1. We must now convert the "IPostRepository" interface into a class in order to use it.
2. In order to do this, we must first build a new class in the repository folder named "PostRepository."
3. Simply use the same processes that we used to build a new class, and then construct the class by implementing "IPostRepository" as we did before.

using DevopsCore.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace DevOpsCoreV1.Repository

{

public class PostRepository : IPostRepository

{

public List<PostViewModel> GetPostData()

{

var postValues = new List<PostViewModel> {

new PostViewModel(){ iId =001, Name = "Demo User 1"},

new PostViewModel(){ iId =002, Name = "Demo User 2"},

new PostViewModel(){ iId =003, Name = "Demo User 3"},

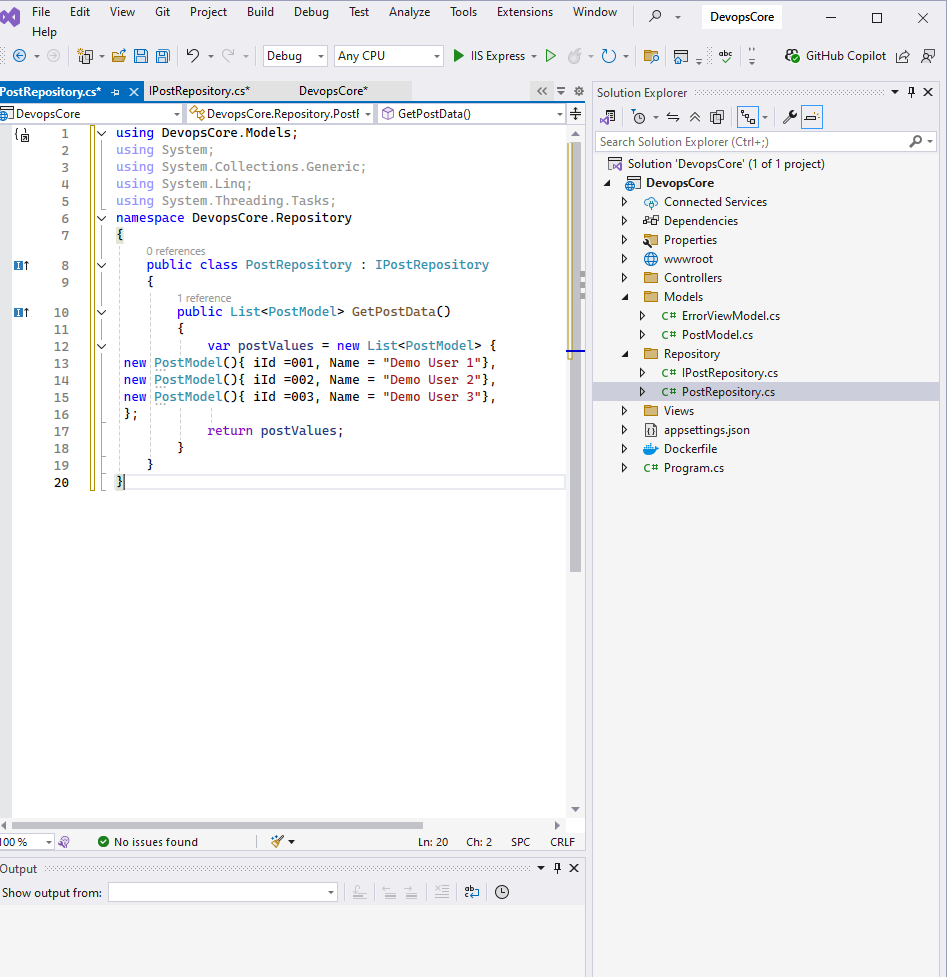
};

return postValues;

}

}

}



1. We should now have a controller class that will retrieve the data from the repository.In the controller folder, we may construct a controller class that we can use.
2. Similarly as previously, right click on the “Controllers” folder and pick Add->Class from the context menu. I'm going to utilize the same Controller class that is already present in the project, which is named "HomeController," for this project, though. You can, on the other hand, construct a new controller and put your implementation in that class if you need to change anything.

using DevopsCore.Models;

using DevopsCore.Repository;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using System.Diagnostics;

namespace DevopsCore.Controllers

{

public class HomeController : Controller

{

private readonly IPostRepository postRepository;

public HomeController()

{

}

public HomeController(IPostRepository \_postRepository)

{

this.postRepository = \_postRepository;

}

public IActionResult Index()

{

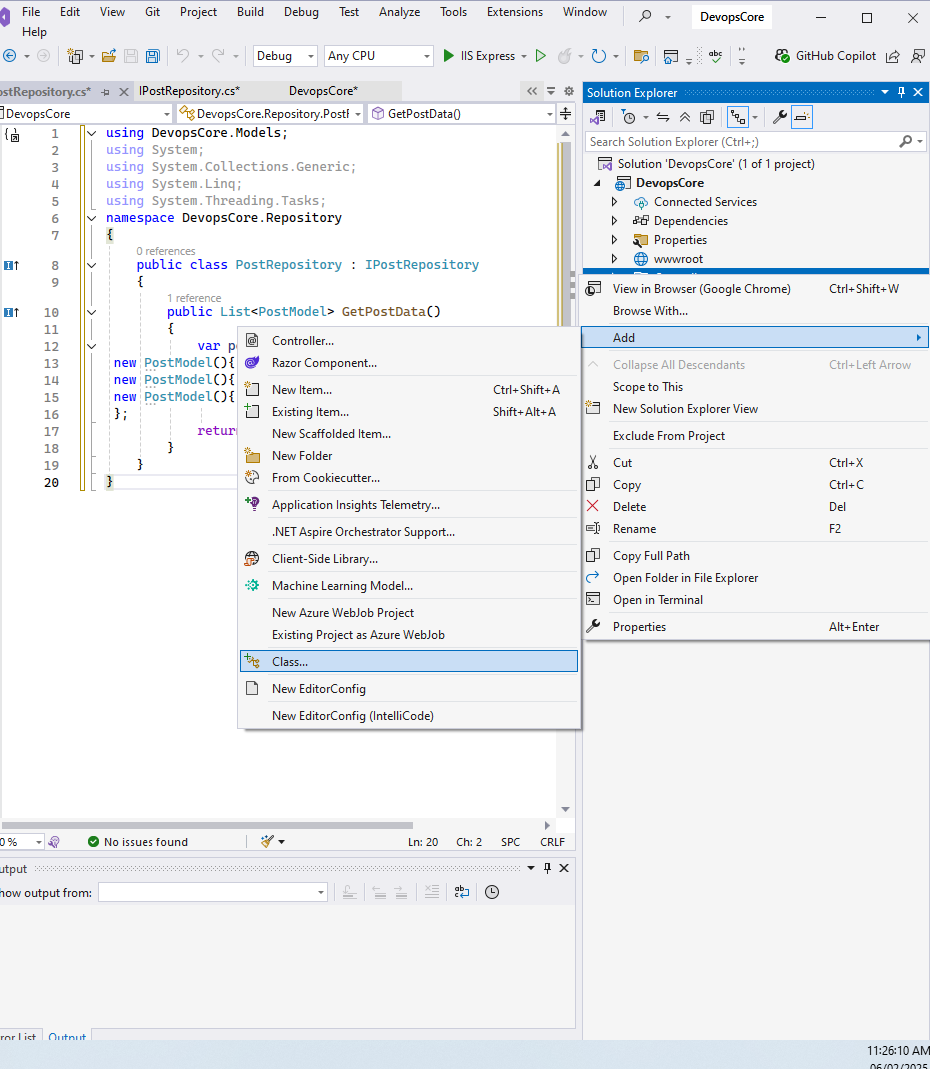
var model = postRepository.GetPostData();

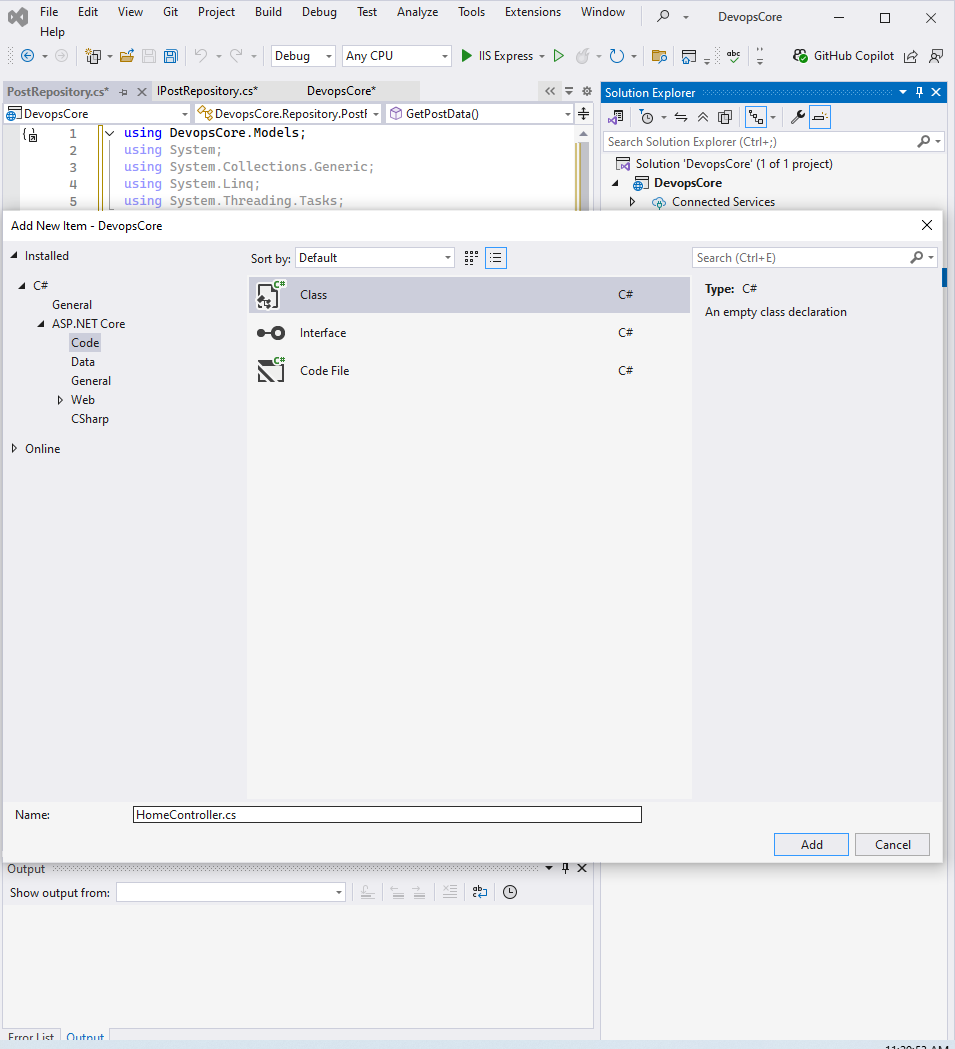
return View(model);

