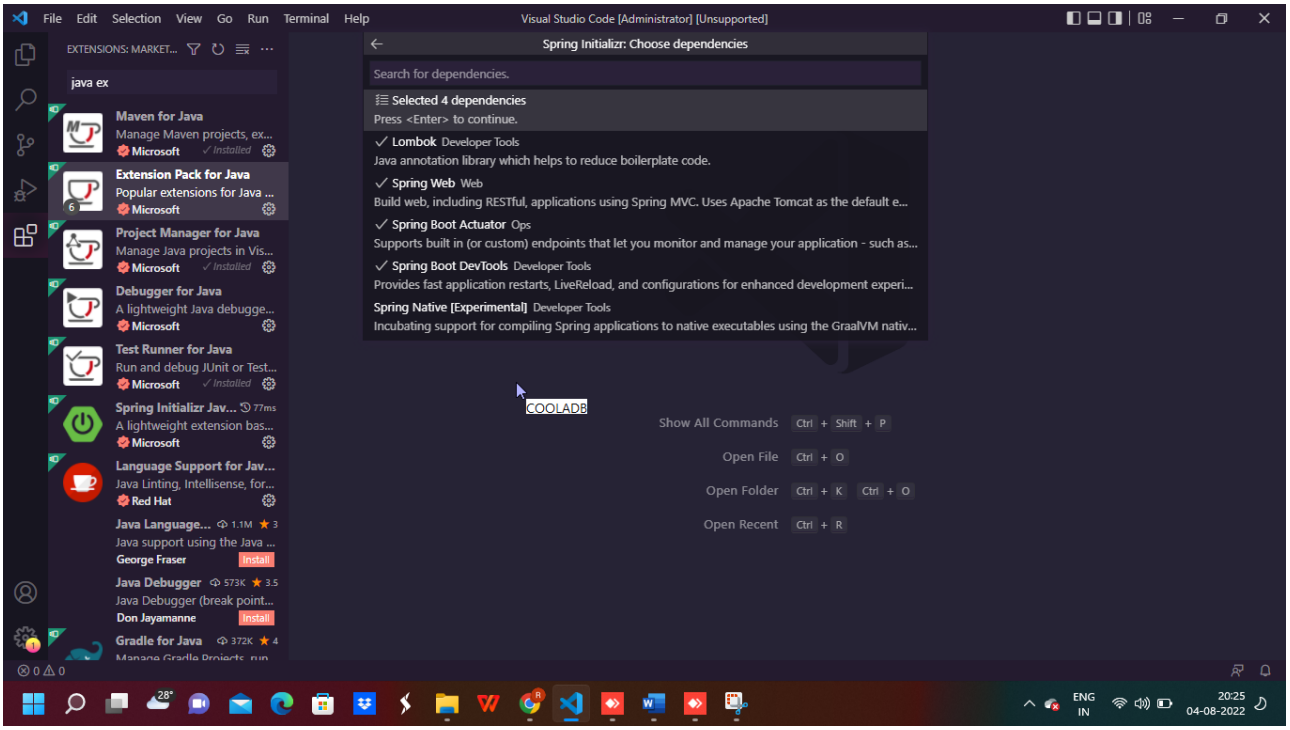


7. Select **Jar** And **8** and press enter .

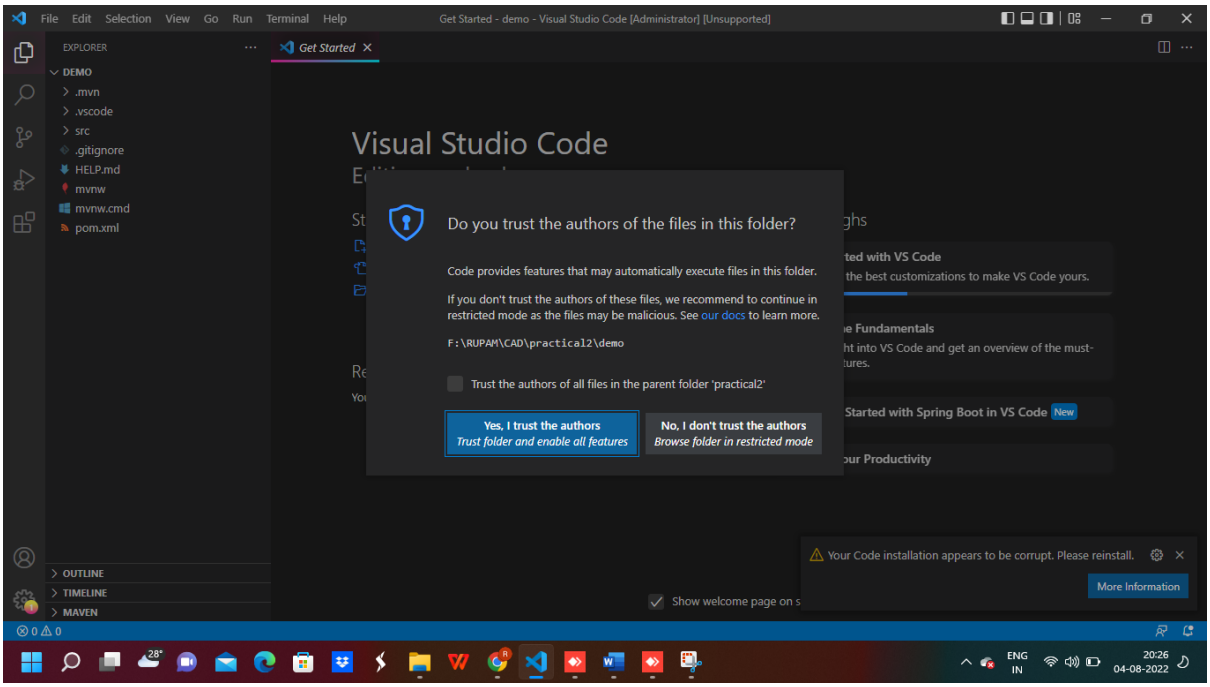




8. Select the four Dependencies :- **Lombok , Spring Web , Spring Boot Actuator , Spring Boot DevTool** and Press Enter .



9. Select the Path To save the Project and open the project



10. Under the Project click src ➤ Main ➤ Java ➤ **DemoApplication.java** Create the new three file **Employee.java , EmployeeService.java , EmployeeController.java** and type the following code :

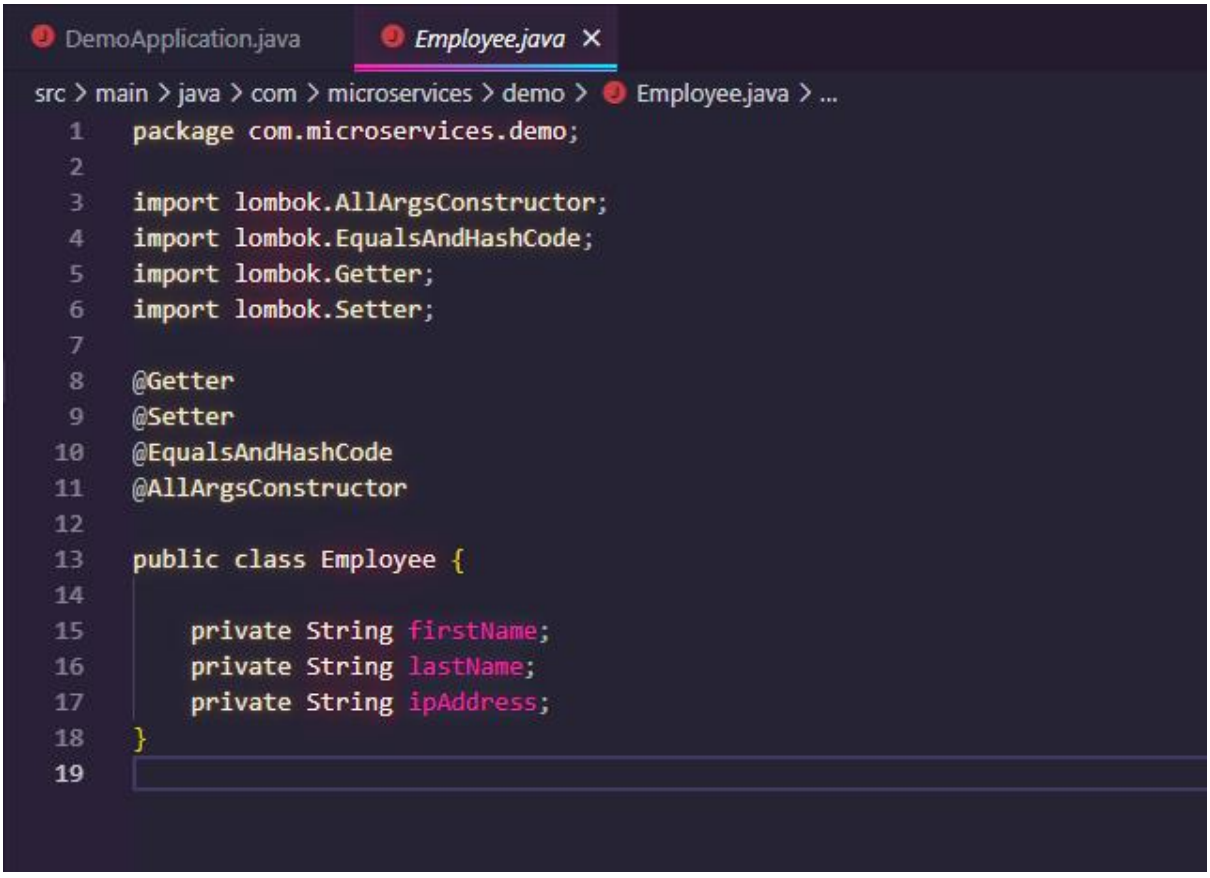
DemoApplication.java

src > main > java > com > microservices > demo > DemoApplication.java > ...

```java
package com.microservices.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class DemoApplication {

    @RequestMapping("/")
    public String home() {
        return "Hello Docker World";
    }

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

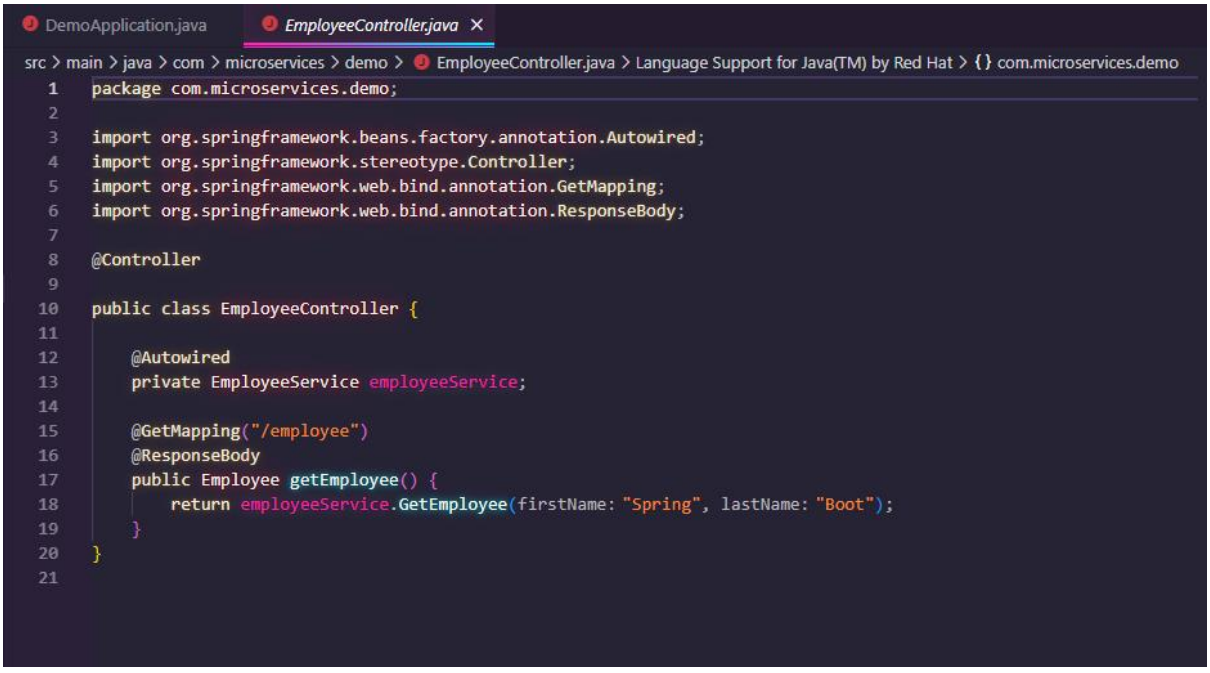
Employee.java

```
package com.microservices.demo;
import lombok.AllArgsConstructor;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;
@Getter
@Setter
@EqualsAndHashCode
@AllArgsConstructor

public class Employee {

   private String firstName;
   private String lastName;
   private String ipAddress;
}
```



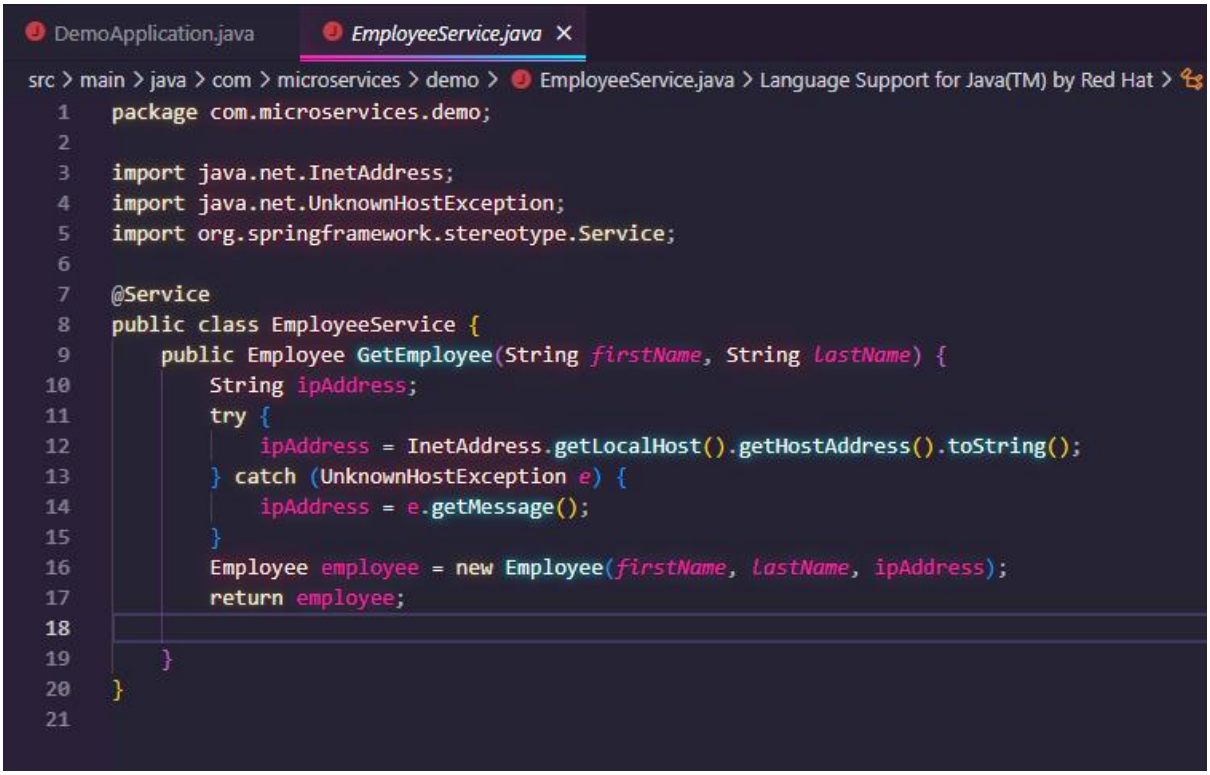
EmployeeController.java

```
package com.microservices.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;
@Controller

public class EmployeeController {

   @Autowired
   private EmployeeService employeeService;
   @GetMapping("/employee")
   @ResponseBody
   public Employee getEmployee(){
      return employeeService.GetEmployee("Spring", "Boot");
   }
}
```

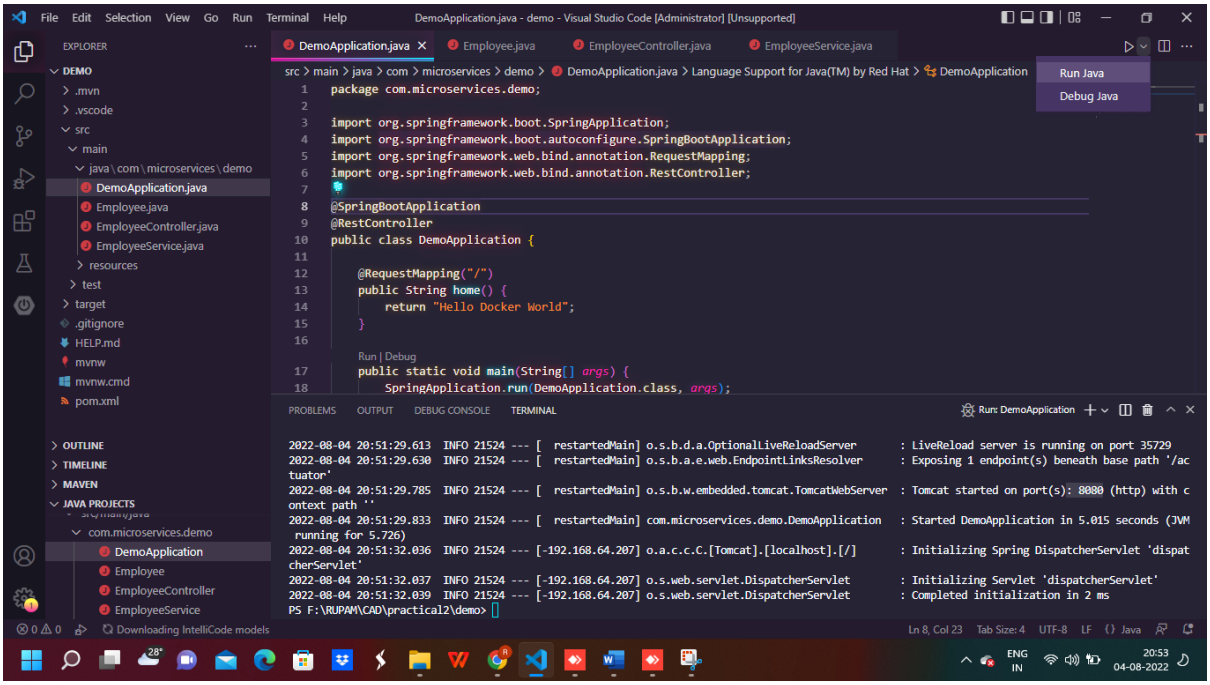


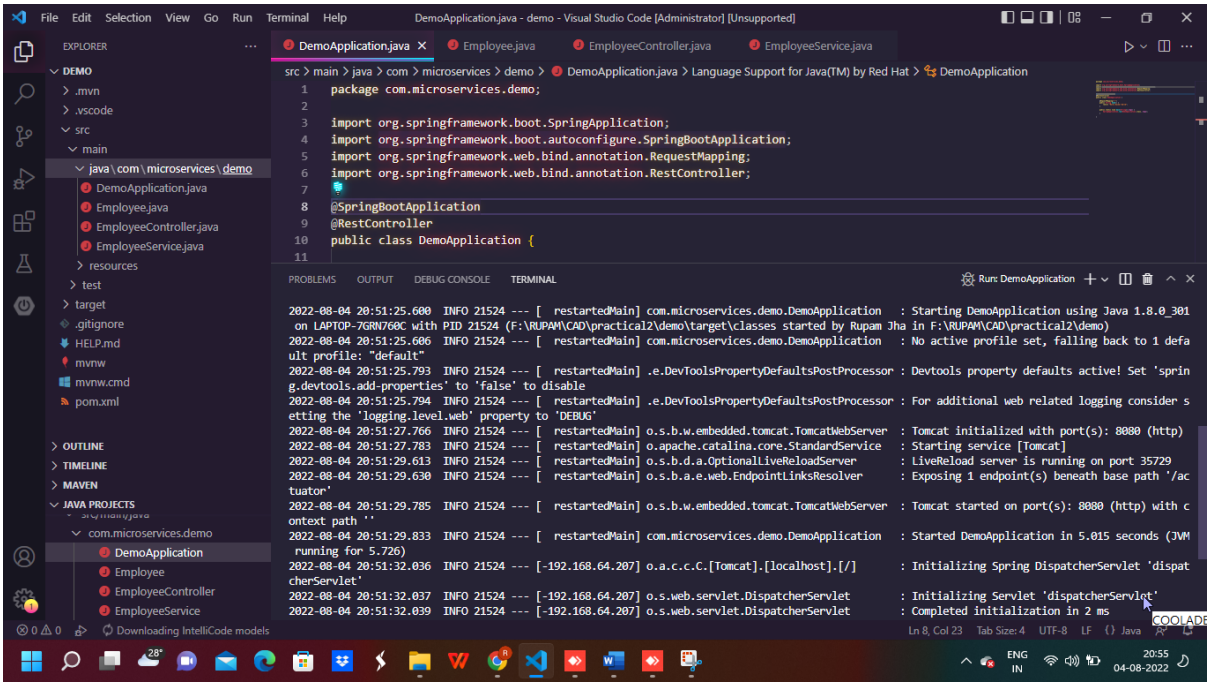
EmployeeService.java

```
package com.microservices.demo;
import java.net.InetAddress;
import java.net.UnknownHostException;
import org.springframework.stereotype.Service;
@Service

public class EmployeeService {
   public Employee GetEmployee(String firstName, String lastName)
{
   String ipAddress;
   try{
      ipAddress=InetAddress.getLocalHost().getHostAddress().toString();
   }
   catch (UnknownHostException e)
   {
      ipAddress=e.getMessage();
   }
   Employee employee= new Employee(firstName,lastName,ipAddress);
   return employee;

}
}
```

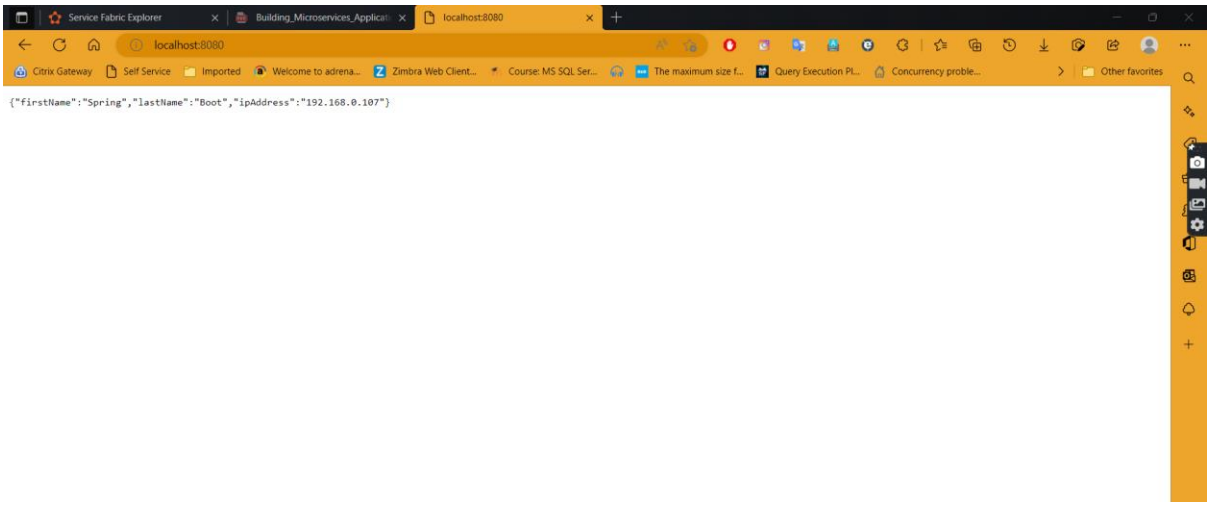


11. Run the **DemoApplication.java** by Clicking on **Run Java** Button .

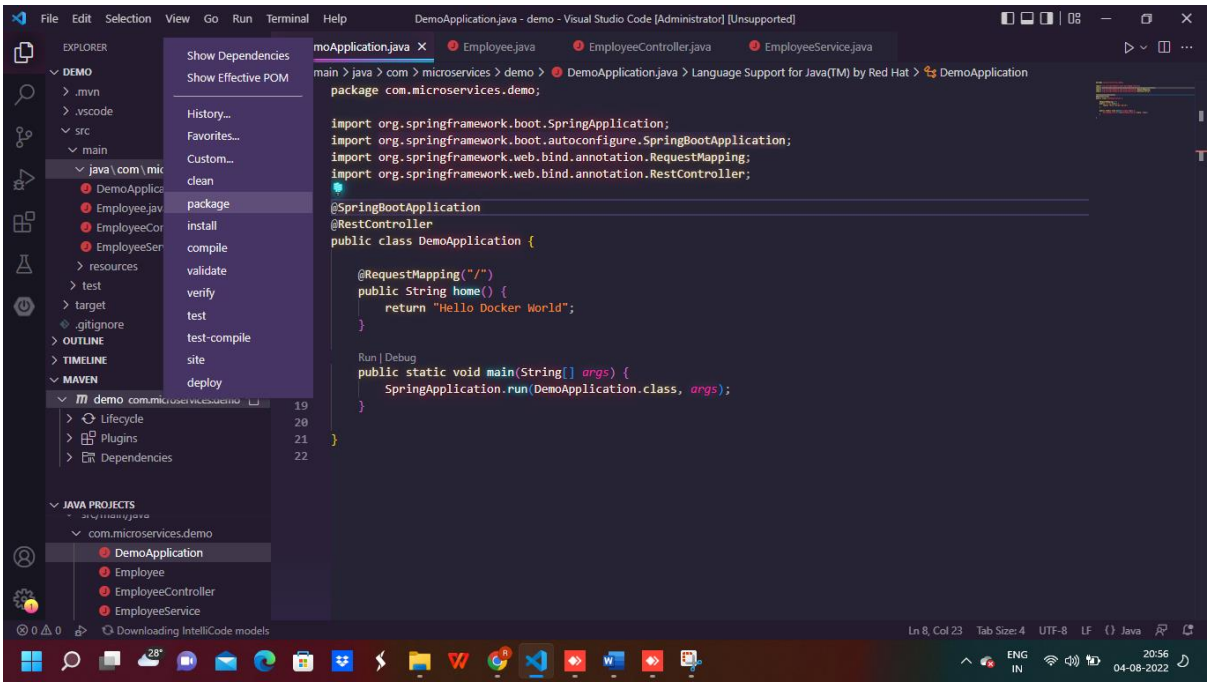


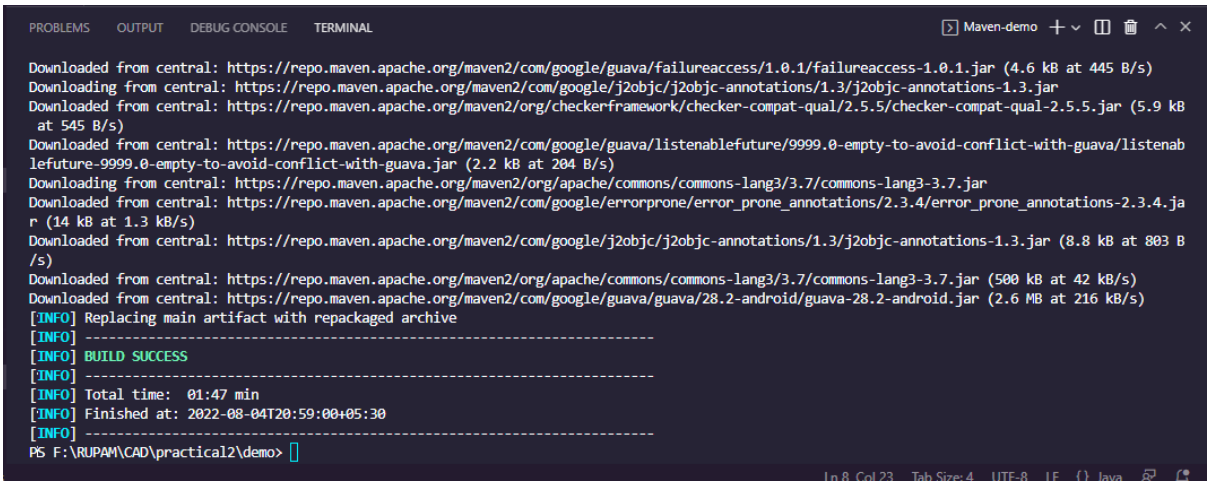


12. After successful running the project you can check the output on **localhost:8080**

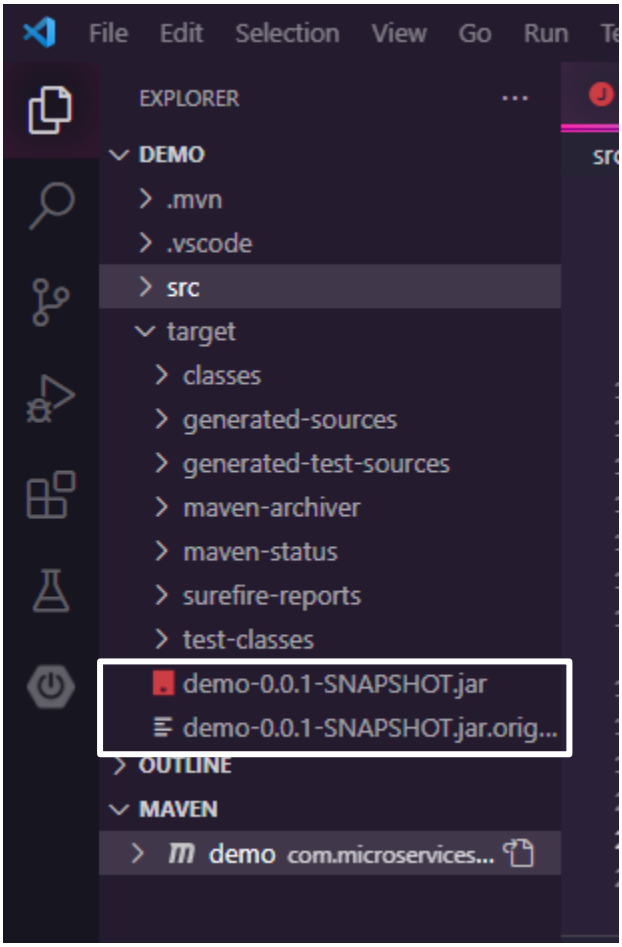


13. Now click on **Right Click on Maven Project** at left hand side of File Explorer Click on **Package** and wait till the project get Build Successfull and **Jar** file get build

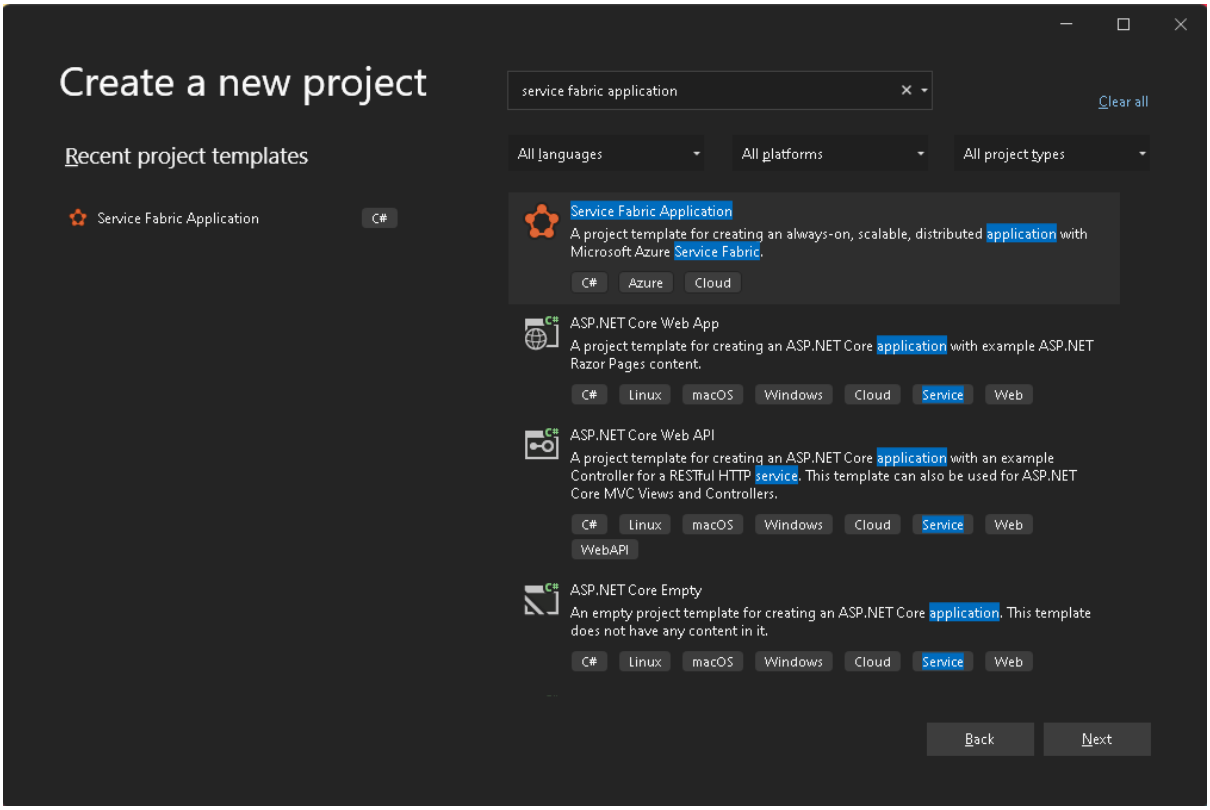




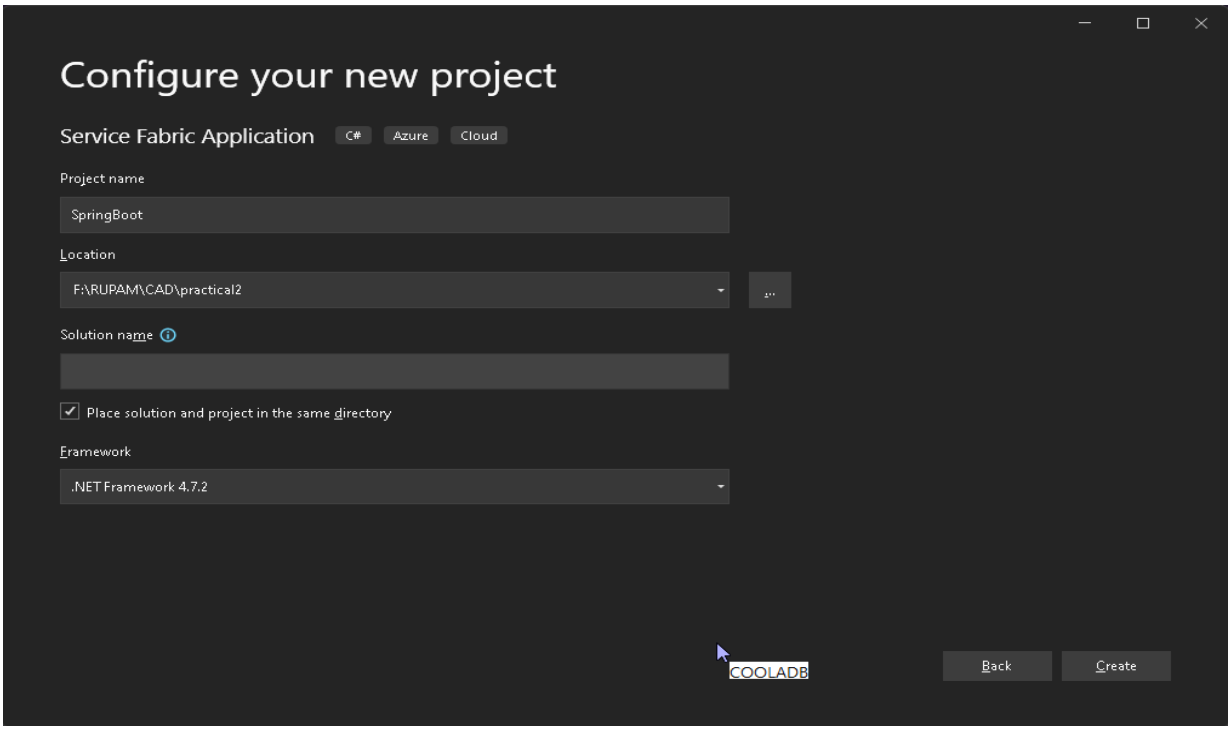
14. **Jar File** has been created successfully .



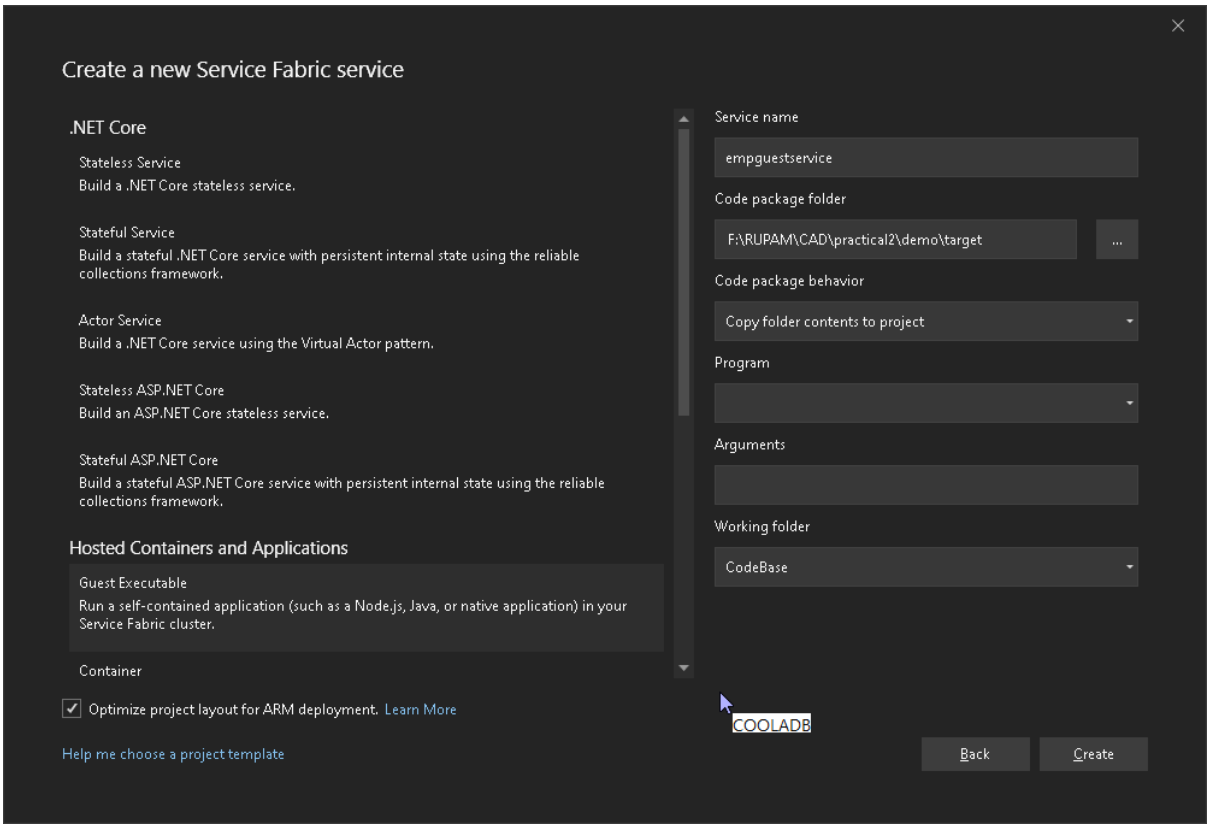
15. Now open the **Visual Studio** 17/19/21 in **Admin** Mode and create a new project and select **Services Fabric Application** and Next



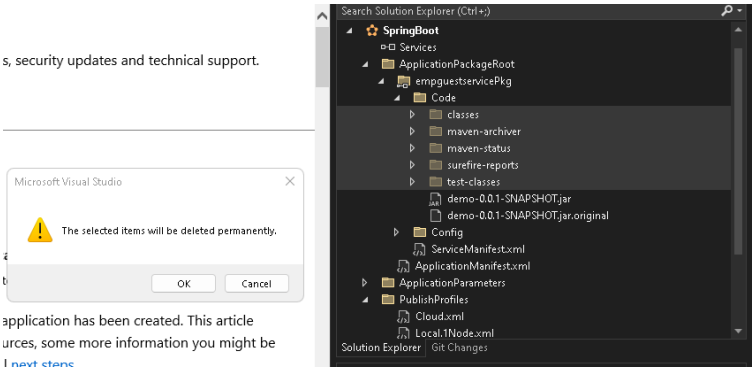
16. Provide the name pf the project "**SpringBoot"** and click in Create



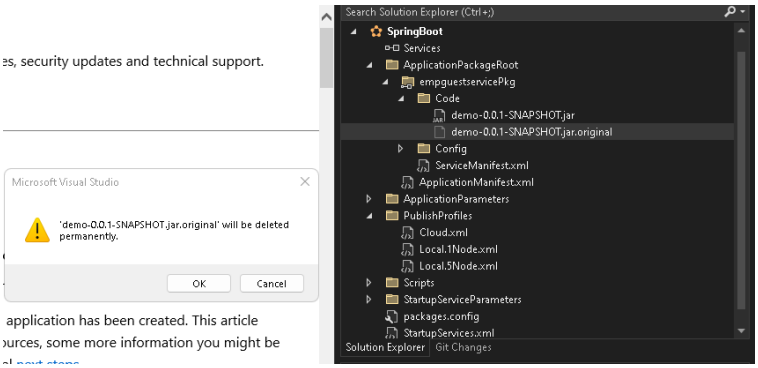
17. Now Select the **Guest Executable** and give the name of the service as "**empguestservice**" & Inside Code package folder, give the path of the folder in which **.jar** file is created.
Code package behaviour will be **Copy folder contents to project** and Working folder will be **CodeBase.**



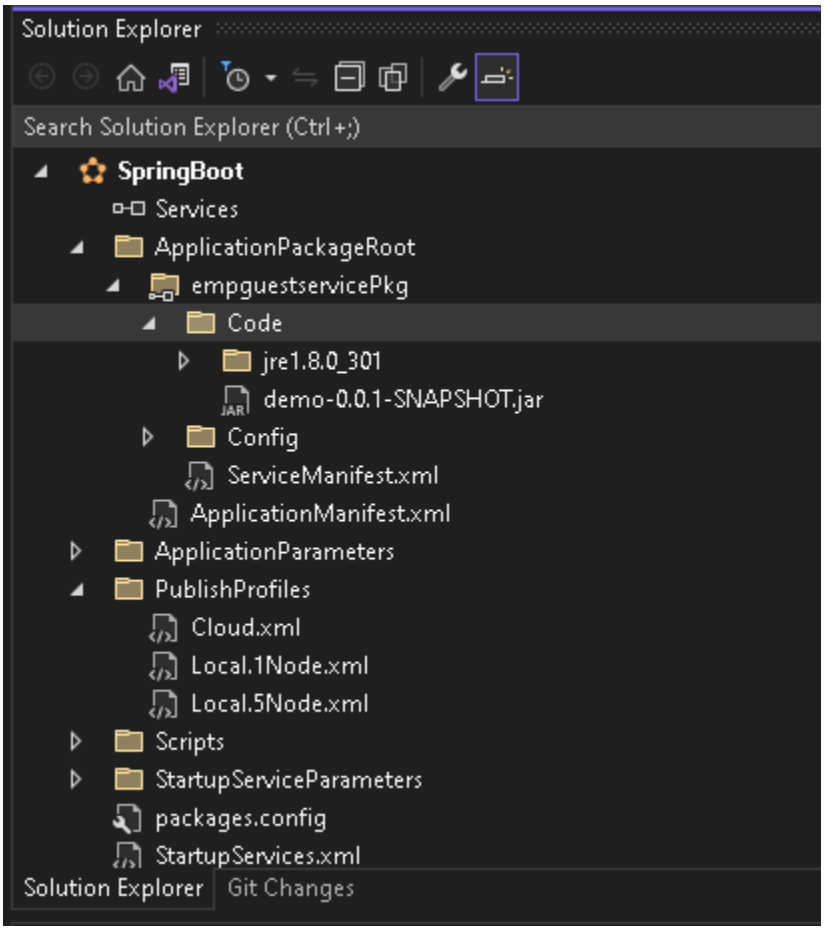
18. Once the project is created, **Delete** the following selected folders.



19. Also **Delete** the **demo-0.0.1-SNAPSHOT.jar.original.**



20. Now, we also need to upload the runtime to run the JAR. Generally, it resides in the JDK installation folder (C:\Program Files\Java). Paste it in the Code folder,



21. Open ServiceManisfest.xml and set the following values.

```
<EntryPoint>
    <ExeHost>
      <Program>jre1.8.0_221\bin\java</Program>
      <Arguments>-jar ..\..\springboot-0.0.1-SNAPSHOT.jar</Arguments>
      <WorkingFolder>CodeBase</WorkingFolder>
            <!-- Uncomment to log console output (both stdout and stderr) to one   of the service's working
            directories. -->
          <!-- <ConsoleRedirection FileRetentionCount="5" FileMaxSizeInKb="2048"/> -->
    </ExeHost>
  </EntryPoint>
 </CodePackage>
```

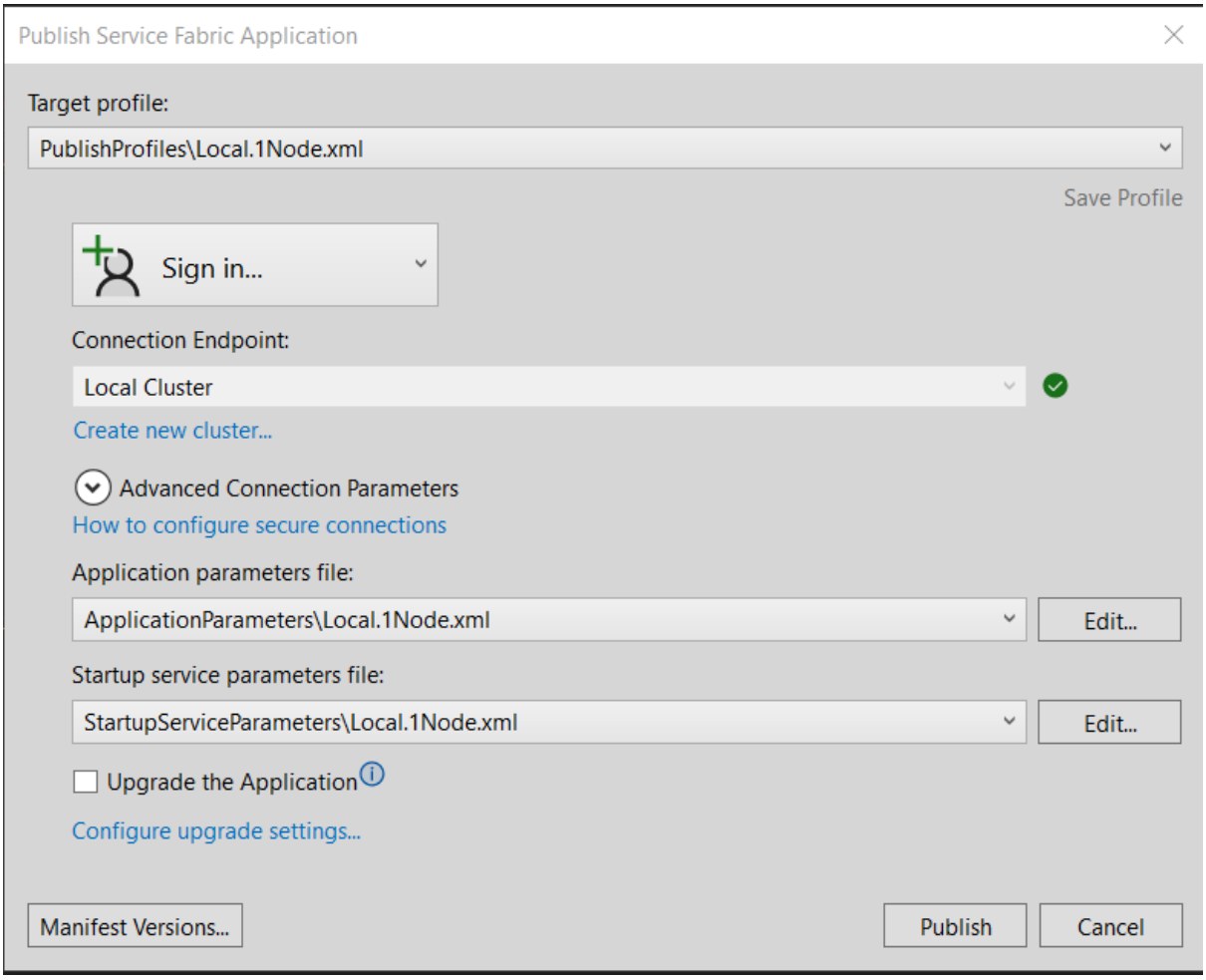
ServiceManifest.xml

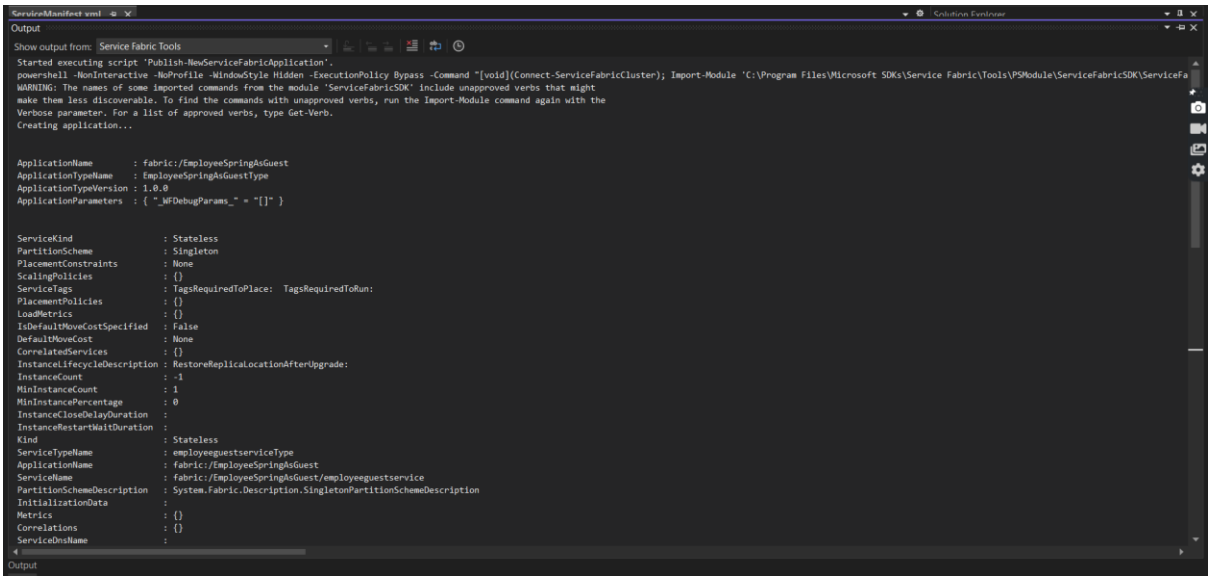
```
    <!-- The SetupEntryPoint is an optional element used to specify a
         program to be executed before the service's code is launched. -->
    <!--
    <SetupEntryPoint>
      <ExeHost>
        <Program></Program>
      </ExeHost>
    </SetupEntryPoint>
    -->
    <EntryPoint>
      <ExeHost>
        <Program>jre1.8.0_221\bin\java</Program>
        <Arguments>-jar ..\..\springboot-0.0.1-SNAPSHOT.jar</Arguments>
        <WorkingFolder>CodeBase</WorkingFolder>
        <!-- Uncomment to log console output (both stdout and stderr) to one of the
             service's working directories. -->
        <!-- <ConsoleRedirection FileRetentionCount="5" FileMaxSizeInKb="2048"/> -->
      </ExeHost>
    </EntryPoint>
  </CodePackage>
```

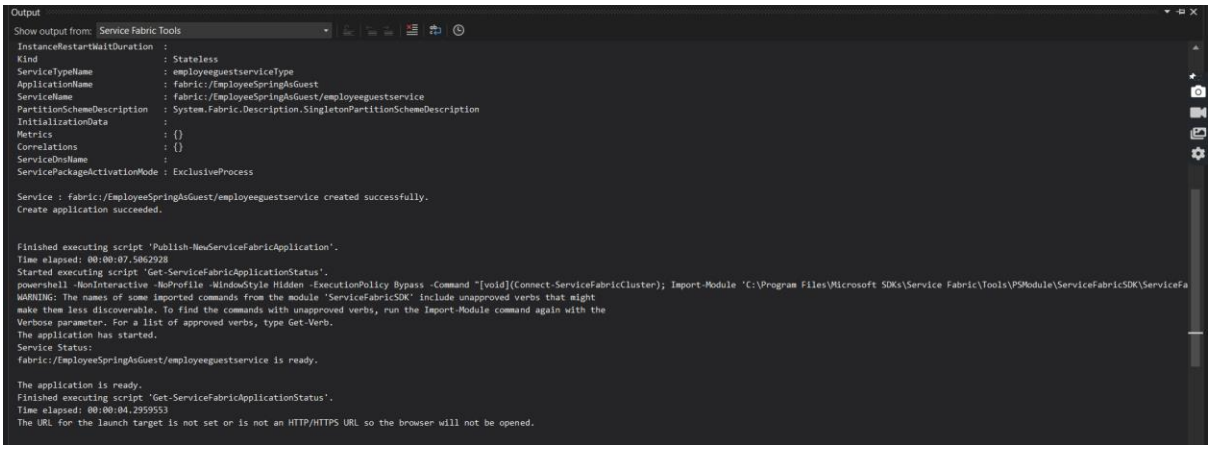
```
<Resources>
  <Endpoints>
        <!-- This endpoint is used by the communication listener to obtain the port on which to listen. Please note
        that if your service is partitioned, this port is shared with replicas of different partitions that are placed in your
        code. -->
       <Endpoint Name="employeeguestserviceTypeEndpoint" Protocol="http"  Port="8080" Type="Input" />
  </Endpoints>
</Resources>
```

```
  <Resources>
    <Endpoints>
      <!-- This endpoint is used by the communication listener to obtain the port on which to
           listen. Please note that if your service is partitioned, this port is shared with
           replicas of different partitions that are placed in your code. -->
      <Endpoint Name="employeeguestserviceTypeEndpoint" Protocol="http" Port="8080" Type="Input" />
    </Endpoints>
  </Resources>
</ServiceManifest>
```

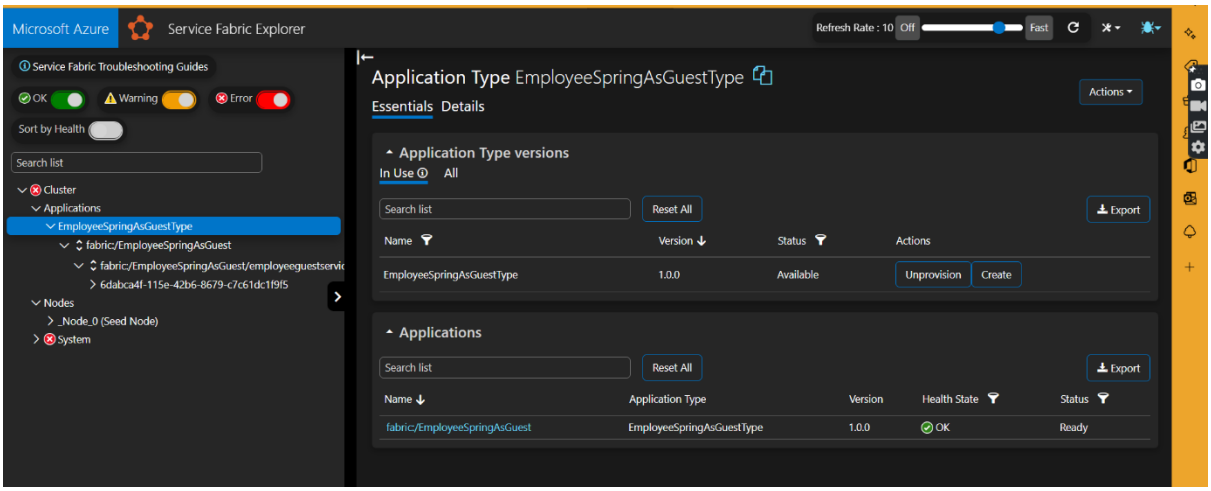
22. Now, Publish your application to Service Fabric Cluster.


Publish Service Fabric Application
Target profile:
PublishProfiles\Local.1Node.xml
Save Profile
Sign in...
Connection Endpoint:
Local Cluster
Create new cluster...
Advanced Connection Parameters
How to configure secure connections
Application parameters file:
ApplicationParameters\Local.1Node.xml
Edit...
Startup service parameters file:
StartupServiceParameters\Local.1Node.xml
Edit...
Upgrade the Application
Configure upgrade settings...
Manifest Versions...
Publish
Cancel

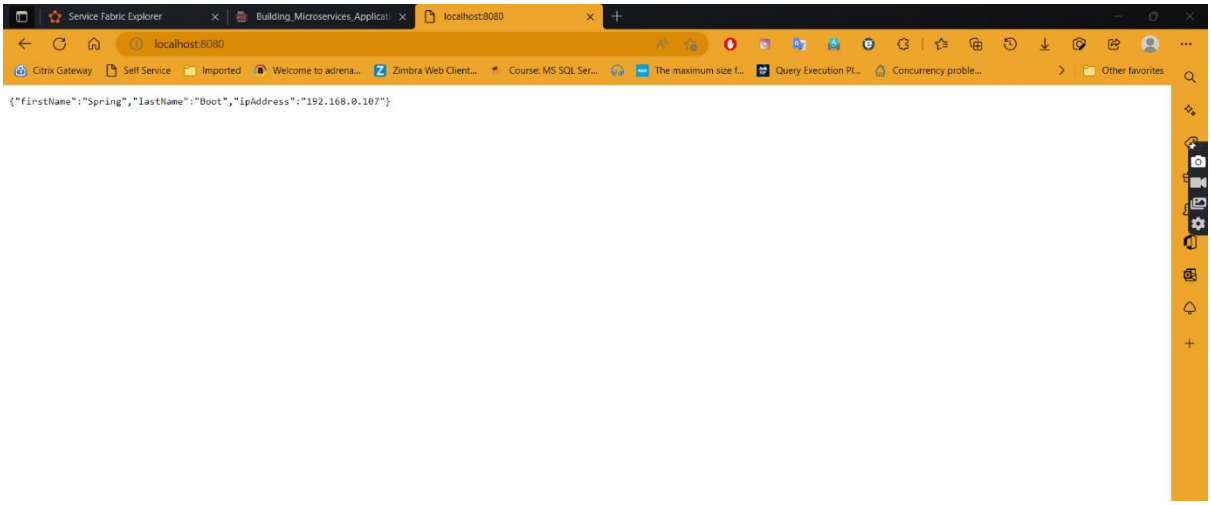

Output
Show output from: Service Fabric Tools
Started executing script 'Publish-NewServiceFabricApplication'.
powershell -NonInteractive -NoProfile -WindowStyle Hidden -ExecutionPolicy Bypass -Command "[void](Connect-ServiceFabricCluster); Import-Module 'C:\Program Files\Microsoft SDKs\Service Fabric\Tools\PSModule\ServiceFabricSDK\ServiceFa
WARNING: The names of some imported commands from the module 'ServiceFabricSDK' include unapproved verbs that might
make them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the
Verbose parameter. For a list of approved verbs, type Get-Verb.
Creating application...
ApplicationName : fabric:/EmployeeSpringAsGuest
ApplicationTypeName : EmployeeSpringAsGuestType
ApplicationTypeVersion : 1.0.0
ApplicationParameters : { "_WFDebugParams_" = "[]" }
ServiceKind : Stateless
PartitionScheme : Singleton
PlacementConstraints : None
ScalingPolicies : {}
ServiceTags : TagsRequiredToPlace: TagsRequiredToRun:
PlacementPolicies : {}
LoadMetrics : {}
IsDefaultMoveCostSpecified : False
DefaultMoveCost : None
CorrelatedServices : {}
InstanceLifecycleDescription : RestoreReplicaLocationAfterUpgrade:
InstanceCount : -1
MinInstanceCount : 1
MinInstanceCount : 0
InstanceCloseDelayDuration :
InstanceRestartWaitDuration :
Kind : Stateless
ServiceTypeName : employeeguestserviceType
ApplicationName : fabric:/EmployeeSpringAsGuest
ServiceName : fabric:/EmployeeSpringAsGuest/employeeguestservice
PartitionSchemeDescription : System.Fabric.Description.SingletonPartitionSchemeDescription
InitializationData :
Metrics : {}
Correlations : {}
ServiceDnsName :
Output



23. Make sure that the local Service Fabric cluster is up and running. Click F5. Browse the Service Fabric dashboard.



24. Browse http://localhost:8080 to access your service. In servicemanifest.xml, we specified the service port as 8080; you can browse the same on 8080

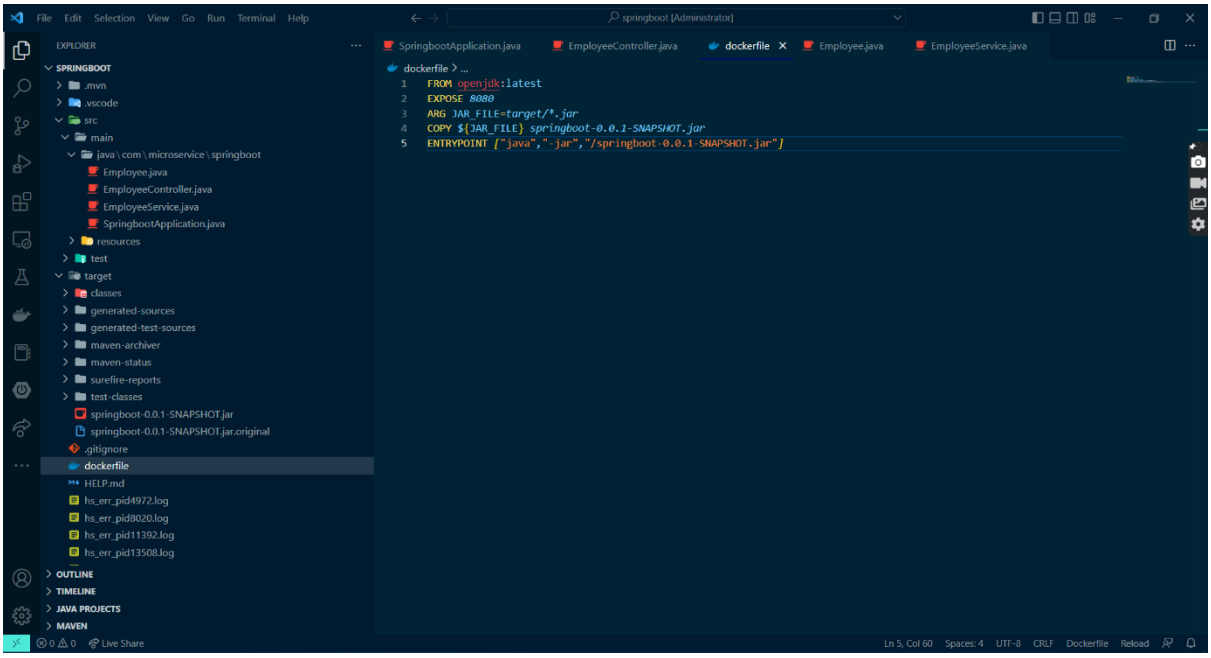


Steps :-

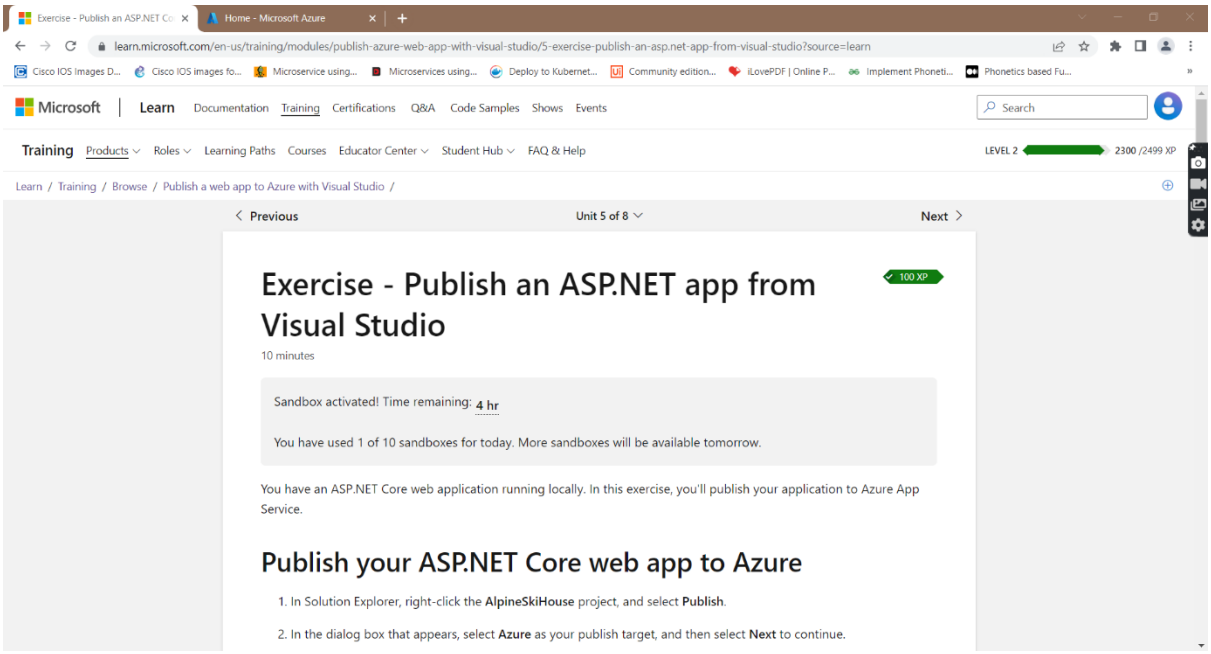
We will work on same **.jar** file that we worked in previous practical.

1. Create **Dockerfile** inside the same Project folder. This **Dockerfile** will have all the contents to build an image. Add the following code in it:

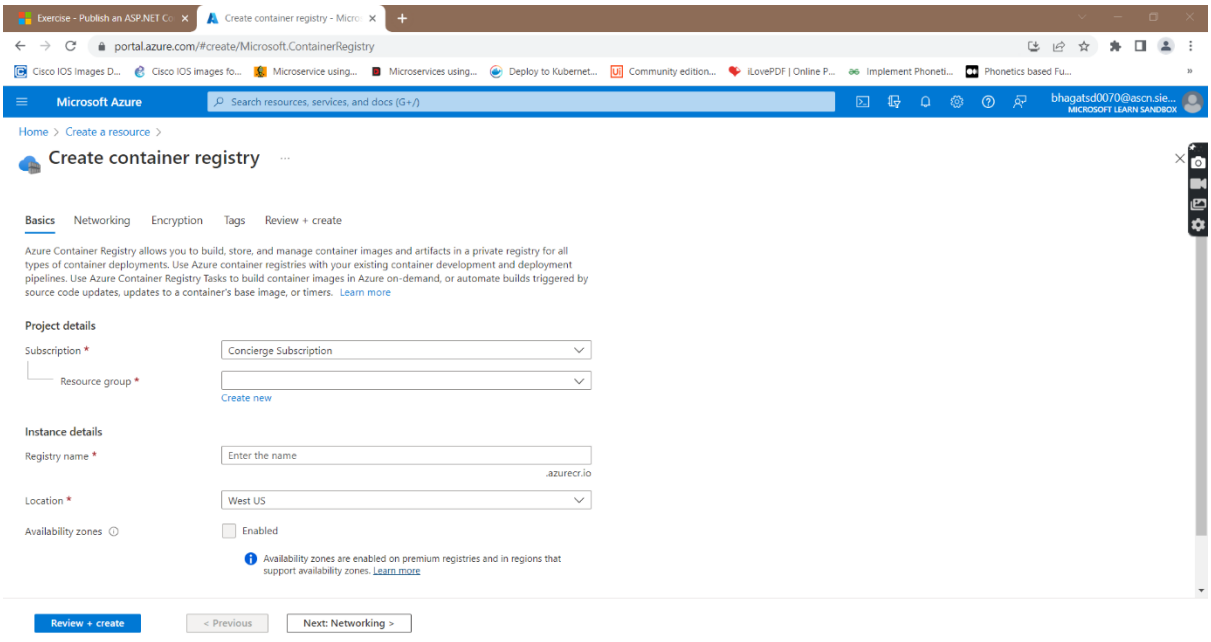
   FROM openjdk:latest
   EXPOSE 8080
   ARG JAR_FILE = target/*.jar
   COPY ${JAR_FILE} springboot-0.0.1-SNAPSHOT.jar
   ENTRYPOINT [“java”, “-jar”, “/} springboot-0.0.1-SNAPSHOT.jar”]



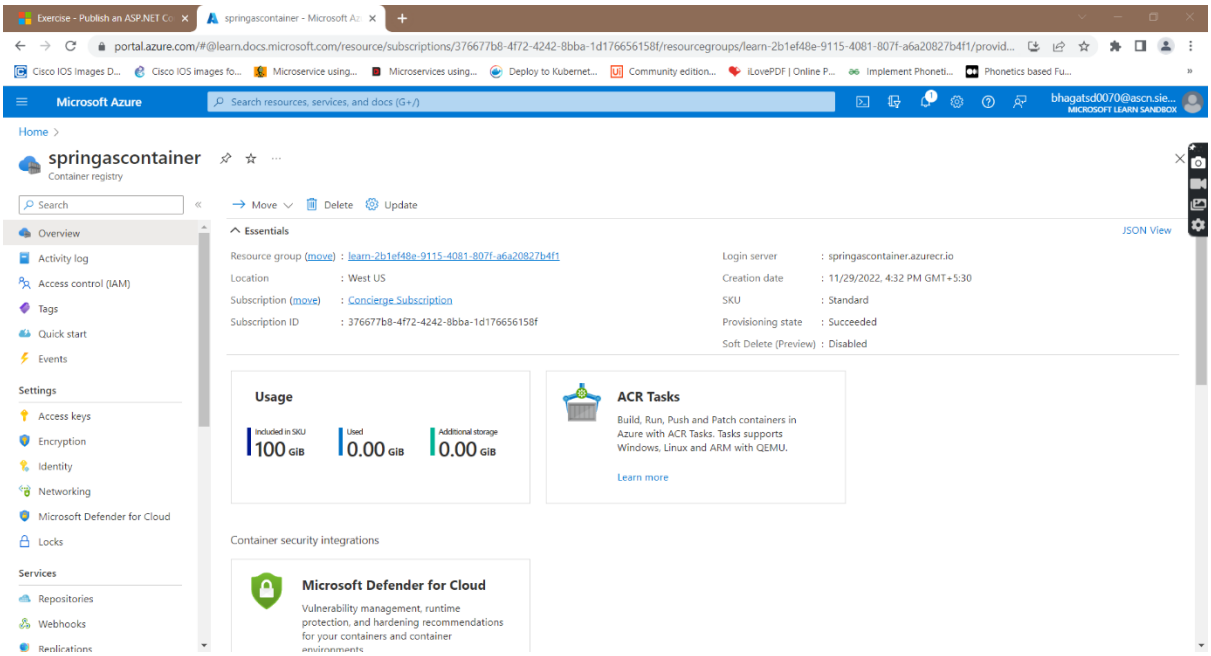
2. Go to **docs.microsoft.com** and Activate Sandbox.



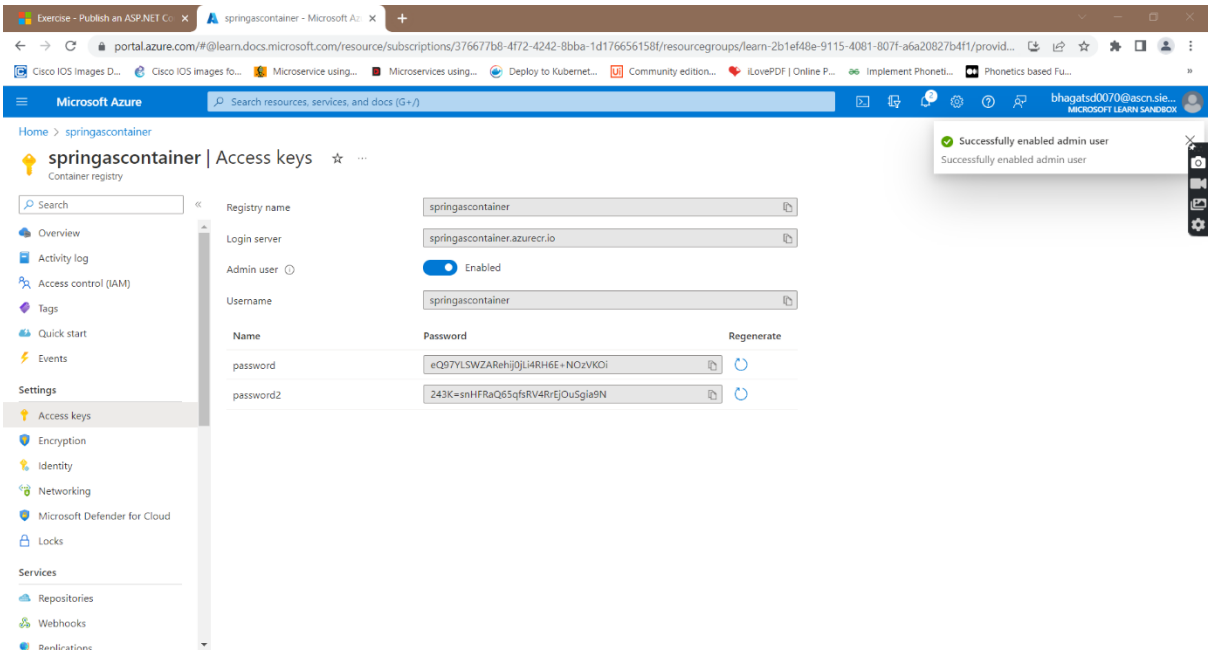
3. Go to **portal.azure.com** and Create a Resource of type **Container Registry.**



4. Go to Resources, you will find your resource of **Container Registry** has been created.

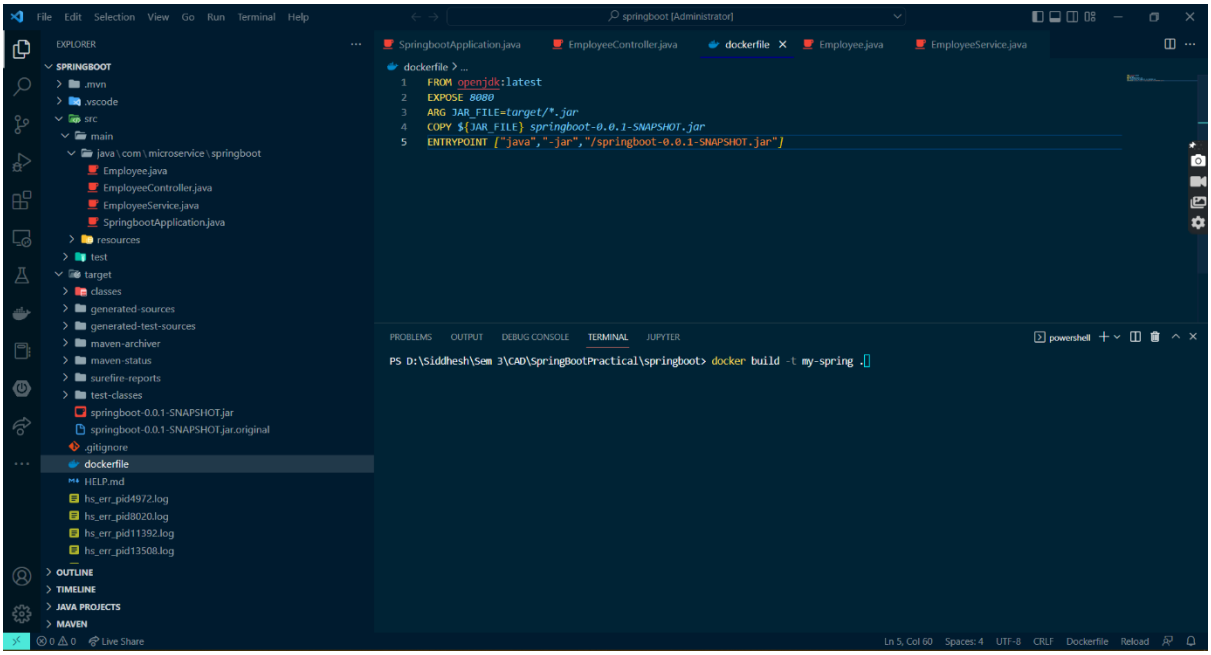


5. Now, Go to **Access keys** and Enable **Admin user.** Here, you will find **Login server, Username** and **Password** to access your **Container Registry.**

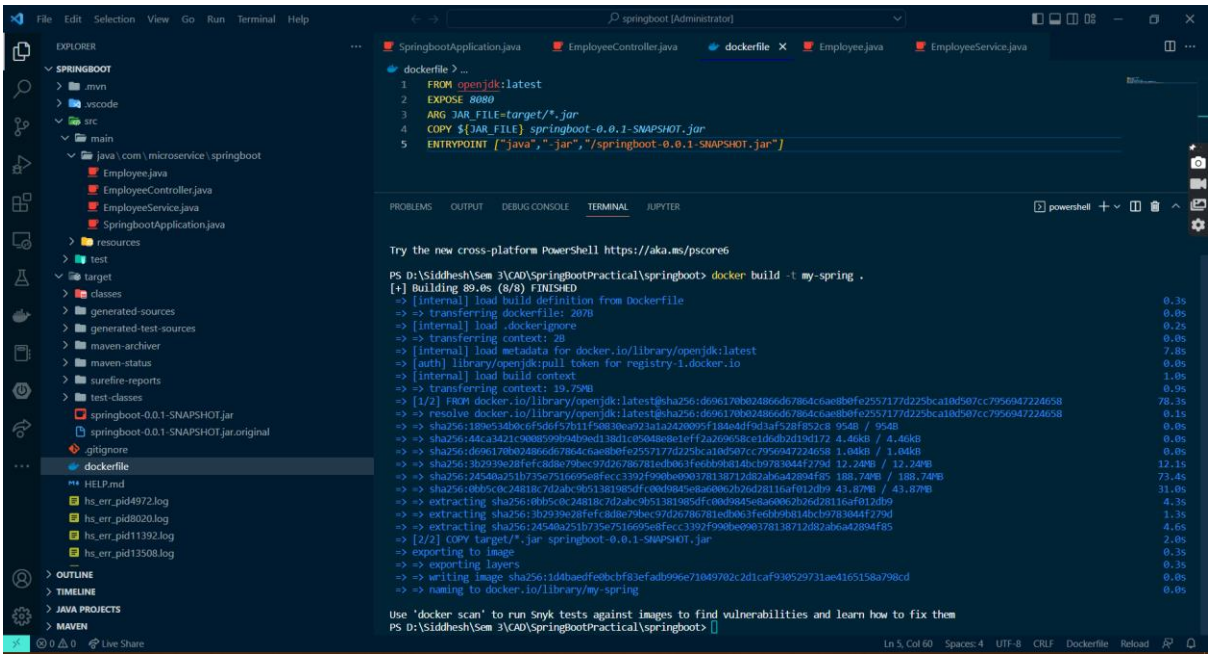


6. Now, to build an image from the **Dockerfile** we created, execute the following command in command prompt where the Dockerfile exists:

   docker build -t my-spring .



7. After clicking Enter, you will find that the image has been successfully created.



8. Now, we will **Login** into our **Container Registry Server** by executing following command:

   docker login springascontainer.azurecr.io -u springascontainer -p eQ97YLSWZARehij0jLi4RH6E+NOzVKOi

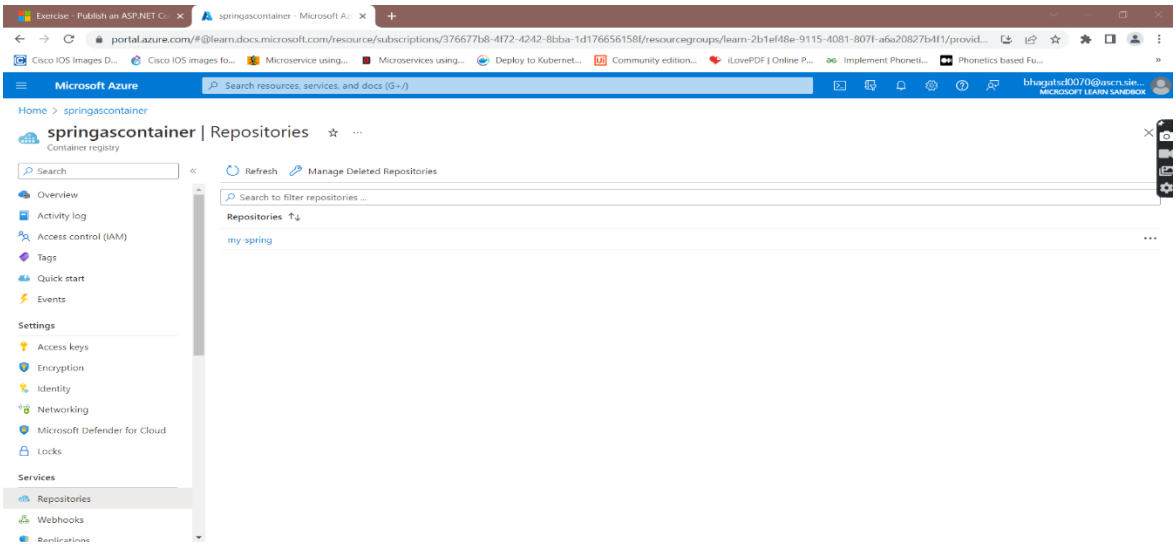
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER                    powershell

PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker login springascontainer.azurecr.io -u springascontainer -p eQ97YLSWZARehij0jLi
4RH6E+NOzVKOi
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot>
```