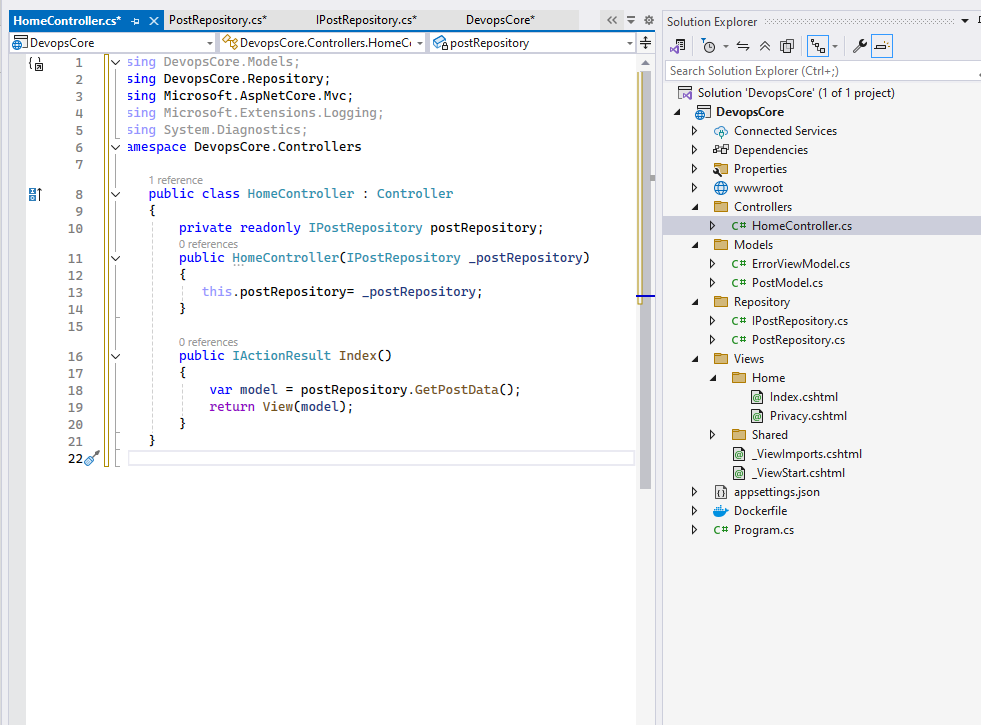
}

}

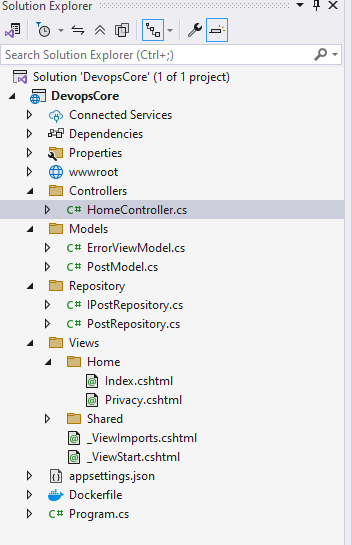
}







1. The home controller is already bound to the index.cshtml class in the Home view folder.



1. Binding the model to the view is what we'll be doing in this stage. As previously noted, HomeController is already tied to the Index.cshtml file located in the home folder of the view folders, which is a good thing. As a result, I'm going to associate the model with the index file. Include the below code to your file.

@model IList<DevopsCore.Models.PostModel>

@{

ViewData["Title"] = "Home Page";

}

<h2>Post List</h2>

<table class="table">

<thead>

<tr>

<th>Id</th>

<th>Name</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@Html.DisplayFor(modelItem => item.iId)</td>

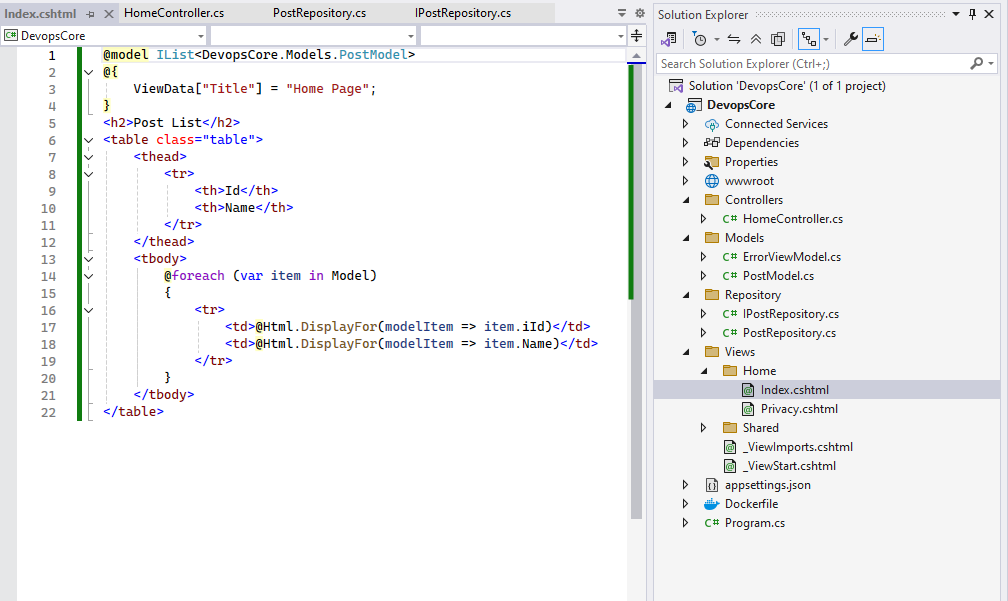
<td>@Html.DisplayFor(modelItem => item.Name)</td>

</tr>

}

</tbody>

</table>



1. On your “C#Program.cs” type this code:

using DevopsCore.Repository;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

builder.Services.AddScoped<IPostRepository, PostRepository>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

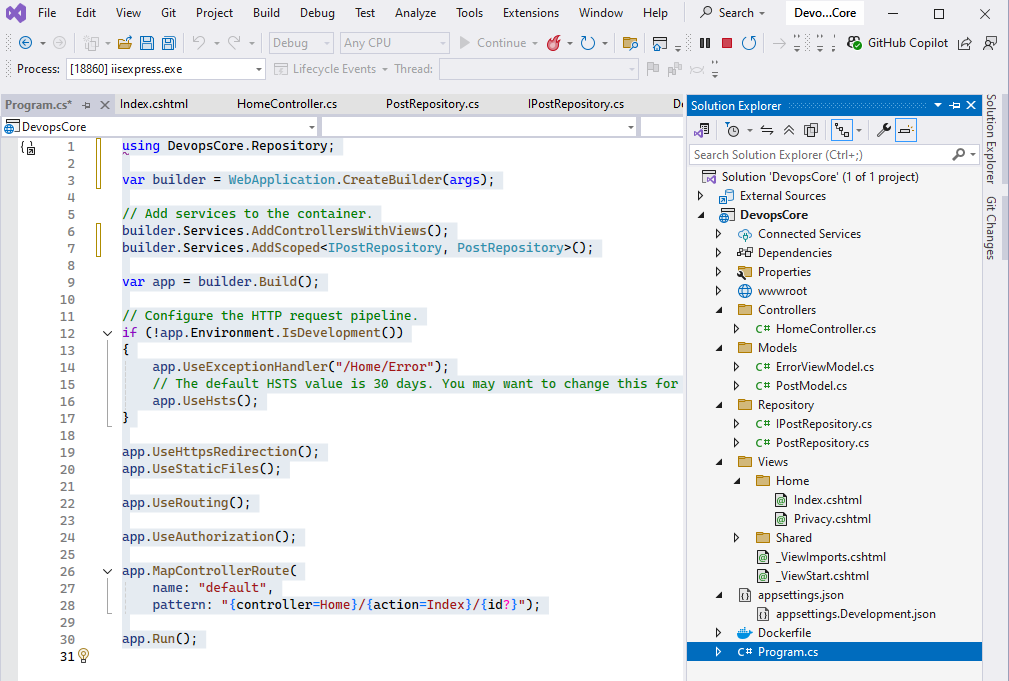
app.UseAuthorization();

app.MapControllerRoute(

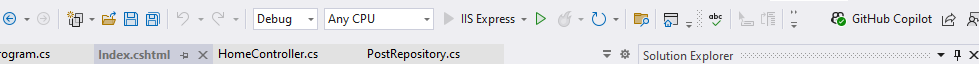
name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