9. Now, we will tag our image with **Container Registry Server Name** and push it into **Azure** by executing following command:

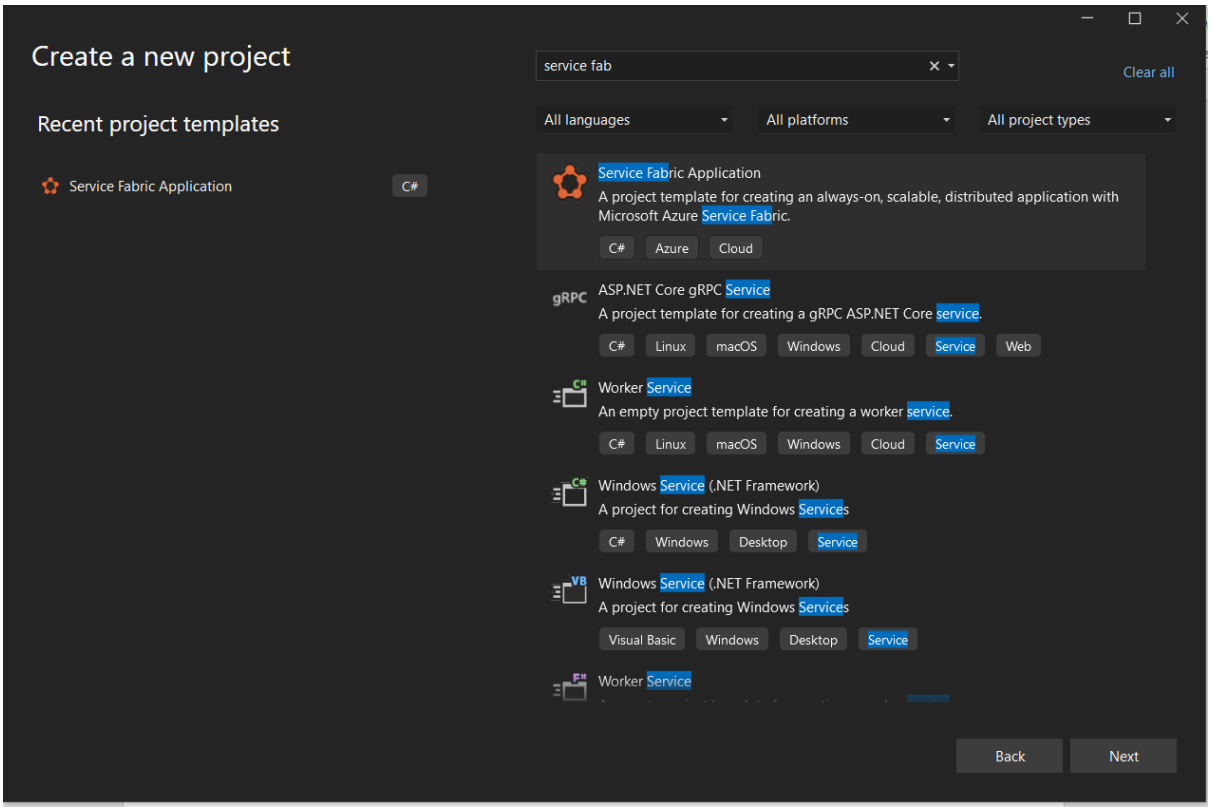
   To Tag - *docker tag my-spring springascontainer.azurecr.io/my-spring*
   To Push - *docker push springascontainer.azurecr.io/my-spring*

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER                    powershell

PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker login springascontainer.azurecr.io -u springascontainer -p eQ97YLSWZARehij0jLi
4RH6E+NOzVKOi
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker tag my-spring springascontainer.azurecr.io/my-spring
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker images
REPOSITORY                              TAG       IMAGE ID       CREATED         SIZE
my-spring                               latest    1d4baedfe0bc   6 minutes ago   486MB
springascontainer.azurecr.io/my-spring  latest    1d4baedfe0bc   6 minutes ago   486MB
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker push springascontainer.azurecr.io/my-spring
Using default tag: latest
The push refers to repository [springascontainer.azurecr.io/my-spring]
21fb78da0c37: Pushed
fbda31bd63cf: Pushed
59a7b98be63f: Pushed
4d0c6342b0f5: Pushed
latest: digest: sha256:a857a7f3d79ff34df26fac5cc40a49327b9d953242ec4579673f9945b5768e77 size: 1166
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot>
```

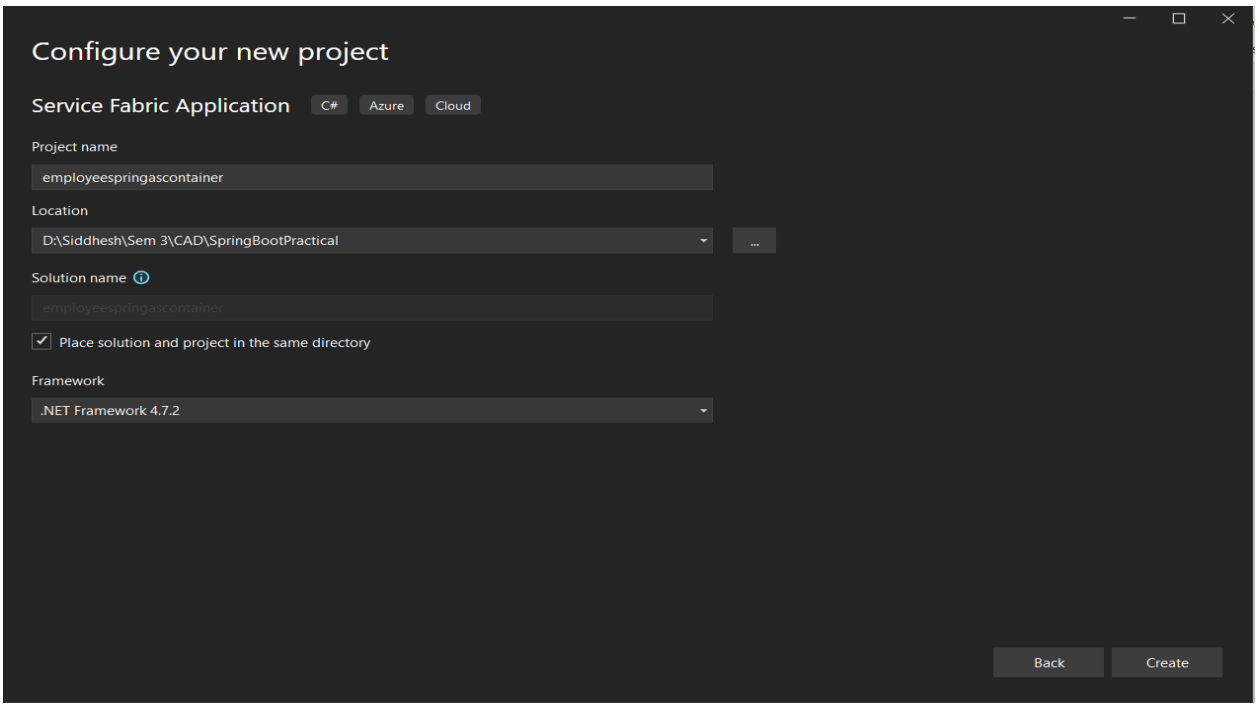
10. Inside **Azure Portal,** navigate to **Repositories** section and you will find that your image has been successfully pushed from local system to **Azure Container Registry.**



11. Since the container image is ready and uploaded in Azure Container Registry, let's create a Service Fabric project to deploy the container to the local Service Fabric cluster.



12. Give your project a meaningful name.



13. In New Service Fabric Service, select the following.

a. Service Name: **employeecontainerservice**

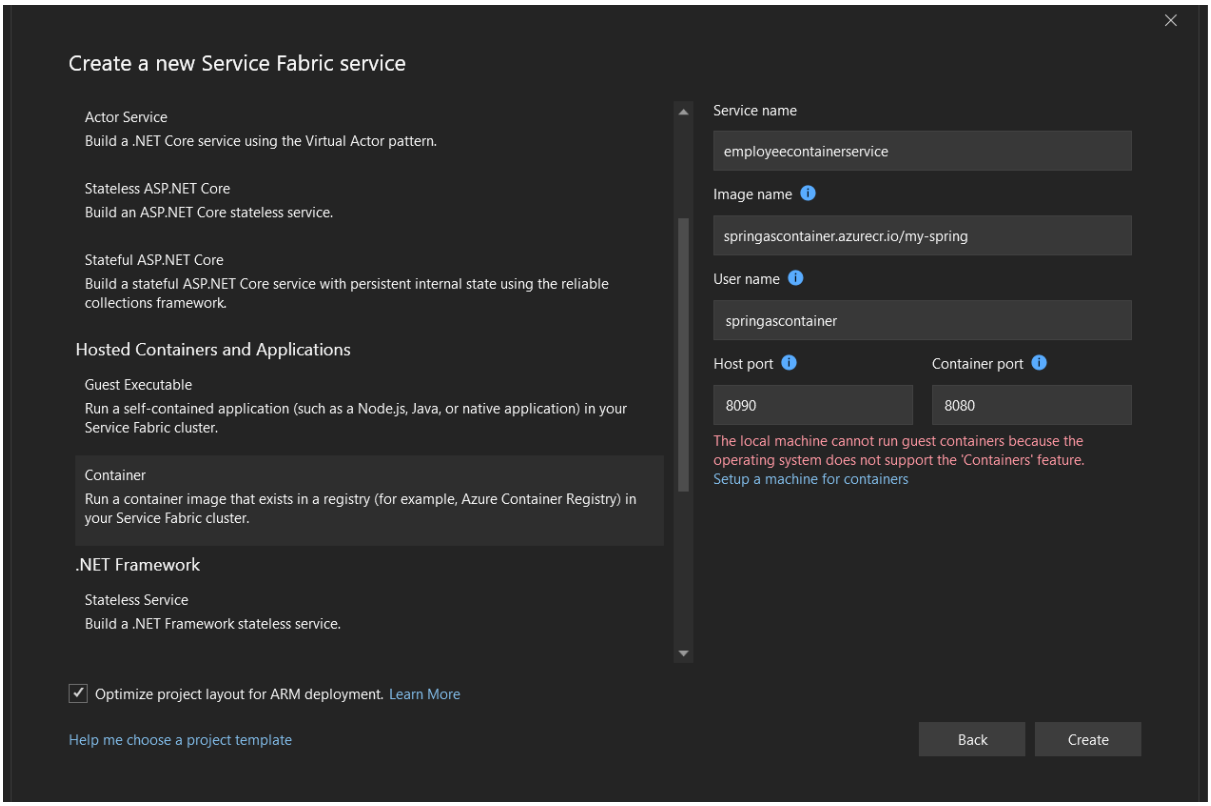
b. Image Name: **springascontainer.azurecr.io/my-spring**

c. User Name: Your username in the Azure Container Registry i.e. **springascontainer**

d. Host Port: **8090**

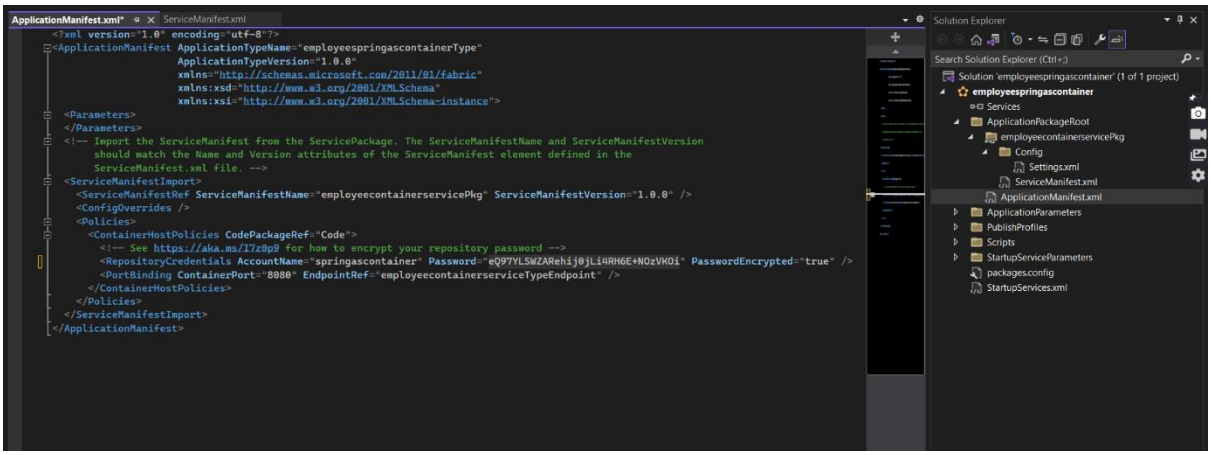
e. Container Port: **8080**

Click on Create

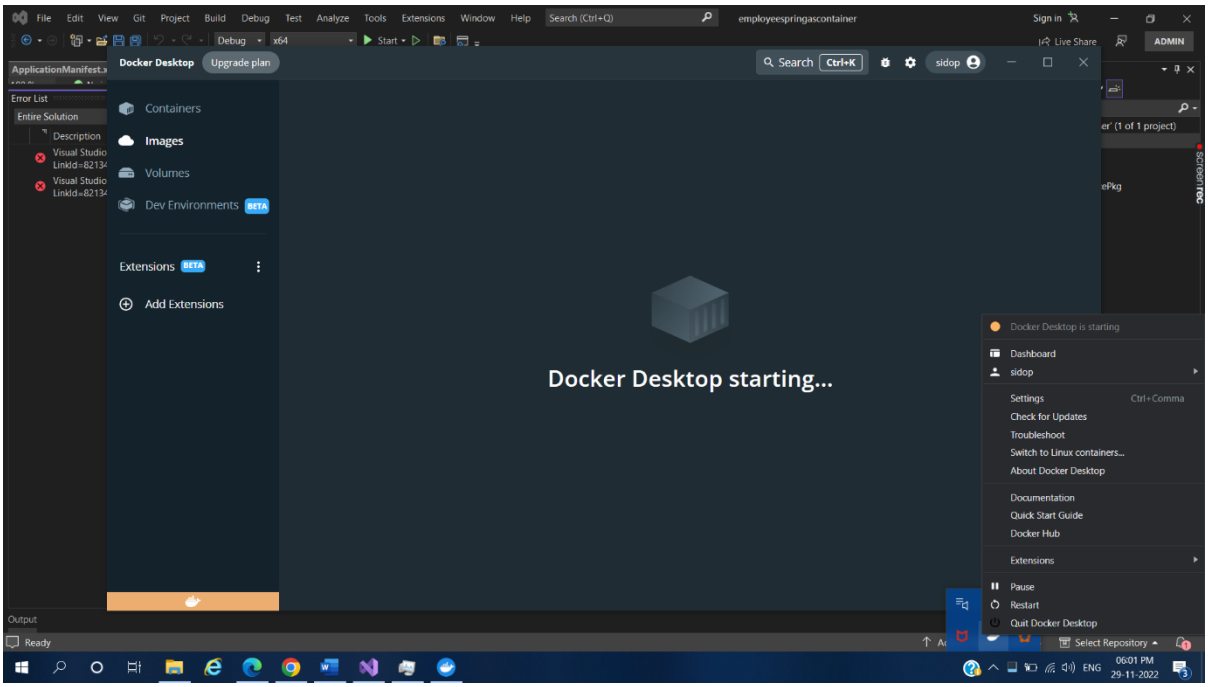


14. Once the solution is created, open the **ApplicationManifest.xml**. Specify the right **password** for the admin user:

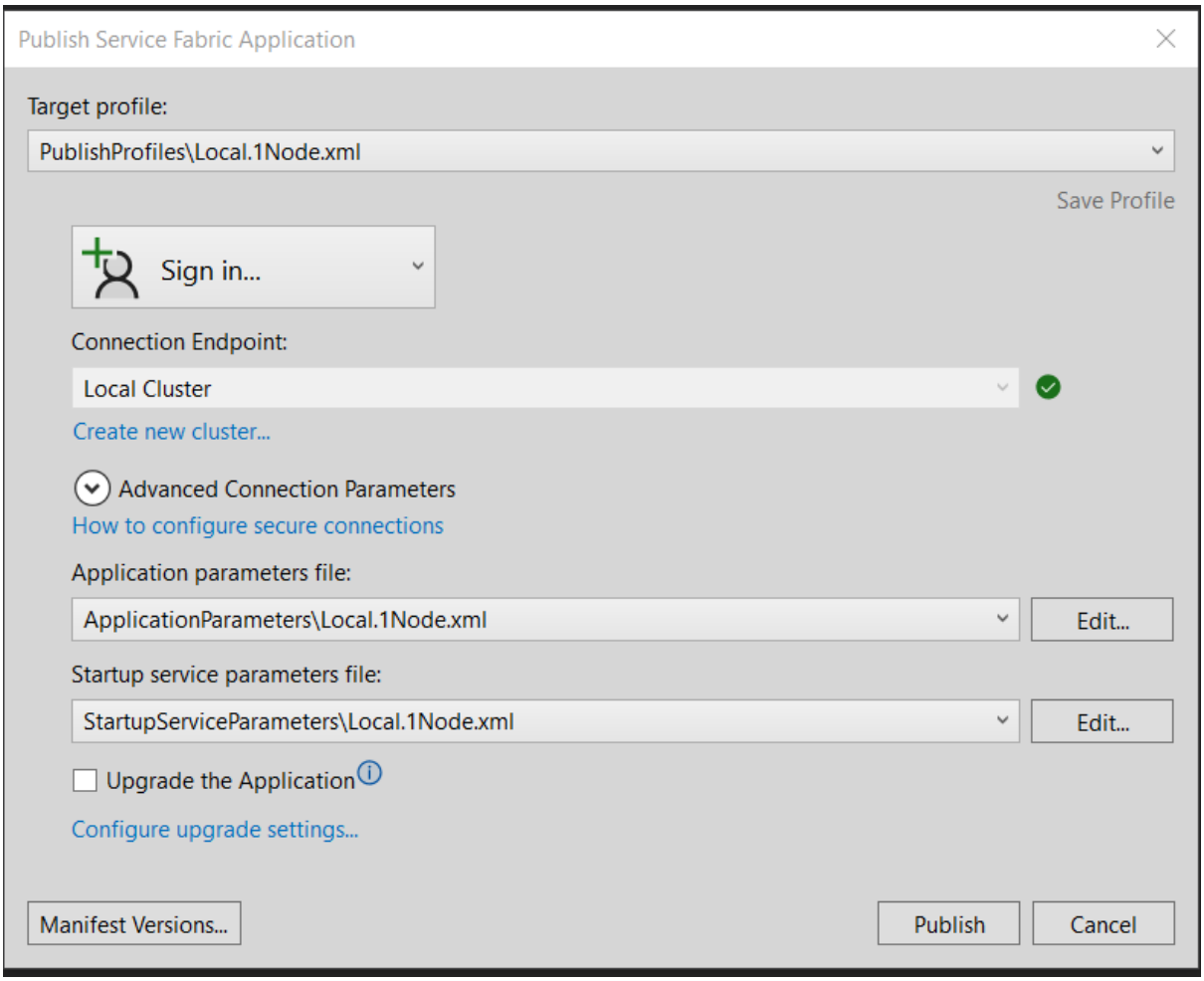
<RepositoryCredentials AccountName="springascontainer"
Password="3TWkwuCxN1aCRa6PeLpoj0pDAdsp5o5giFe1j5YNG0+ACRDoEnfM"
PasswordEncrypted="true" />

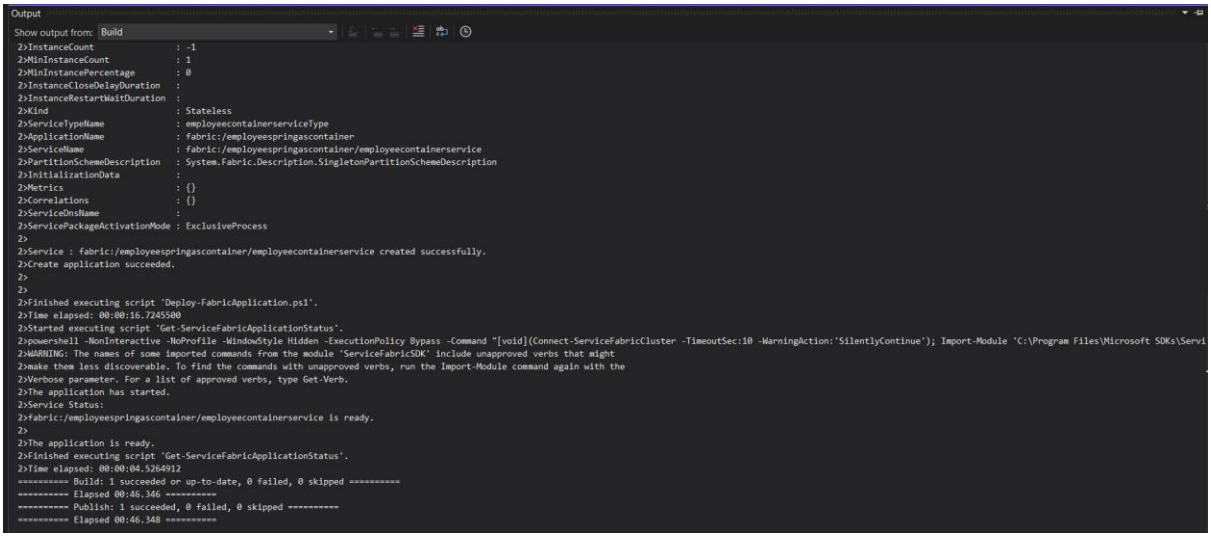


15. Select **Switch to Windows container…** in Docker Desktop.

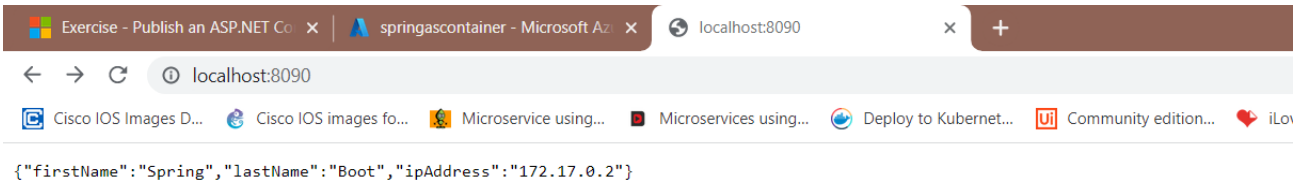


16. Now, **Publish** your project to the **Local Service Fabric Cluster.**



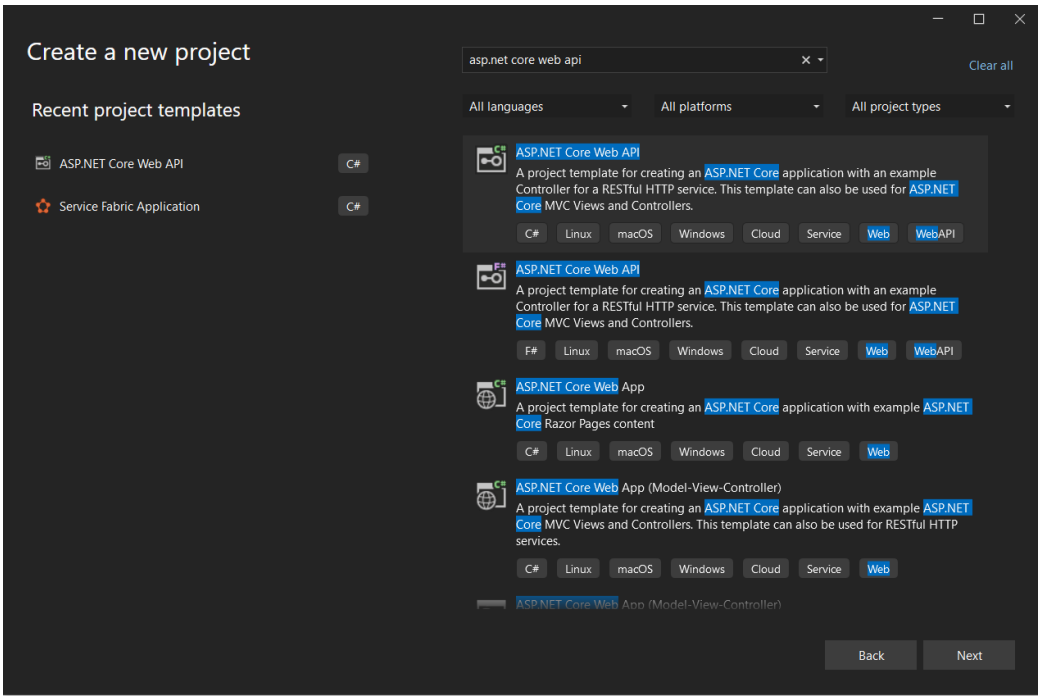


17. Browse to **http://localhost:8090/** to access your service. You get the response as shown below, which is served from the container run by Service Fabric.

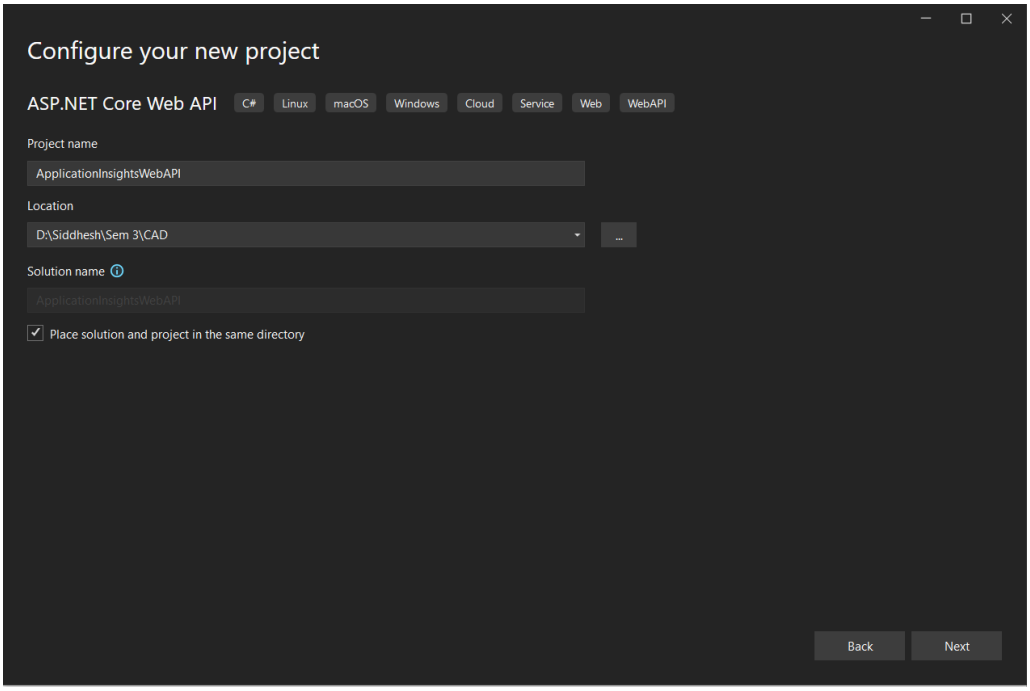


Steps :-

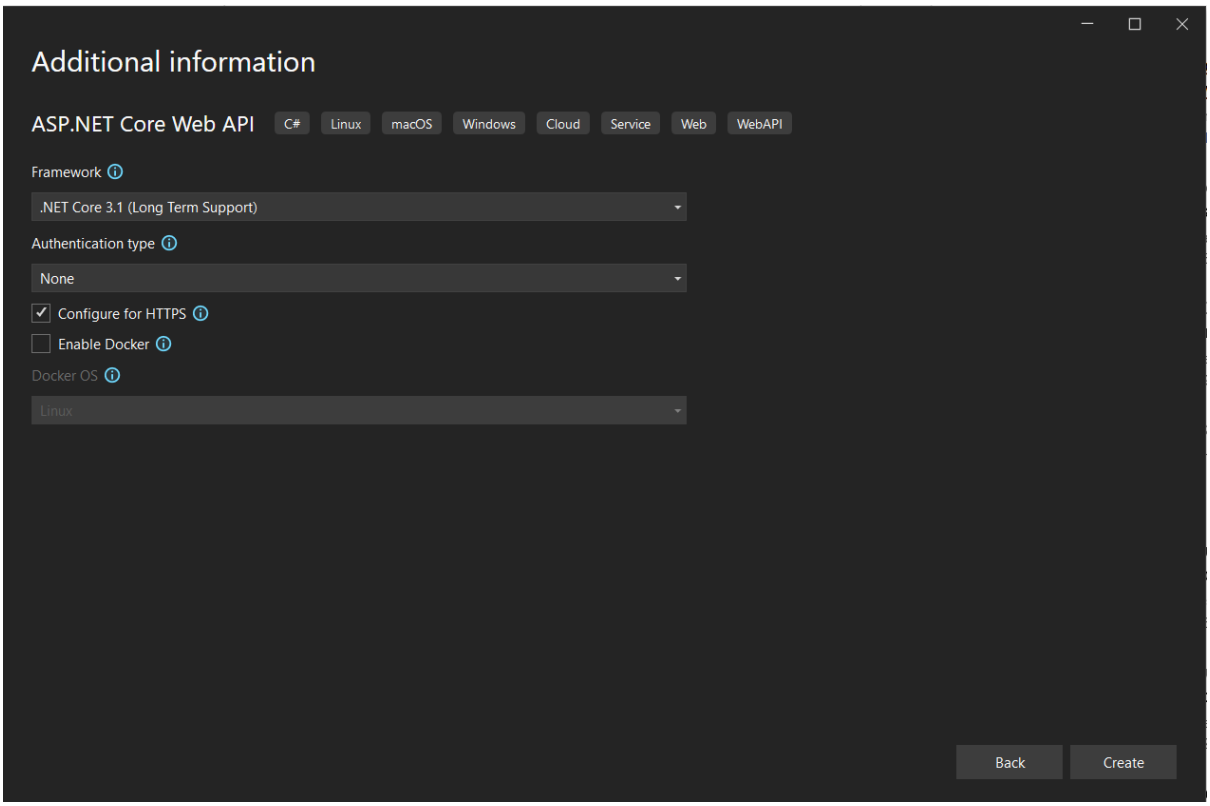
1. Open Visual Studio and Cretae a new project of the type **ASP.NET Core Web API**.



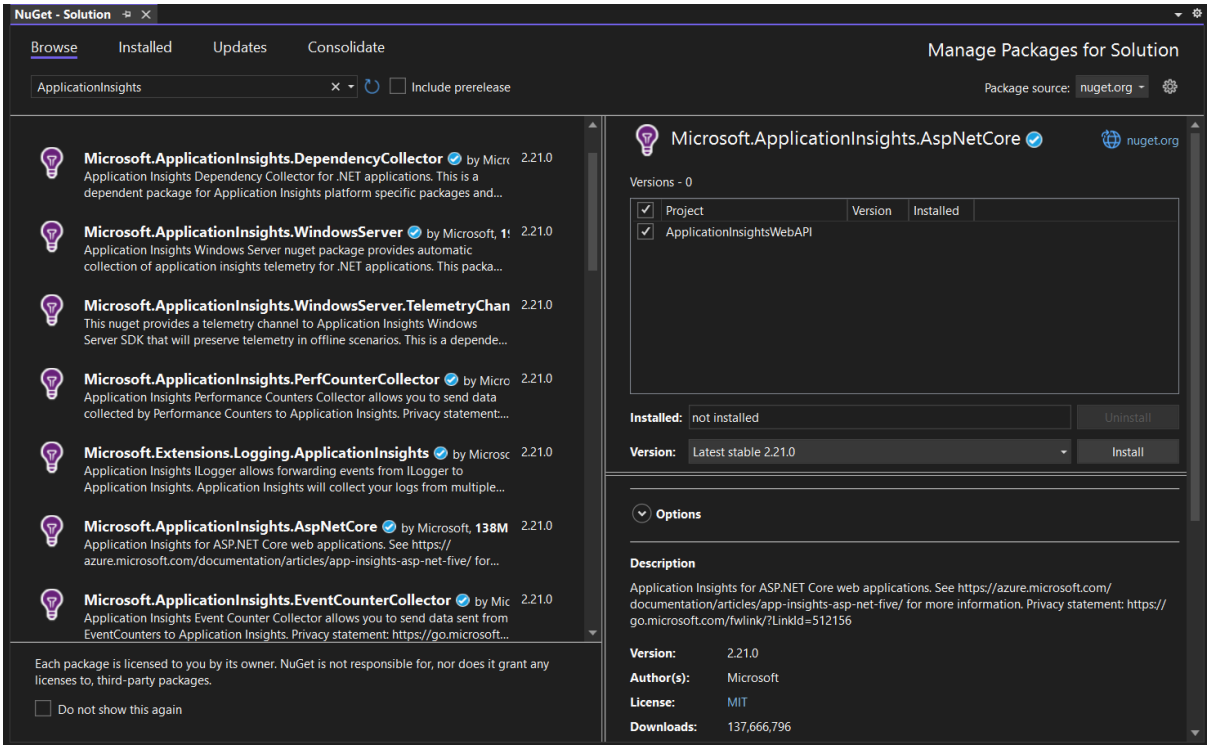
2. Give it a Meaningful name.



3. Select **.NET Core 3.1** as Framework for this project and Click on Create.

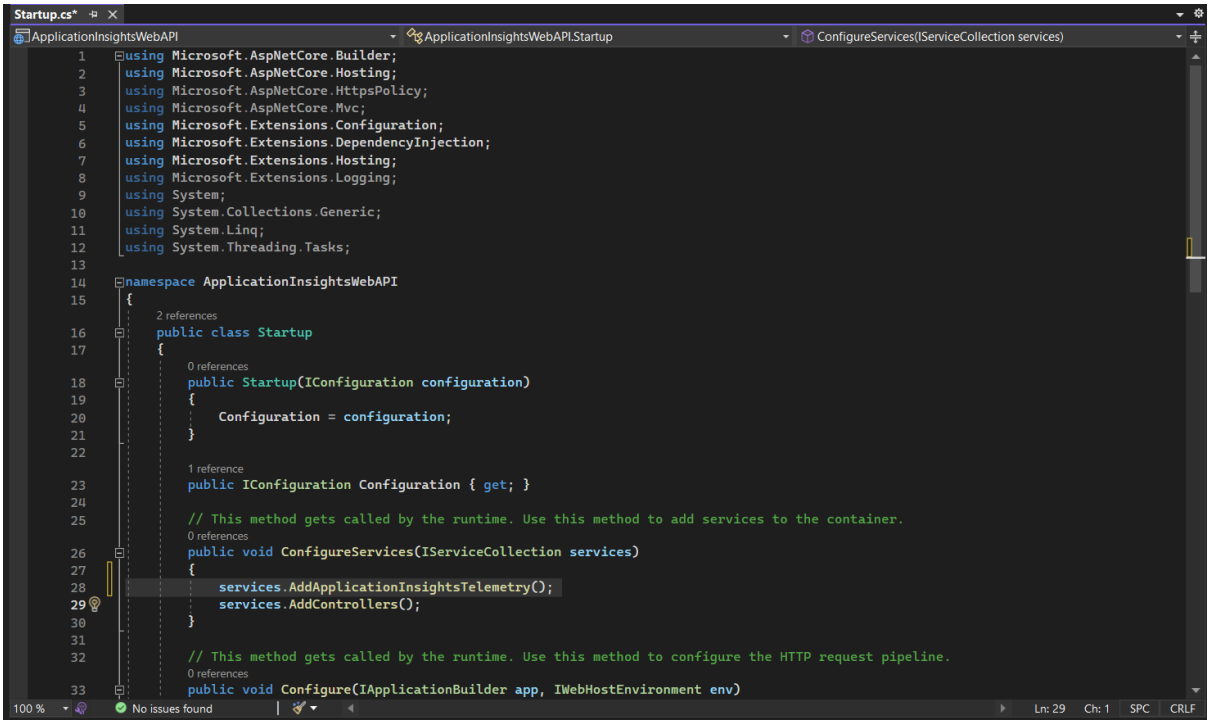


4. Go inside **Nuget Package Manager** and Install **Microsoft.ApplicationInsights.AspNetCore.**

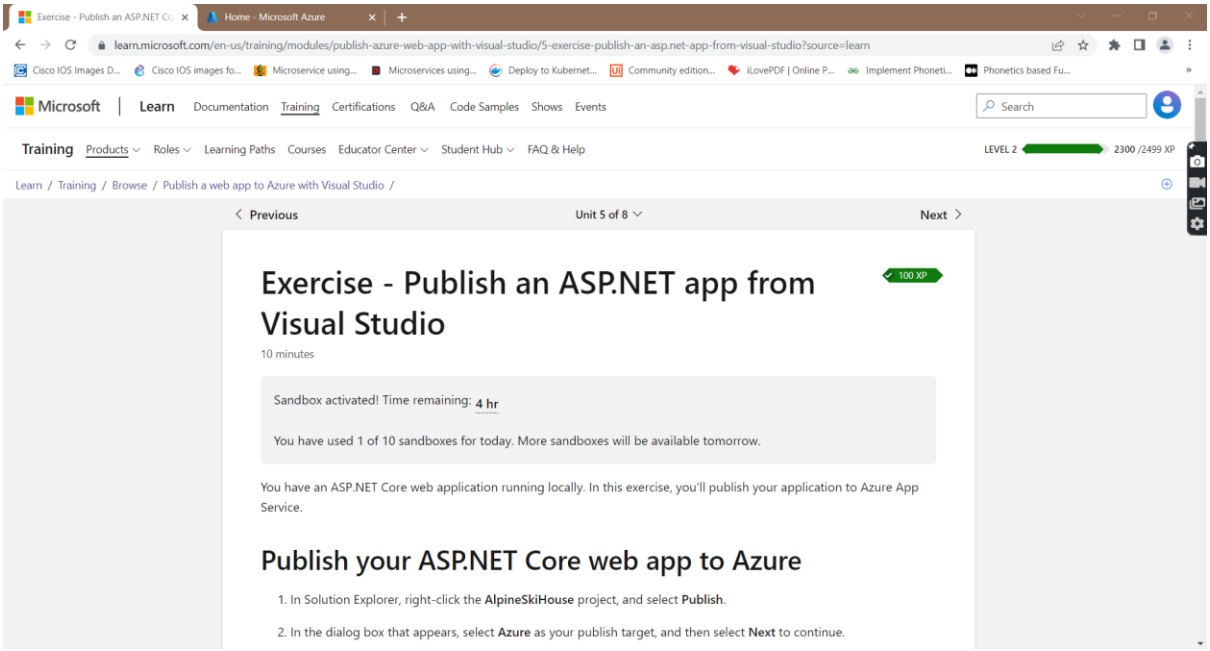


5. Inside **Startup.cs** file add the Following Code:

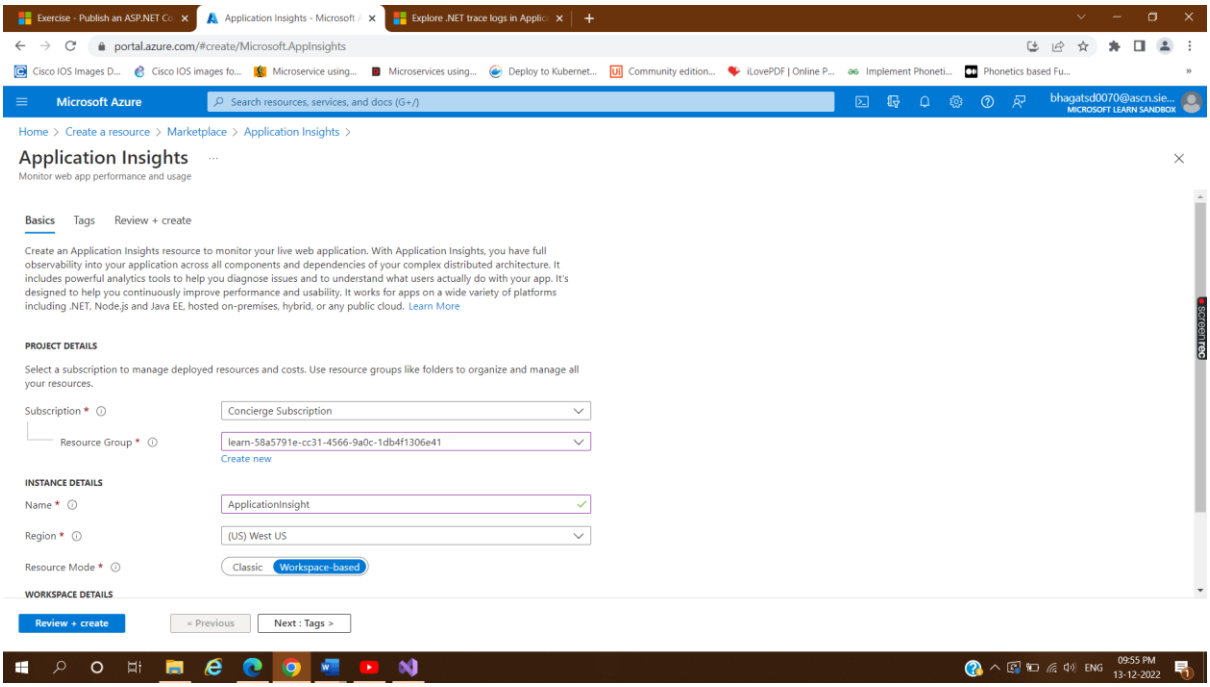
services.AddApplicationInsightsTelemetry();



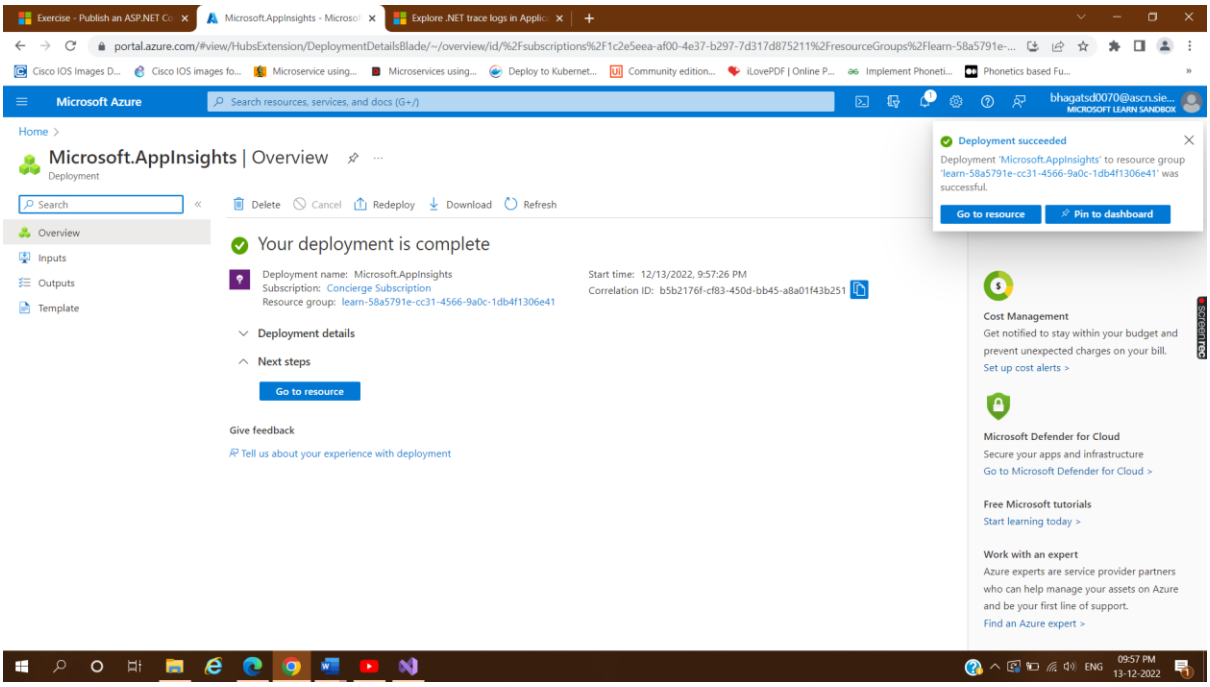
6. Go to **docs.microsoft.com** and Activate Sandbox.



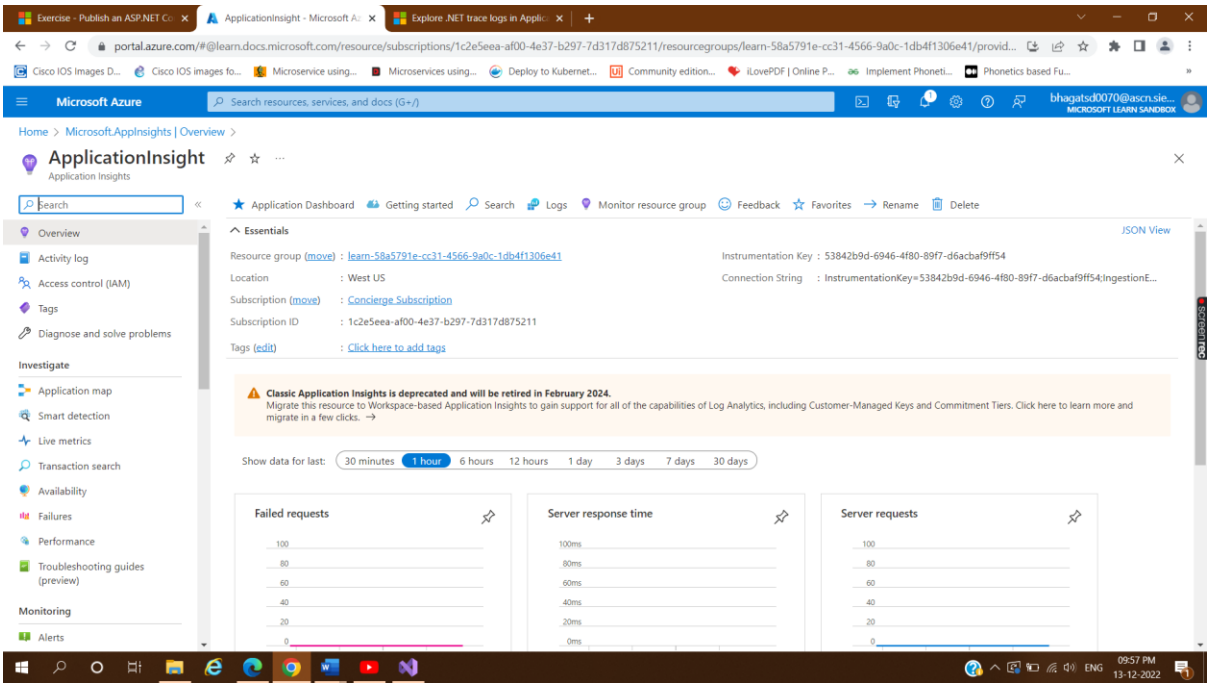
7. Go to **portal.azure.com** and Create a Resource of type **Application Insights.**



8. Click on **Go To Resource**.

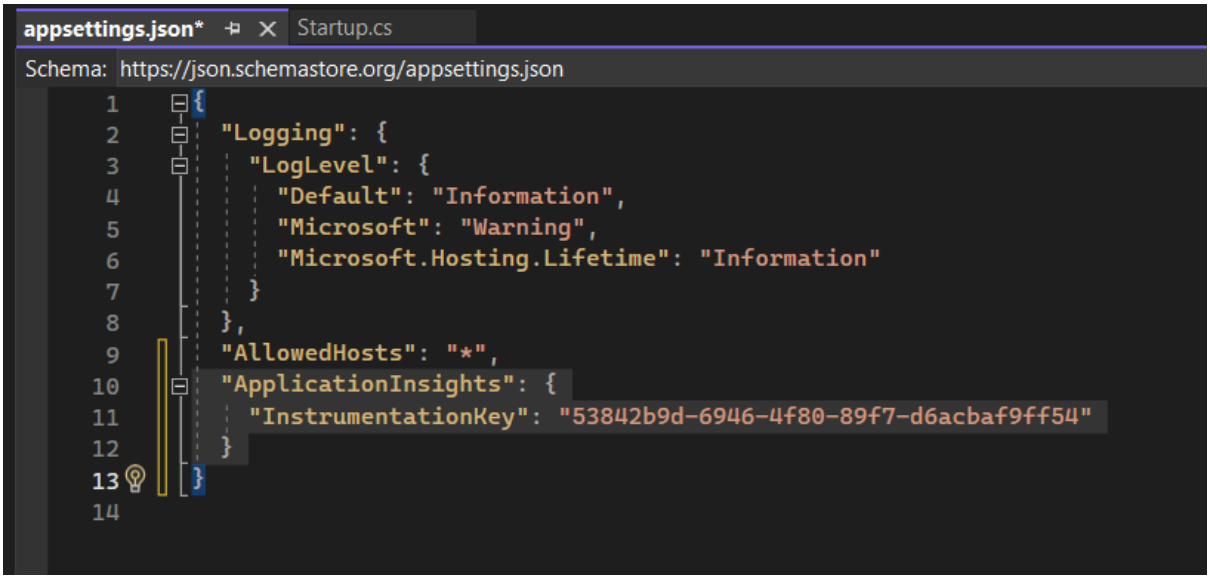


9. You will find your resource of **Application Insights** has been created.



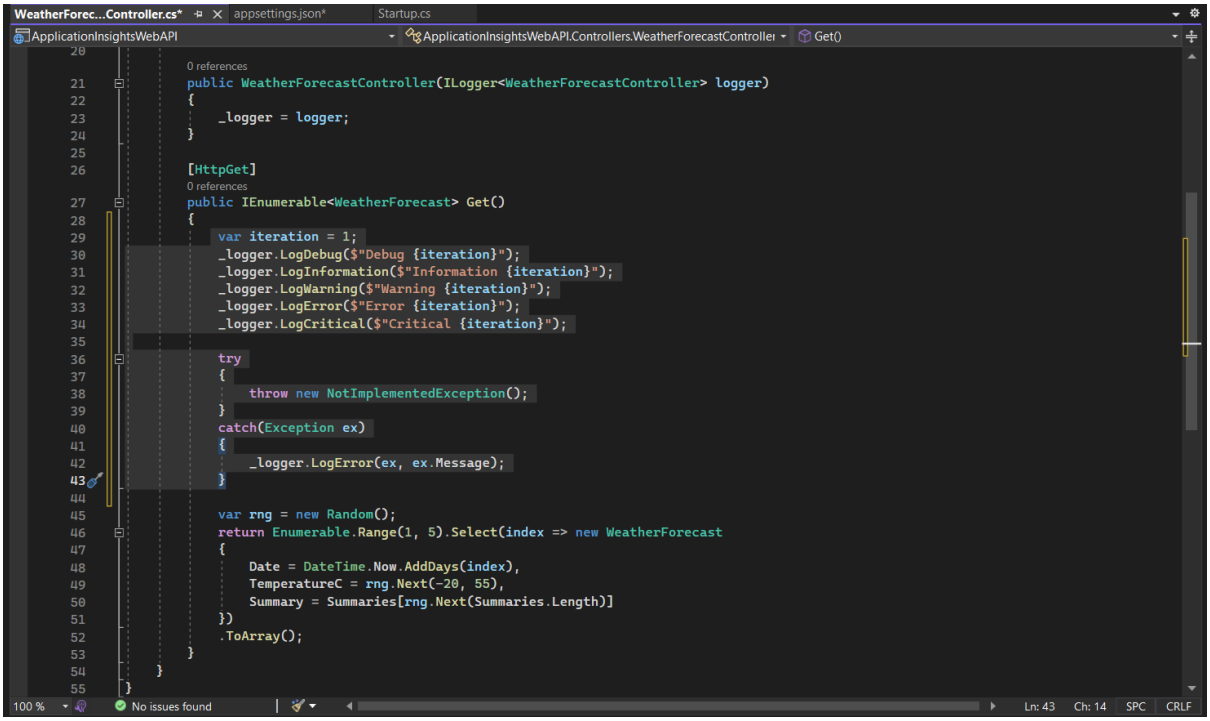
10. On the Azure portal, inside **Application Insight** resource you will find **Instrumentation Key;** copy and paste it into **appsettings.json** file as:

"ApplicationInsights": {
"Instrumentation Key" : "***Your Key***"
}

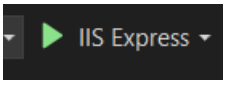


11. Inside **WeatherForecastController.cs** file add the following code:

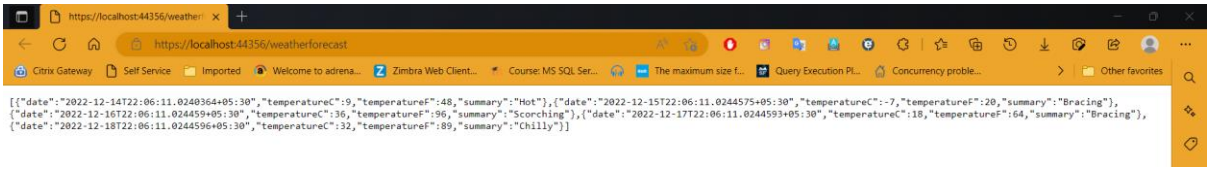
```
var iteration = 1;
_logger.LodDebug($”Debug {iteration}”);
_logger.LodInformation($”Information {iteration}”);
_logger.LodWarning($”Warning {iteration}”);
_logger.LodError($”Error {iteration}”);
_logger.LodCritical($”Critical {iteration}”);