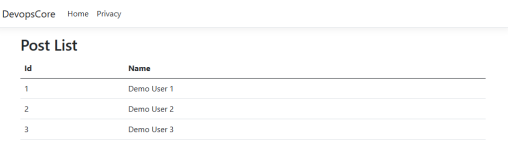
app.Run();



1. Run the code by clicking “IIS Express” and then your project has been completed.

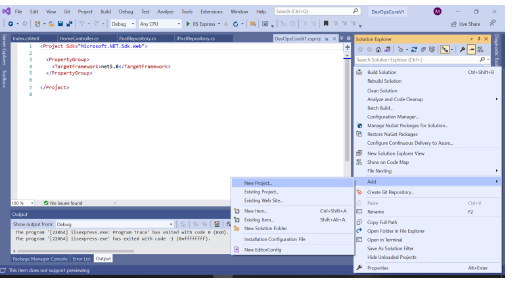


1. This will be you output.

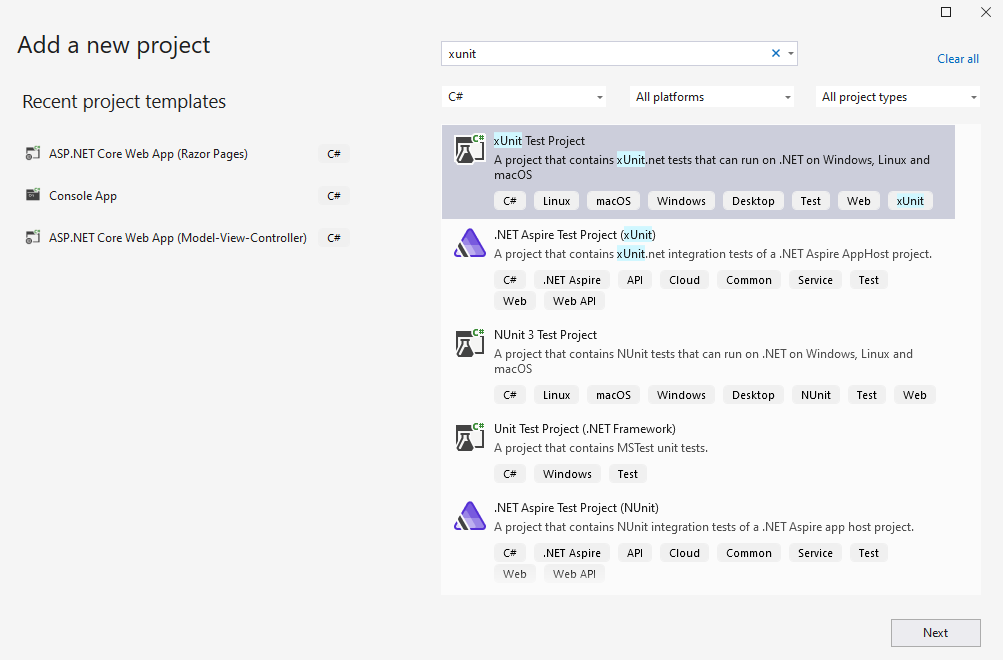


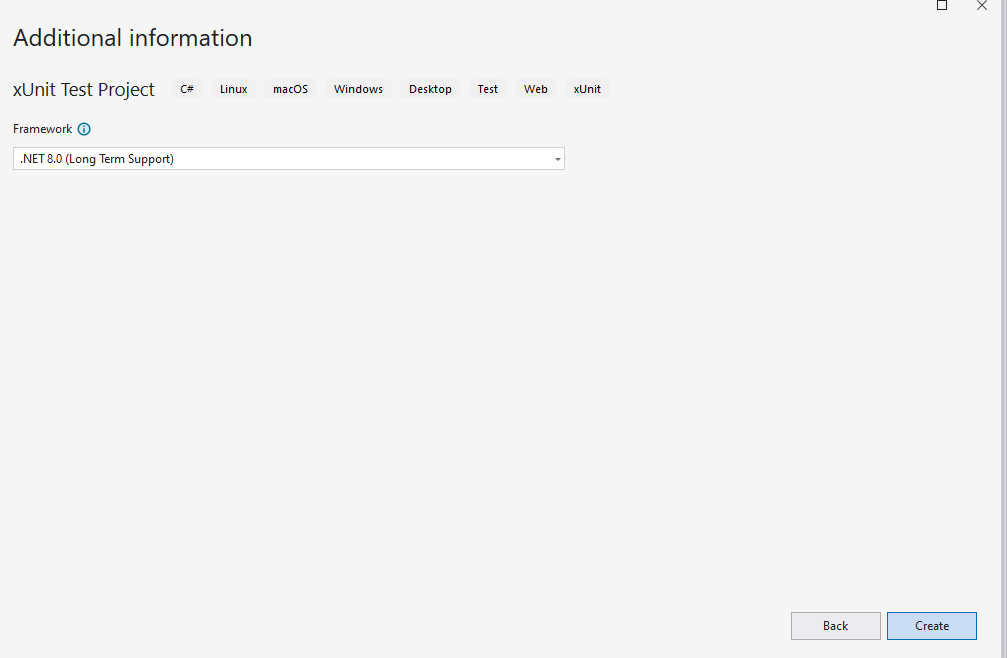
**4.4 Add Unit Test Project**

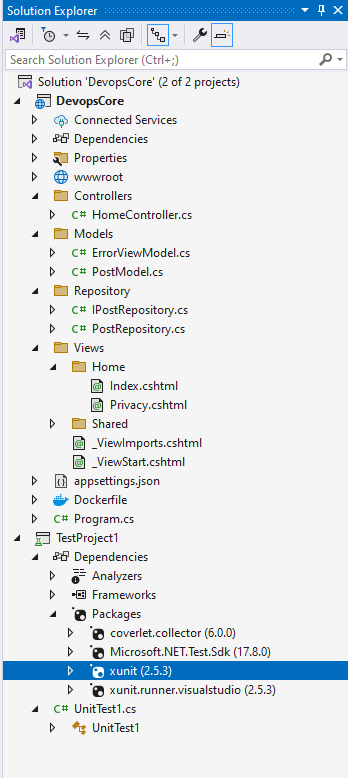
1. Unit testing is a critical component of the software development process. We can put our system into a production environment without having to test it before hand.Testing our functionality would be easier if we utilize a basic piece of code. However,apart from unit testing, there are a plethora of other approaches to test our product.
2. Unit tests allow us to determine whether or not our code is functioning properly. It also aids in the discovery of faults or defects in our code before it is released to the public.
3. We have a test project for our project that we use for testing reasons. We'll be working on a "xUnit" test project for the time being. Let's see how we may include a test project into our existing project to do this.
4. If the code running means stop it.
5. Right-click on your Project solution (Ex:Solution ‘DevopsCore’) and select "Add"->"New Project".



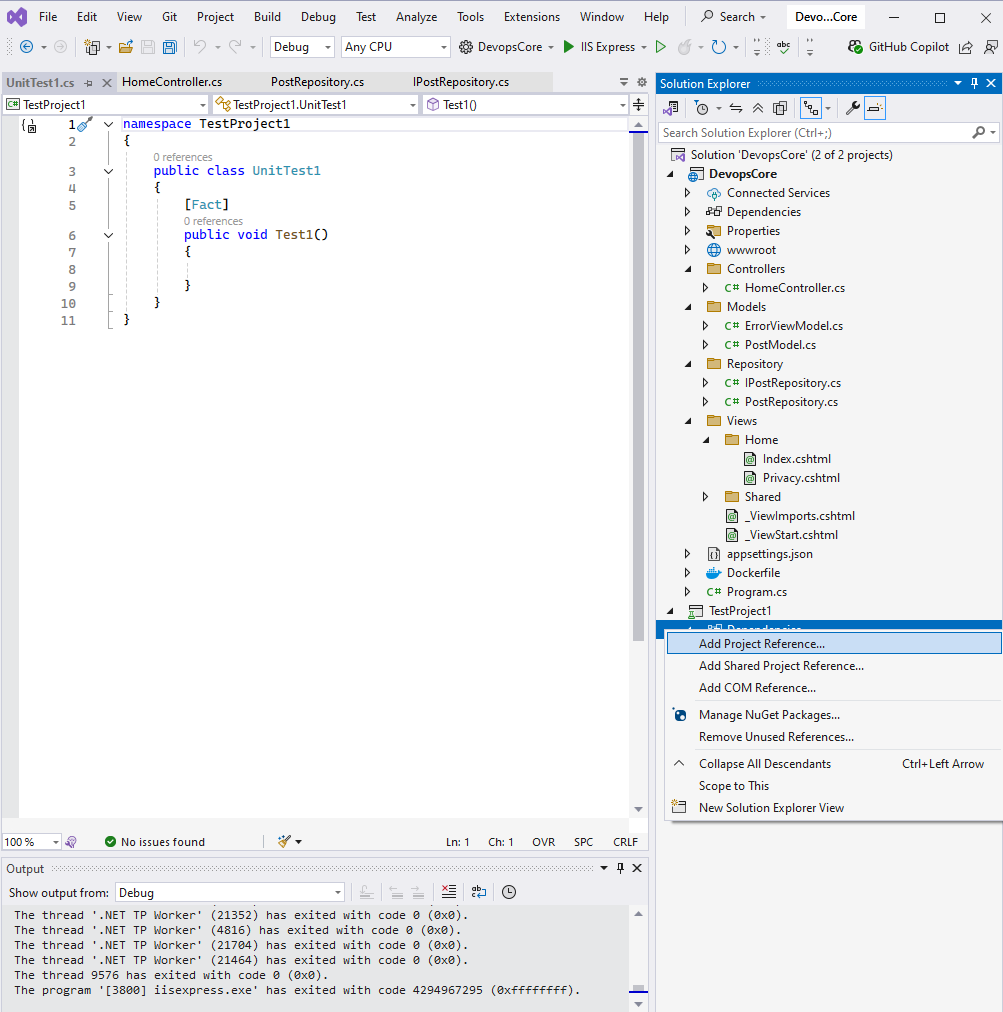
1. There is a search option available here, and you can just search for xunit from there to get the xunit option for the c#.Then, select “xUnit Test Project “ , click “Next” and click create.

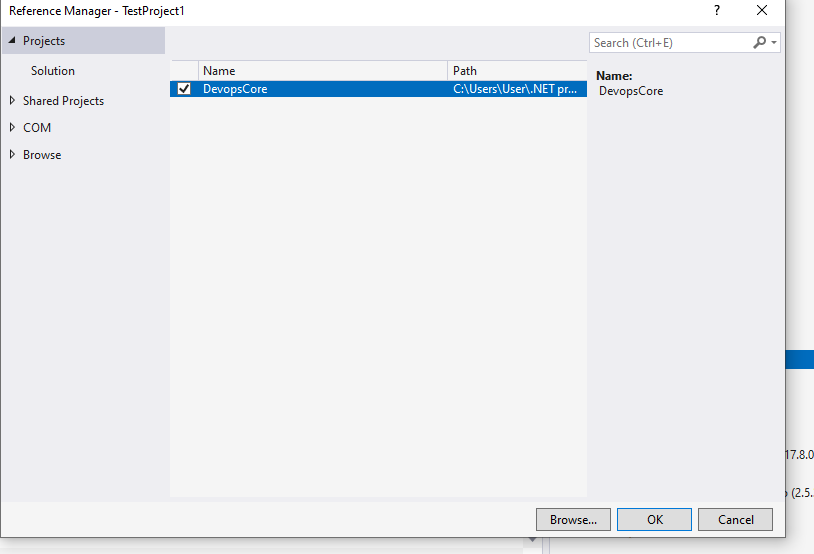






1. These test classes will be written for our current project, which is currently under development. As a result, we must include our project in our unit test class.
2. To do so,right-click on the dependencies in the test project and choose Add Project Reference from the context menu. All of the projects that we currently have open in solution explorer will be listed in a new window.
3. Select the project to which you want to add the test project, check the box next to it, and then click OK.





1. Now we can modify the test class and use it to validate our app.
2. Lets add below code to your unit test class in test project:

using DevopsCore.Controllers;

using DevopsCore.Repository;

using DevopsCore.Models;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using Xunit;

namespace TestProject1

{

public class UnitTest1

{

private PostRepository repository;

public UnitTest1()

{

repository = new PostRepository();

}

[Fact]

public void Test\_Index\_View\_Result()

{

//Arrange

//var controller = new HomeController(this.repository);

var controller = new HomeController();

//Act

var result = controller.Index();

//Assert

Assert.IsType<ViewResult>(result);

}

[Fact]

public void Test\_Index\_Return\_Result()

{

//Arrange

//var controller = new HomeController(this.repository);

var controller = new HomeController();

//Act

var result = controller.Index();

//Assert

Assert.NotNull(result);

}

[Fact]

public void Test\_Index\_GetPosts\_MatchData()

{

//Arrange

// var controller = new HomeController(this.repository);

var controller = new HomeController();

//Act

var result = controller.Index();

//Assert

var viewResult = Assert.IsType<ViewResult>(result);

var model =

Assert.IsAssignableFrom<List<PostModel>>(viewResult.ViewData.Model);

Assert.Equal(3, model.Count);

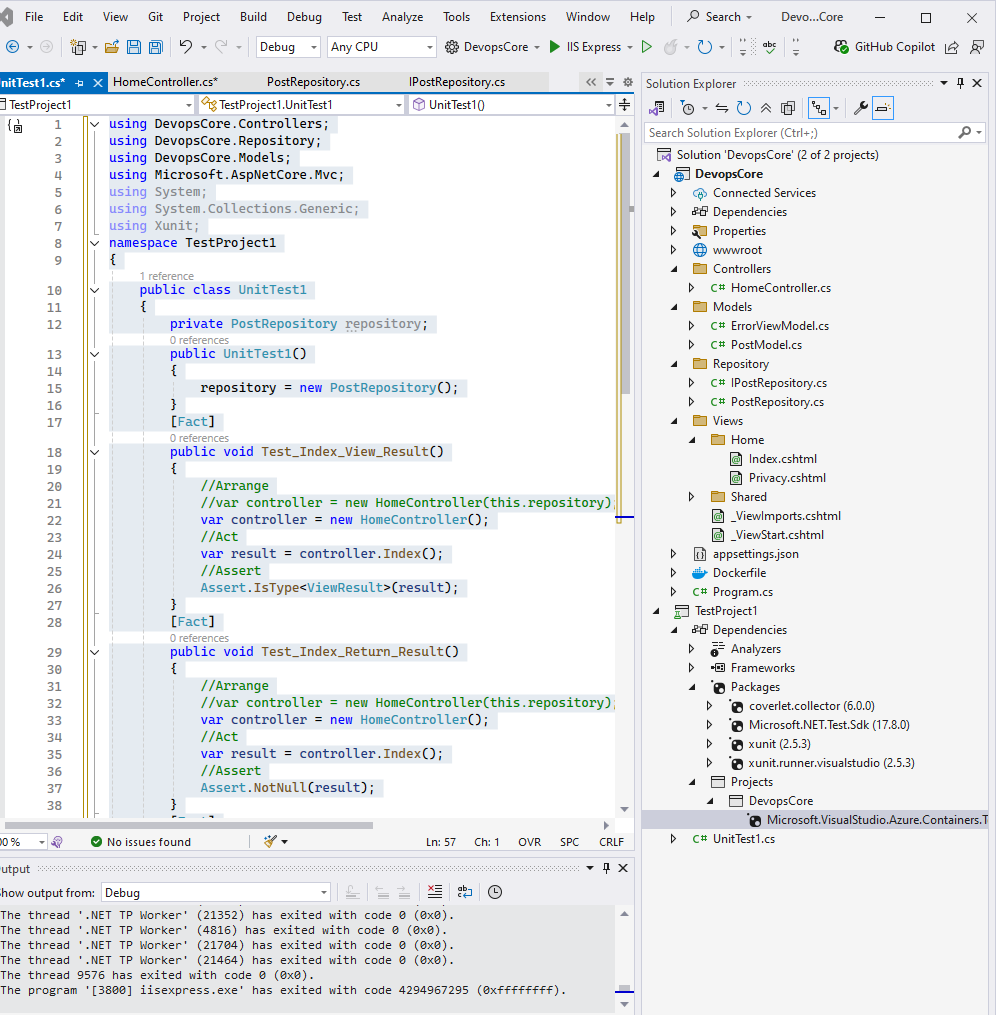
Assert.Equal(101, model[0].iId);

Assert.Equal("DevOps Demo Title 1", model[0].Name);