try
{
    throw new NotImplementedException();
}
catch(Exception ex)
{
    _logger.LogError(ex, ex.Message);
}
```



12. Click on **Run**.

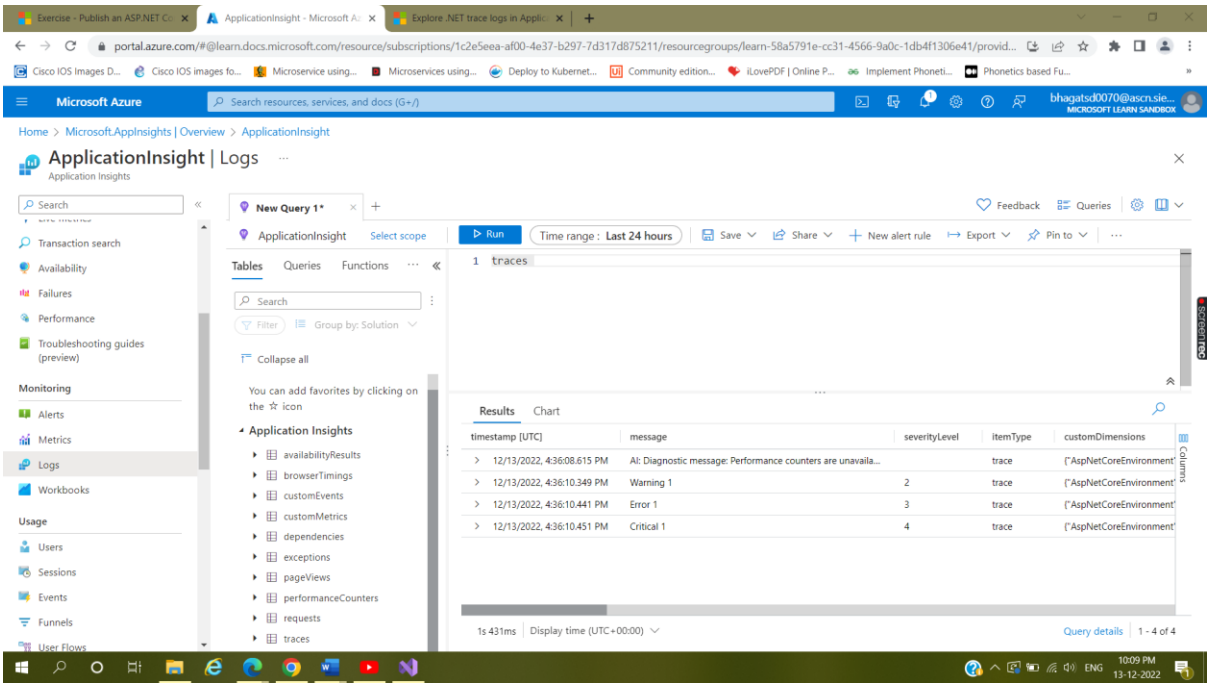


OUTPUT:



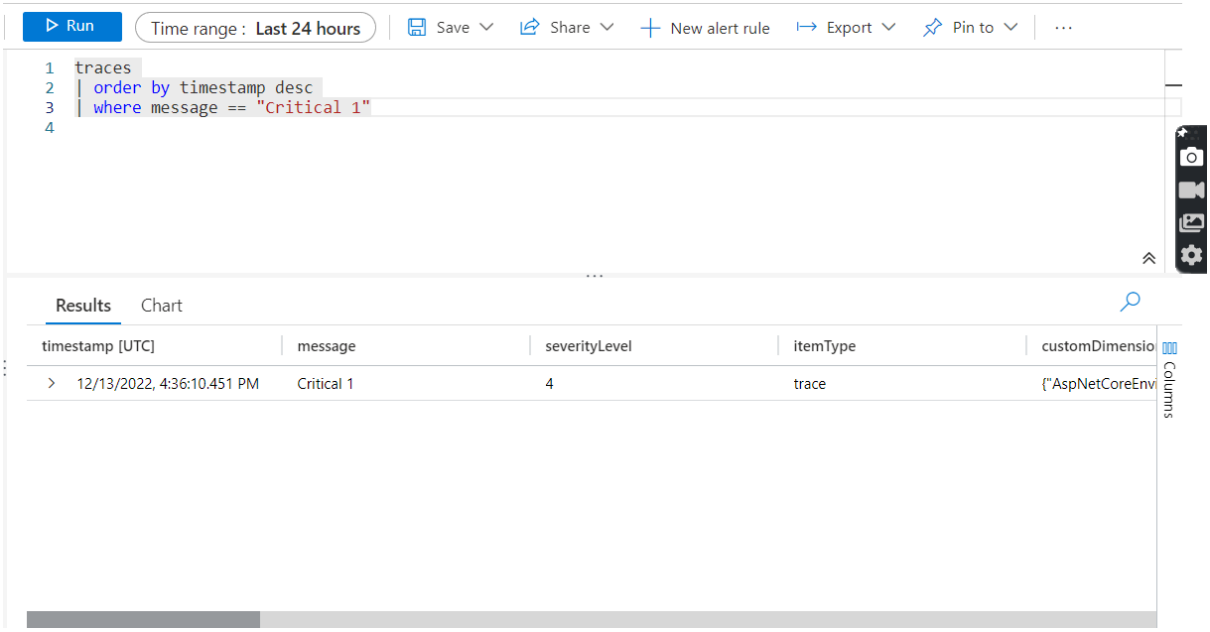
13. Inside **ApplicationInsight** resource on **Azure Portal**, inside **Monitoring > Logs > Query** section; type

traces and click on Run

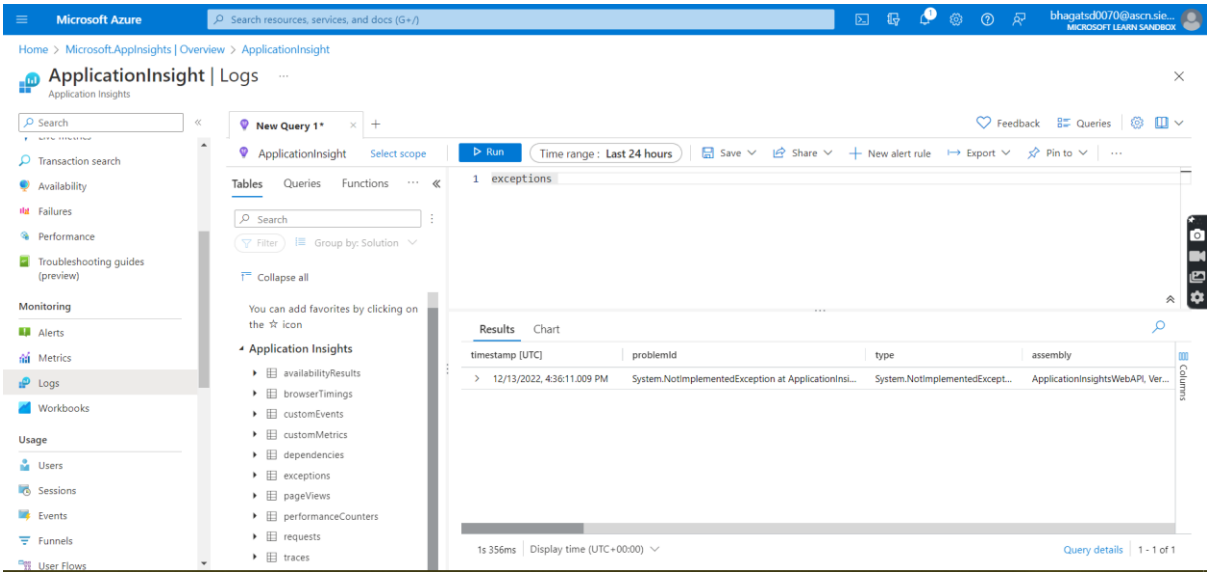


14. Type the following and click on Run:

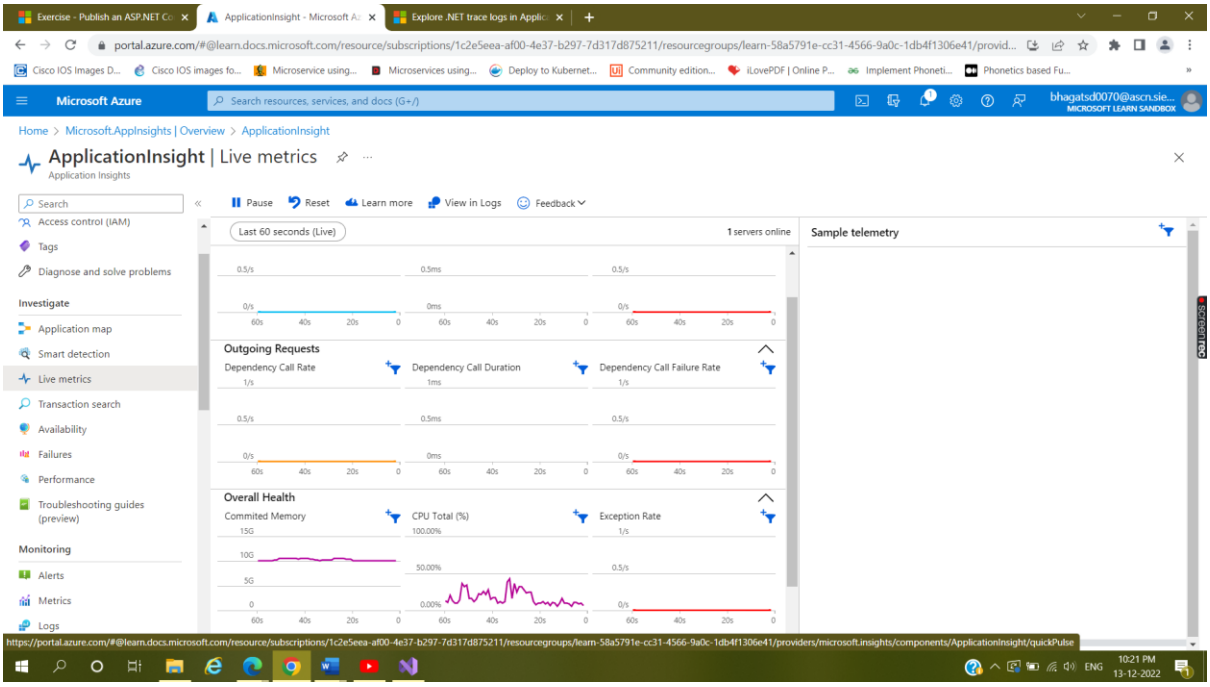
traces
| order by timestamp desc
| where message == “Critical 1”



15. Type **exceptions** and click on Run

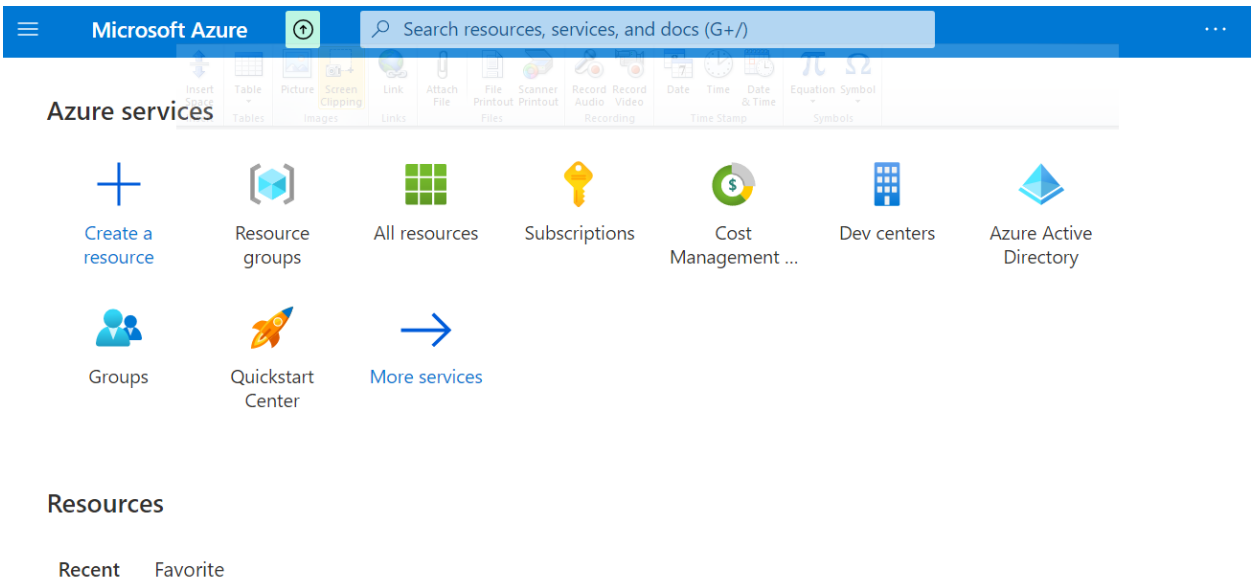


16. Go inside **Investigate > Live Metrics** section.

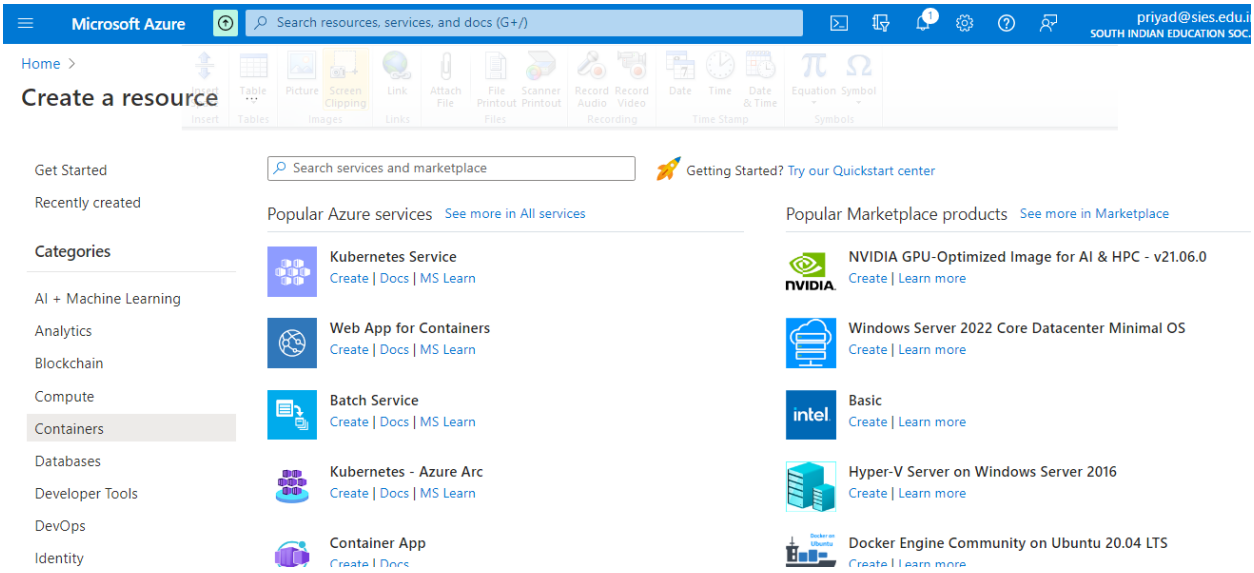


Steps :-

1. Login to Azure portal and click on **“Create a resource**”.

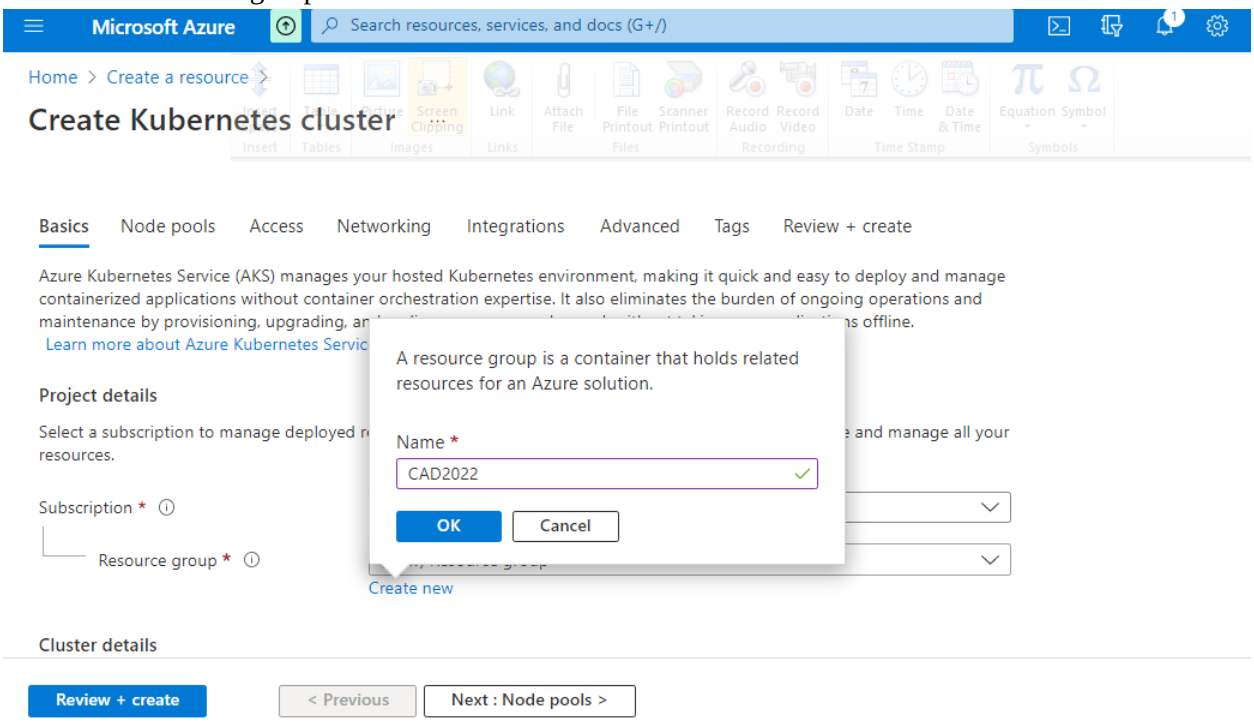


2. Click on “**Containers**” tab and select **Kubernetes Services** and click **create.**

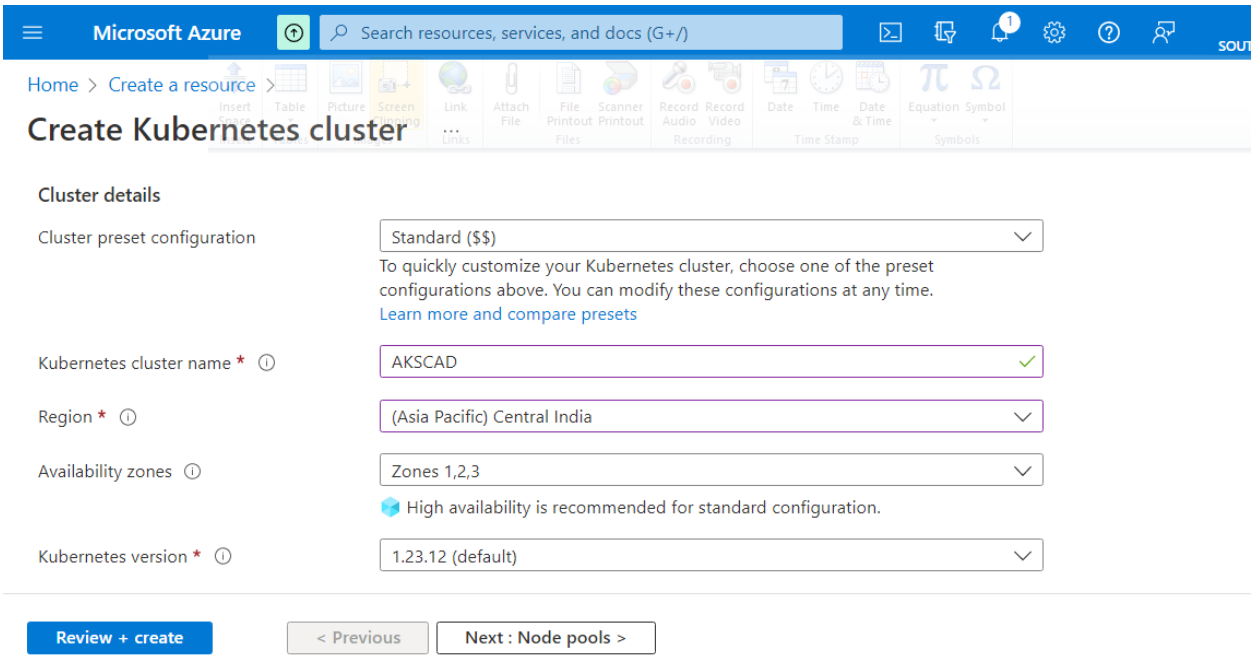


3. Fill all the details as follows:

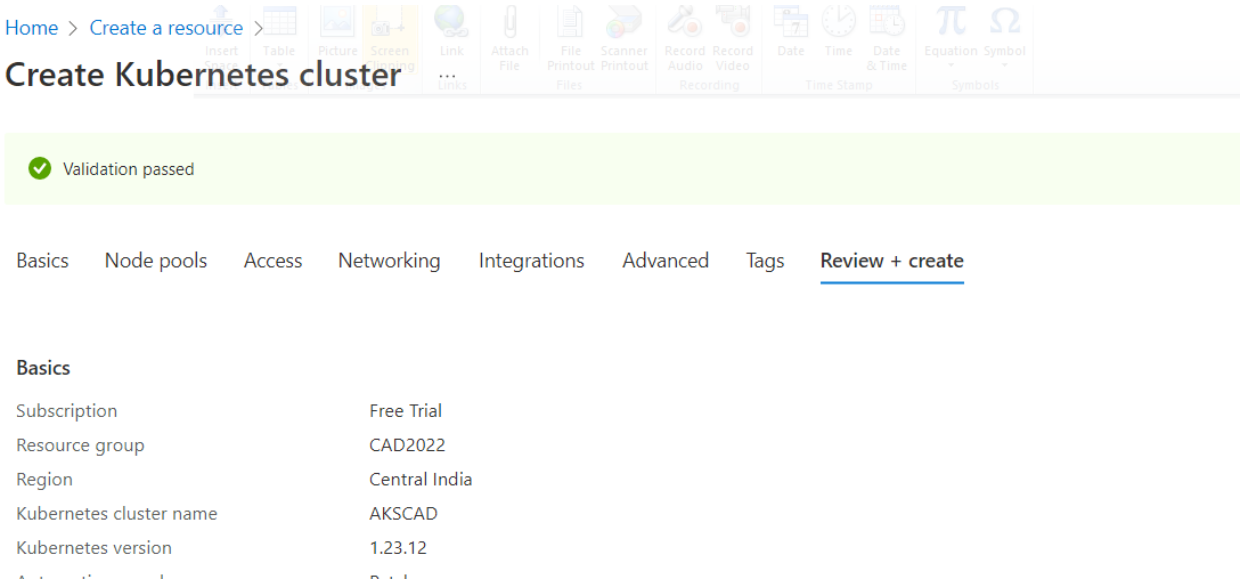
Create new resource group.



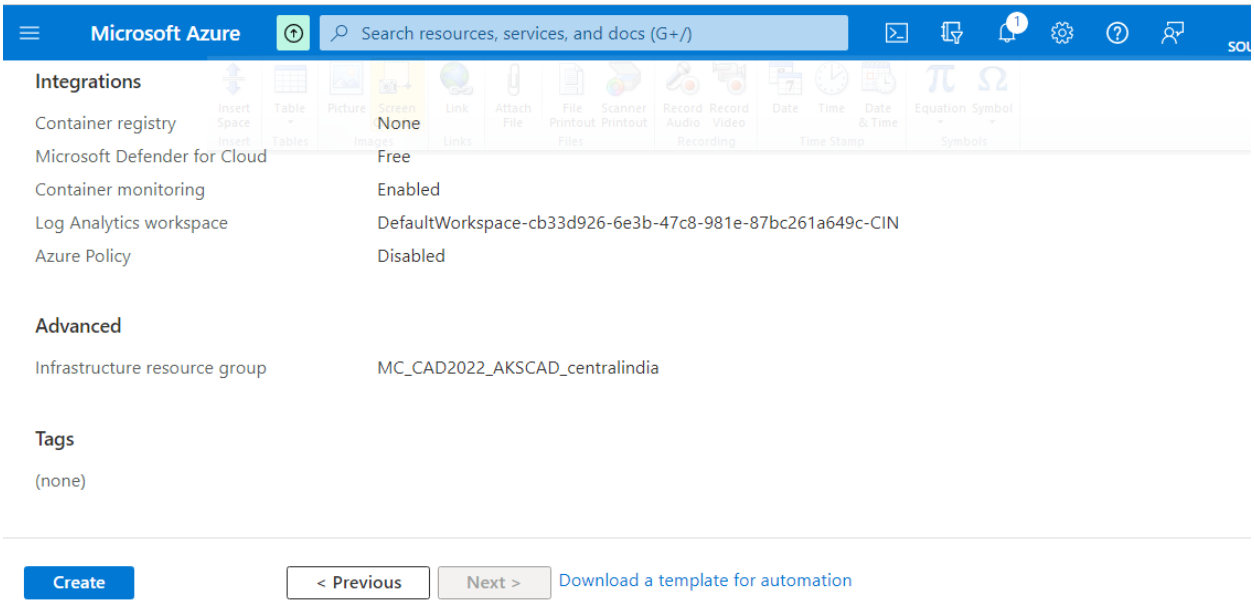
4. Name the cluster, select the region. Keep remaining details as default.Click **review+create**



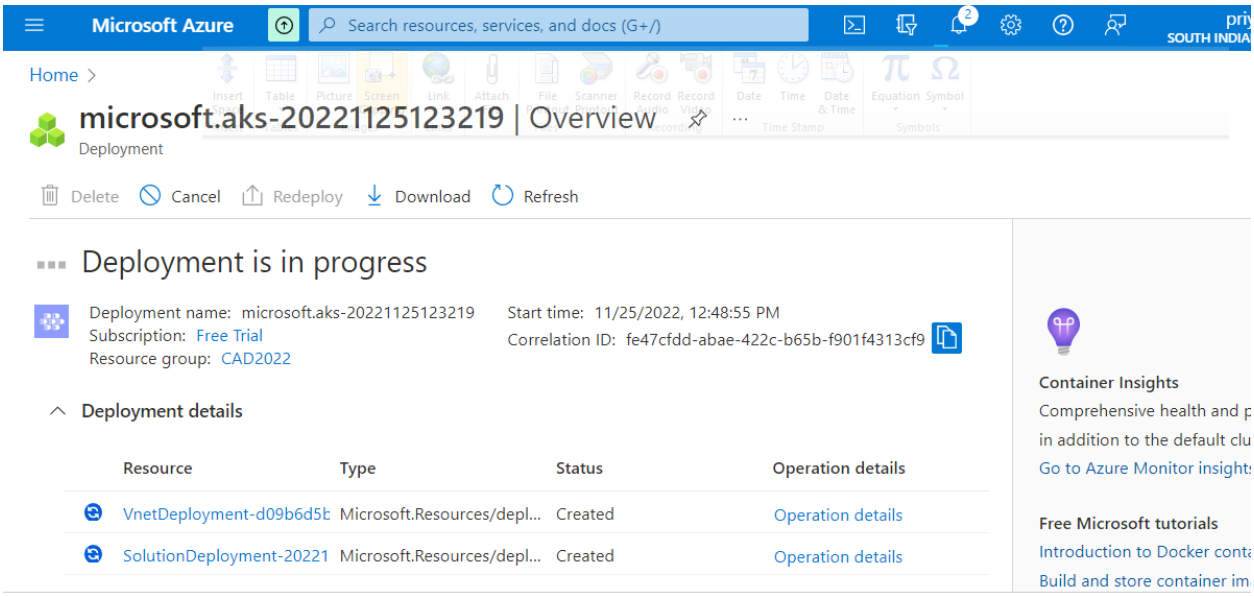
5. **Validation passed** will be displayed.

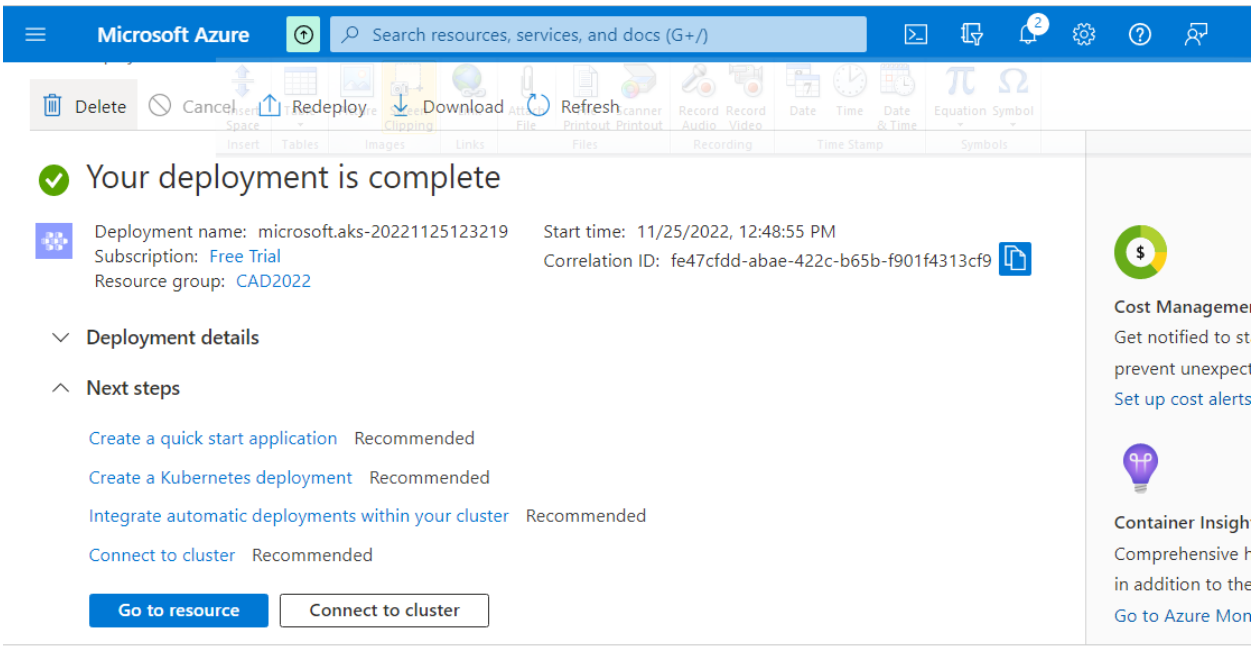


6. Click **Create**

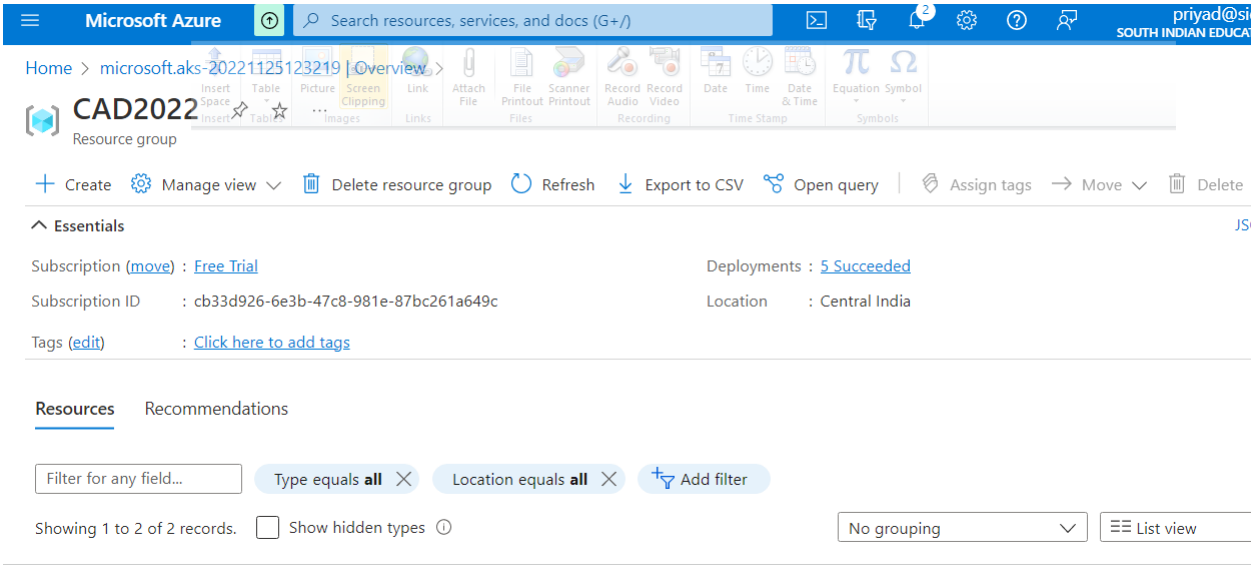


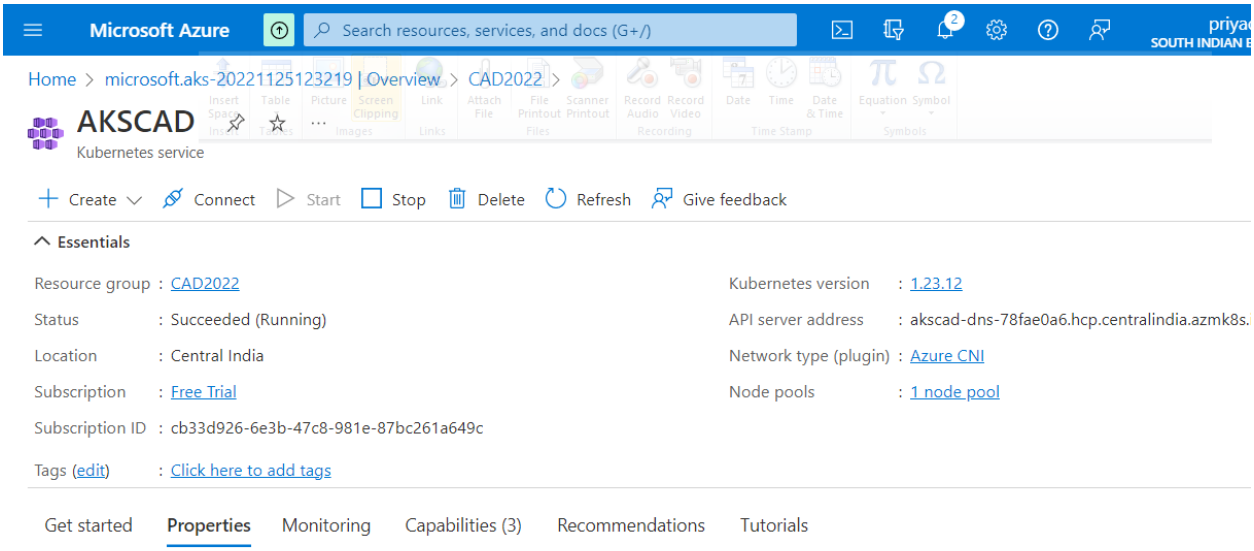
7. The cluster will proceed for deployment.





8. You can check the details of your resource and Kubernetes Cluster by clicking on them.





Get started | Properties | Monitoring | Capabilities (3) | Recommendations | Tutorials

Kubernetes services

Encryption type	Encryption at-rest with a platform-managed key
Virtual node pools	Enabled

Node pools

Node pools	1 node pool
Kubernetes versions	1.23.12
Node sizes	Standard_DS2_v2

Configuration

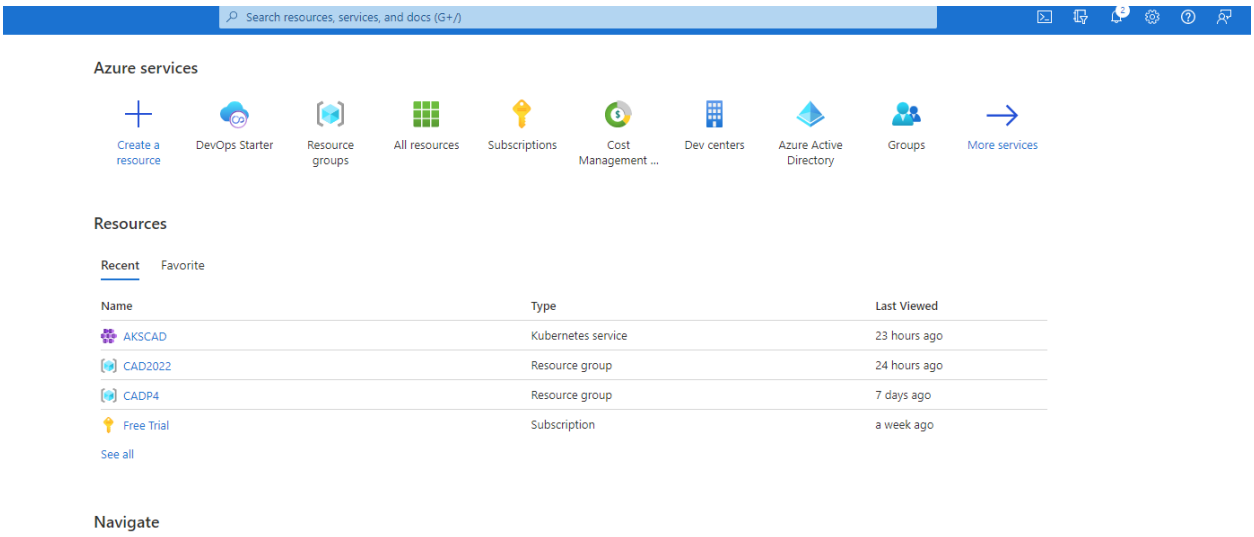
Kubernetes version	1.23.12
Auto Upgrade Type	Patch
Authentication and Authorization	Local accounts with Kubernetes RBAC
Local accounts	Enabled

Networking

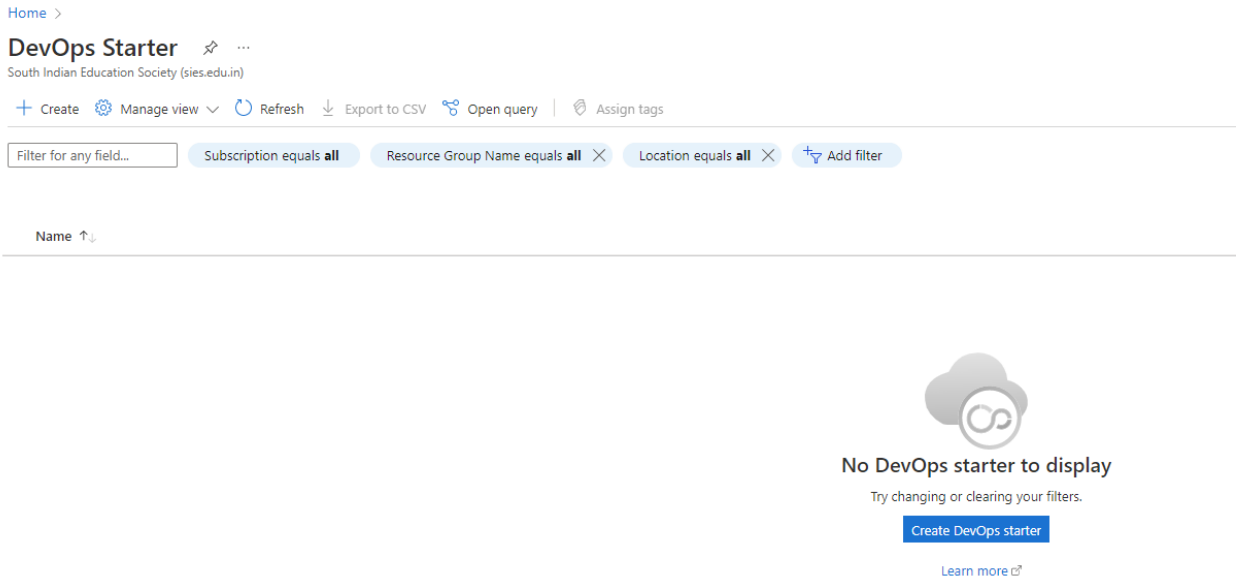
API server address	akscad-dns-78fae0a6.hcp.centralindia.azmk8s.io
Network type (plugin)	Azure CNI
Pod CIDR	-
Service CIDR	10.0.0.0/16
DNS service IP	10.0.0.10
Docker bridge CIDR	172.17.0.1/16
Network Policy	None
Load balancer	Standard
HTTP application routing	Not enabled
Private cluster	Not enabled
Authorized IP ranges	Not enabled
Application Gateway ingress controller	Not enabled

Steps :-

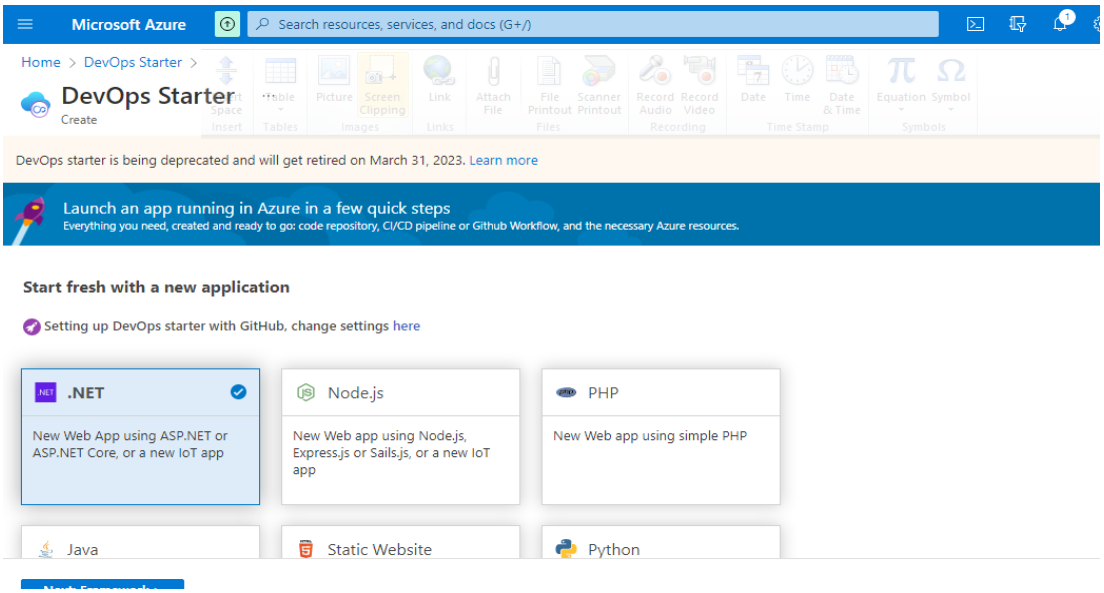
1. In the azure portal search **devops starter.**



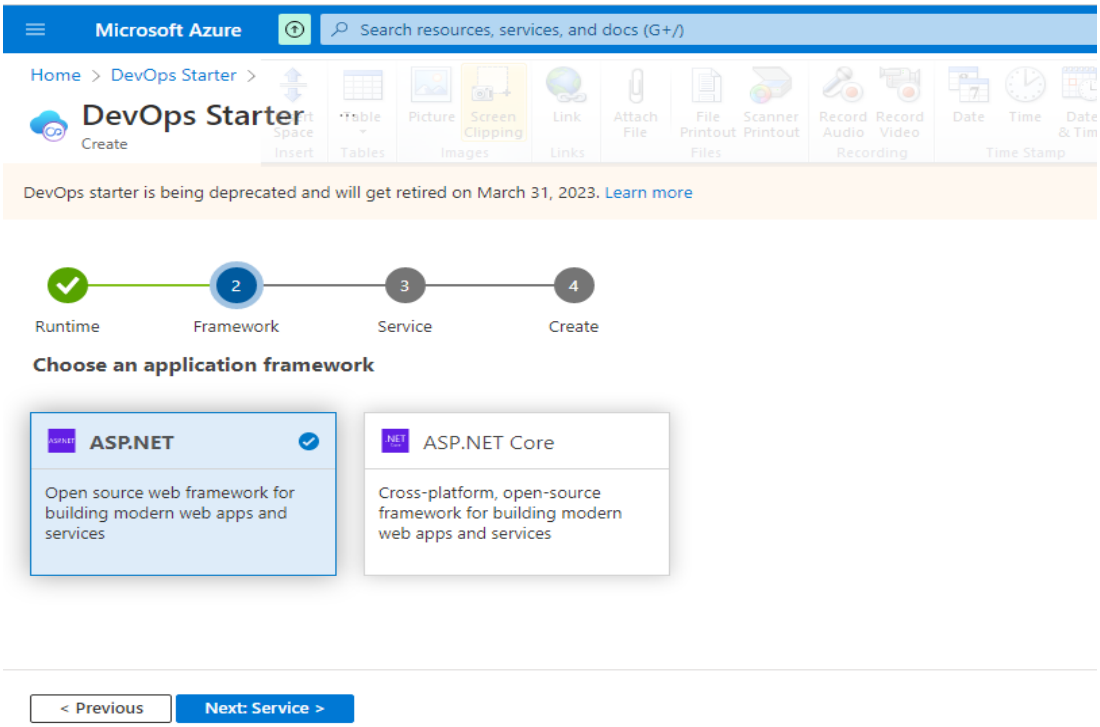
2. Click on create.



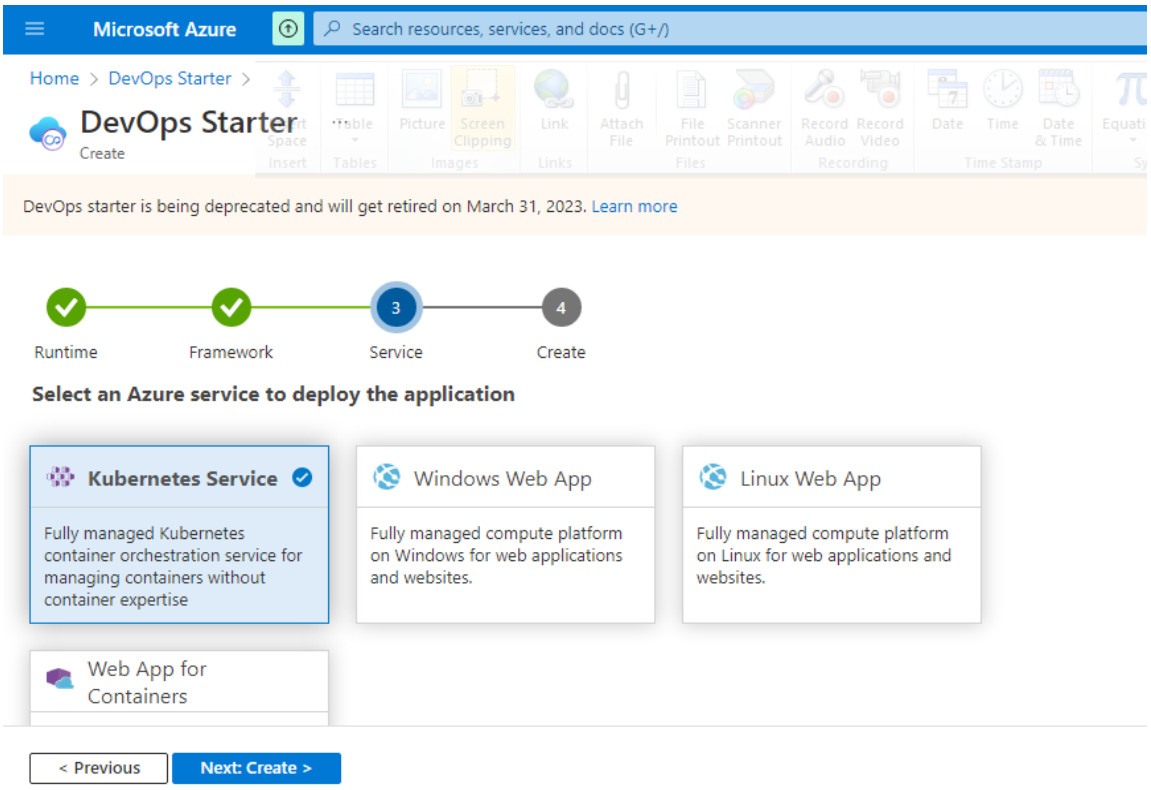
3. Select .NET and click next.



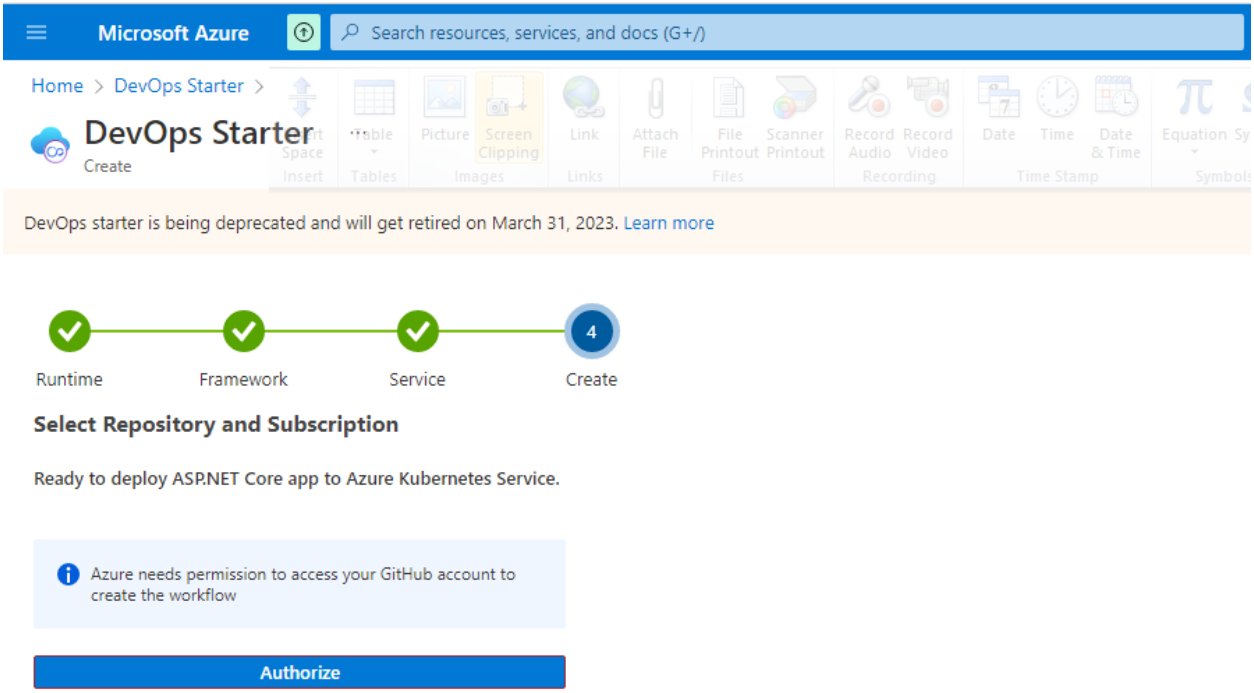
4. Under Choose an application framework**, select ASP.NET Core** and then select Next



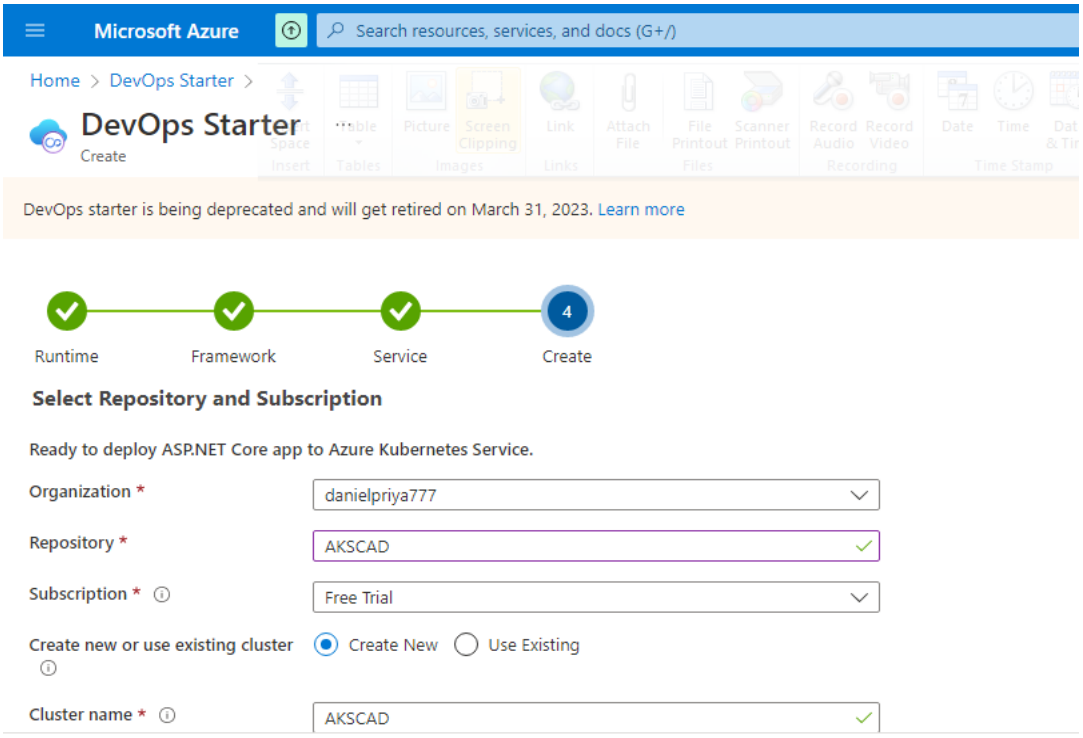
5. Select **Kubernetes Service,** and then select Next**.**