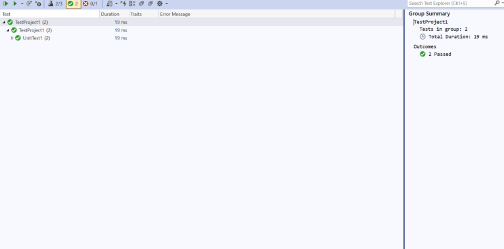
}

}

}

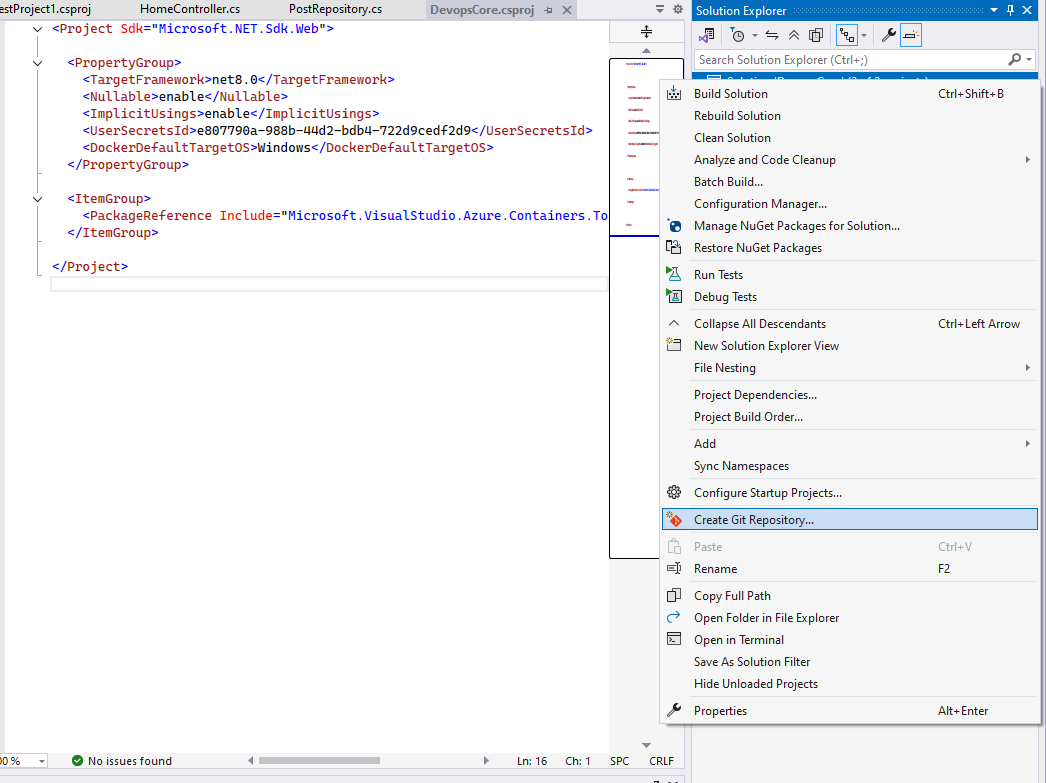


1. You can launch your unit test project by right-clicking on “TestProject1” it and selecting "run test,"which will cause the code to be executed.
2. The final result will be presented in the test explorer when all unit cases have been completed

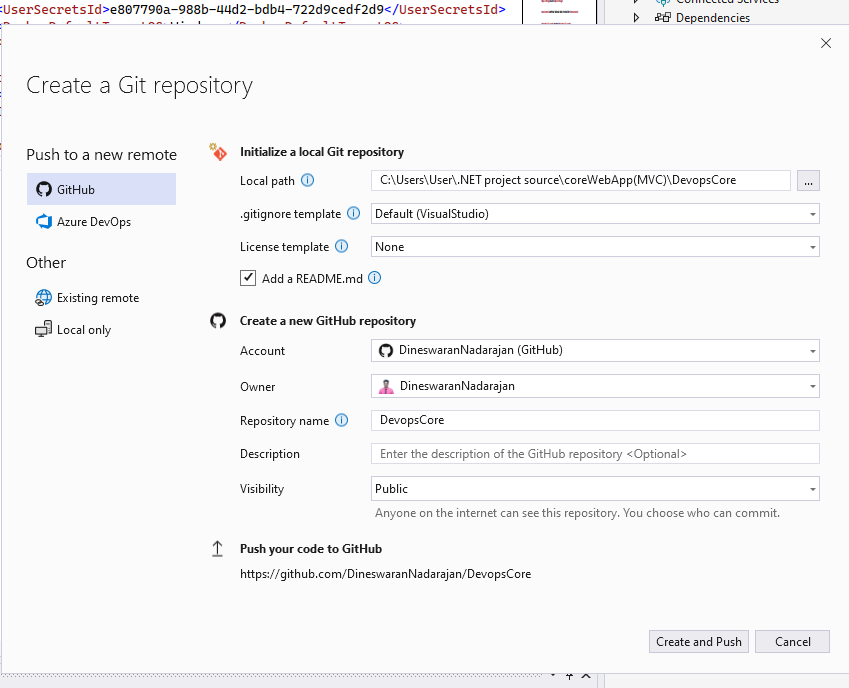


**4.5 Add Project to the GitHub**

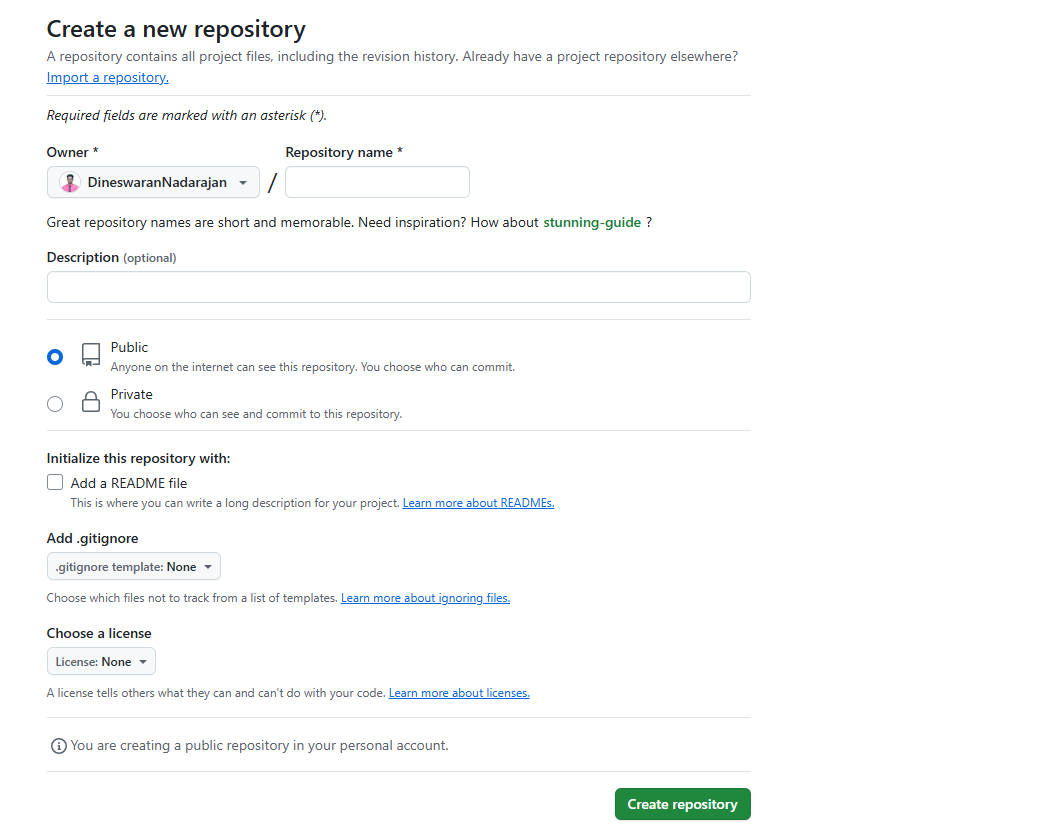
1. You may upload your project to any repository, such as Bitbucket or GitHub, and you can access it from Azure DevOps at any time. However, for the time being, I'll be using GitHub to upload my project
2. First and foremost, if you do not already have a GitHub account, you must establish one right now.There is few option have submit your code in github.
3. Option 01 : You will discover an option named "Create Git Repository" if you right-click on your project solution(Ex:Solution ‘DevopsCore’) and choose it from the context menu. Select that item from the dropdown menu.



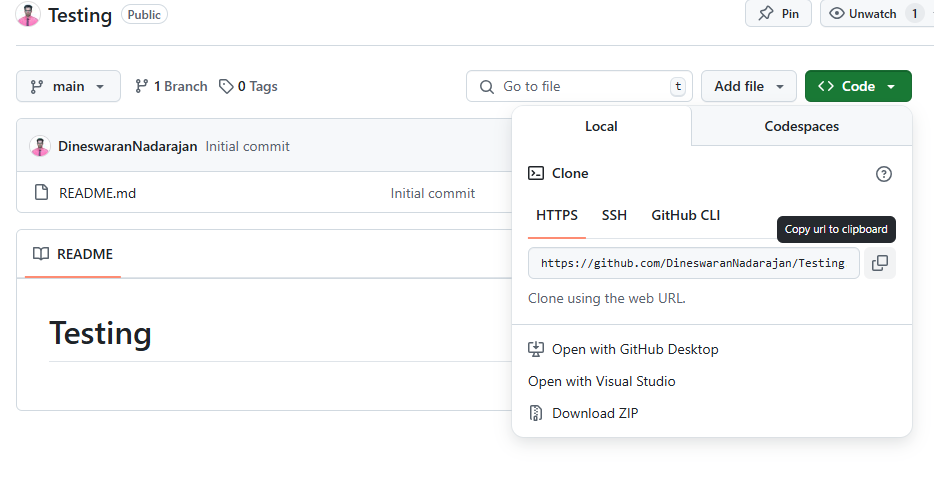
* 1. Choose "GitHub" from the drop-down menu in the next new window.In this window, it will show you the path of your project's local directory. If you choose the incorrect option, you will be able to choose the proper way from there.
  2. Check that the Git account you've chosen is accurate; you may be required to provide your login credentials from this page for verification. The name of the repository will be automatically associated with the name of your project.
  3. They construct your project on GitHub using the repository name as a starting point. If you choose, you may alter the name of your project, and it will show on GitHub under the new name you choose.Make sure that the "Private Repository" option is selected if you wish to keep your project private.
  4. In Git, adding a README.md helps with version control and ensures that anyone who clones or forks the repository can easily understand what the project is about and how to use it.
  5. By adding the README.md to your repository, you also ensure that it will be visible to anyone who views your project on GitHub, GitLab, or other Git platforms.



1. Option 2 :You may do this by creating a new repository on your GitHub account.
   1. It includes all of the same choices that we had before when we were using Visual
   2. Studio for development. You have the option of making your project public or private.
   3. If you choose, you may include a readme file with your project, and you can provide explanations or instructions in that file as well. There is an option called "git ignore" that may be used. You may use this to disregard certain undesired files, such as dlls that have been committed to GitHub.



* 1. Copy the URL that will be used to clone your repository to a local folder.



* 1. Now go to the folder where your project is stored on your computer's hard drive. If you already have Git installed on your computer, you can simply right-click inside of your folder and pick "git clone" or open the git shell and write "git clone URL" to clone the file.
  2. First and foremost, if you have not already done so, you must configure your git hub on your local workstation. Following that, you can add your project to the GitHub selection by choosing the
  3. "Add" and "Push" choices from the drop-down menu.

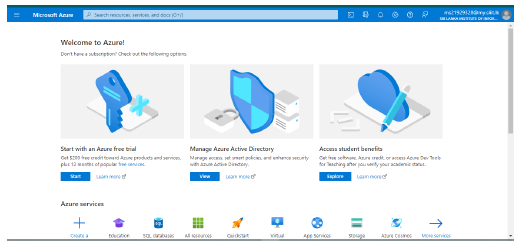
1. Option 3: Using Visual Studio Code

**5.0 CI/CD Pipeine for .NET with Azure Devops**

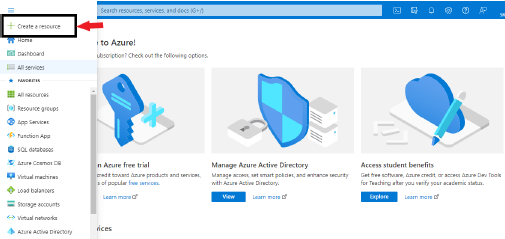
1. Create Organization
2.  Create a new Project
3. Create the .Net Core Pipeline
4. Managing Pipeline using Azure CLI
   1. Run a Pipeline
   2. Update Pipeline
   3. Show Pipeline
5. Update Project Details
6. Add/Update Project Teams
7. Checking and Granting Permissions

**5.1 Setting up a .Net Core application from GitHub**

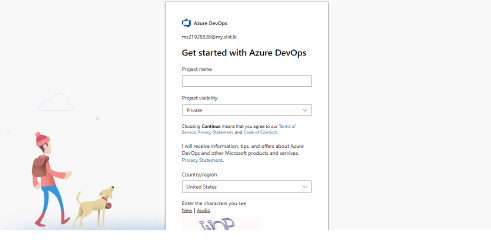
1. Log into your Azure DevOps portal by clicking on the link below.
2. [https://portal.azure.com/](CI CD Azure.docx)



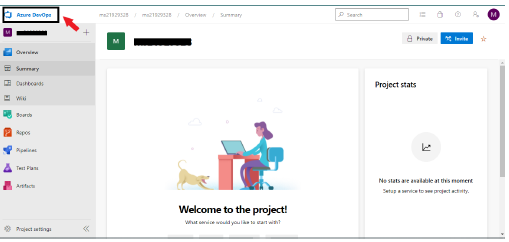
1. Navigate to the navigation bar and click on the '+ Create a resource' button.



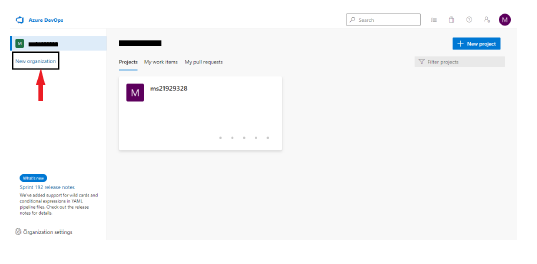
1. In the search box, type "DevOps Services" and press the Enter key to search.If didn’t populate click this link <https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/> . As for now, under Azure Pipeline you can click start free to build pipeline.
2. Please Watch this Video:
   1. <https://www.youtube.com/watch?v=aonA7Kb7WGE> .



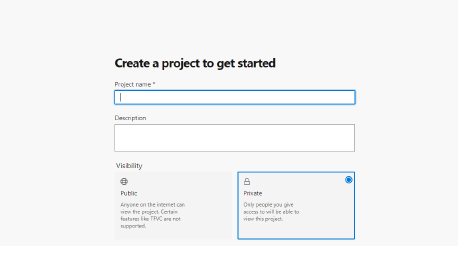
1. Click on the ‘Azure DevOps' button.



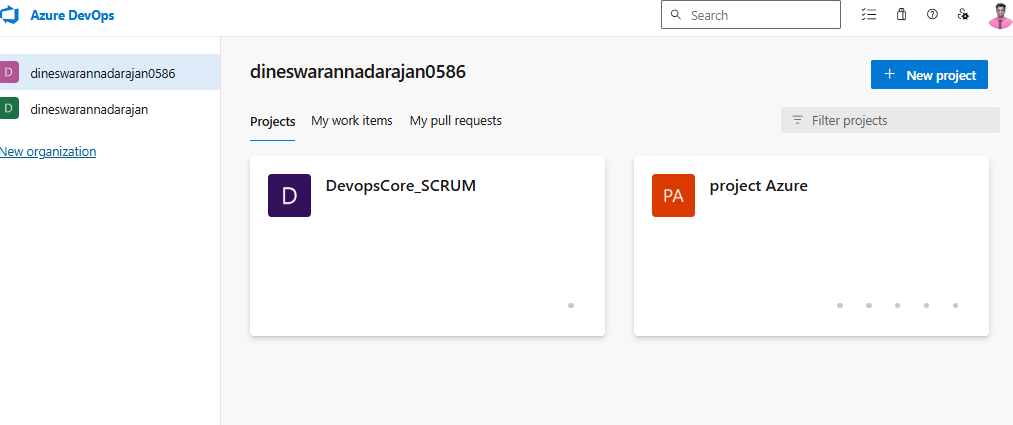
1. There is an organization there that has been formed by the system automatically.
2. However, if you so choose, you may form a new organization.



1. Immediately after the creation of a new organization, you will be routed to the newly established organization. Following the creation of an organization, you will get an email asking you to confirm your participation. Additionally, it will take you to the interface shown below.

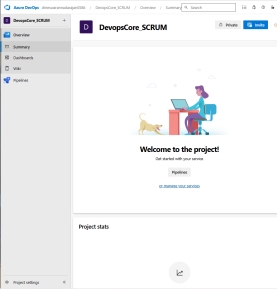


1. Now you can create a new project
   1. Project name: The project's official name is
   2. Description: A brief overview of the project
   3. Visibility: Your project's exposure is important.
   4. From the Advanced menu, choose
   5. Version control: Git
   6. Work item process: Scrum
   7. In this form, the project name is a required field, while the other elements are optional to complete.
   8. In this case, I choose to build a secret project, which means that it will not be available to anybody until I share it.
   9. My version controller will be Git, and my work item process will be scrum. It is entirely up to you the options you want to use for visibility, version control, and work item processing.
   10. Now click on the Create Project
   11. You will be led to the organization's dashboard, which looks like the one shown below.