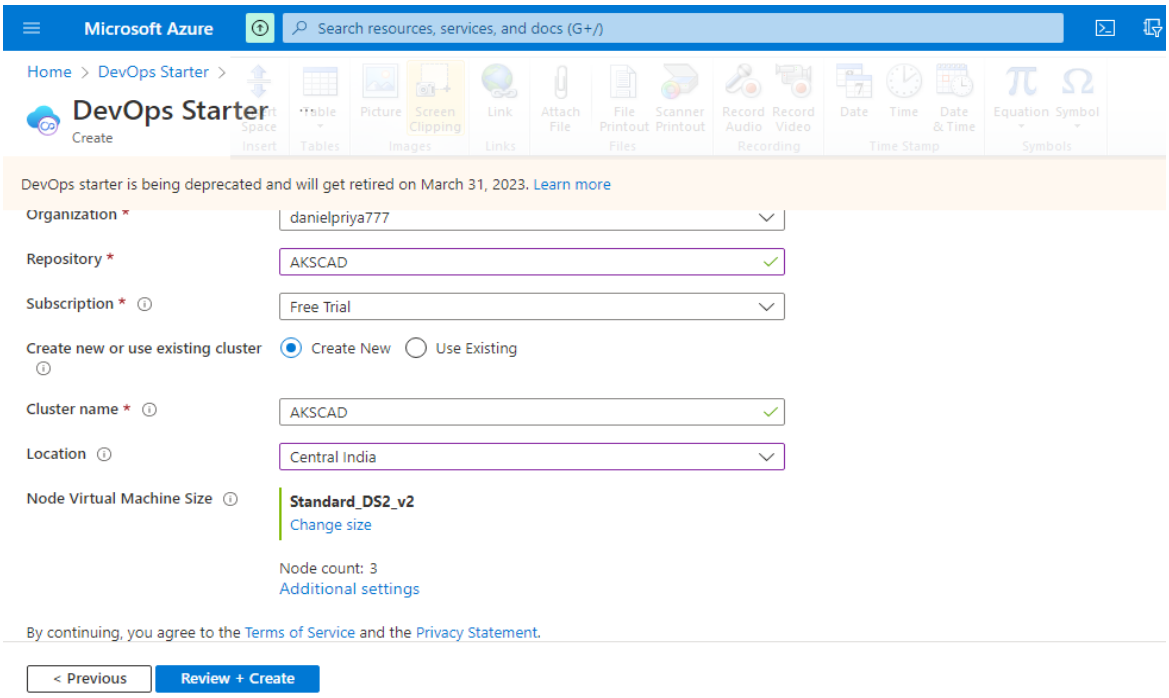
6. Click on **Authorize** and provide credentials for **Github Account.**

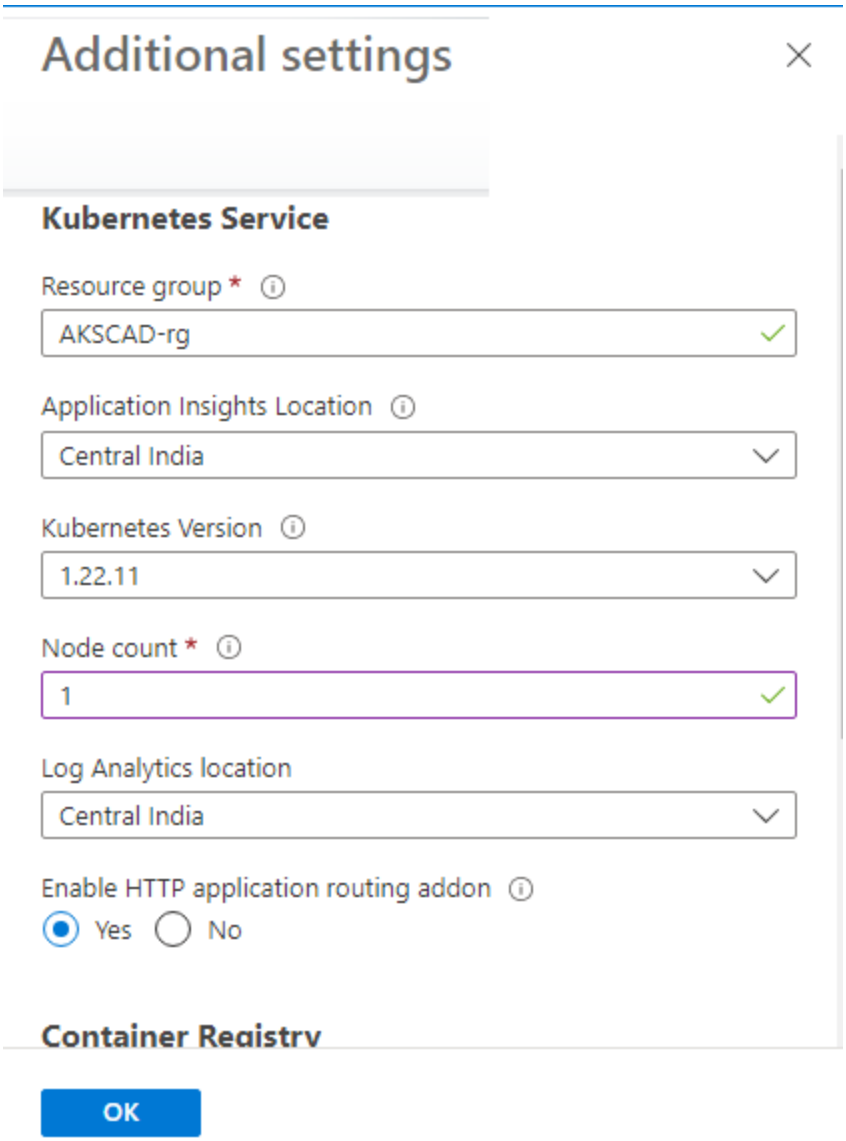


7. Fill the details.

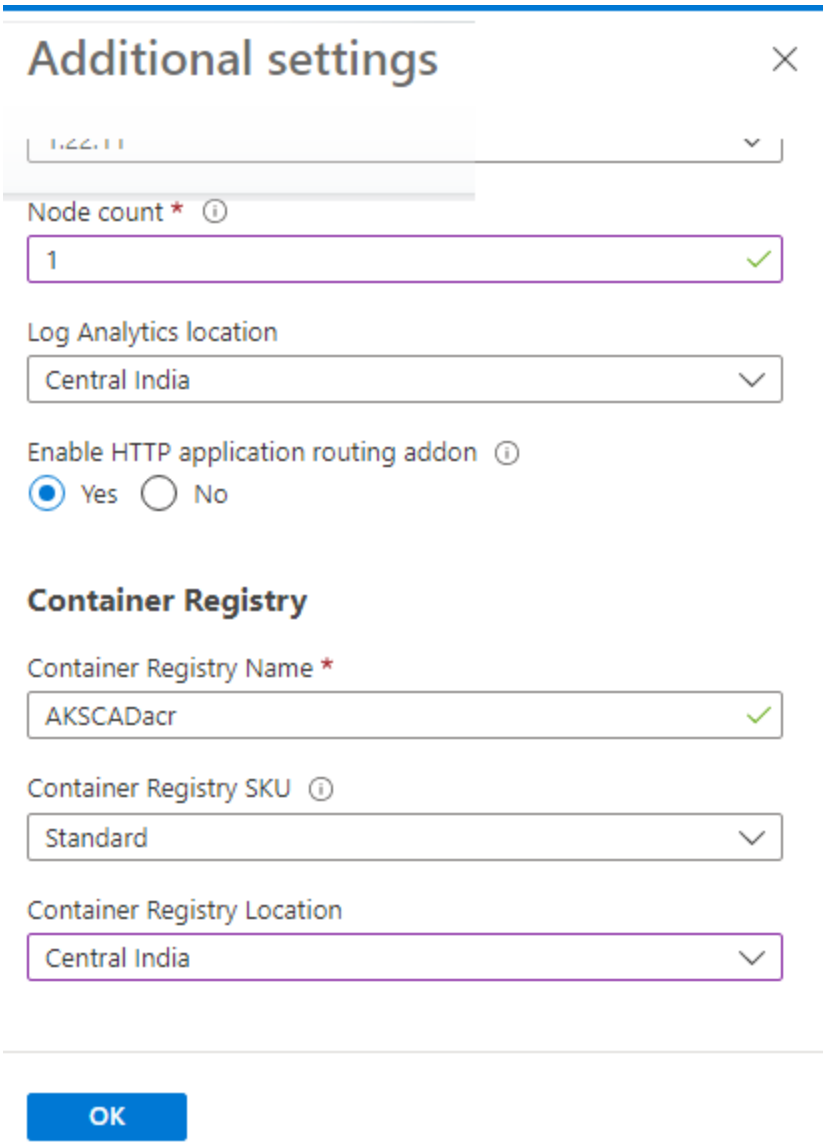


8. Click on **Additional settings** under Node **Virtual Machine** to reduce node count to 1.

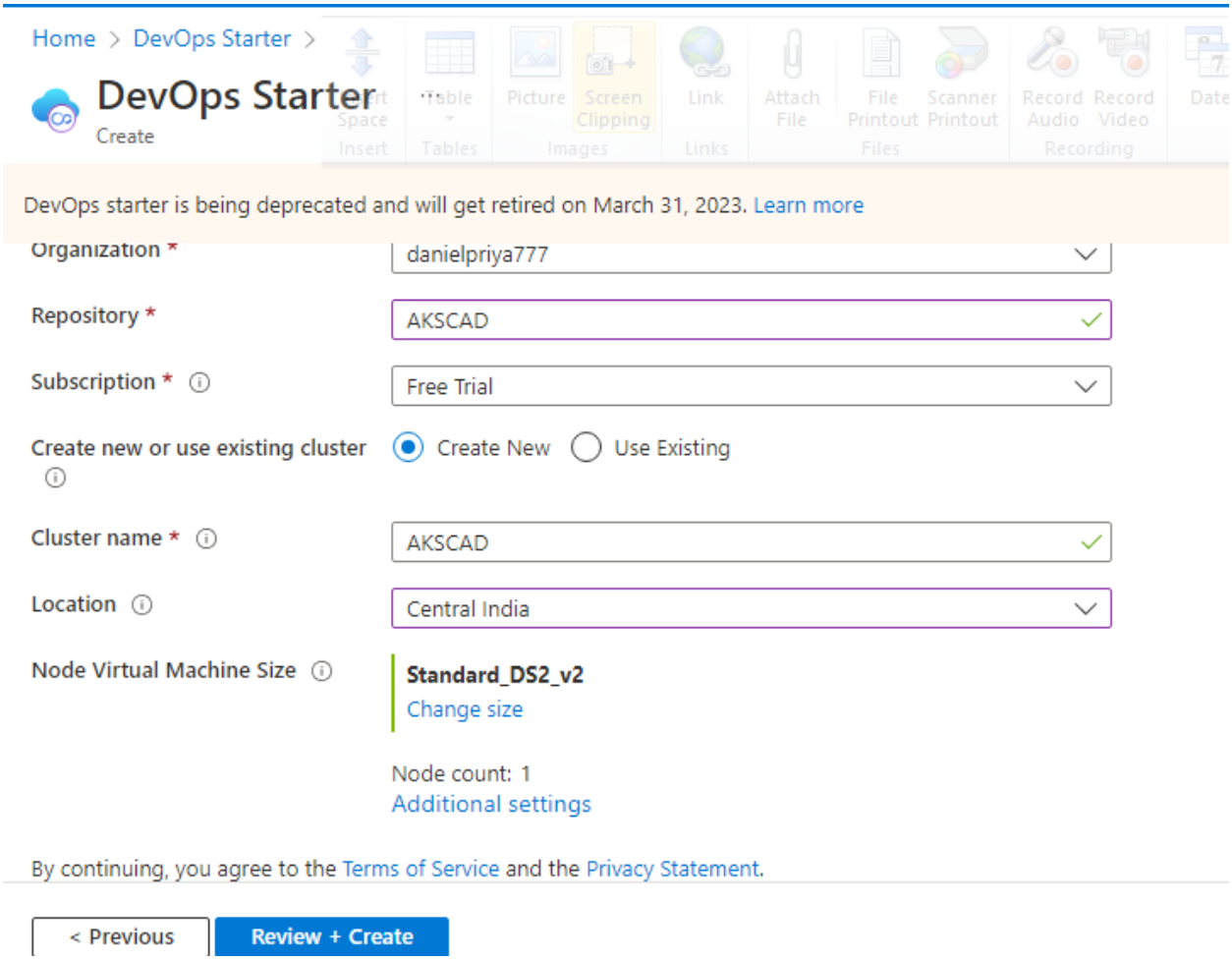


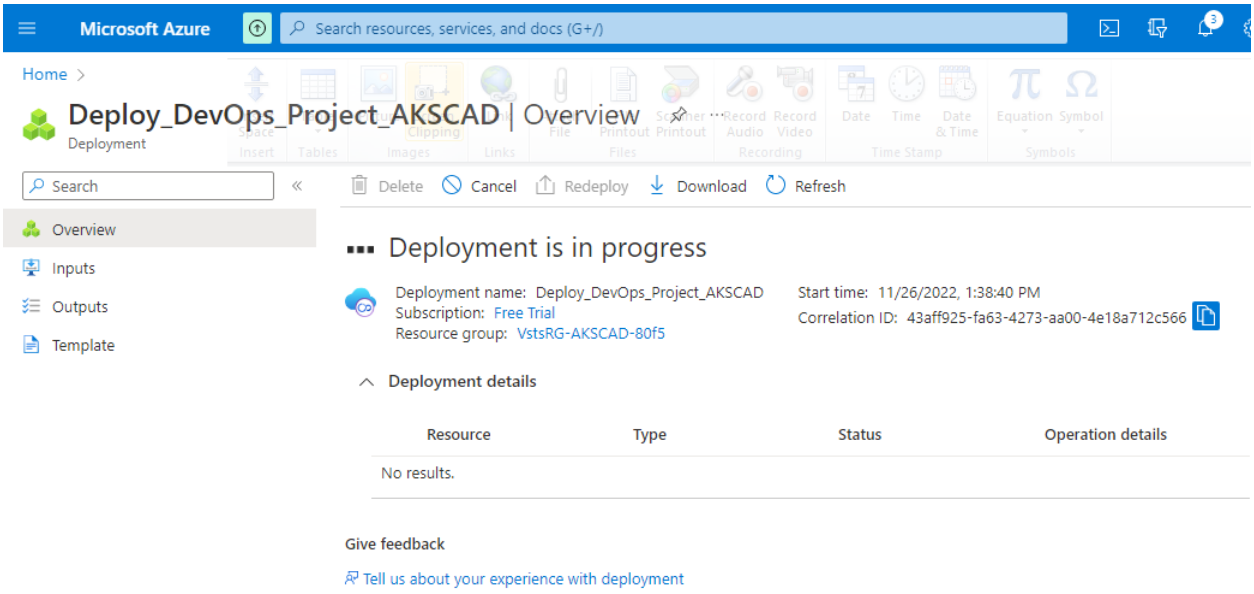

Additional settings
Kubernetes Service
Resource group *
AKSCAD-rg
Application Insights Location
Central India
Kubernetes Version
1.22.11
Node count *
1
Log Analytics location
Central India
Enable HTTP application routing addon
Yes
No
Container Registry
OK

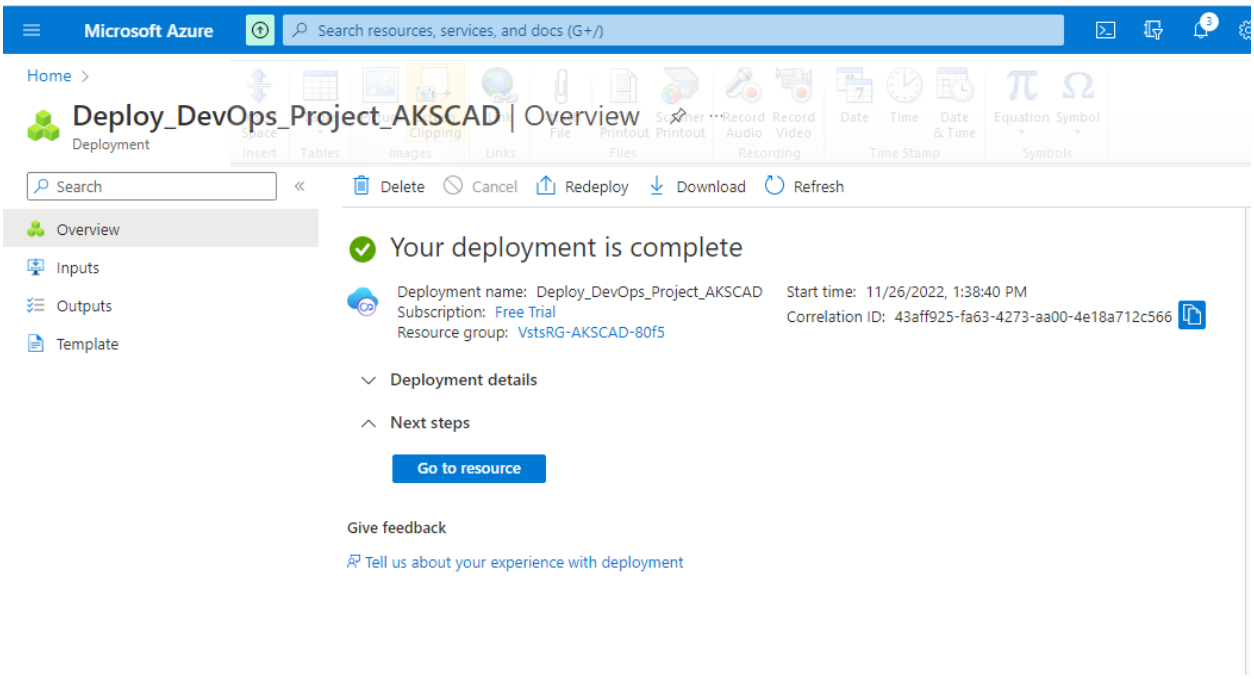
Continued.

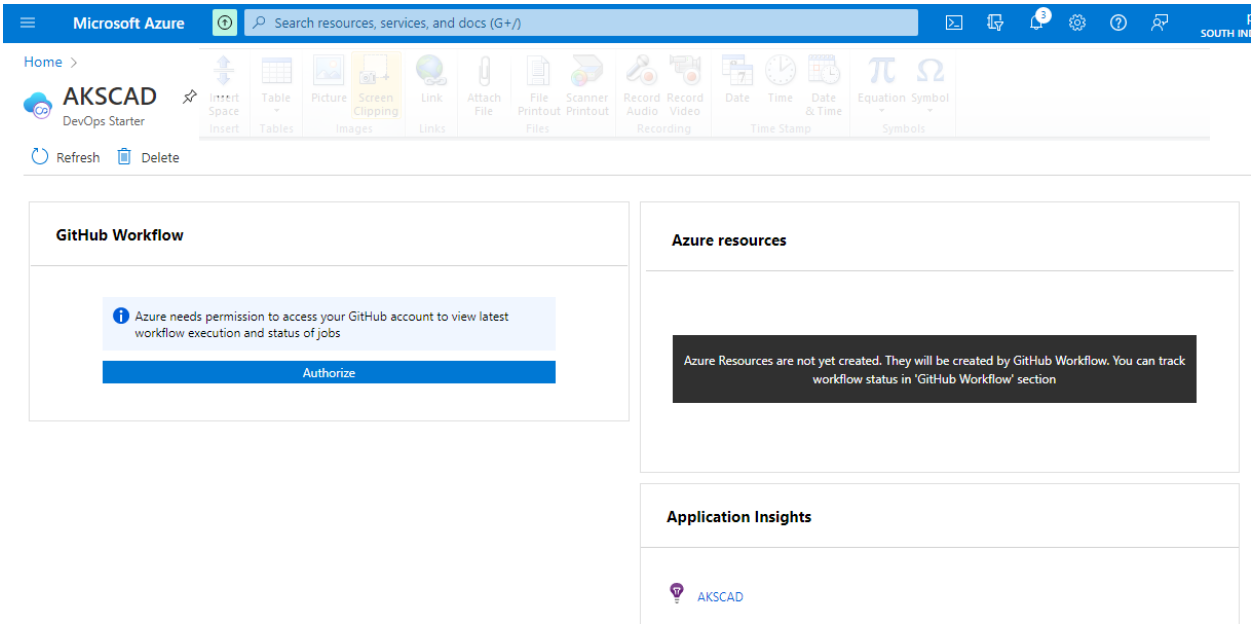

Additional settings
Node count *
1
Log Analytics location
Central India
Enable HTTP application routing addon
Yes
No
Container Registry
Container Registry Name *
AKSCADacr
Container Registry SKU
Standard
Container Registry Location
Central India
OK

9. **C**lick OK**.** Then click **review+create.**

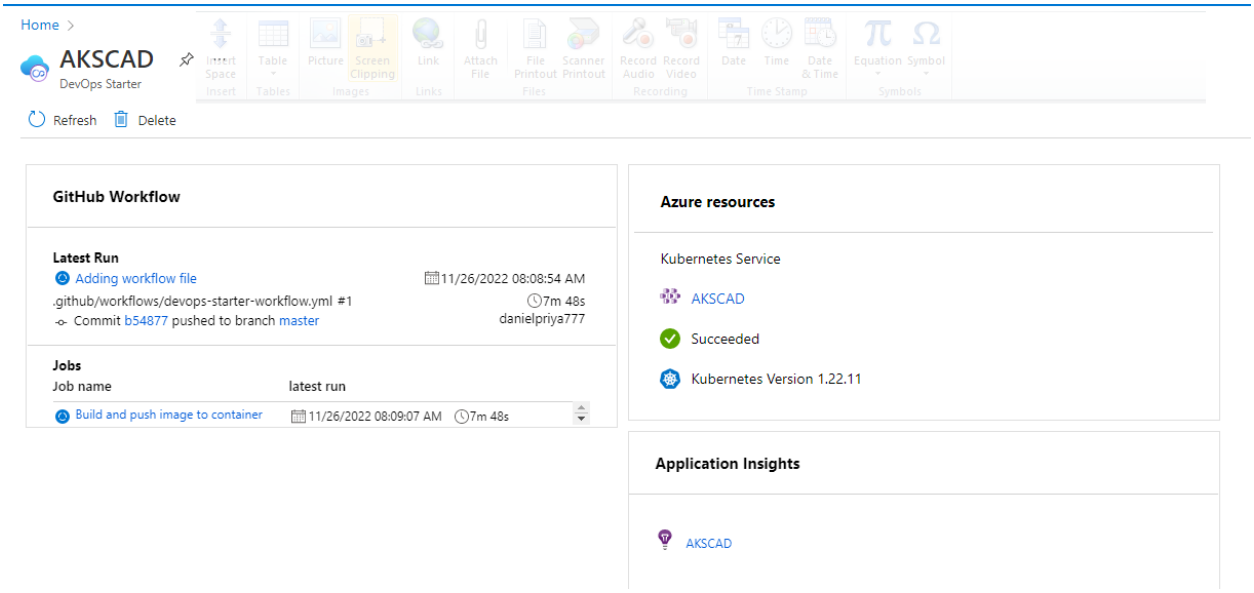



Microsoft Azure
Search resources, services, and docs (G+/)
Home
Deploy_DevOps_Project_AKSCAD | Overview
Deployment
Search
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Download
Refresh
Deployment is in progress
Deployment name: Deploy_DevOps_Project_AKSCAD
Subscription: Free Trial
Resource group: VstsRG-AKSCAD-80f5
Start time: 11/26/2022, 1:38:40 PM
Correlation ID: 43aff925-fa63-4273-aa00-4e18a712c566
Deployment details
Resource
Type
Status
Operation details
No results.
Give feedback
Tell us about your experience with deployment


Microsoft Azure
Search resources, services, and docs (G+/)
Home
Deploy_DevOps_Project_AKSCAD | Overview
Deployment
Search
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Download
Refresh
Your deployment is complete
Deployment name: Deploy_DevOps_Project_AKSCAD
Subscription: Free Trial
Resource group: VstsRG-AKSCAD-80f5
Start time: 11/26/2022, 1:38:40 PM
Correlation ID: 43aff925-fa63-4273-aa00-4e18a712c566
Deployment details
Next steps
Go to resource
Give feedback
Tell us about your experience with deployment



10. Click on **Authorize.** The Kubernetes service will be **created.**



Continued…

Home >

AKSCAD

DevOps Starter

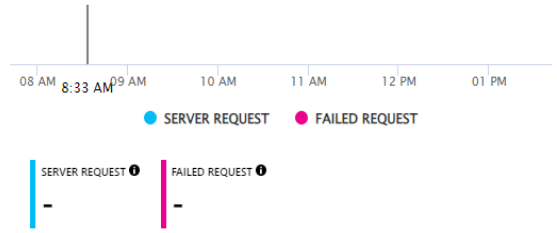
Refresh Delete

Job name	latest run
Build and push image to container	11/26/2022 08:09:07 AM 8m 17s

Kubernetes Version 1.22.11

Application Insights

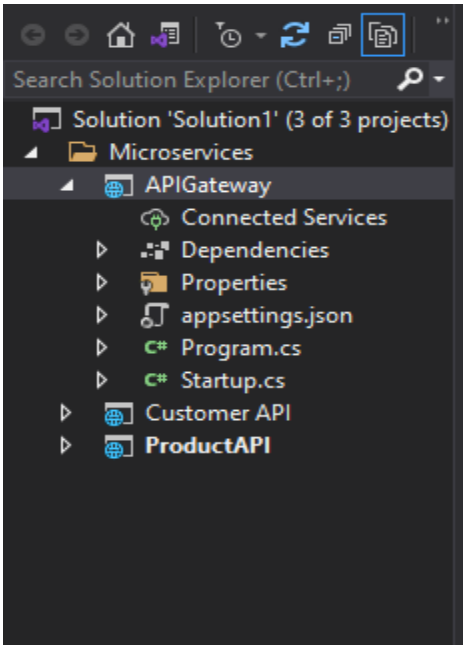
AKSCAD



Steps :-

Create a blank solution -> Solution1
Add the following projects in it.

ProductAPI (Web API)
CustomerAPI (Web API)
APIGateway(Asp.net core empty project)



1. Add a co**ntroller to ProductAPI** and add the following Code:

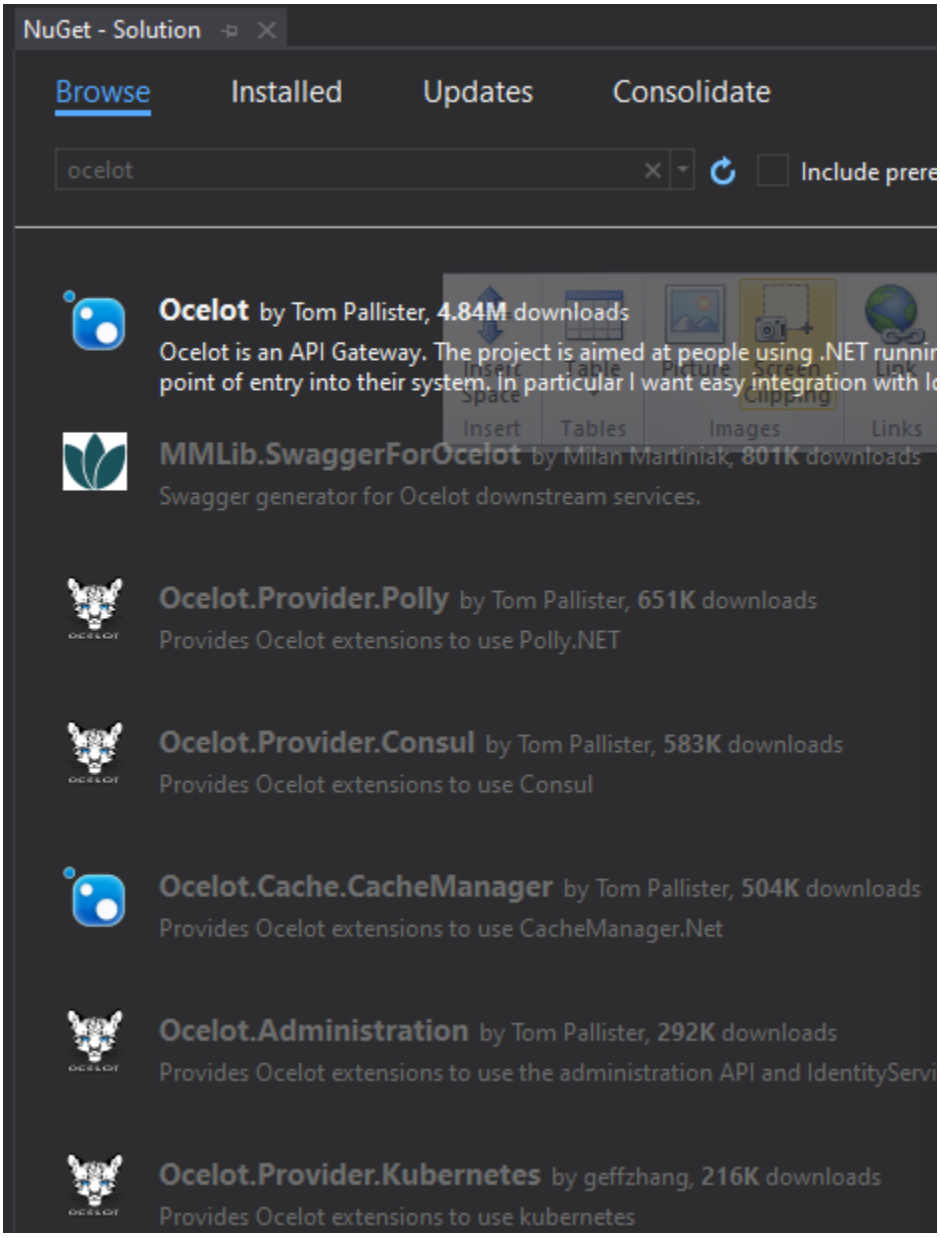
```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ProductAPI.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class ProductController : ControllerBase
   {
      public ProductController()
```

```
        { }
        [HttpGet]
        public List<String> Get()
        {
            var productList = new List<String> { "Product1", "Product2", "Product3", "Product4" };
            return productList;
        }
        [HttpDelete]
        [Route("ProductVersion/{id}")]
        public string ProductVersion(int id)
        {
            return $"Product with id {id} is removed";
        }


    }
}
```

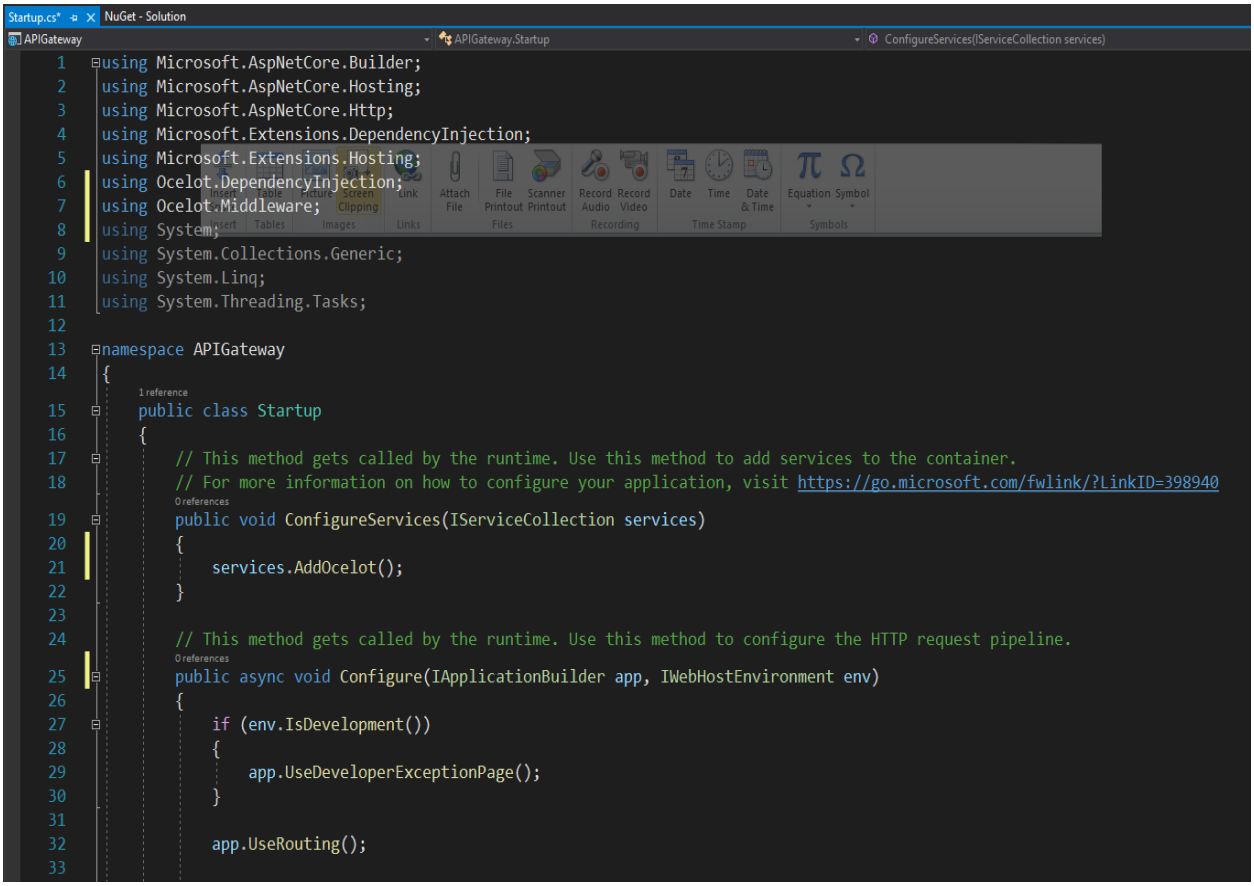
2. **Add controller to** customerAPI and add the following code**.**

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

namespace Customer_API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CustomerController : ControllerBase
    {
        public List<string> Get()
        {
            var productList = new List<string> { "Customer 1","Customer 2","Customer 3","Customer 4" };
            return productList;
        }
    }
}
```

3. **Install OCELOT** using nuget package manager.



4. Open **Startup.cs** for APIGateway. Add **Ocelot** in services under Configure Service.

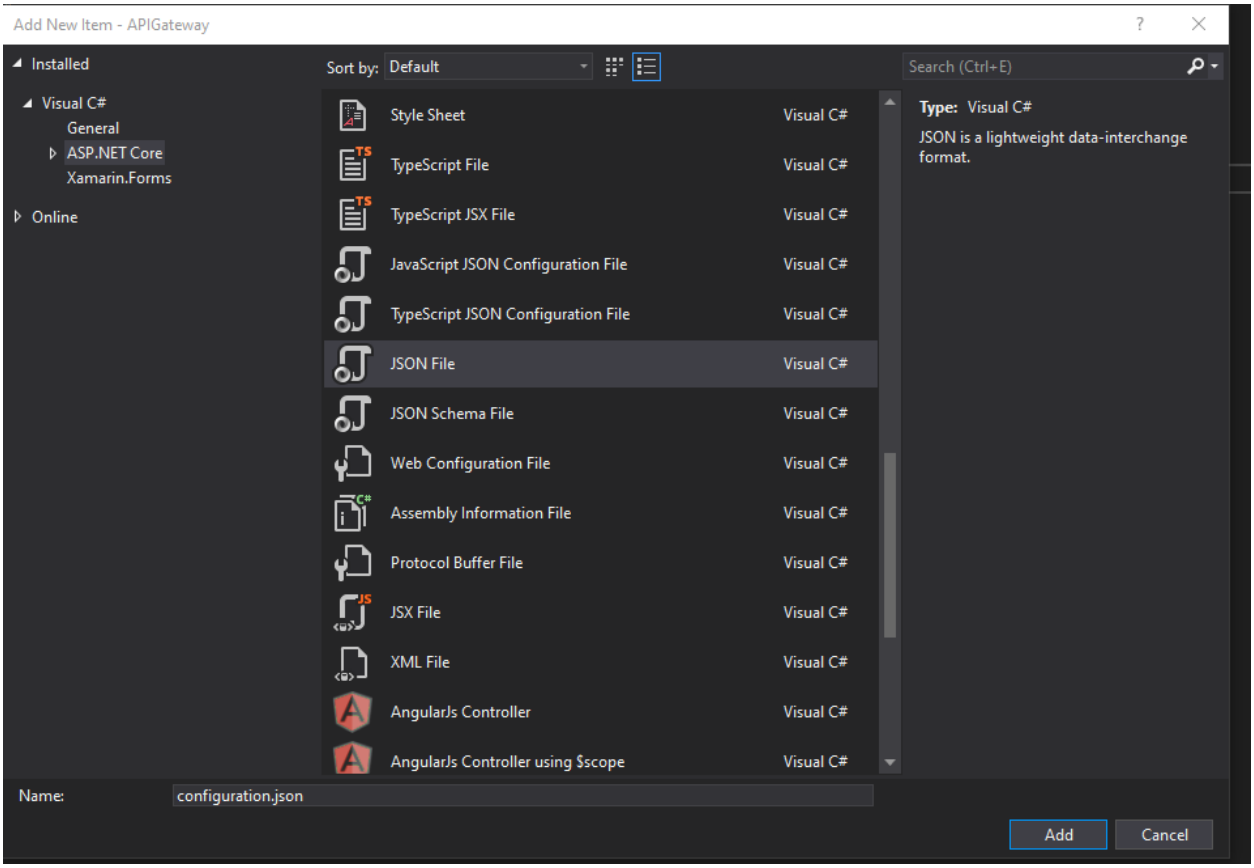


5. At the end , write await app.UseOcelot to make sure that the app uses the Ocelot Middleware. So the final Code looks like this for **APIGateway -> Startup.cs**

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Ocelot.DependencyInjection;
using Ocelot.Middleware;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace APIGateway
{
   public class Startup
   {
```

```
        /*
This method gets called by the runtime. Use this method to add services to the container.
        For more information on how to configure your application, visit
https://go.microsoft.com/fwlink/?LinkID=398940
*/
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddOcelot();
        }

        /* This method gets called by the runtime. Use this method to configure the HTTP request
pipeline.*/
        public async void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapGet("/", async context =>
                {
                    await context.Response.WriteAsync("Hello World!");
                });
            });
            await app.UseOcelot();
        }
    }
}
```

6. Next, right click on **APIGateway in solution->New Item->JSON file. Name it Configuration.json file.**



7. After that, open **program.cs file** of APIGateway and make the following changes.

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace APIGateway
{
   public class Program
   {
      public static void Main(string[] args)
      {
         CreateHostBuilder(args).Build().Run();
      }
```

```
        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                })
            .ConfigureAppConfiguration((hostingContext, config) =>
            {
                config.AddJsonFile("configuration.json");
            });
    }
}
```

Microservices is a new software architecture. It is based on the Web Services (Web APIs). Microservices are the architectural approach to build applications from small to large scale applications.

An API gateway is an API management tool that sits between a client and a collection of backend services.

Ocelot is a lightweight, open-source, scalable, and fast API Gateway based on . NET Core and specially designed for microservices architecture. Basically, it is a set of middleware designed to work with ASP.NET Core.
It has several features such as routing, caching, security, rate limiting, etc.

Configuration File Syntax

```
{
  "Routes": [
    {
      "UpstreamPathTemplate": "/gateway/product",
      "UpstreamHttpMethod": [ "POST", "PUT", "GET" ],
      "DownstreamPathTemplate": "/api/product",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44339
        }
      ],
      "DangerousAcceptAnyServerCertificateValidator": true

    },
```

```
  {
    "UpstreamPathTemplate": "/gateway/product/{id}",
    "UpstreamHttpMethod": [ "GET", "DELETE" ],
    "DownstreamPathTemplate": "/api/product/ProductVersion/{id}",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 44339
      }
    ],

    "DangerousAcceptAnyServerCertificateValidator": true

  },
  {
    "UpstreamPathTemplate": "/gateway/customer",
    "UpstreamHttpMethod": [ "POST", "PUT", "GET" ],
    "DownstreamPathTemplate": "/api/customer",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 44398
      }
    ],

    "DangerousAcceptAnyServerCertificateValidator": true

  },
  {
    "DownstreamPathTemplate": "/api/customer/{id}",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 44398
      }
    ],
    "UpstreamPathTemplate": "/gateway/customer/{id}",
    "UpstreamHttpMethod": [ "GET", "DELETE" ],
    "DangerousAcceptAnyServerCertificateValidator": true
  }
],
```

```
  "GlobalConfiguration": {
    "BaseUrl": "http://localhost:44353"
  }
}
```

Change the SSL Port in the launchsetting.json file of ProductAPI and CustomerAPI to the value in configuration.jso file of API Gateway.

ProductAPI->Launchsetting.json

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:35108",
      "sslPort": 44339
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "ProductAPI": {
      "commandName": "Project",
      "launchBrowser": true,
      "applicationUrl": "https://localhost:5001;http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

CustomerAPI->Launchsetting.json

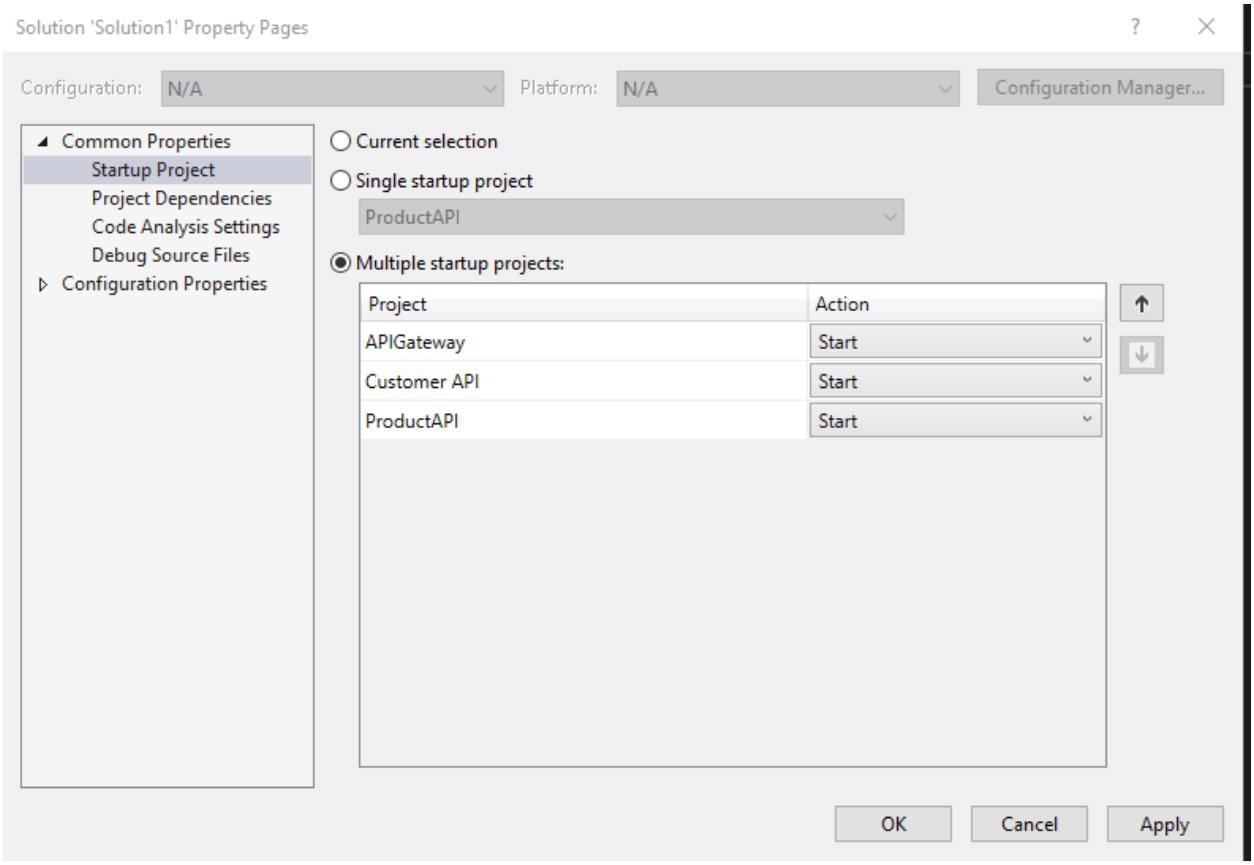
```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
```

```
    "applicationUrl": "http://localhost:6080",
    "sslPort": 44398
   }
  },
  "profiles": {
   "IIS Express": {
    "commandName": "IISExpress",
    "launchBrowser": true,
    "environmentVariables": {
     "ASPNETCORE_ENVIRONMENT": "Development"
    }
   },
   "CustomerAPI": {
    "commandName": "Project",
    "launchBrowser": true,
    "applicationUrl": "https://localhost:5001;http://localhost:5000",
    "environmentVariables": {
     "ASPNETCORE_ENVIRONMENT": "Development"
    }
   }
  }
}
```

8. Set up multiple startup in **VS Studio**

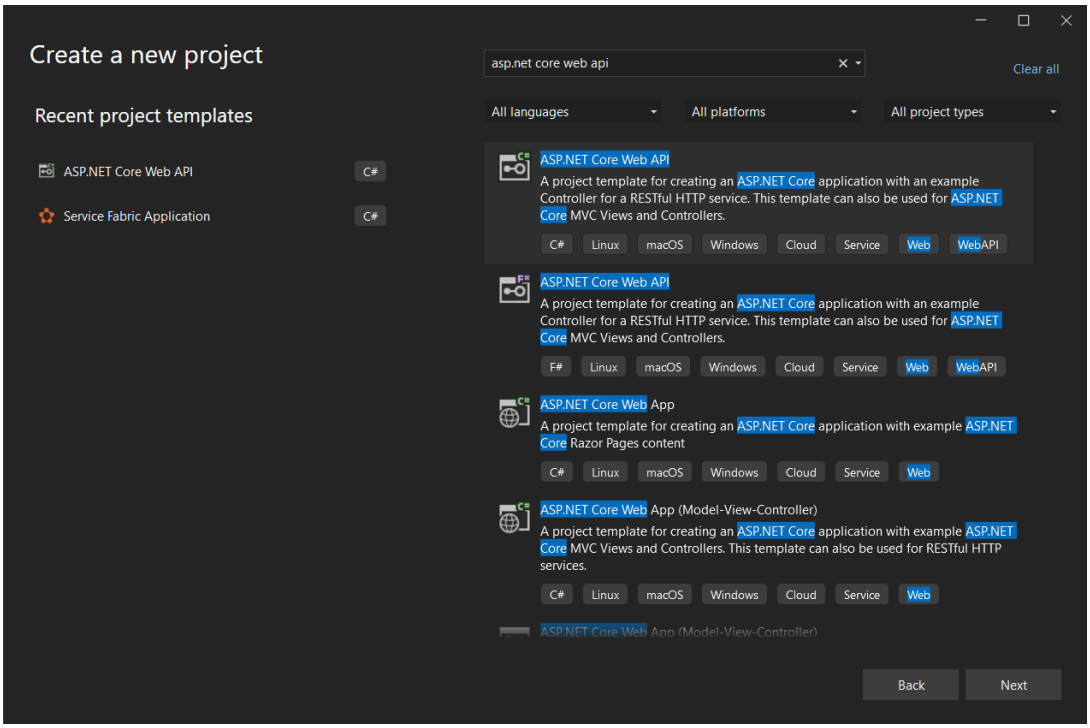
Right click on solution-> properties->Select “Multiple startup projects”. Then set all 3 projects to start.

9. Now run the project.

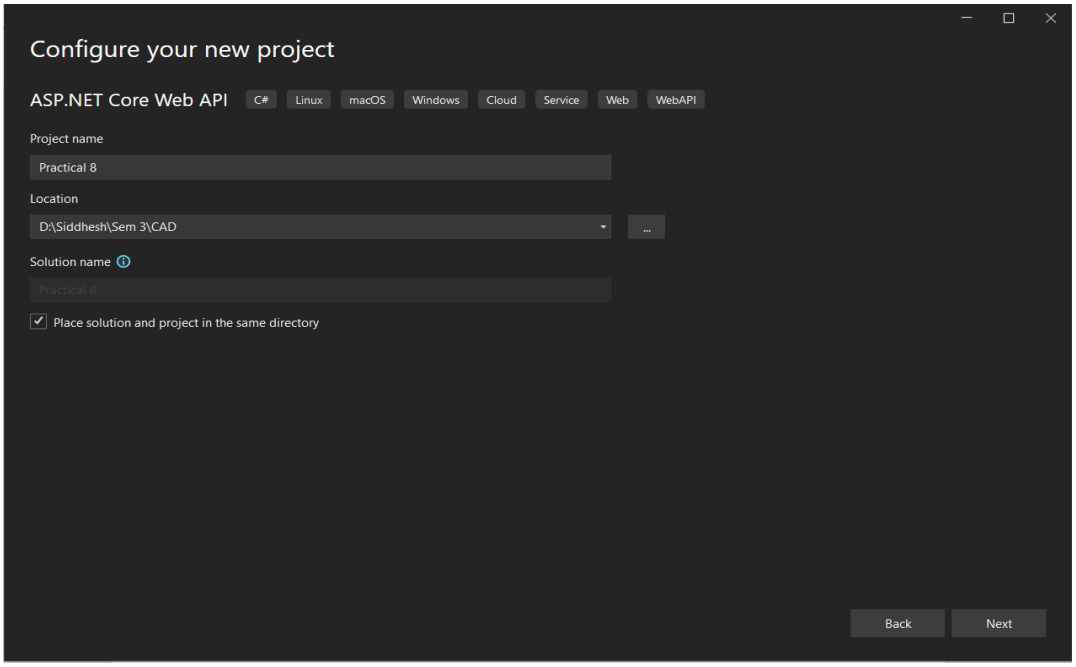


Steps :-

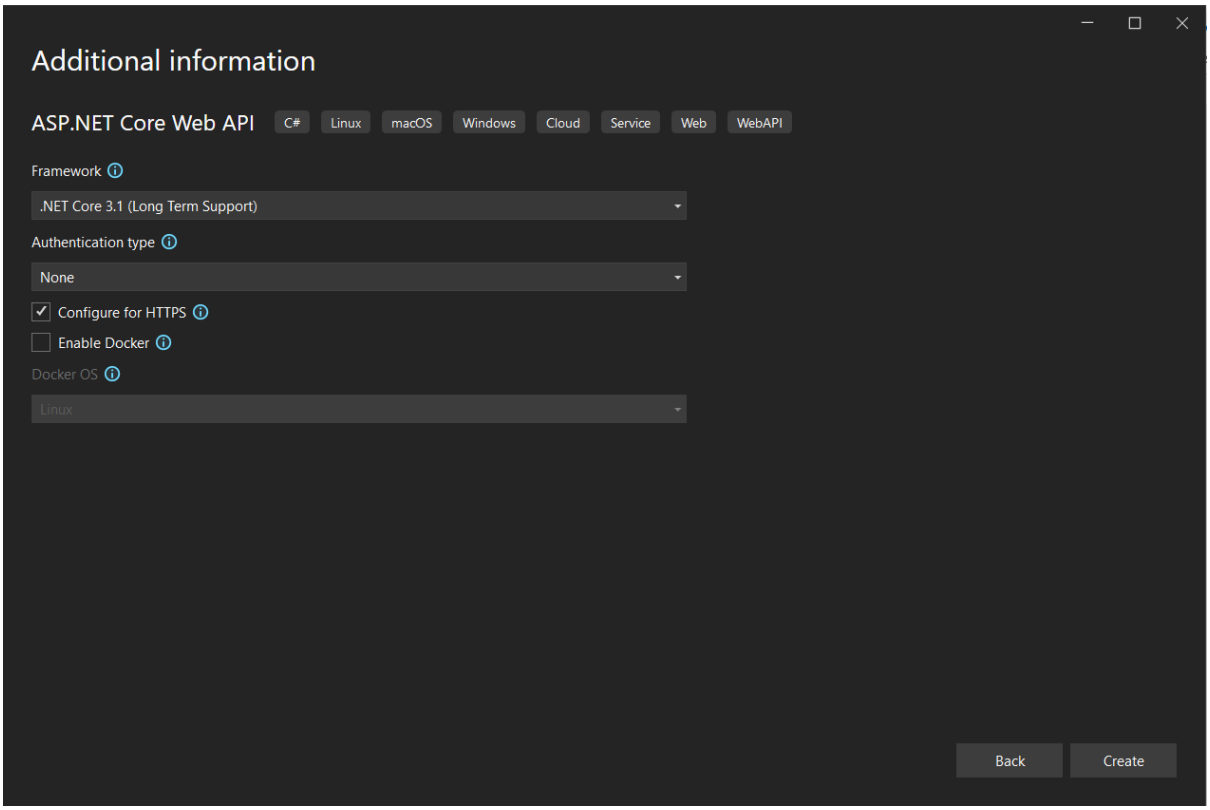
1. Open Visual studio and Create a new project of the type **ASP.NET Core Web API.**



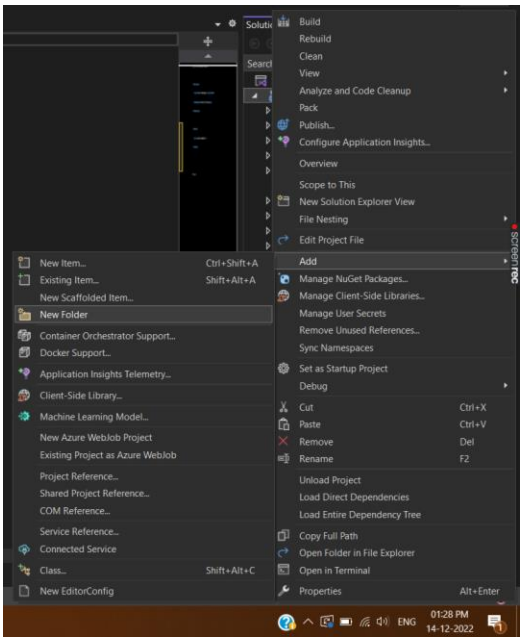
2. Give it a Meaningful name.



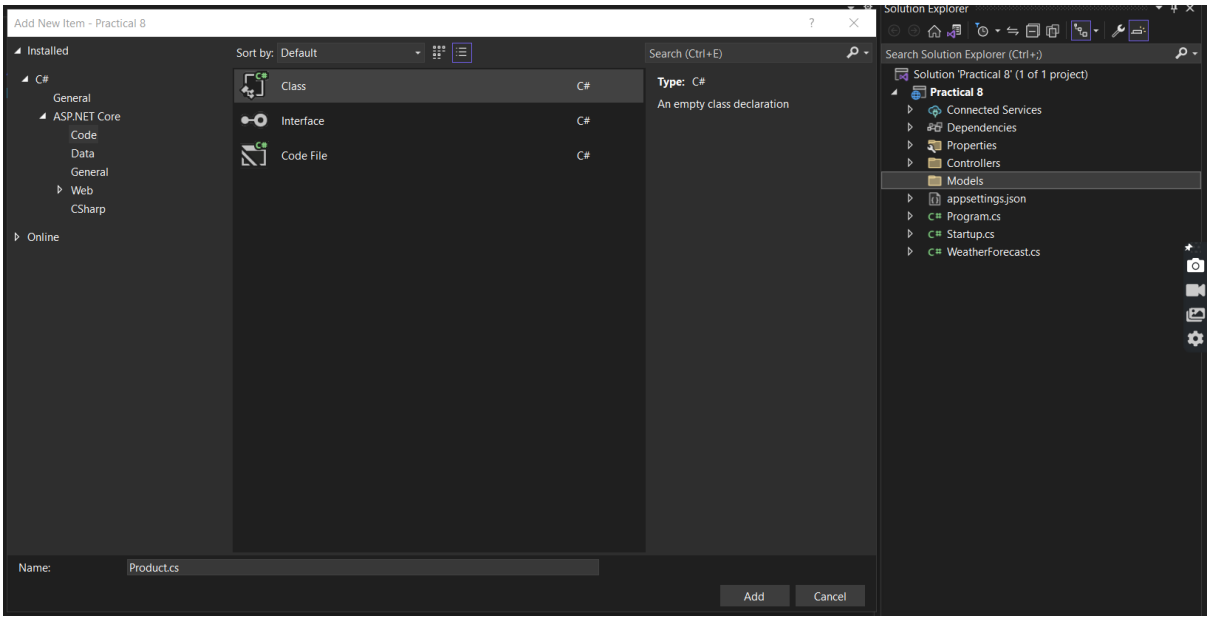
3. Select **.NET Core 3.1** as Framework for this project and Click on Create.



4. Add a folder in the project and name it **"Models".**

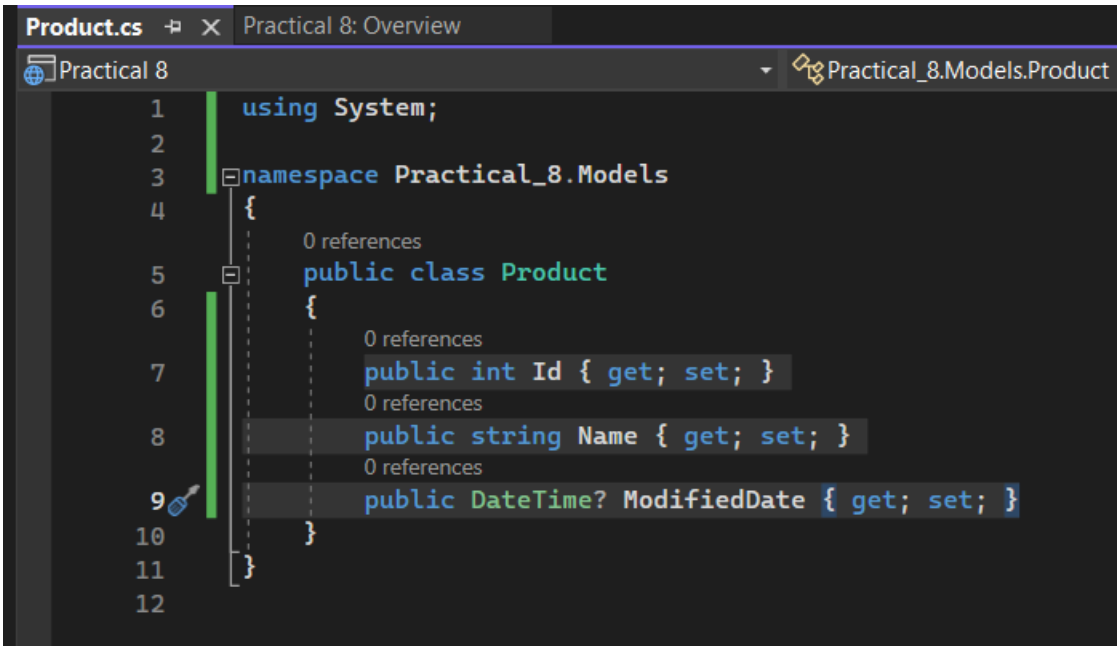


5. Create the Product model in this folder. Give it name as **Product.cs.**

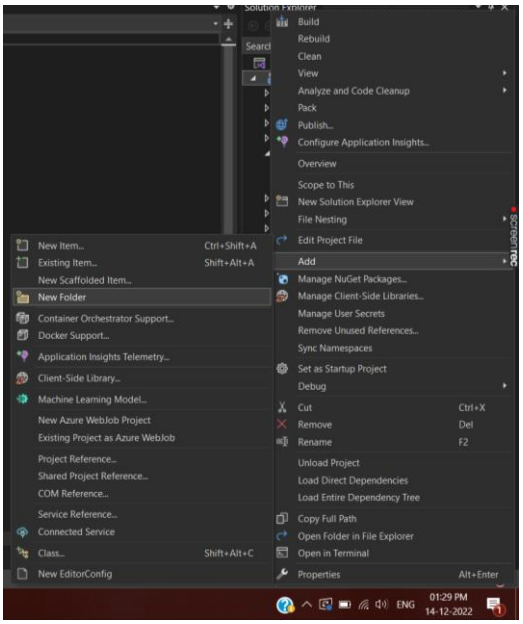


6. Add the following code inside **Product.cs** file:

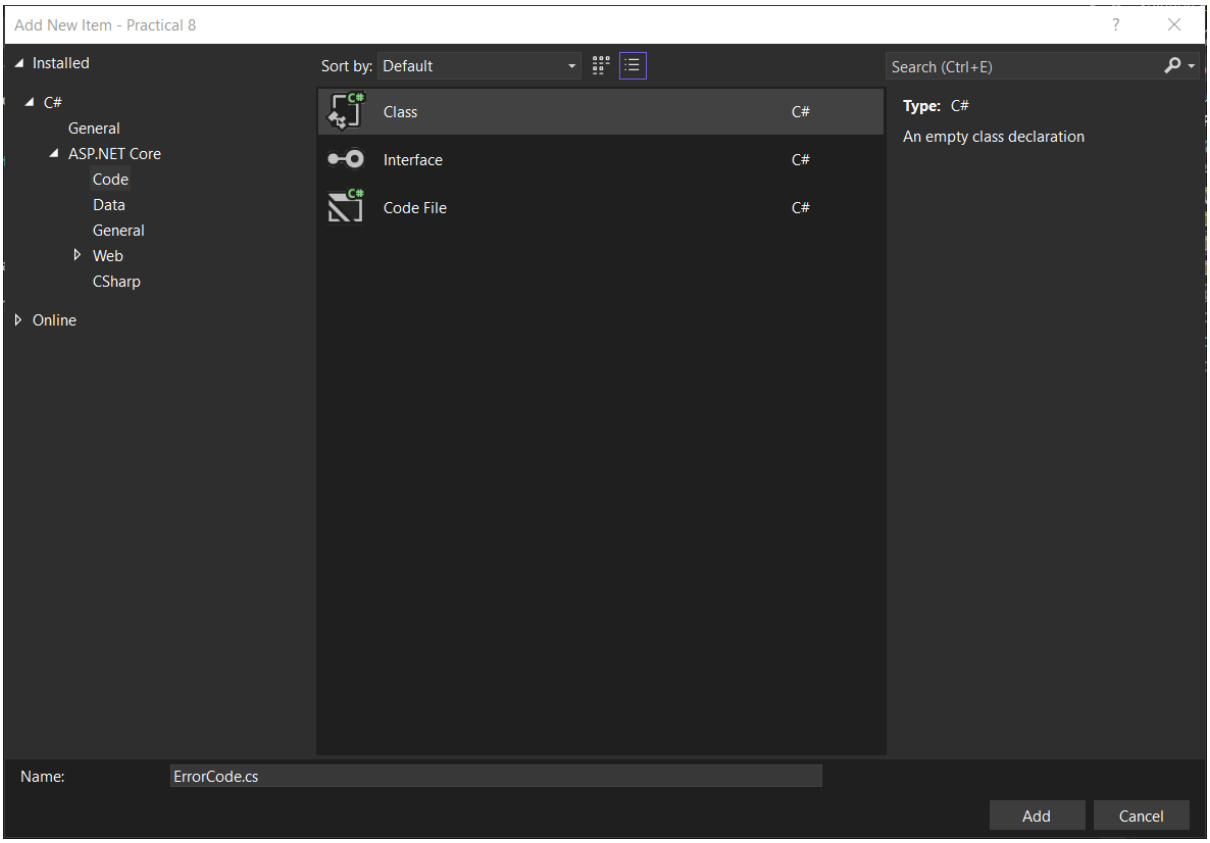
```
public class Product
 {
     public int Id { get; set; }
     public string Name { get; set; }
     public DateTime? ModifiedDate { get; set; }
 }
```



7. Add another folder and name it **“Errors”.**

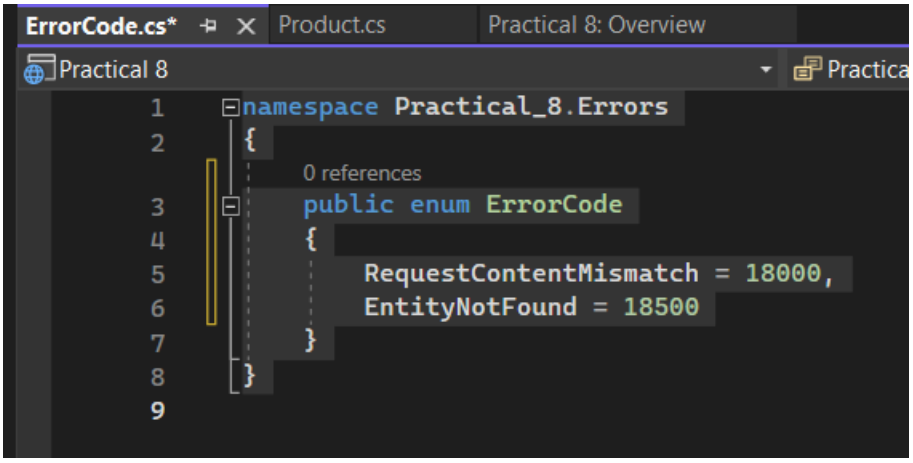


8. Add a class inside **Errors** folder and Name it as **ErrorCode.cs.**

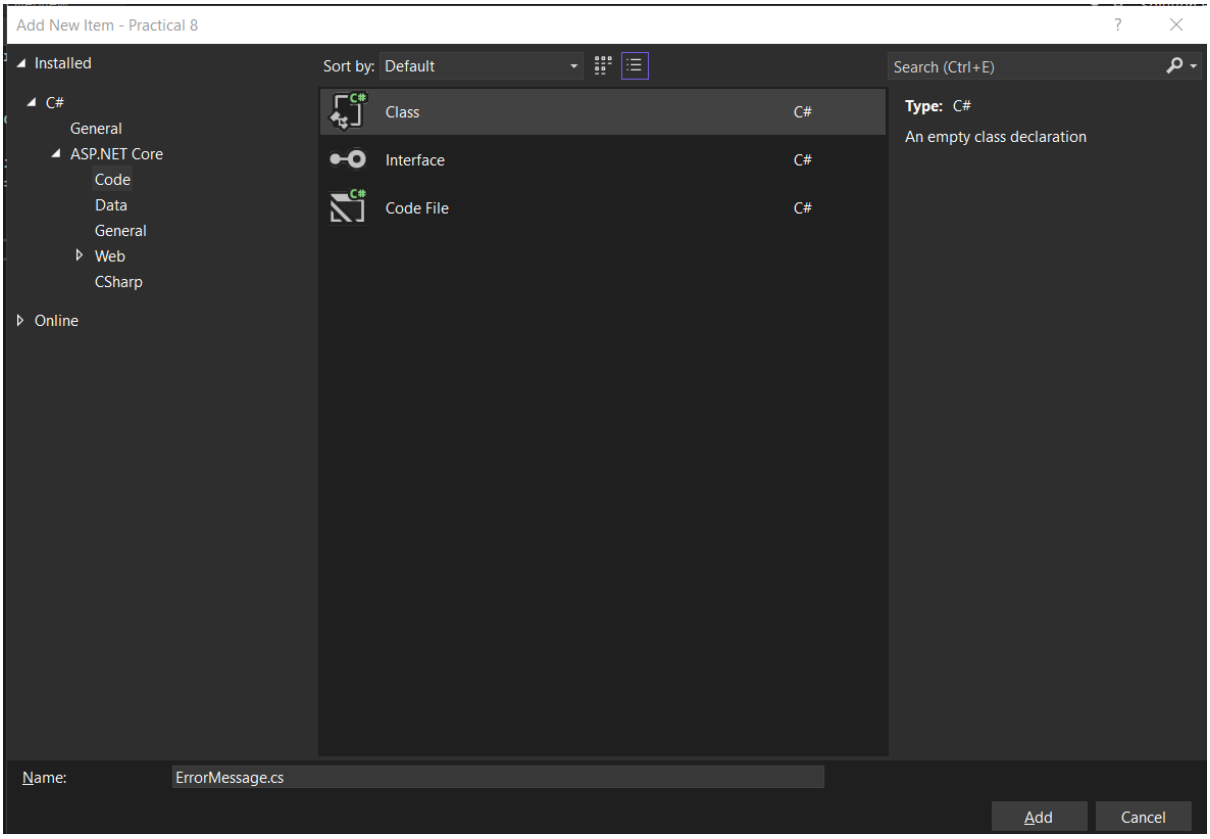


9. Add the following code inside **ErrorCode.cs** file:

```
public enum ErrorCode
{
    RequestContentMismatch = 18000,
    EntityNotFound = 18500
}
```

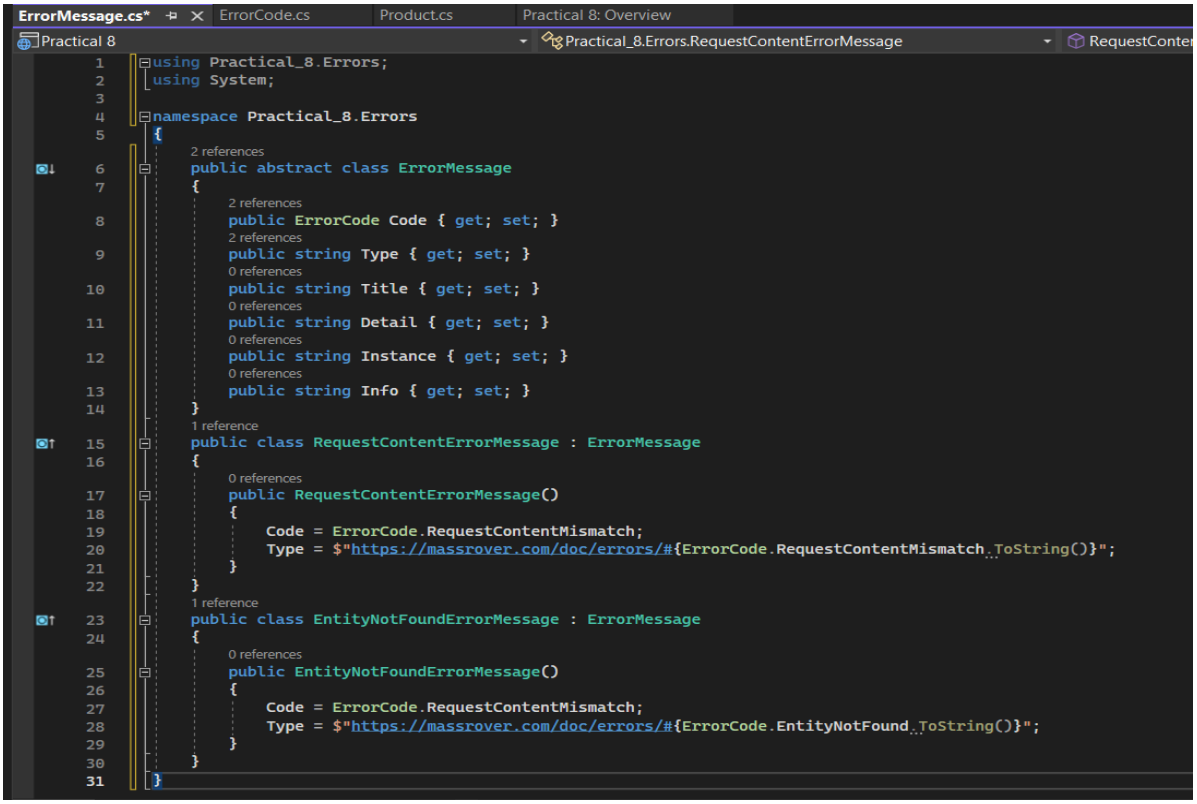


10. Add an abstract class in the **Errors** folder, named **"ErrorMessage.cs"**

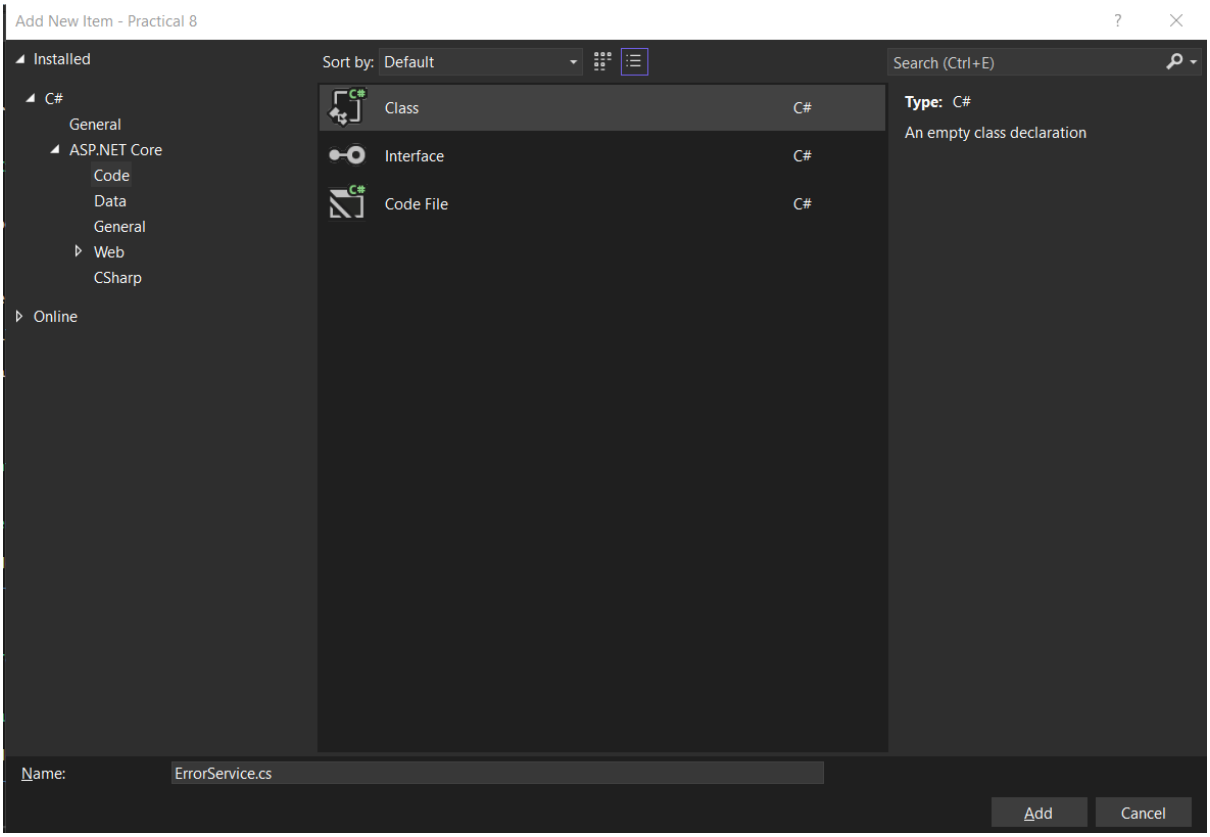


11. Add the following code inside **ErrorMessage.cs** file:

```
public abstract class ErrorMessage
{
    public ErrorCode Code { get;set; }
    public string Type { get; set; }
    public string Title { get; set; }
    public string Detail { get; set; }
    public string Instance { get; set; }
    public string Info { get; set; }
}
public class RequestContentErrorMessage : ErrorMessage
{
    public RequestContentErrorMessage()
    {
            Code = ErrorCode.RequestContentMismatch;
            Type=$"https://massrover.com/doc/errors/#{ErrorCode.RequestContentMismatch.ToString()}";
    }
}
public class EntityNotFoundErrorMessage : ErrorMessage
{
   public EntityNotFoundErrorMessage()
   {
       Code = ErrorCode.RequestContentMismatch;
              Type=$"https://massrover.com/doc/errors/#{ErrorCode.EntityNotFound.ToString()}";
   }
}
```



12. Add another class to return the correct **ErrorMessage** instance in the right context and name it as **ErrorService.cs.**



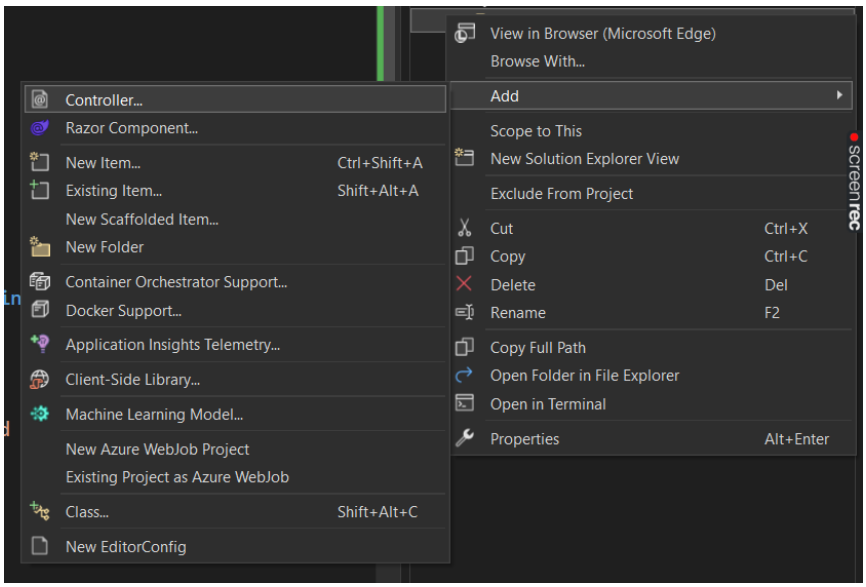
13. Add the following code in the **ErrorService.cs** file:

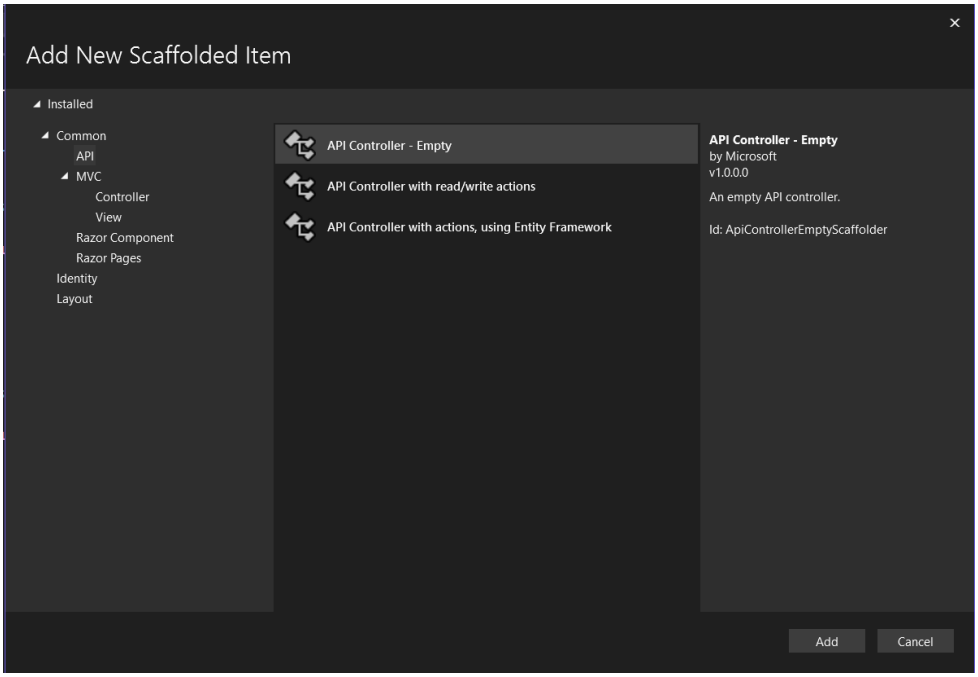
```
public static class ErrorService
{
	public static ErrorMessage GetRequestContentMismatchErrorMessage()
	{
		return new RequestContentErrorMessage
		{
			Title = $"Request content mismatch",
			Detail = $"Error in the request context."
		};
	}
	public static ErrorMessage GetEntityNotFoundErrorMessage(Type entity, int     id)
	{
		return new EntityNotFoundErrorMessage
		{
			Title = $"{entity.Name} not found",
			Detail = $"No {entity.Name.ToLower()} found for the supplied id {id}"
		};
	}
}
```

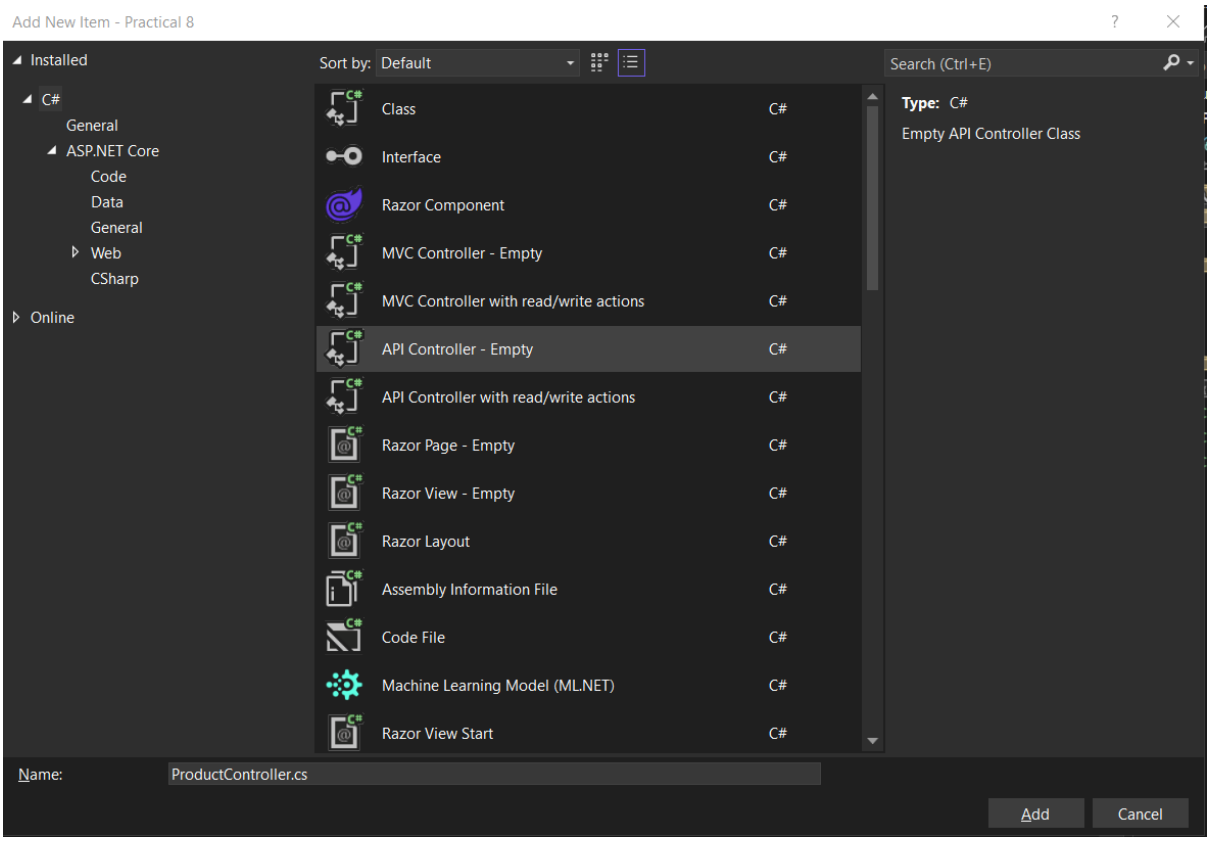
```
ErrorService.cs | ErrorMessage.cs* | ErrorCode.cs | Product.cs | Practical 8: Overview
Practical 8 | Practical_8.Errors.ErrorService | GetRequestContentMismatchErr
```

```
1  using System;
2
3  namespace Practical_8.Errors
4  {
       0 references
5      public static class ErrorService
6      {
           0 references
7          public static ErrorMessage GetRequestContentMismatchErrorMessage()
8          {
9              return new RequestContentErrorMessage
10             {
11                 Title = $"Request content mismatch",
12                 Detail = $"Error in the request context."
13             };
14         }
           0 references
15         public static ErrorMessage GetEntityNotFoundErrorMessage(Type entity, int id)
16         {
17             return new EntityNotFoundErrorMessage
18             {
19                 Title = $"{entity.Name} not found",
20                 Detail = $"No {entity.Name.ToLower()} found for the supplied id - {id}"
21             };
22         }
23     }
24 }
25
```

14. Create an **ASP.NET Core controller** with the actions for the CRUD operations of Product entity. In orde to this, add an **empty Web API** controller named **"ProductsController.cs"**

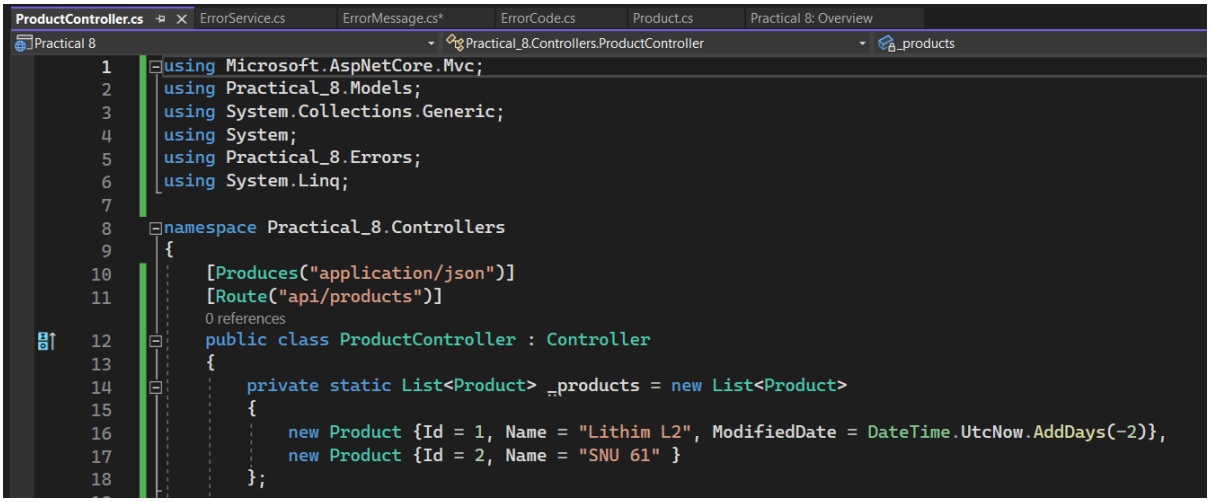



Add New Scaffolded Item
Installed
Common
API
MVC
Controller
View
Razor Component
Razor Pages
Identity
Layout
API Controller - Empty
API Controller with read/write actions
API Controller with actions, using Entity Framework
API Controller - Empty
by Microsoft
v1.0.0.0
An empty API controller.
Id: ApiControllerEmptyScaffolder
Add
Cancel


Add New Item - Practical 8
Installed
C#
General
ASP.NET Core
Code
Data
General
Web
CSharp
Online
Sort by: Default
Search (Ctrl+E)
Class C#
Interface C#
Razor Component C#
MVC Controller - Empty C#
MVC Controller with read/write actions C#
API Controller - Empty C#
API Controller with read/write actions C#
Razor Page - Empty C#
Razor View - Empty C#
Razor Layout C#
Assembly Information File C#
Code File C#
Machine Learning Model (ML.NET) C#
Razor View Start C#
Type: C#
Empty API Controller Class
Name: ProductController.cs
Add
Cancel

15. . First, add a product collection (hardcoded) in the controller as shown below.

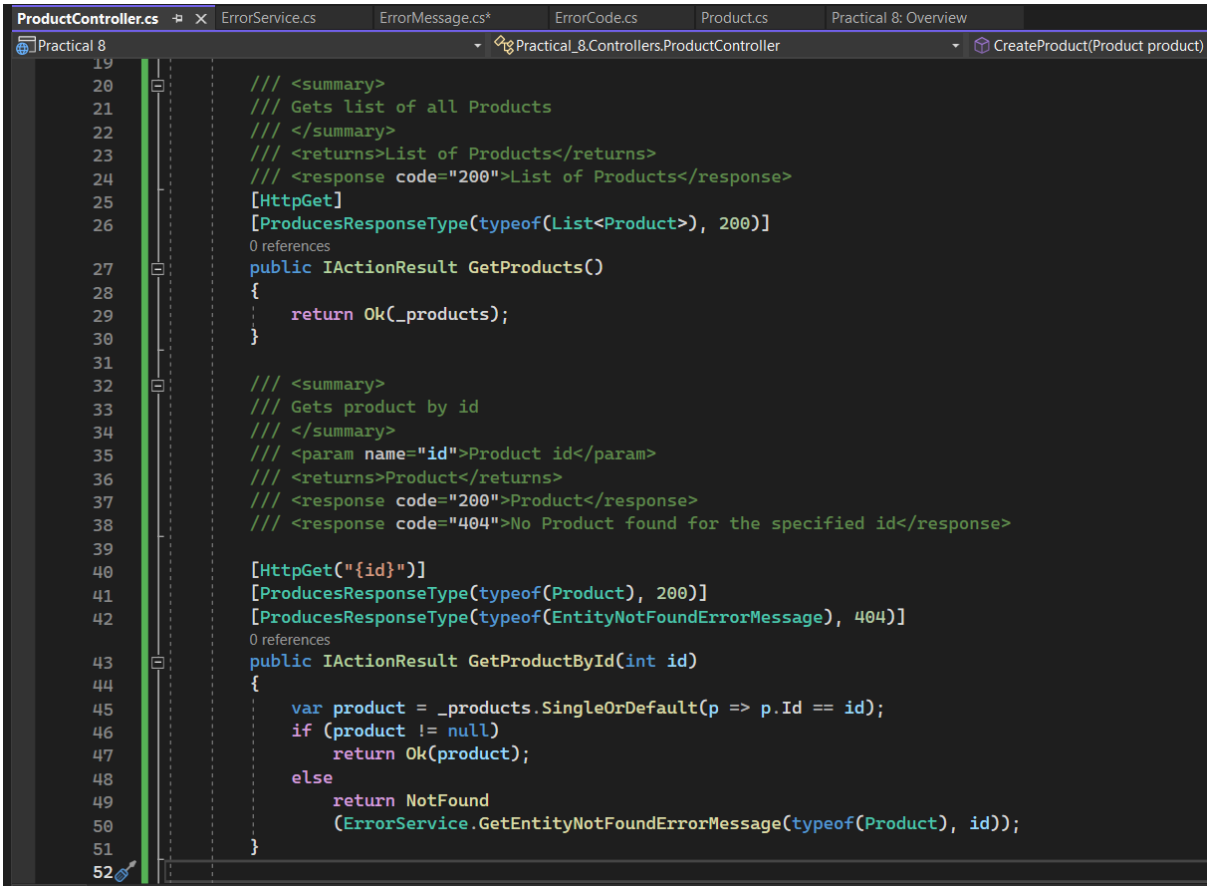
```
[Produces("application/json")]
[Route("api/products")]
public class ProductController : Controller
{
        private static List<Product> _products = new List<Product>
    {
                new Product {Id = 1, Name = "Lithim L2", ModifiedDate = DateTime.UtcNow.AddDays(-2)},
        new Product {Id = 2, Name = "SNU 61" }
     };
```



16. Next, let's add **two HTTP GET** actions: one to retrieve all the products, and another to retrieve a product by its ID, parameter is passed via URI path.

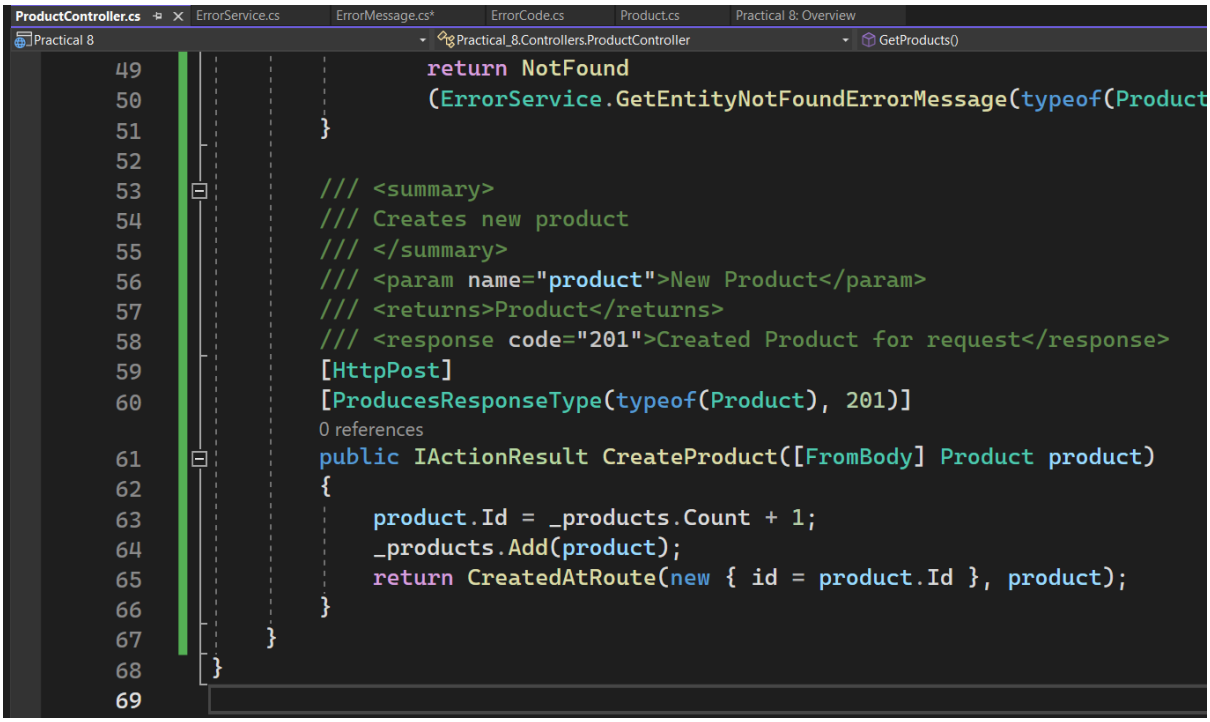
```
/// <summary>
/// Gets list of all Products
/// </summary>
/// <returns>List of Products</returns>
/// <response code="200">List of Products</response>
[HttpGet]
[ProducesResponseType(typeof(List<Product>), 200)]
public IActionResult GetProducts()
{
      return Ok(_products);
}

/// <summary>
/// Gets product by id
/// </summary>
/// <param name="id">Product id</param>
/// <returns>Product</returns>
/// <response code="200">Product</response>
/// <response code="404">No Product found for the specified id</response>
[HttpGet("{id}")]
[ProducesResponseType(typeof(Product), 200)]
[ProducesResponseType(typeof(EntityNotFoundErrorMessage), 404)]
public IActionResult GetProductById(int id)
{
      var product = _products.SingleOrDefault(p => p.Id == id);
```

```
        if (product != null)
                return Ok(product);
        else
                return NotFound
                (ErrorService.GetEntityNotFoundErrorMessage(typeof(Product), id));
}
```



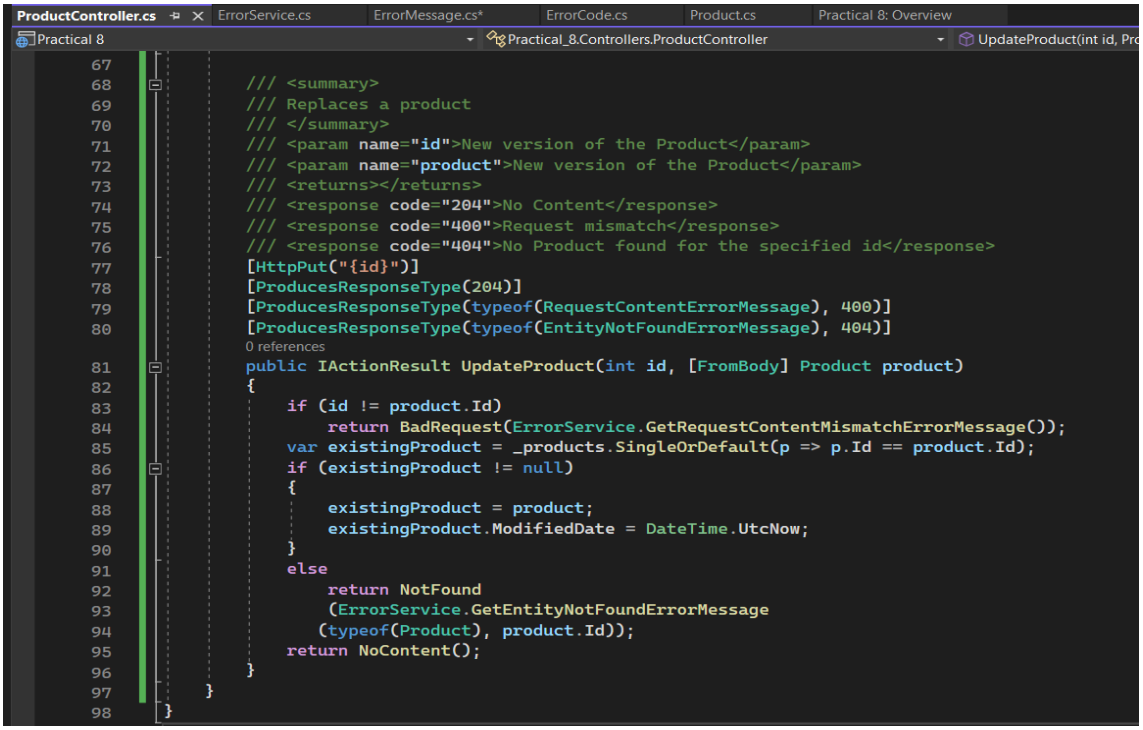
17. Add an action to create new products. **HTTP POST** creates a new product, taking the parameter in the request body.

```
    /// <summary>
/// Creates new product
/// </summary>
/// <param name="product">New Product</param>
/// <returns>Product</returns>
/// <response code="201">Created Product for request</response>
[HttpPost]
[ProducesResponseType(typeof(Product), 201)]
public IActionResult CreateProduct([FromBody] Product product)
{
   product.Id = _products.Count + 1;
   _products.Add(product);
   return CreatedAtRoute(new { id = product.Id }, product);
 }
```



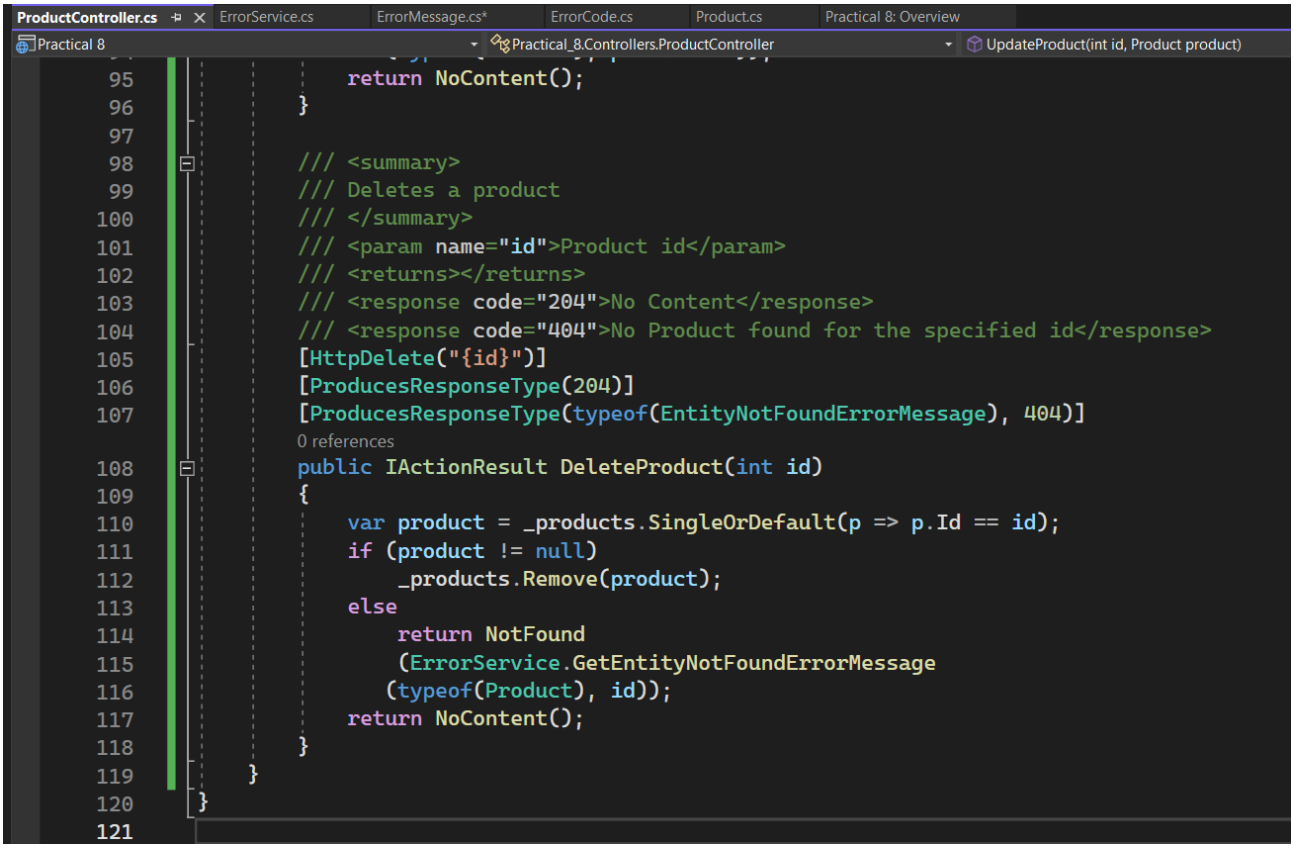
18. Add the **PUT** method for replacing products with the specified ID.

```
    /// <summary>
/// Replaces a product
/// </summary>
/// <param name="id">New version of the Product</param>
/// <param name="product">New version of the Product</param>
/// <returns></returns>
/// <response code="204">No Content</response>
/// <response code="400">Request mismatch</response>
/// <response code="404">No Product found for the specified id</response>
[HttpPut("{id}")]
[ProducesResponseType(204)]
[ProducesResponseType(typeof(RequestContentErrorMessage), 400)]
[ProducesResponseType(typeof(EntityNotFoundErrorMessage), 404)]
public IActionResult UpdateProduct(int id, [FromBody] Product product)
{
   if (id != product.Id)
                  return BadRequest(ErrorService.GetRequestContentMismatchErrorMessage());
   var existingProduct = _products.SingleOrDefault(p => p.Id == product.Id);
   if (existingProduct != null)
   {
      existingProduct = product;
      existingProduct.ModifiedDate = DateTime.UtcNow;
   }
   else
      return NotFound
      (ErrorService.GetEntityNotFoundErrorMessage
     (typeof(Product), product.Id));
   return NoContent();
 }
```



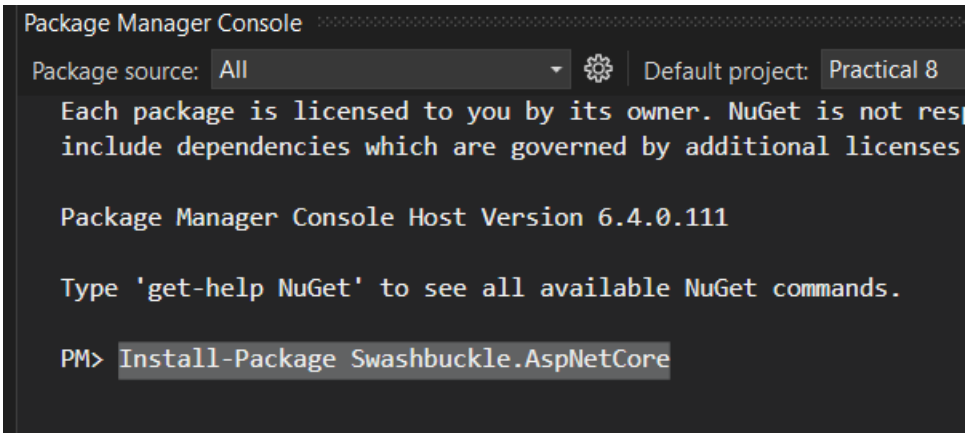
19. Add a delete endpoint using the **HTTP DELETE** action.

```
    /// <summary>
  /// Deletes a product
  /// </summary>
  /// <param name="id">Product id</param>
  /// <returns></returns>
  /// <response code="204">No Content</response>
  /// <response code="404">No Product found for the specified id</response>
  [HttpDelete("{id}")]
  [ProducesResponseType(204)]
  [ProducesResponseType(typeof(EntityNotFoundErrorMessage), 404)]
  public IActionResult DeleteProduct(int id)
  {
     var product = _products.SingleOrDefault(p => p.Id == id);
     if (product != null)
        _products.Remove(product);
     else
        return NotFound
       (ErrorService.GetEntityNotFoundErrorMessage
      (typeof(Product), id));
     return NoContent();
   }
}
```



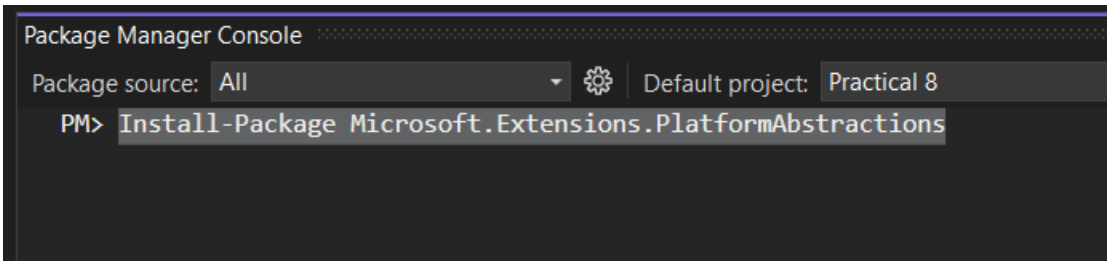
20. Install the package **Swashbuckle.AspNetCore** in the project by executing the following command in the **Package Manager Console (PMC)**:

Install-Package Swashbuckle.AspNetCore



21. In the sample, in order to provide the XML path, install the package **Microsoft.Extensions.PlatformAbstractions** using **PMC**. Execute the following:

Install-Package Microsoft.Extensions.PlatformAbstractions



22. In the **Startup.cs**, update the **ConfigureServices** method, as shown below by Adding the following code:

```
public void ConfigureServices(IServiceCollection services)
{
   services.AddControllers();
   services.AddMvc(options => options.EnableEndpointRouting = false);
   services.AddSwaggerGen(c =>
   {
                        c.SwaggerDoc("v1", new OpenApiInfo { Title = "MassRover API",     Version = "v1" });
                        c.IncludeXmlComments(Path.Combine(PlatformServices.Default.Application.ApplicationBas
                        ePath,"Practical 8.xml"));

   });
}
```

```
Practical 8 | Startup.cs | ProductController.cs | ErrorService.cs | ErrorMessage.cs | ErrorCode.cs | Product.cs | Practical 8: Overview | WeatherFore
Practical 8 | Practical_8.Startup | ConfigureServices(IServiceCollection services)

21      0 references
        public Startup(IConfiguration configuration)
22      {
23          Configuration = configuration;
24      }
25
        1 reference
26      public IConfiguration Configuration { get; }
27
28      // This method gets called by the runtime. Use this method to add services to the container.
        0 references
29      public void ConfigureServices(IServiceCollection services)
30      {
31          services.AddControllers();
32          services.AddMvc(options => options.EnableEndpointRouting = false);
33          services.AddSwaggerGen(c =>
34          {
35              c.SwaggerDoc("v1", new OpenApiInfo { Title = "MassRover API", Version = "v1" });
36
37              c.IncludeXmlComments(Path.Combine(PlatformServices.Default.Application.ApplicationBasePath,"Practical 8.xml"));
38          });
39      }
```

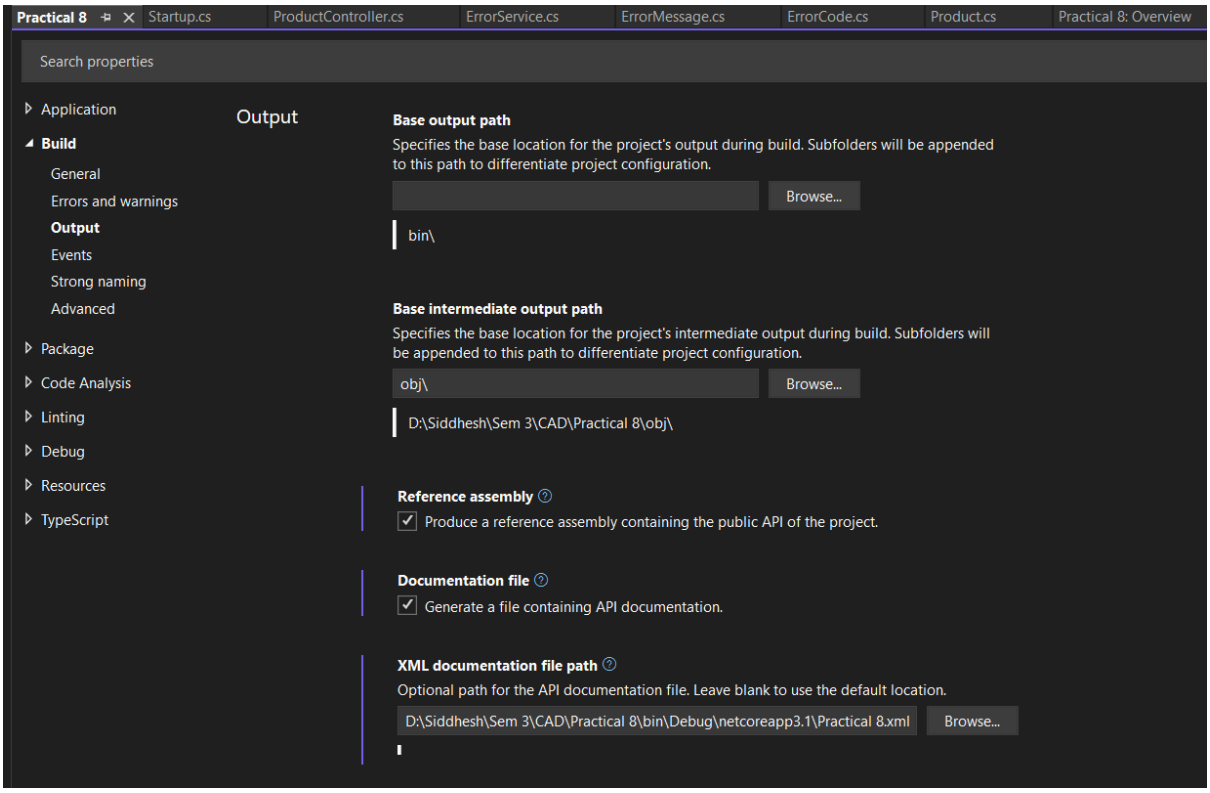
23. Inside **Startup.cs** file, Update the **Configure** method as shown below. This enables the Swagger UI and sets the Swagger definition endpoint.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseSwagger();
    app.UseSwaggerUI(s =>
    {
                s.SwaggerEndpoint("/swagger/v1/swagger.json","MassRover Open                API");
    });

    app.UseMvc();
```

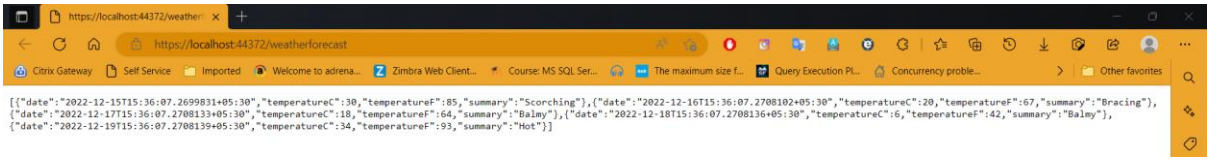
```
Startup.cs* | ProductController.cs | ErrorService.cs | ErrorMessage.cs | ErrorCode.cs | Product.cs | Practical 8
Practical 8 | Practical_8.Startup

41          // This method gets called by the runtime. Use this method to configur
            0 references
42          public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
43          {
44              if (env.IsDevelopment())
45              {
46                  app.UseDeveloperExceptionPage();
47              }
48
49              app.UseSwagger();
50              app.UseSwaggerUI(s =>
51              {
52                  s.SwaggerEndpoint("/swagger/v1/swagger.json",
53                "MassRover Open API");
54              });
55
56              app.UseMvc();
```

24. Finally, we will instruct Visual Studio to generate **XML documentation** based on the **comments**. Navigate to **Project Properties**, then the **Build tab**, and enable the **XML documentation file**. Give it the path of your **'.XML'** file which will be inside your project path i.e.

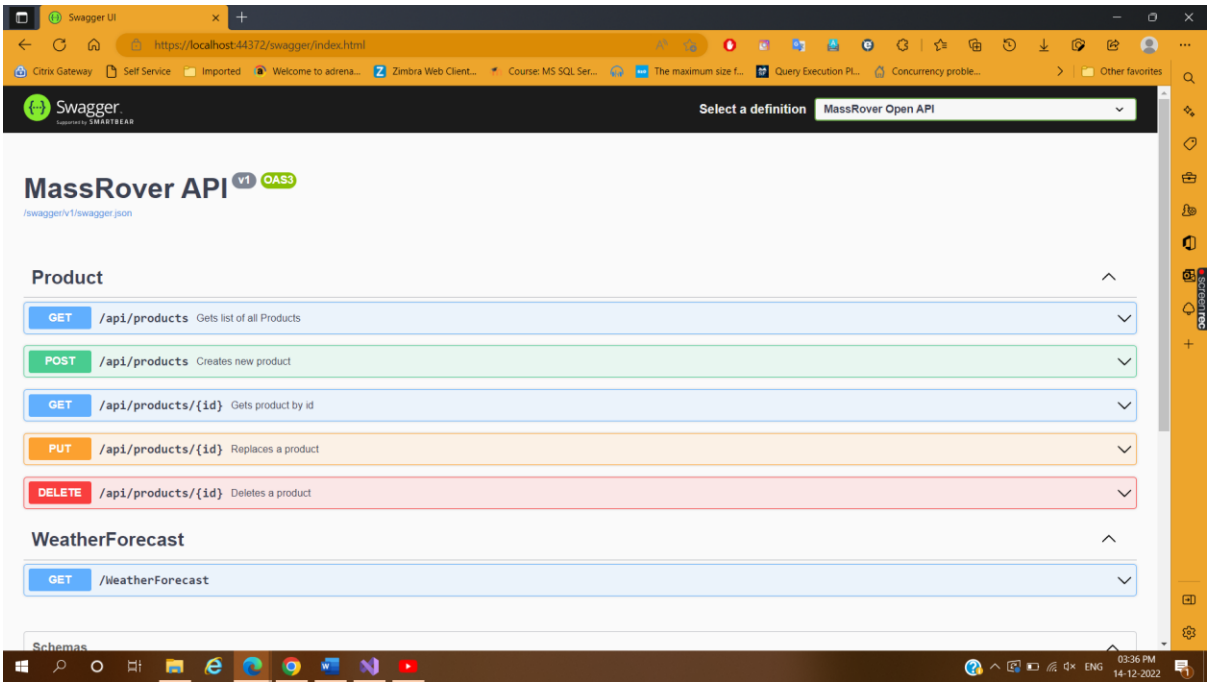
….\Project\bin\Debug\netcoreapp3.1\Project.xml



25. Now you can **Run** the application.



26. Navigate to the **URL**:
http://{host}:{specified port}/swagger.
In This case, **http://localhost:44372/swagger**



https://stackoverflow.com/questions/21175781/a-namespace-cannot-directly-contain-members-such-as-fields-or-methods - It Shlould be Inside Class

https://stackoverflow.com/questions/56502869/info-class-not-recognized-with-swashbuckle-and-swagger-libraries - It has been replaced with OpenApiInfo so you have to reflect that in your swagger setup:

```
c.SwaggerDoc("v1", new OpenApiInfo { Version = "1.0", Title = "My API" });
```

https://stackoverflow.com/questions/57684093/using-usemvc-to-configure-mvc-is-not-supported-while-using-endpoint-routing - 3. Disable endpoint Routing. As the exception message suggests and as mentioned in the following section of documentation: use mvcwithout endpoint routing

```
services.AddMvc(options => options.EnableEndpointRouting = false);
//OR
services.AddControllers(options => options.EnableEndpointRouting = false);
```