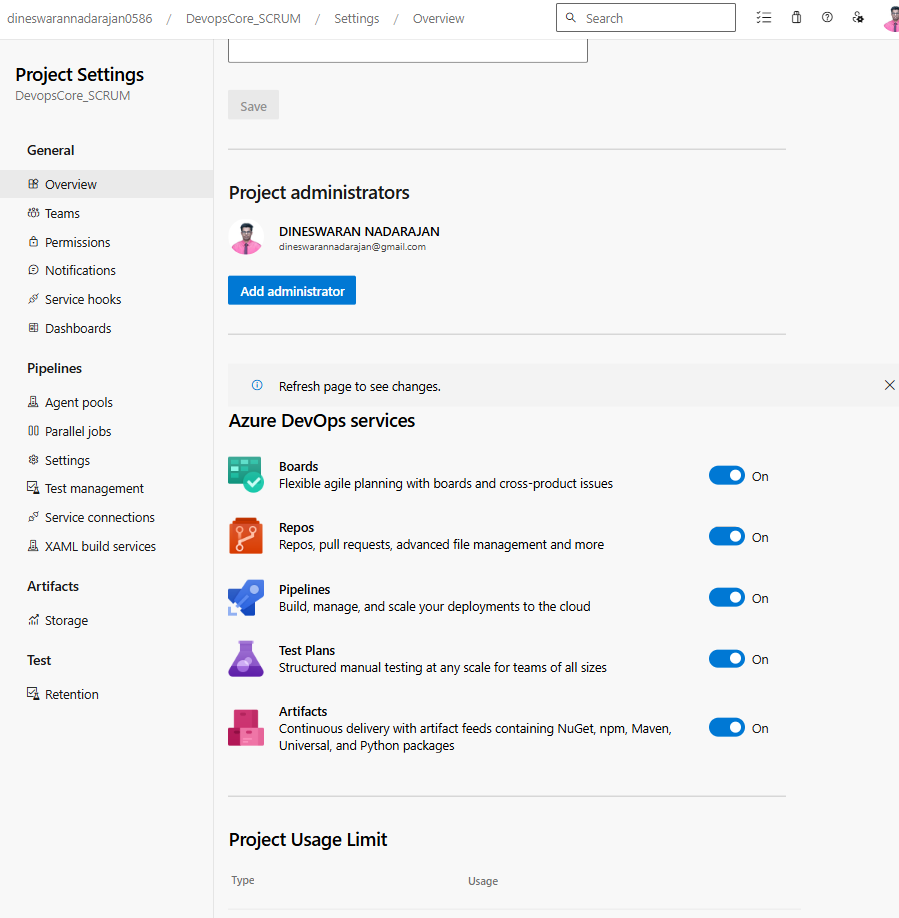


**5.1.1 Develop .Net Core Continuous Integration/Continuous Delivery Pipeline**

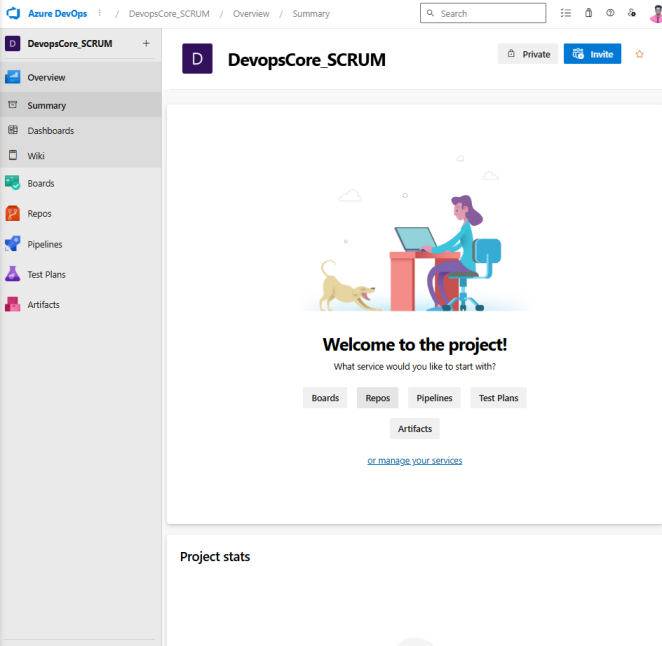
1. On the dashboard, choose the project you want to work on. In my instance, the project name is "DevopsCore\_SCRUM."
2. On the left pannel bottom , select “Project Setting”



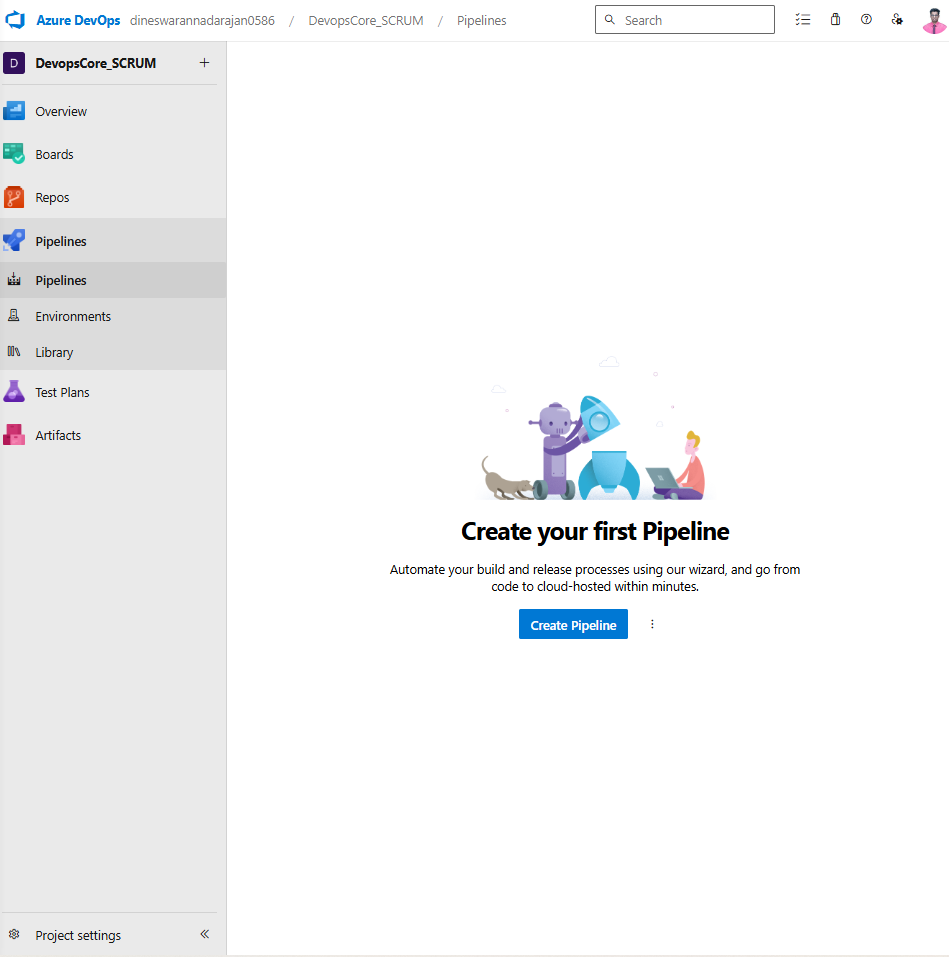
1. Enable all Azure DevOps services.



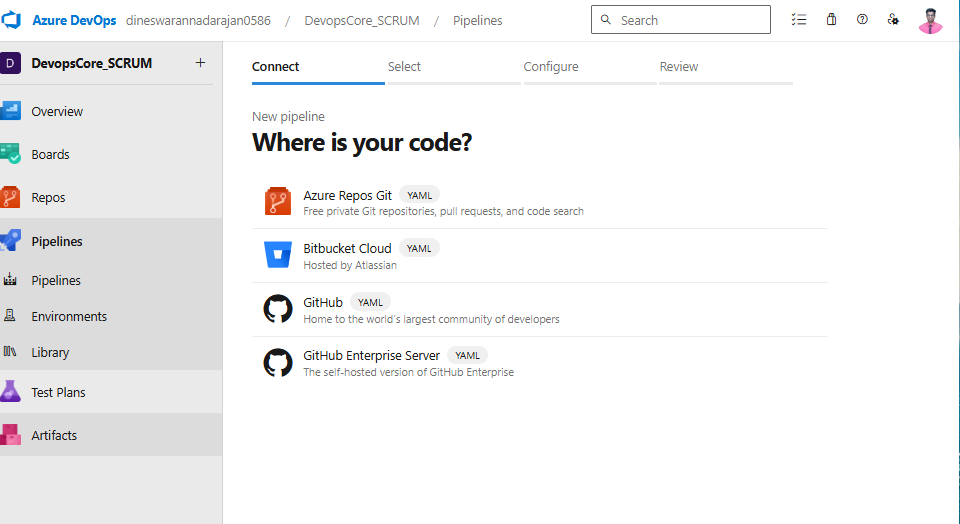
1. To construct a new pipeline, choose "Pipeline" from the drop-down menu in the window



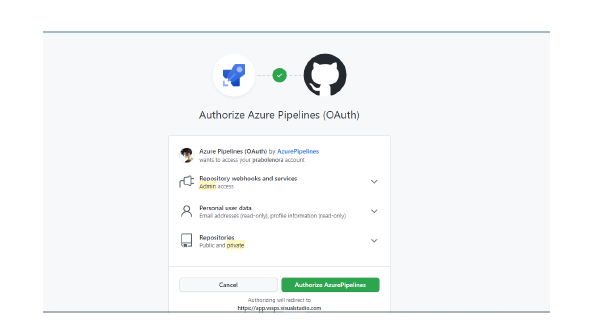
1. Click on the “Create Pipeline”**.**



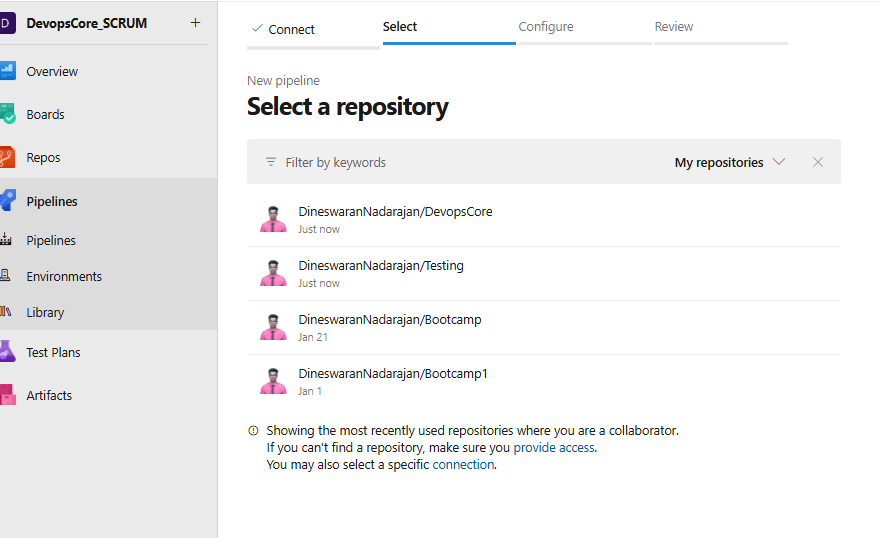
1. Choose GitHub as the repository for your source code.



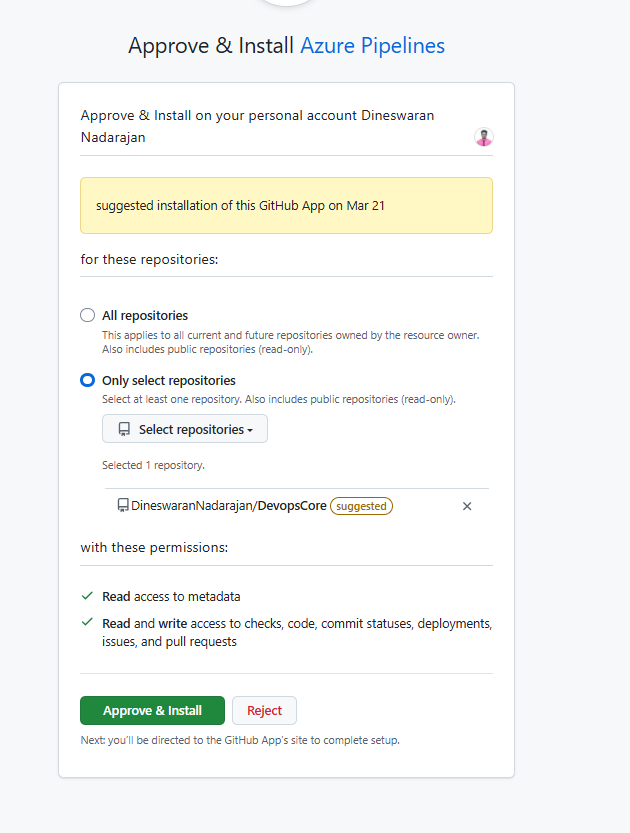
1. It is necessary to upload your .netCore project to your GitHub account before proceeding. Next that, proceed with the following steps.
2. You will be sent to a GitHub login screen after that. As a result, before you can utilize the GitHub Azure pipeline, you must first get permission access to GitHub servers.
3. By using the Azure portal, you may input your GitHub credentials and connect to your account.Then click authorize pipeline.



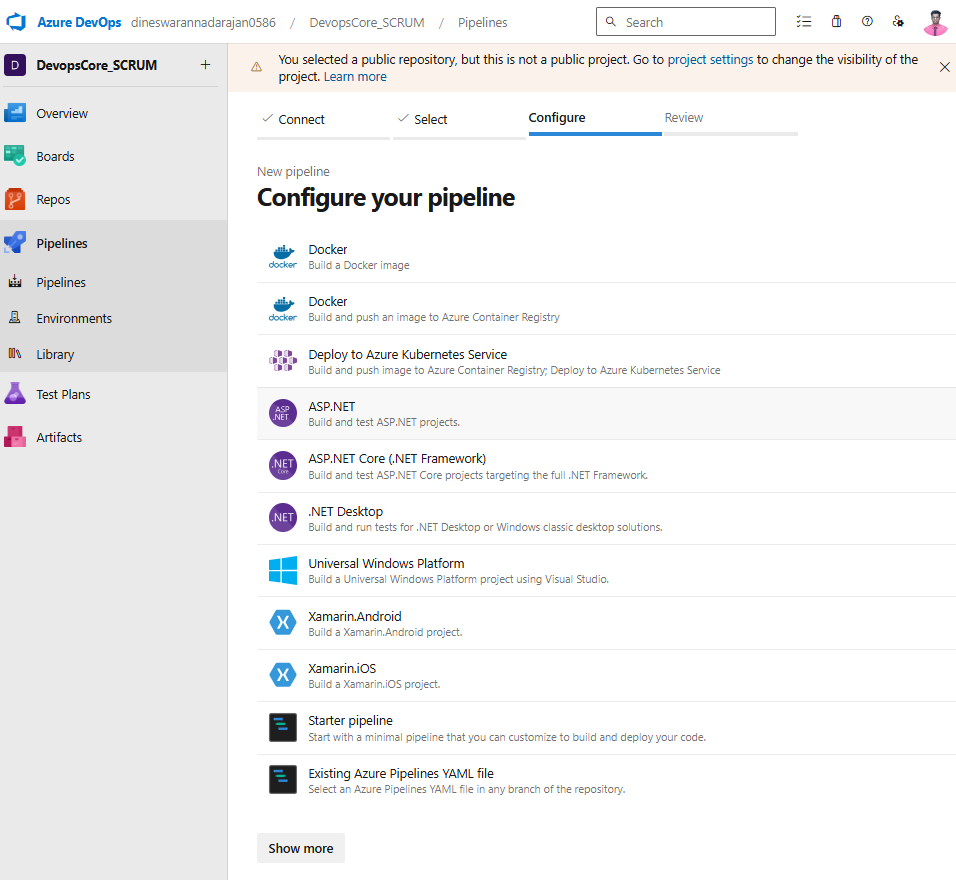
1. Select your project from GitHub repositories.



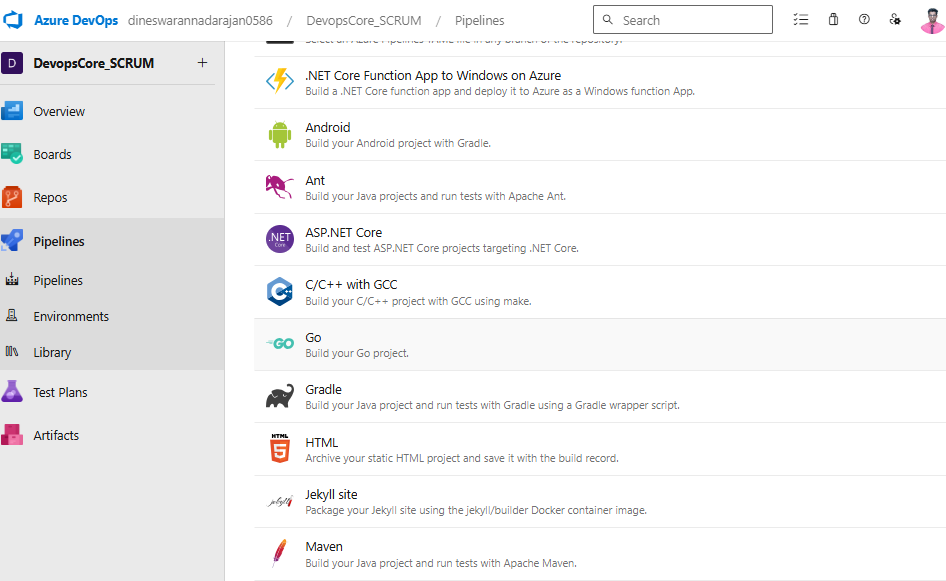
1. Then you'll be sent to a new window where you can install the Azure pipeline application.
2. Select "Choose and Approve" from the drop-down menu and select apporove&Install.



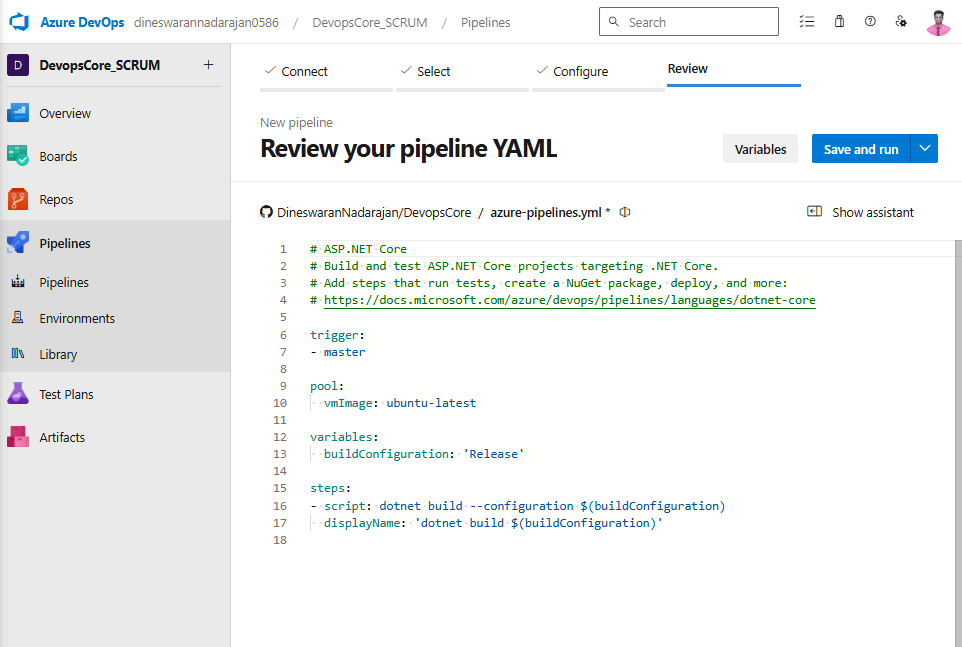
1. The configure window will appear when the installation has been completed successfully.



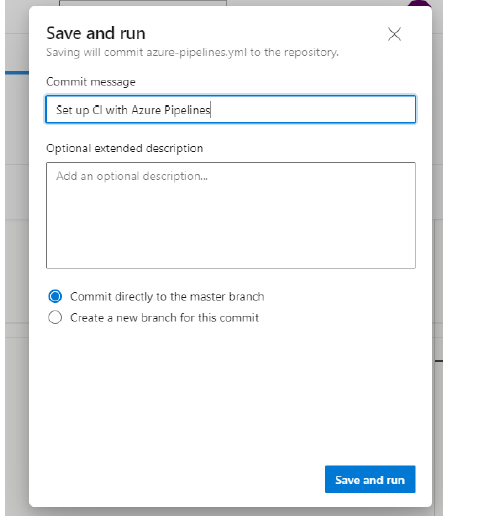
1. From the configure window choose the ASP .Net Core



1. Your pipeline file will now display as a YAML file on the review tab of the application.
2. To make the YAML file executable, choose "Save & Run."



1. A new popup window will open before the program can be launched. Simply choose "Commit directly to the master branch" from the drop-down menu and hit "Save and Run."
2. Otherwise, your project will be saved in a new GitHub branch if you choose any of the other choices available.



1. The fundamentals of pipeline development are now complete, and the pipeline maybe used to execute the program and create the program.
2. Following the deployment, you will be presented with the window shown below:

