Azure DevOps CI/CD Pipeline for .Net Core

Book · October 2023

CITATIONS
0

READS
3,070

1 author:

R. Prabodha Dilini Lenora
Sri Lanka Institute of Information Technology
5 PUBLICATIONS **3** CITATIONS

SEE PROFILE

Azure DevOps

CI/CD Pipeline for .Net Core

Prabodha Lenora

This book is dedicated to my family and friends.
Cover page image from https://free4kwallpapers.com/
Chapter images from https://www.cleanpng.com/

Table of Contents

About the Author

Prabodha Lenora is a senior software developer with over four years of experience working with.Net and Angular. She earned a Bachelor of Science degree and is now pursuing a Master of Science degree.

Chapter 01

Introduction

1.1 What is DevOps

DevOps is a term that refers to the merger of development (Dev) and operations (Ops) (Ops). The combination of people, processes, and technology that adds value to the customer experience is also important. The cooperation between the development and operations teams results in a high-quality product and a streamlined deployment procedure. DevOps is capable of continuous integration and continuous development, which means that application development and deployment may be fully automated using automation tools.

1.2 Why DevOps

There are numerous advantages to embracing DevOps practices.

1.2.1 Stability

Our work environment has been stabilized, balanced, and improved as a result of the implementation of DevOps.

1.2.2 Speed

Continuous delivery and continuous integration help to make the product more responsive, and it will also enable the company to outperform the competition.

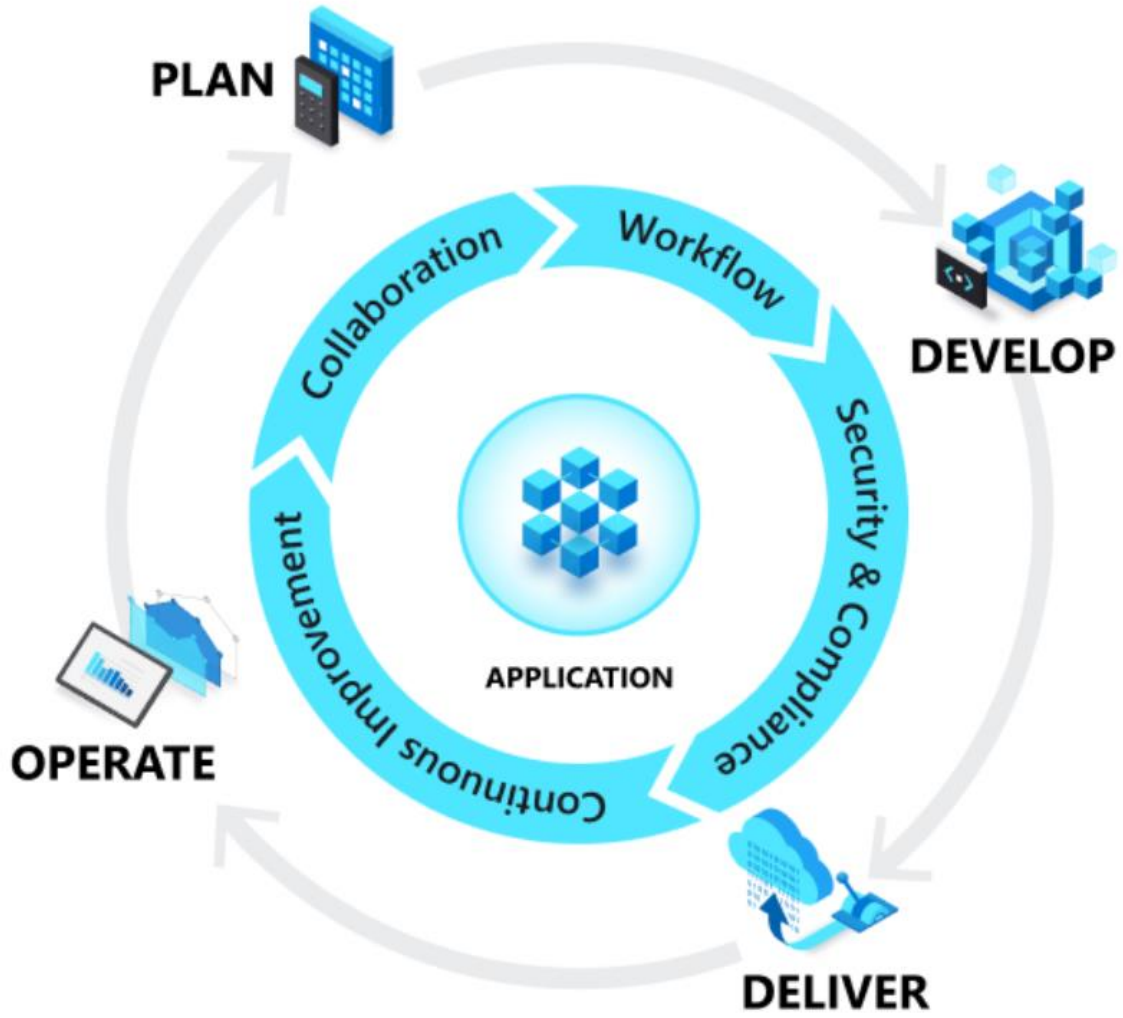
1.2.3 Transparency

Users will be able to interact with team members much more readily as a result, and the team members will be able to focus on their own specialties.

1.2.4 Cost

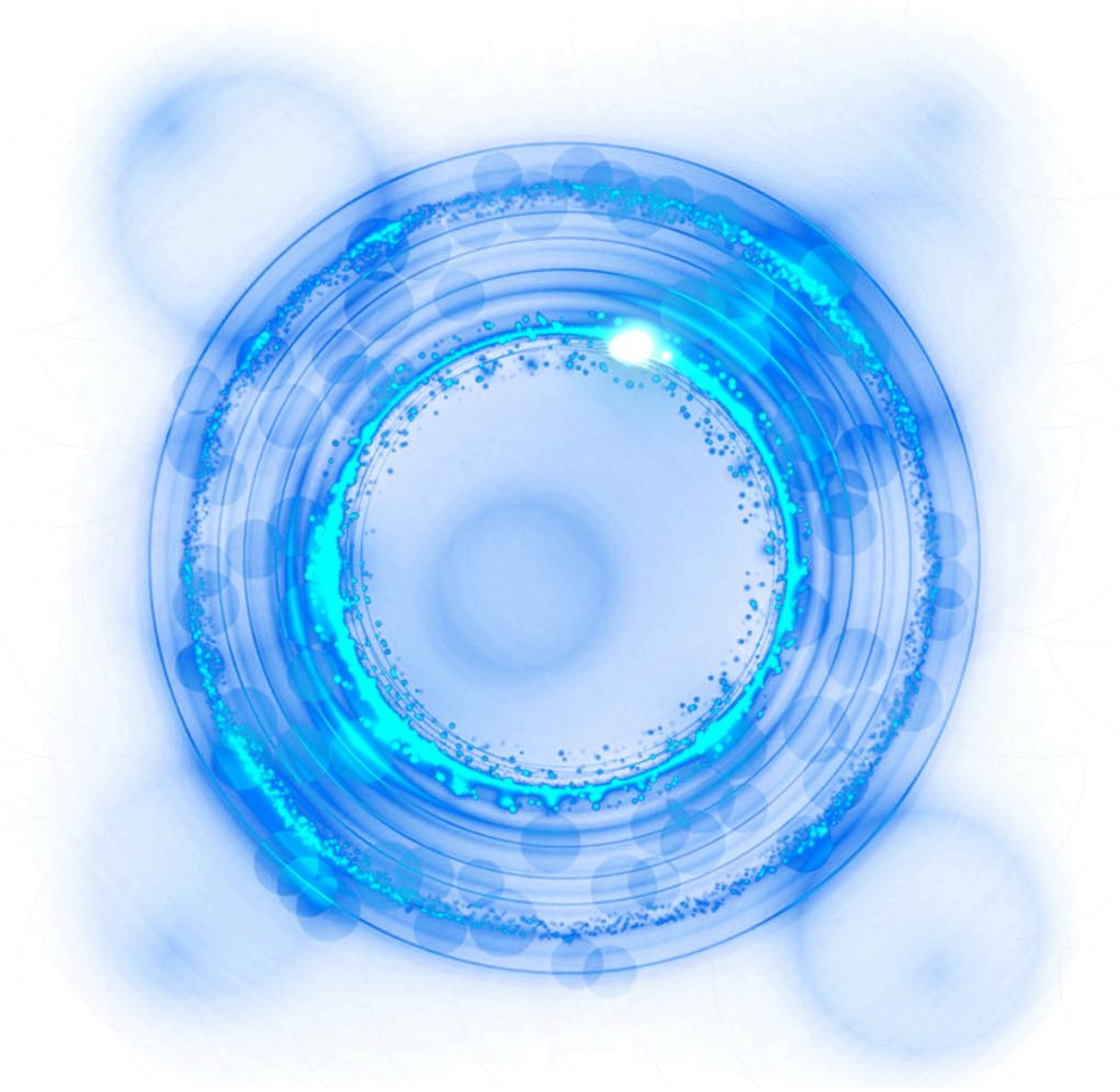
We may reduce the cost of our department's product and management by implementing DevOps practices. All of this is included in the DevOps package, so we don't have to look for separate substitutes for each activity.

1.3 Application life cycle with DevOps



[W. de Kort, “What Is DevOps?,” in *DevOps on the Microsoft Stack*, Berkeley, CA: Apress, 2016, pp. 3–8.]

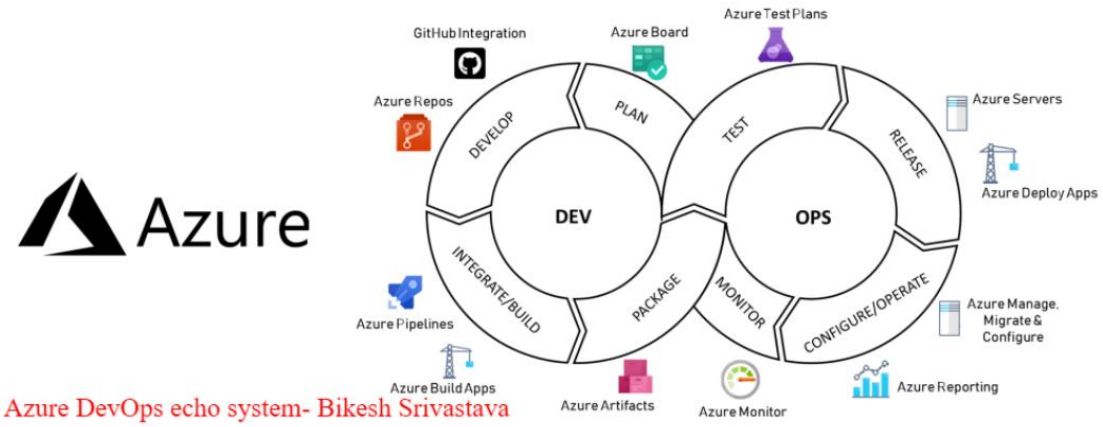
Chapter 02



Azure DevOps

2.1 Introduction

Azure DevOps is a single platform that enables all software development teams on the universe to design projects using the Agile approach, manage code using Git, test the application, and release code using the CI/CD architecture, all in a single environment. Azure DevOps is all you need to get started building your product (formerly known as VSTS-Visual Studio Team Service).



B. Srivastava, “What is Azure DevOps and why should we use it?,” *C-sharpcorner.com*. [Online]. Available: https://www.c-sharpcorner.com/article/what-is-azure-devops/. [Accessed: 12-Nov-2021].

2.2 Online and On-Premises

Azure DevOps is comprised of two distinct components. The first is located on-premises and is referred to as "Server," while the second is located online and is referred to as "Services." All of the cloud services are part of the Microsoft Azure cloud, and they are built with exactly the same code as the on-premises services. It makes a few tiny tweaks before running the most recent code.

2.3 History of Azure DevOps

Product name	Form	Release year
Visual Studio 2005 Team System	On-premises	2006
Visual Studio Team System 2008	On-premises	2008
Team Foundation Server 2010[26]	On-premises	2010
Team Foundation Service Preview	Cloud	2012
Team Foundation Server 2012	On-premises	2012
Visual Studio Online[27]	Cloud	2013
Team Foundation Server 2013	On-premises	2013
Team Foundation Server 2015	On-premises	2015
Visual Studio Team Services	Cloud	2015
Team Foundation Server 2017	On-premises	2017
Team Foundation Server 2018	On-premises	2017
Azure DevOps Services[28]	Cloud	2018
Azure DevOps Server 2019[29]	On-premises	2019

Source: Wikipedia

2.4 Keywords

2.4.1 Organization

It is possible to change the name of the Azure DevOps organization, which is by default an account name/domain name. We will often have one Azure DevOps account that will deal with several domains, which means that we can construct many organizations with a single Microsoft account in the majority of cases. Individual organizations can also have their access control and security configurations customized by us.

2.4.2 Projects

A single Azure DevOps organization can be divided into numerous projects, each with its own access control, pipeline, build process, board, and code, and each with its own code base.

2.4.3 Azure Board

The Azure board is primarily used for task planning and tracking, as well as for Backlog and Sprint management. When it comes to application life cycle management, Azure Board is categorized among technologies such as JIRA, SpiraTeam, and other similar solutions. We may develop workflows, issue kinds, epics, and a variety of other ALM components within the Azure board.

2.4.4 Azure Repos

In Azure Repos, we can create, manage, and save different versions of our codebase. It may be used for code review and other functions of version control, among other things. GIT (Distributed Version Control) and TFVC (Transferable File Versioning) are both supported by cloud-based repository services such as Azure Repos (centralized version control).

2.4.5 Azure Pipeline

A DevOps activity such as building, releasing, testing, and deploying to a target machine is one that may be automated with the use of Azure pipelines. For the uninitiated, Azure Pipeline is a blend of Continuous Integration and Continuous Deployment technologies.

2.4.6 Azure Artifact

It is possible that we may use Azure Artifacts as an extension of Azure DevOps to produce, host, manage, and share packages around the team. Azure Artifacts supports a wide range of programming languages, including NPM, Nuget, Maven, Python, and others. Additionally, there are a variety of different types of files that can be discovered in Azure Artifacts as well. DLLs, rpms, jars, and other types of files fall into this category. One of those files with an extension-based structure may contain meta-data.

2.5 Tasks

2.5.1 Planning

As previously noted, Azure is an application lifecycle management (ALM) tool. As a result, using kanban and scrum, Azure DevOps assists us in planning, tracking the development history of individual developers, and identifying problems.

2.5.2 Develop

When we use Azure DevOps, we can manage our workspace and code repository using a variety of version control methods, which is really convenient. Almost all of the capabilities associated with DevOps are already incorporated into the platform.

2.5.3 Integration and Build

Using the Azure pipeline, we can automate the process of continuous integration and delivery (CI/CD). A number of different job agent templates are already included in the package.

2.5.4 Package

There is an integrated extension, Azure Artifacts, so we don't have to go looking for packages outside of Microsoft Azure.

2.5.5 Testing

DevOps relies heavily on testing, as we all know. Azure DevOps has built-in testing templates and management tools for a variety of scenarios.

2.5.6 Release

In the world of DevOps, the software release process is crucial. As a result, Azure DevOps already has a number of built-in capabilities that make the process as simple and risk-free as possible.

2.5.6 Reporting

Reports, a summary, and the ability to create an analytical report utilizing BI integration are all available on the Azure DevOps dashboard.

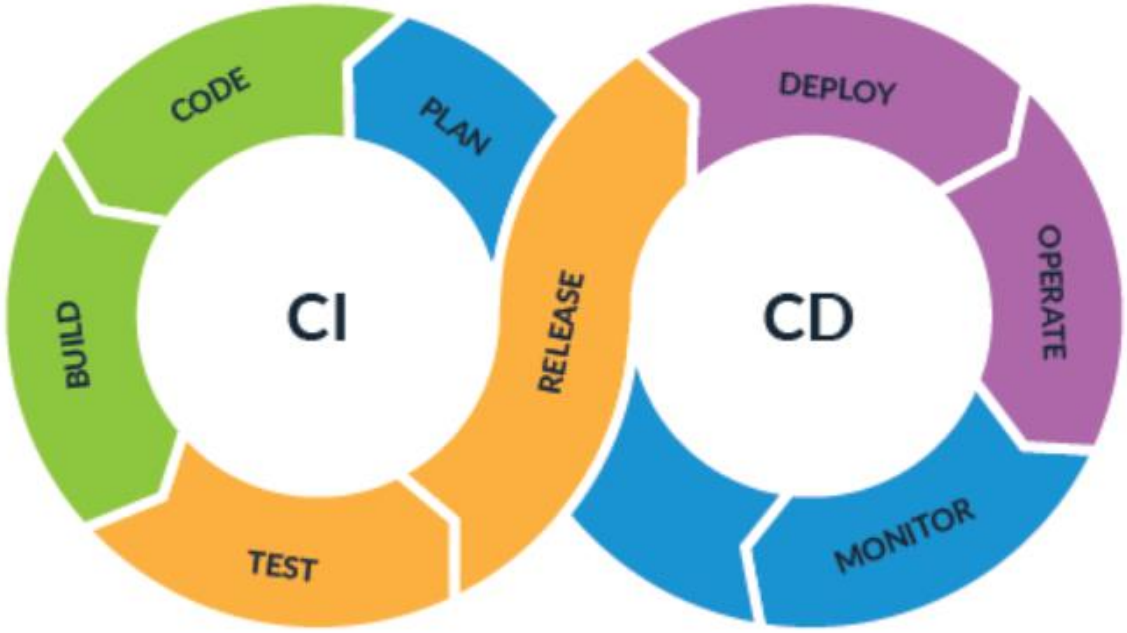
Chapter 03

CI/CD Pipeline

3.1 What is CI/CD pipeline

A continuous integration and delivery pipeline (CI/CD pipeline) streamlines your software delivery process. The pipeline creates code, runs tests (CI), and releases a new version of the application in a secure environment (CD).

Automated pipelines eliminate the need for manual mistakes, offer developers with consistent feedback loops, and allow for rapid product development.



D. Moonat, “A step by step guide to create a CI/CD Pipeline with AWS Services,” *Analyticsvidhya.com*, 15-Jul-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/a-step-by-step-guide-to-create-a-ci-cd-pipeline-with-aws-services/.

3.2 Continuous Integration

The objective of contemporary application development is to have numerous developers working on various parts of the same app at the same time. An organization, on the other hand, may have been set up to merge all branching source code together on the same day. This is because when a developer working in isolation makes a change to an application, there is a chance that it will conflict with different changes being made by other developers at the same time by another developer. When developers use continuous integration (CI), they may easily integrate their code changes back into a common branch. If automated testing uncovers a conflict between new and old code, continuous integration (CI) makes it easy to address such errors more rapidly and often than before.

3.2 Continuous Delivery

Continuous delivery (CD) automates the distribution of verified code to a repository once the builds and unit and integration testing have been automated in continuous integration (CI). In order to have a successful continuous delivery process, it is critical that continuous integration (CI) has been integrated into your product pipeline. With continuous delivery, the aim is to always have a working codebase that is ready for deployment into a production environment.

Every step of continuous delivery, from the merging of code changes through the delivery of production-ready builds, is automated, with test automation and code release automation being two of the most important. Following completion of this procedure, the operations team is able to swiftly and efficiently deploy an application to production.

Chapter 04

.Net Core and GitHub

In this chapter, we'll construct a.NET core application and publish it to GitHub for everyone to use. I'm going to construct the program using Visual Studio 2019, however you may use Visual Studio 2017 or a higher version if you like. First and foremost, let us establish a fundamental understanding of what it is. Net Core and GitHub are two examples.

4.1 .Net Core

.NetCore is a modified version of the software. NET framework that is provided free of charge and is an open-source framework that is maintained by Microsoft Cross-platform means that it will work on several operating systems, including Windows, Mac, and Ubuntu. Together with the. Building web apps, mobile applications, cloud applications, machine learning models (for IoT), and other devices is possible using the Net Core framework. Using the NuGet packages, this framework delivers capabilities in a lightweight manner and with very quick speed.

Only the Windows operating system is capable of running network apps. But .NetCore is compatible with a variety of operating systems, which is a significant benefit. You have the ability to create it. This supports a variety of programming languages, including C#, F#, and Visual Basic. For development, you may use Visual Studio 2017 or a higher version of Visual Code or Sublime Text as your integrated development environment (IDE).

Version	Latest Version	Visual Studio	Release Date	End of Support
.NET 5	Preview 1	VS 2019	16th March, 2020	
.NET Core 3.x - latest	3.1.3	VS 209	24th March, 2020	12th March, 2022
.NET Core 2.x	2.1.17	VS 2017, 2019	24th March, 2020	21st August, 2021
.NET Core 1.x	1.1.13	VS 2017	14th May, 2019	27th May, 2019

4.2 GitHub

GitHub is a high-level website and cloud service provider for developers that allows them to store and manage their code. GitHub is comprised of two major components: the version controller and the Git repository management system.

Version Controller

Through the use of a version controller, developers may keep track of and manage code changes. Branching and merging are the two choices provided by the version controller. Developers may now commit their work to the git repository without making any modifications to the main code as a result of these enhancements. In order to incorporate your modifications into the repository, you will need to establish a new branch, which will include a copy of the original code already present in the repository. After you've committed valid and clean code to the branch, you may merge your work into the main source code base of the project.

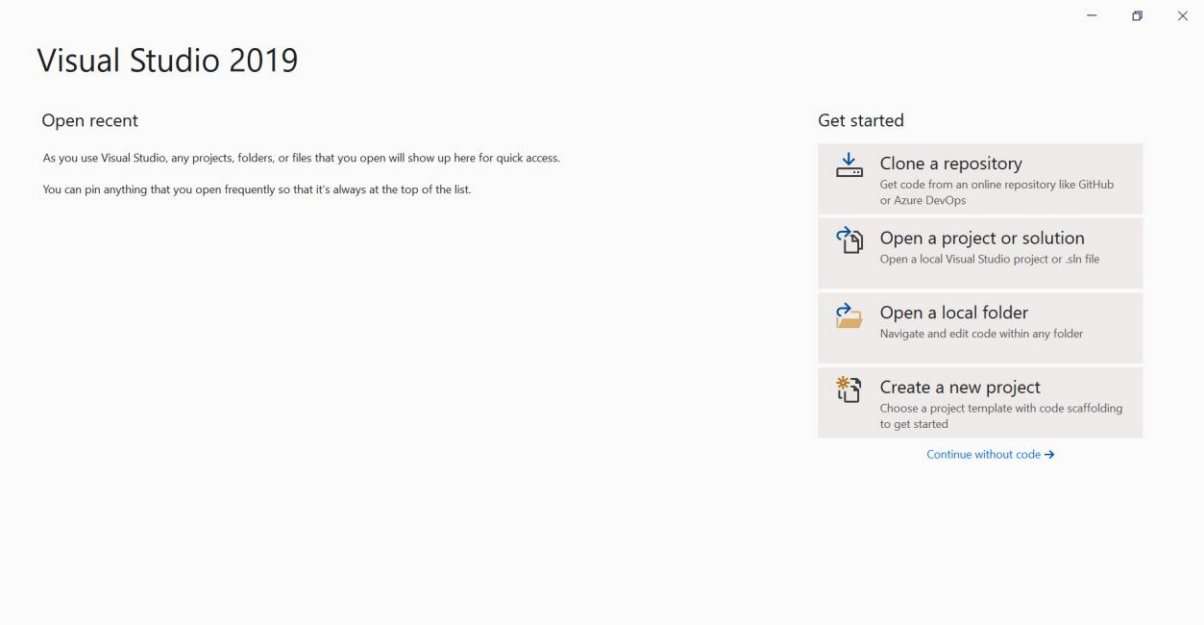
Git

Git is a version control system that is available as free software. This provides developers with access to the whole code base as well as a history of any modifications made to the code over time. It makes the process of combining simpler.

4.3 Create .Net Core Application

Open the visual studio 2017 or upper version .I'm going to use visual studio 2019 an .Net Core version 5

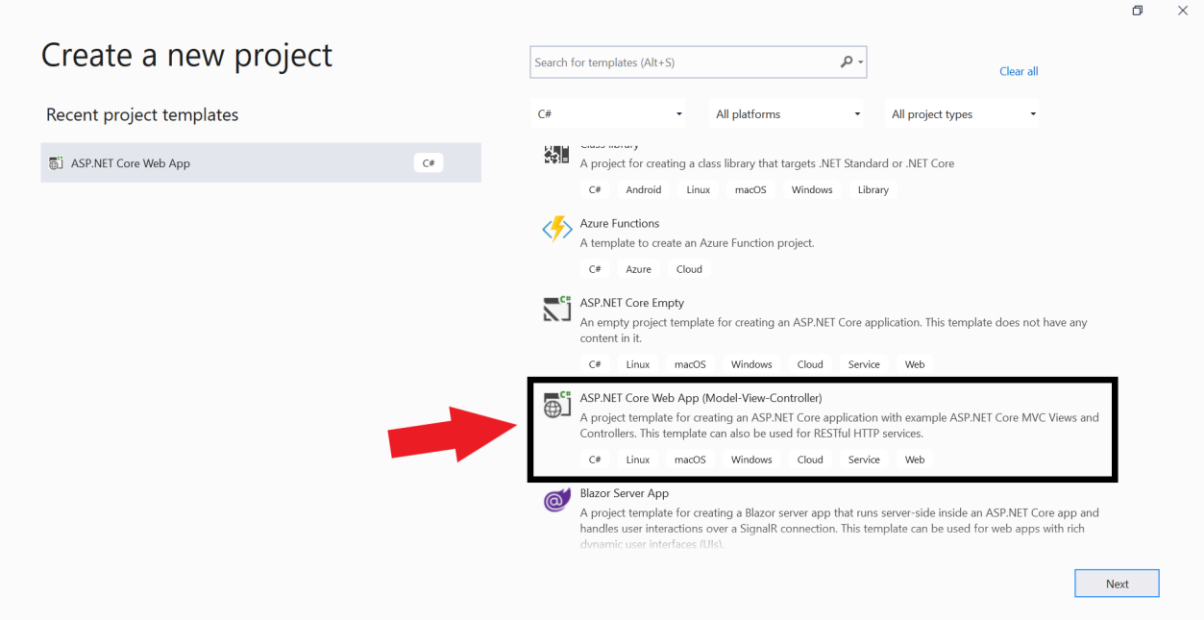
1) Select the "Create the new project" option



2) I'm using C# to create my project. If you want, you can use another language that .Net Core support.

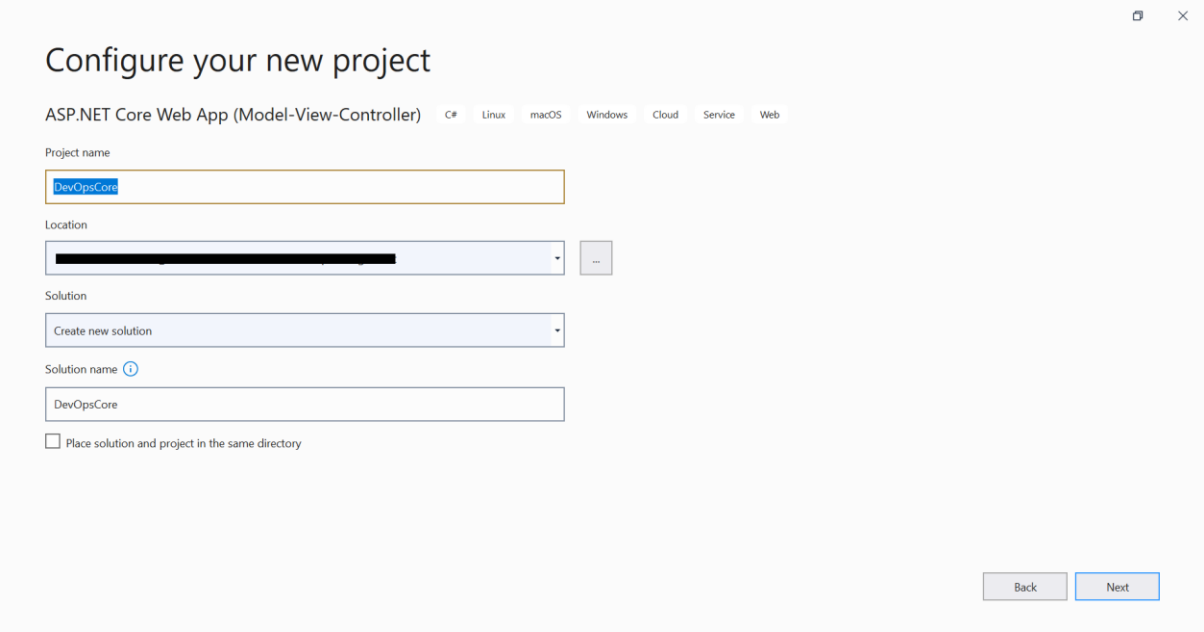
   You can search for languages and platforms using drop-down menus.

   If you wish to continue with C # MVC, select the option that I have already selected like in this image and click the Next.



3) Give a name for the project and select a location to save your project

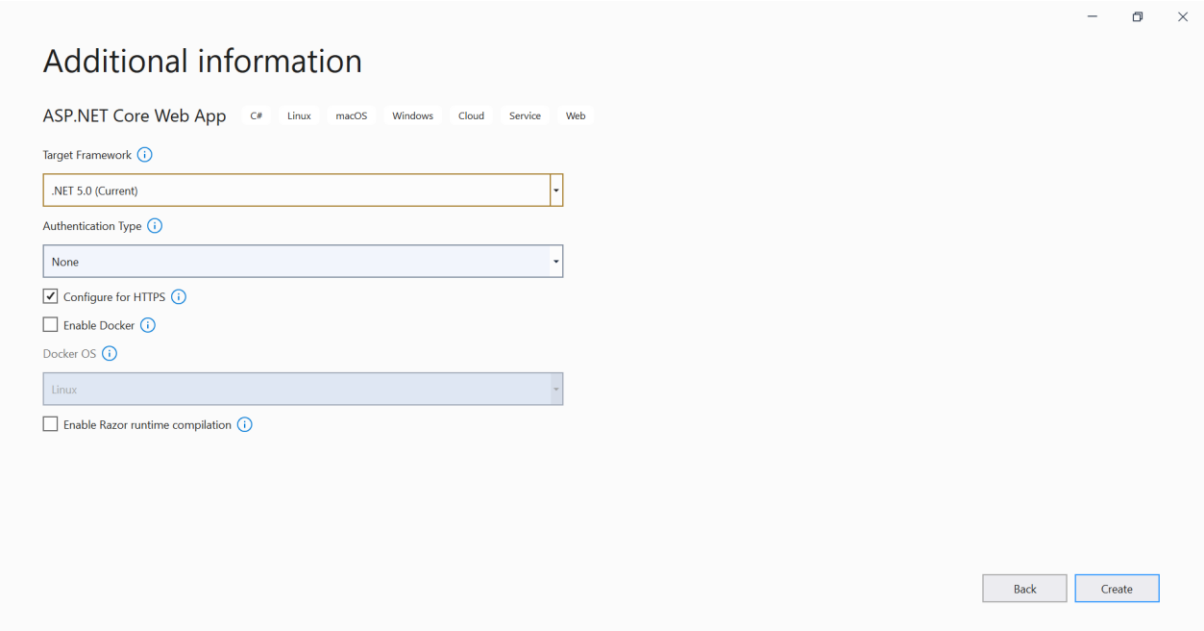
   After that click the Next button.



4) From target framework drop down you can select which Core framework you are going to use. I am going to use .Net Core version 5.

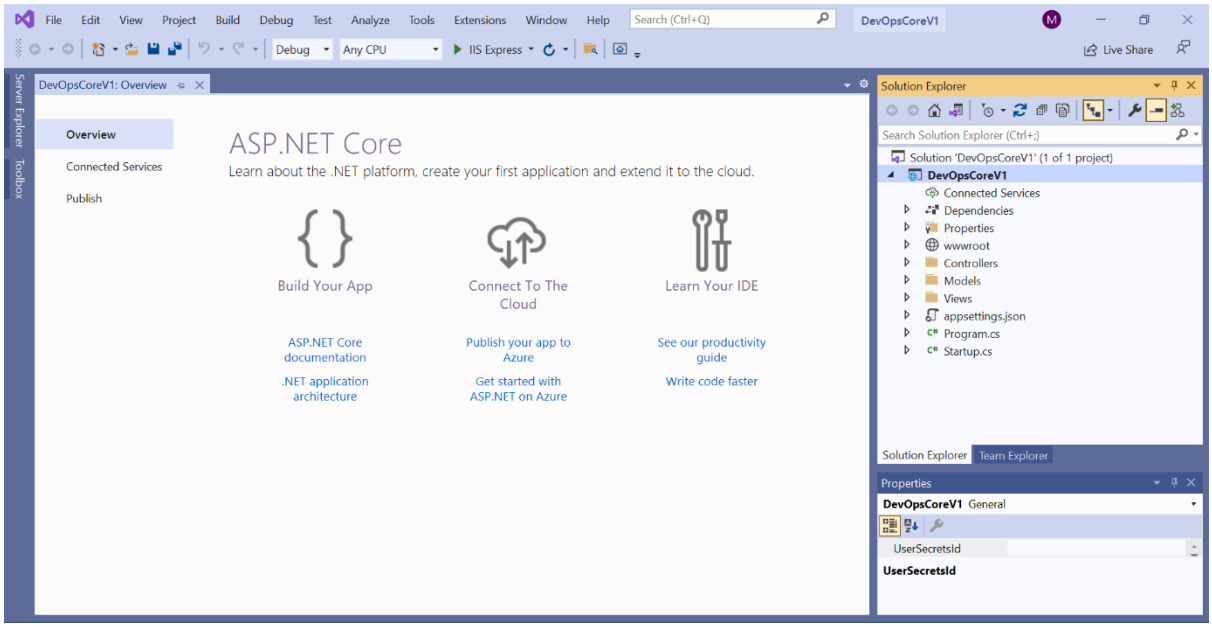
   You are free to choose any version from the drop-down menu that you like to work with while creating your document.

   Then click the Create button.



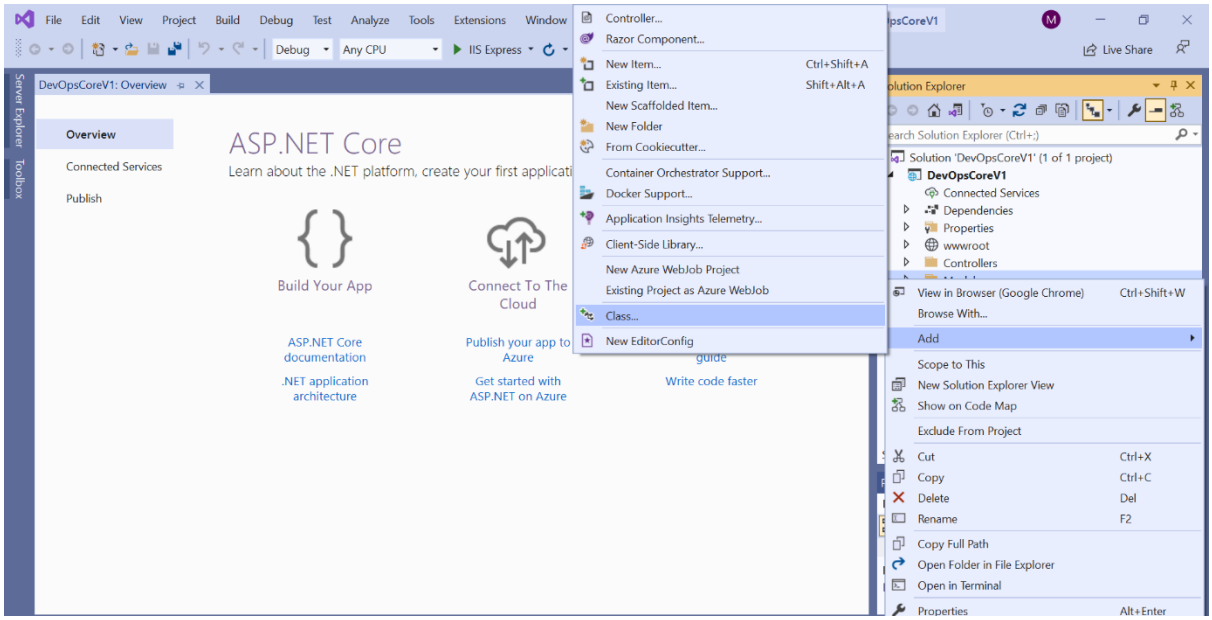
5) Following the preceding step, you will be sent to the Visual Studio IDE.

You will be able to view your new project from there.



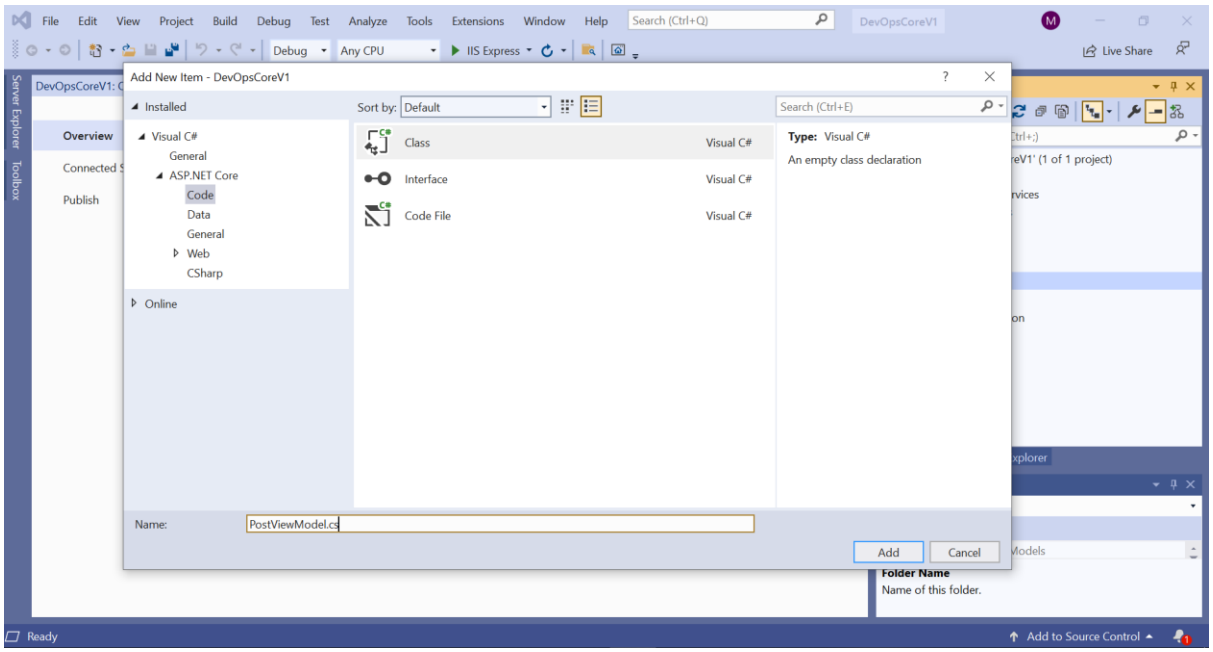
6) Let's make a few changes to the Model, View, and Controller to see how they work together.

   Let's add a "Post" class to the model to make it more user-friendly. Right-click on the model folder and pick "add"-> "class" from the drop-down menu.



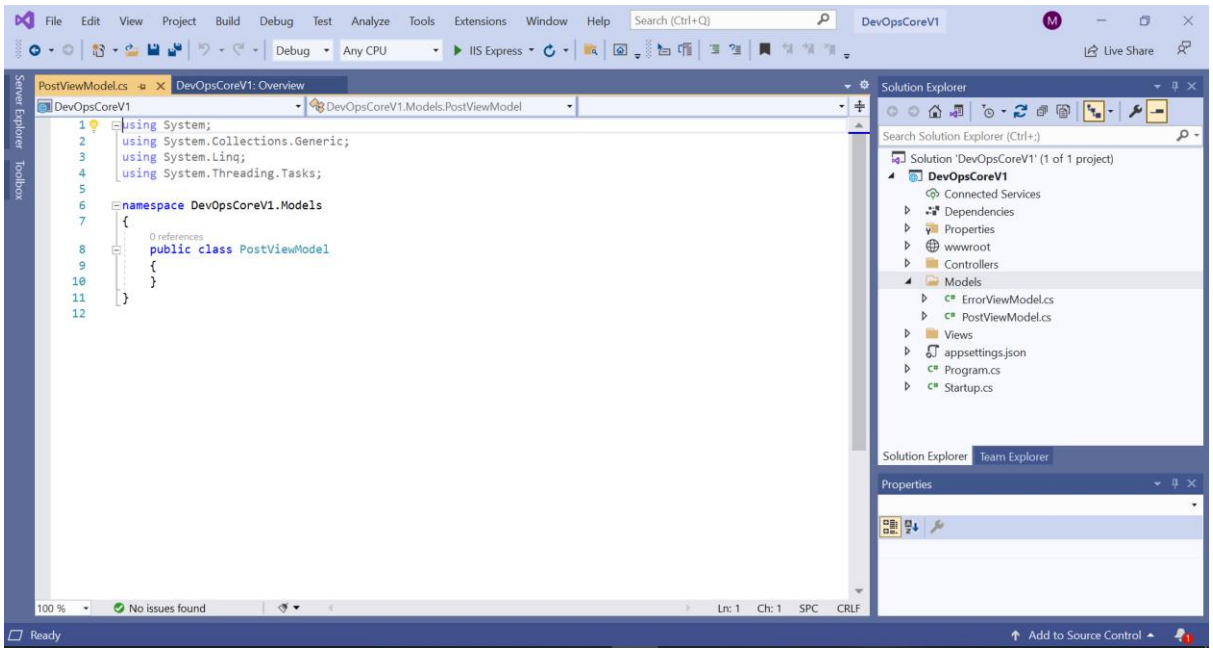
You have the option of giving the model class a name. My model class will be named "PostModel," and it will be implemented in the project

Then click the Add button.



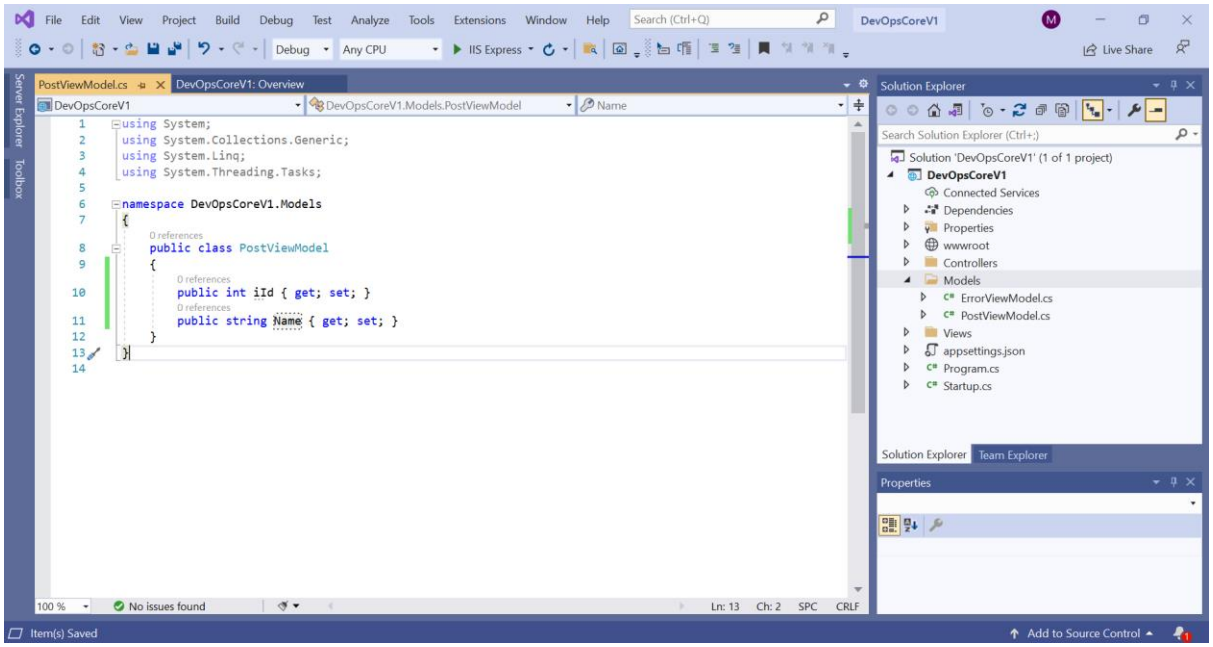
7) Lets add some properties to our model class

Your model class will resemble the picture below. It is currently empty, and we will add additional characteristics to it.



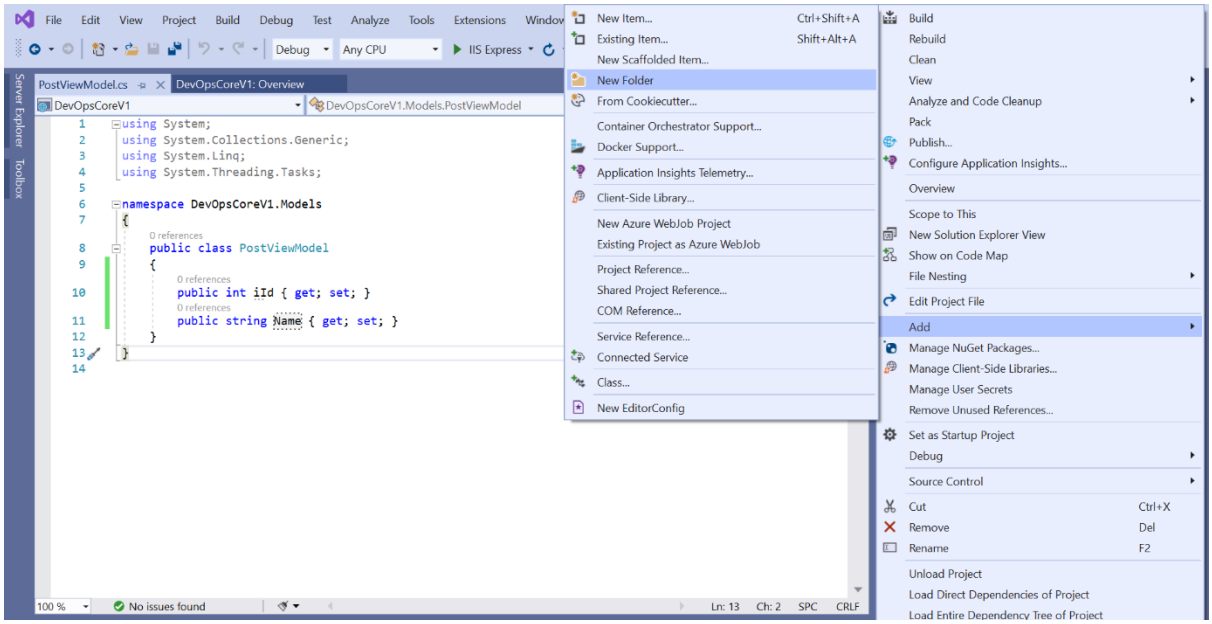
Let's begin by adding the following attributes in the model class.

```
namespace DevOpsCoreV1.Models
{
    public class PostViewModel
    {
        public int iId { get; set; }
        public string Name { get; set; }
    }
}
```

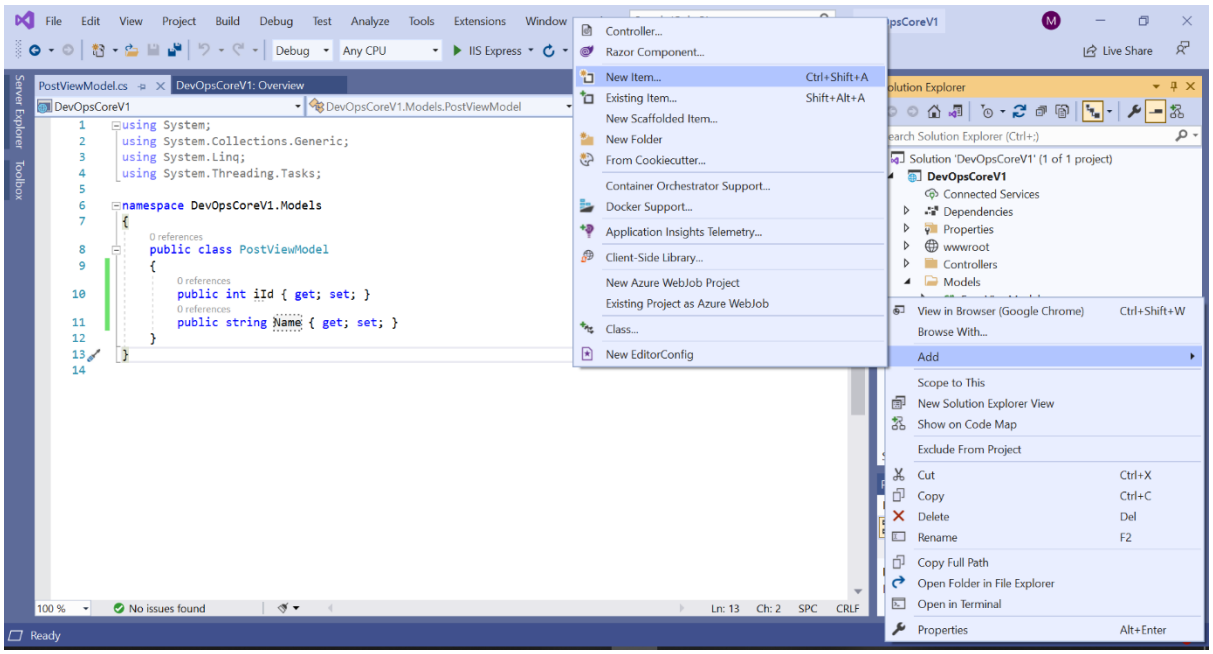


8) The next step is to establish a folder in the repository for the repository classes. We construct repositories via the usage of interfaces.

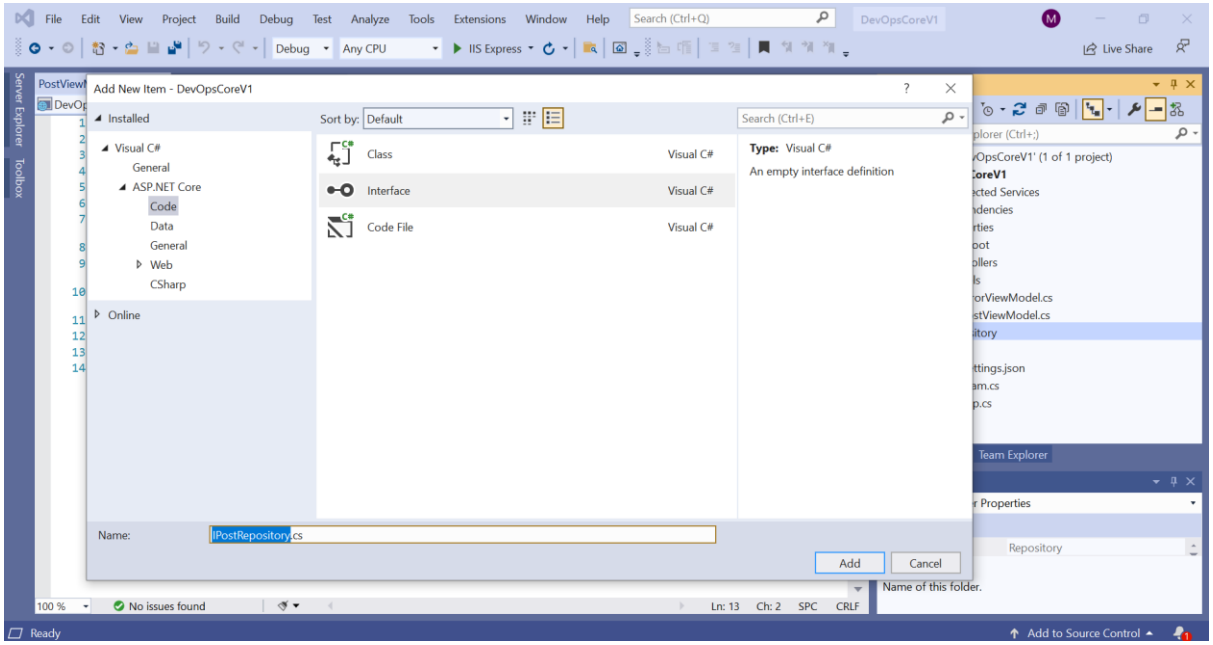
   To create a new folder, right click on the folder structure and pick Add->New Folder. After the new window, you have to provide a folder name and I'm going to call it "Repository."



9) We can now add an interface to the repository folder, which is a great step forward. Right-click on the "Repository" folder and choose "add"->"new item" from the context menu.

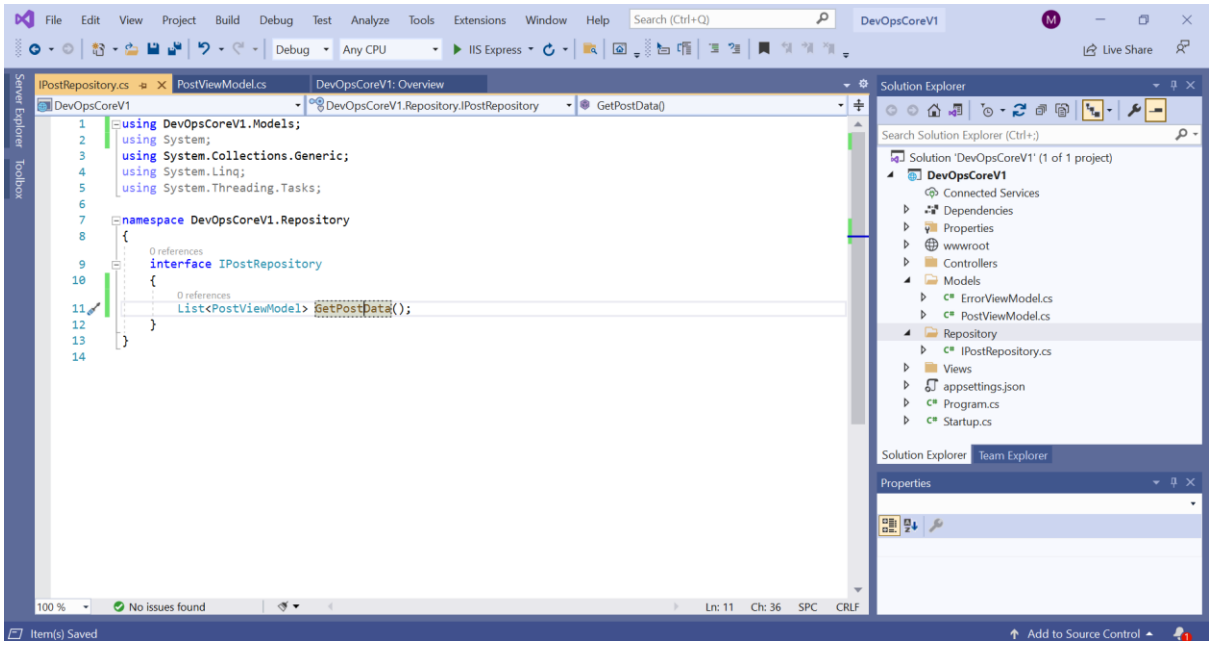


It is necessary to pick an interface and give it a name in the next box. "IPostRepository" is the name I've chosen for my interface.



10) Let's make some changes to the interface by adding some scripts. You may modify the code in the next section so that we get a list as a result of calling this function. in this case it will provide a collection of " PostViewModel."

```
using DevOpsCoreV1.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace DevOpsCoreV1.Repository
{
    interface IPostRepository
    {
        List<PostViewModel> GetPostData();
    }
}
```

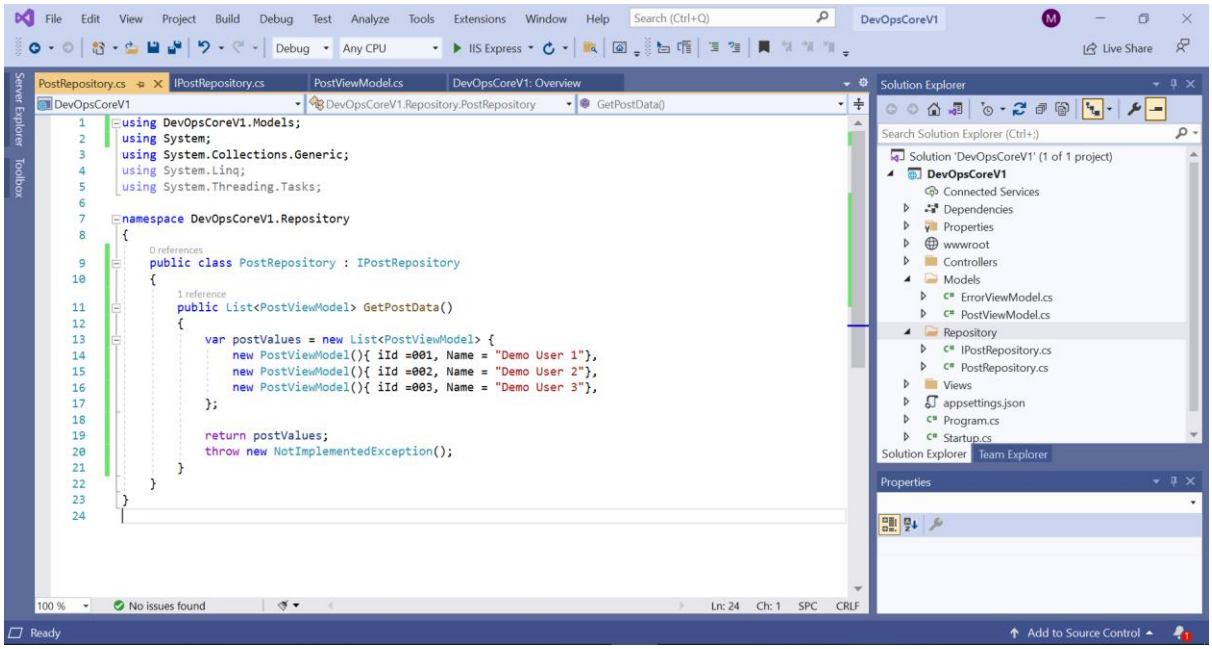


11) We must now convert the "IPostRepository" interface into a class in order to use it.

In order to do this, we must first build a new class in the repository folder named "PostRepository."

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.

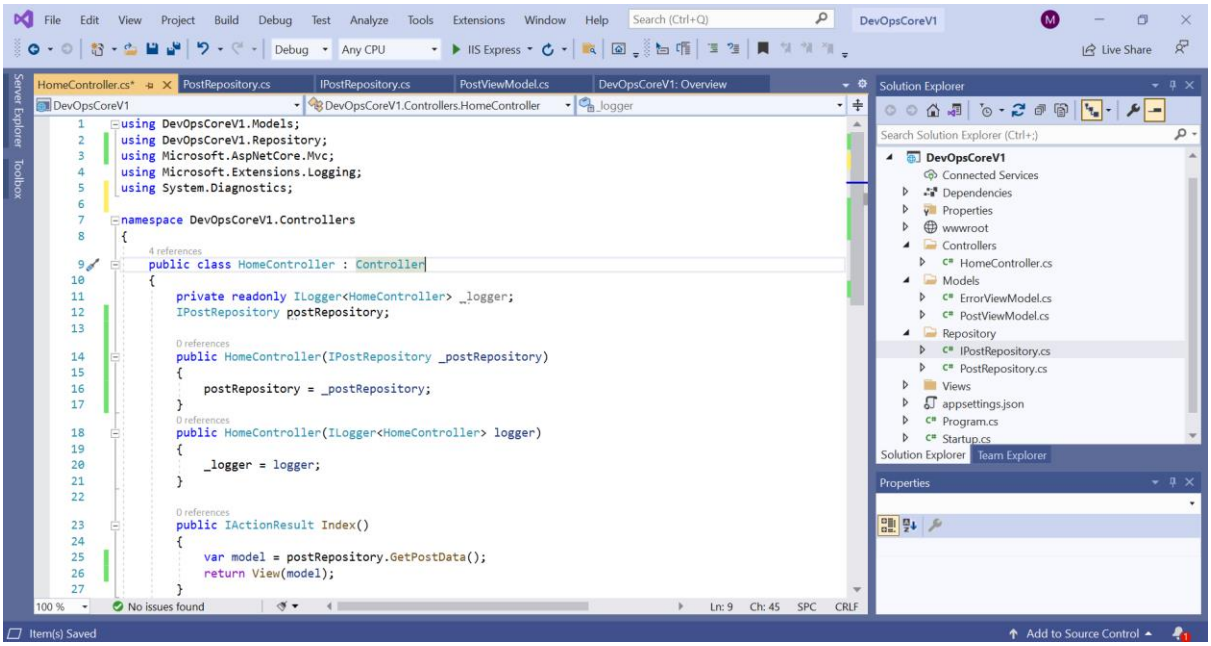
```
using DevOpsCoreV1.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

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;
        }
    }
}
```

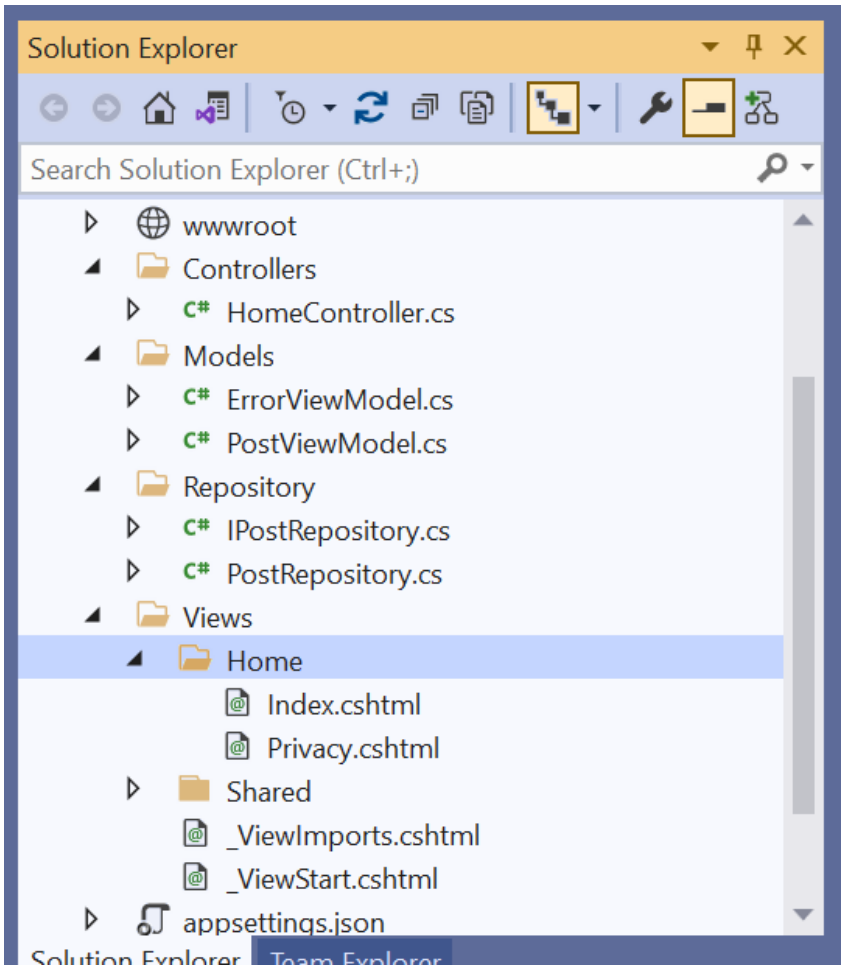

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
DevOpsCoreV1
Debug
Any CPU
IIS Express
Live Share
PostRepository.cs
IPostRepository.cs
PostViewModel.cs
DevOpsCoreV1: Overview
DevOpsCoreV1
DevOpsCoreV1.Repository.PostRepository
GetPostData()
using DevOpsCoreV1.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace DevOpsCoreV1.Repository
{
0 references
public class PostRepository : IPostRepository
{
1 reference
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;
throw new NotImplementedException();
}
}
}
100 %
No issues found
Ln: 24 Ch: 1 SPC CRLF
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'DevOpsCoreV1' (1 of 1 project)
DevOpsCoreV1
Connected Services
Dependencies
Properties
wwwroot
Controllers
Models
ErrorViewModel.cs
PostViewModel.cs
Repository
IPostRepository.cs
PostRepository.cs
Views
appsettings.json
Program.cs
Startup.cs
Solution Explorer
Team Explorer
Properties
Server Explorer
Toolbox
Ready
Add to Source Control

12) 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. Similarly as previously, right click on the controller 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 DevOpsCoreV1.Models;
using DevOpsCoreV1.Repository;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using System.Diagnostics;

namespace DevOpsCoreV1.Controllers
{
    public class HomeController : Controller
    {
        IPostRepository postRepository;

        public HomeController(IPostRepository _postRepository)
        {
            postRepository = _postRepository;
        }

        public IActionResult Index()
        {
            var model = postRepository.GetPostData();
            return View(model);
        }
    }
}
```





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

13) Binding the model to the view is what we'll be doing in this stage.

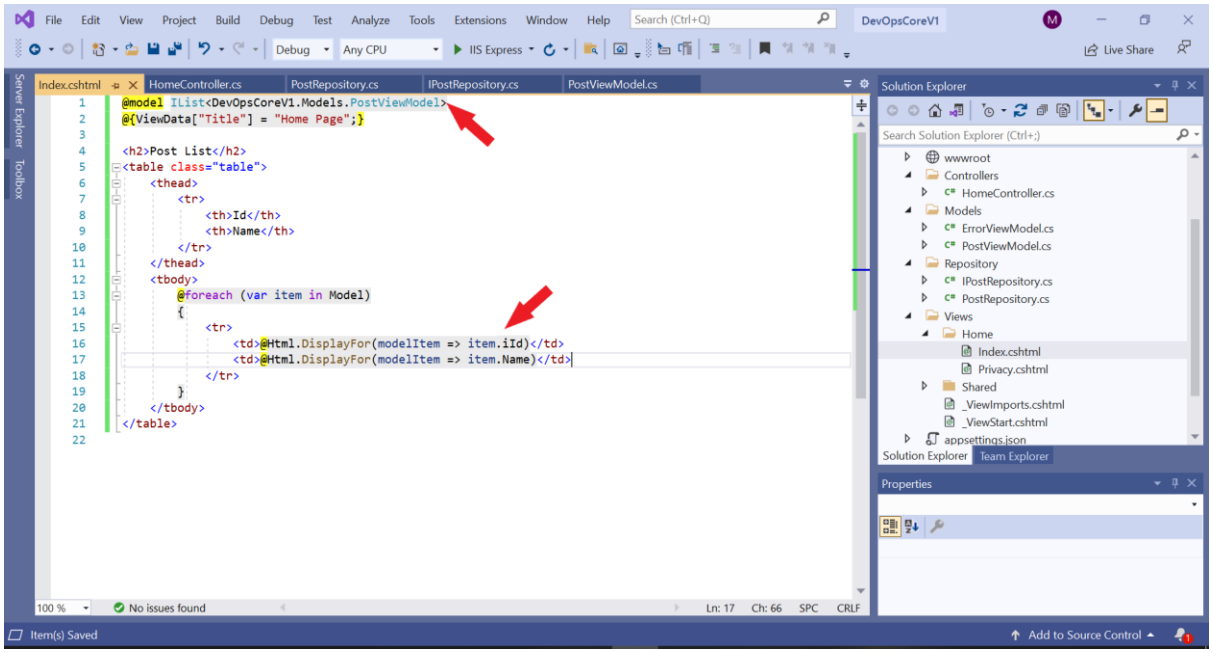
As previously noted, HomeController is already tied to the Index.cshtml file located in the home folder of the view folders, which is a good thing. As a result, I'm going to associate the model with the index file.

Include the below code to your file.

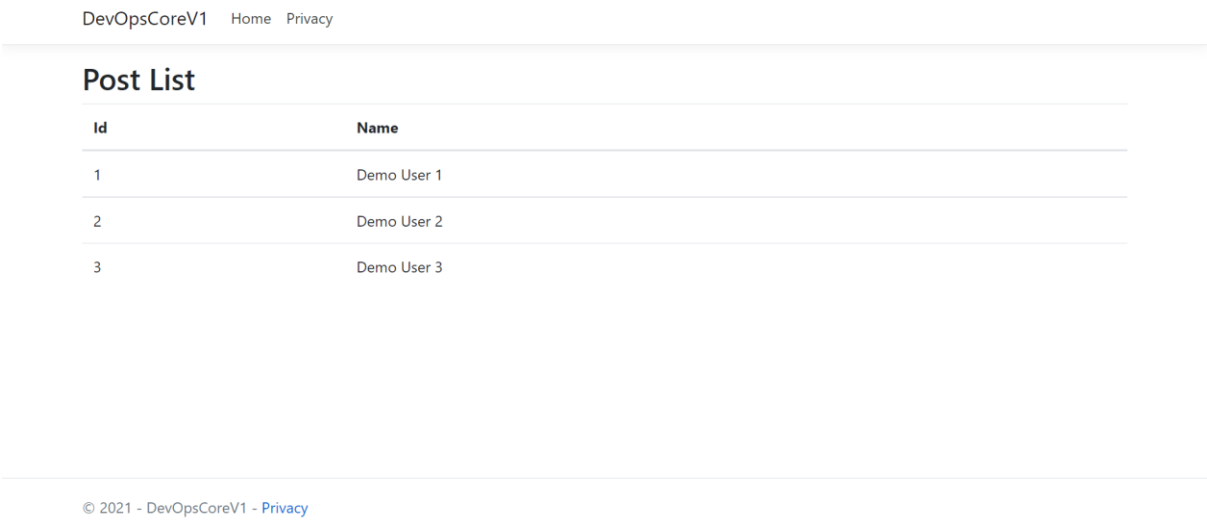
```
@model IList<DevOpsCoreV1.Models.PostViewModel>
@{ViewData["Title"] = "Home Page";}

<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>
```

You must import the model into your page by clicking on it in the upper.

Once you've done that, you can read the attributes from the model and bind them to the view, just as I did within the html table.



Your project has now been finished.



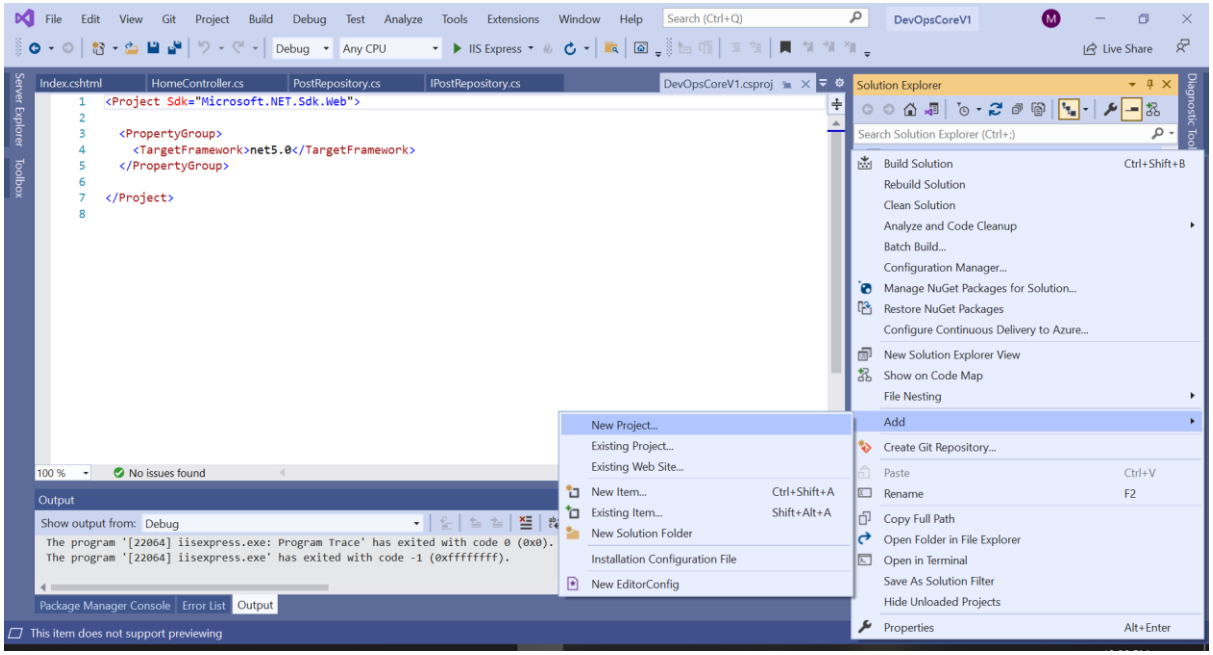
4.4 Add Unit Test Project

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 beforehand. 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.

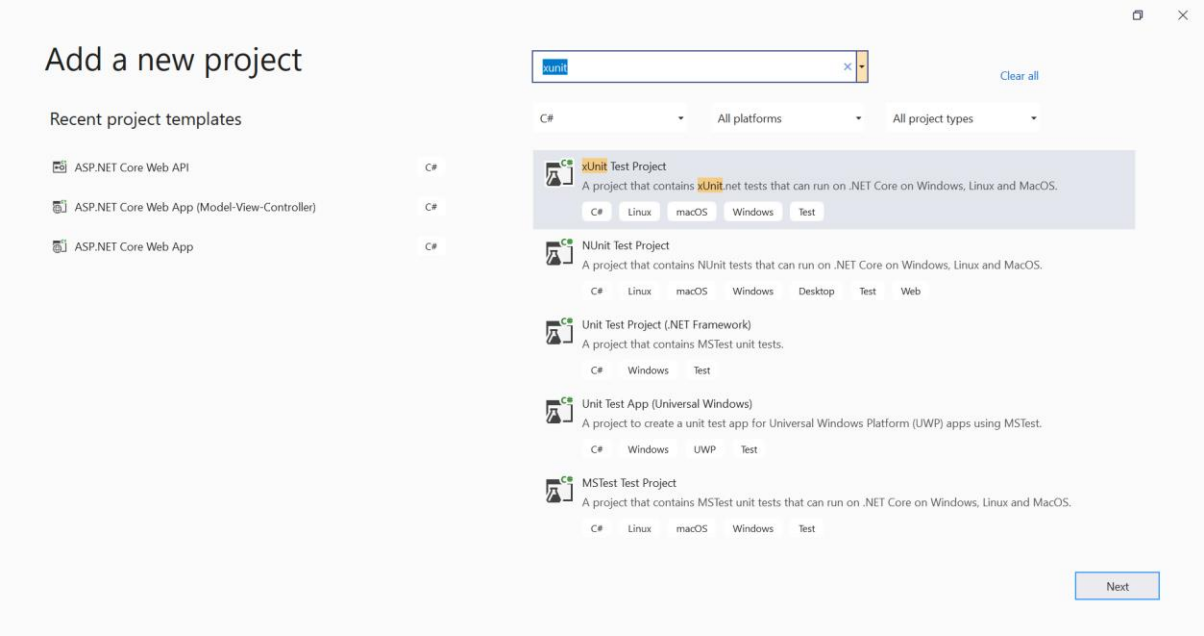
Unit tests allow us to determine whether or not our code is functioning properly. It also aids in the discovery of faults or defects in our code before it is released to the public.

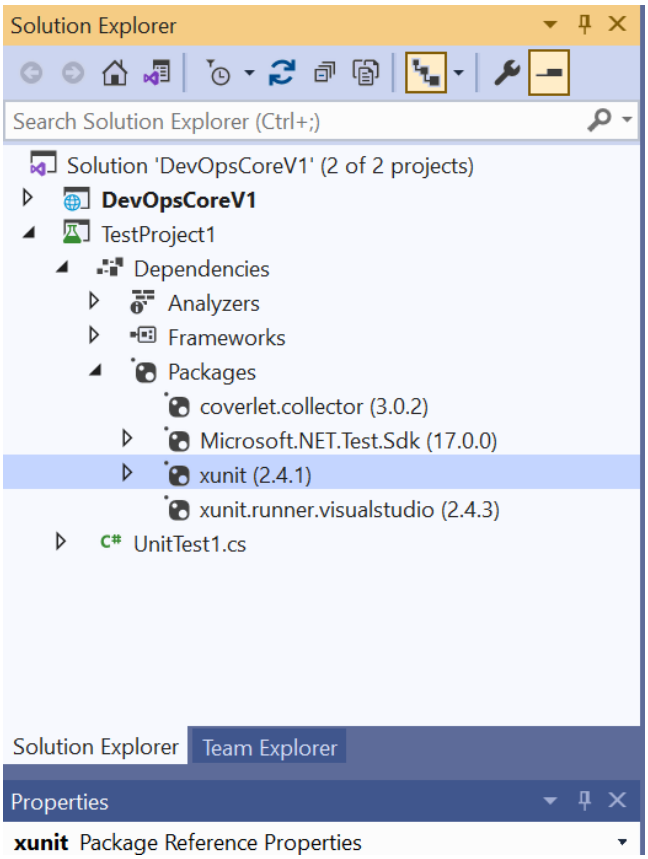
We have a test project for our project that we use for testing reasons. We'll be working on a "xUnit" test project for the time being. Let's see how we may include a test project into our existing project to do this.

Right-click on your project solution and select "Add"->"New Project".

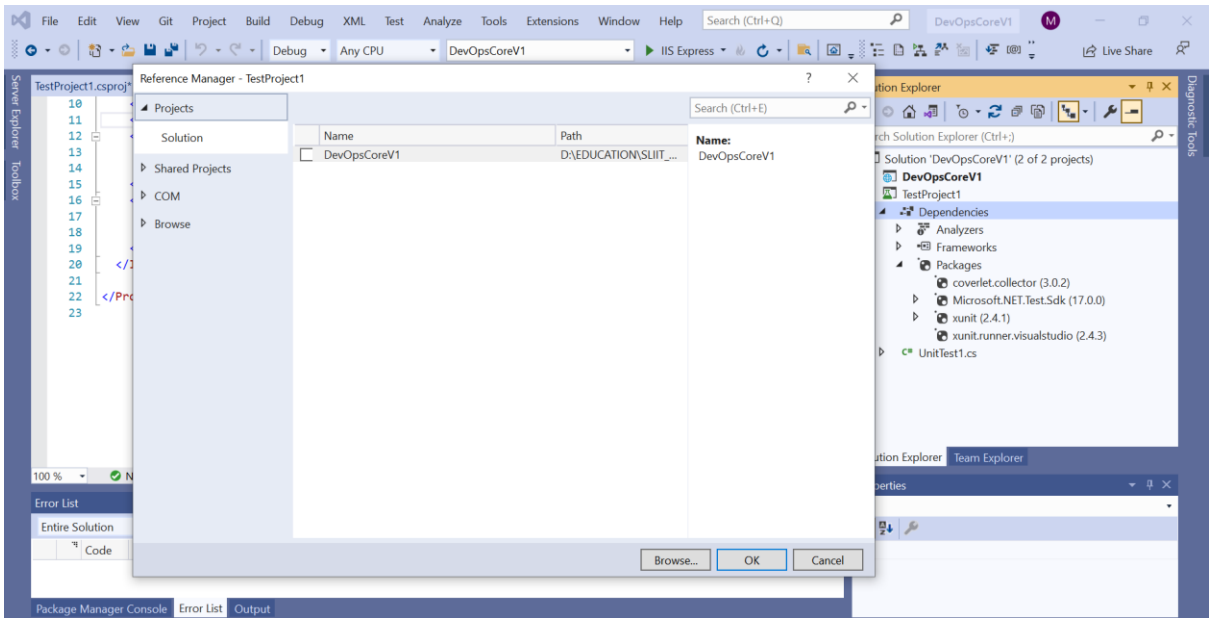


A large number of project choices will be provided in the next window.
There is a search option available here, and you can just search for xunit from there to get the xunit option for the c#.





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. To do so, right-click on the dependencies in the test project and choose Add New Reference from the context menu. All of the projects that we currently have open in solution explorer will be listed in a new window. Select the project to which you want to add the test project, check the box next to it, and then click OK.



Now we can modify the test class and use it to validate our app.

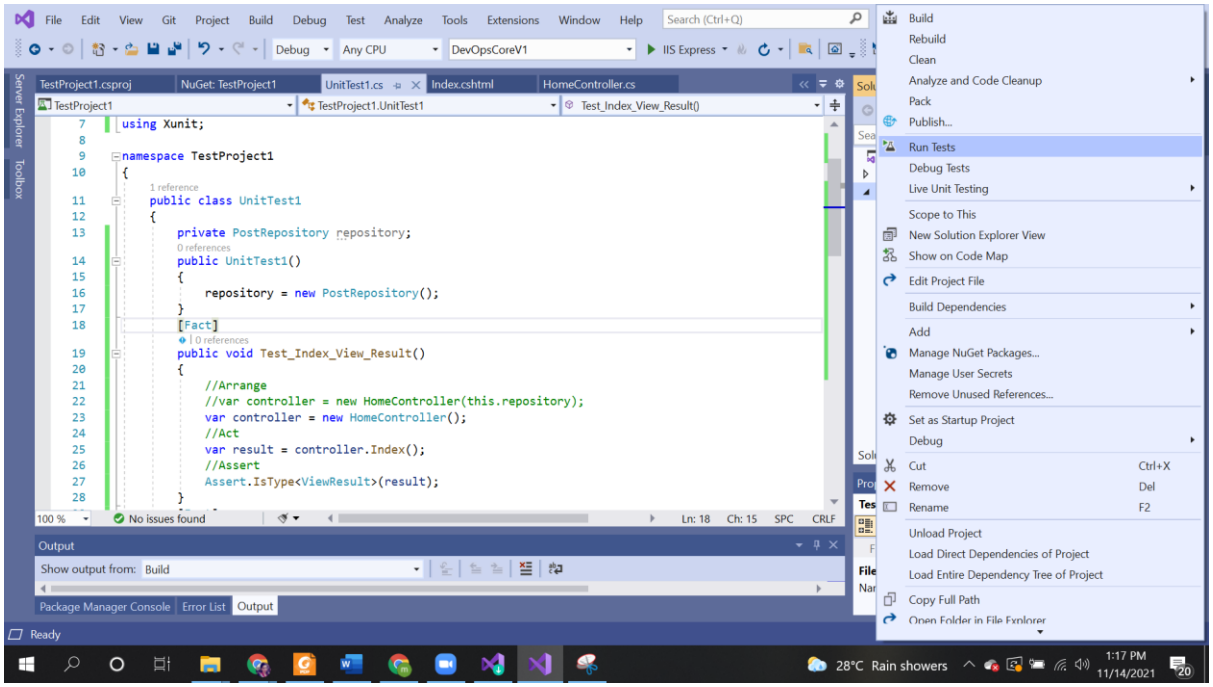
Lets add below code to your unit test class in test project.

```
using DevOpsCoreV1.Controllers;
using DevOpsCoreV1.Models;
using DevOpsCoreV1.Repository;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using Xunit;

namespace TestProject1
{
    public class UnitTest1
    {
        private PostRepository repository;
        public UnitTest1()
        {
```

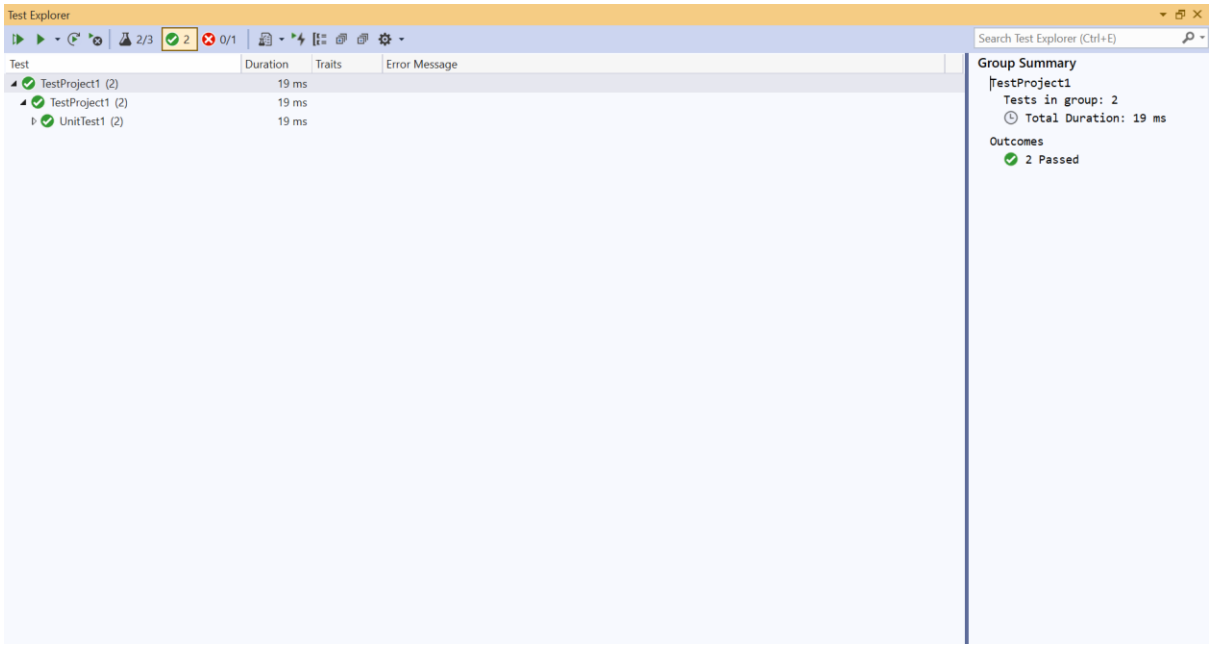
```
            repository = new PostRepository();
        }
        [Fact]
        public void Test_Index_View_Result()
        {
            //Arrange
            //var controller = new HomeController(this.repository);
            var controller = new HomeController();
            //Act
            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<PostViewModel>>(viewResult.ViewData.Model);
            Assert.Equal(3, model.Count);
            Assert.Equal(101, model[0].iId);
            Assert.Equal("DevOps Demo Title 1", model[0].Name);
        }

}
}
```

You may now execute the code for your unit testing. You can determine whether or not your code is functional by running it through the test code.

You can launch your unit test project by right-clicking on it and selecting "run test," which will cause the code to be executed.



The final result will be presented in the test explorer when all unit cases have been completed.



Let's try if we can figure out how to get this project into GitHub.

4.5 Add project to the GitHub

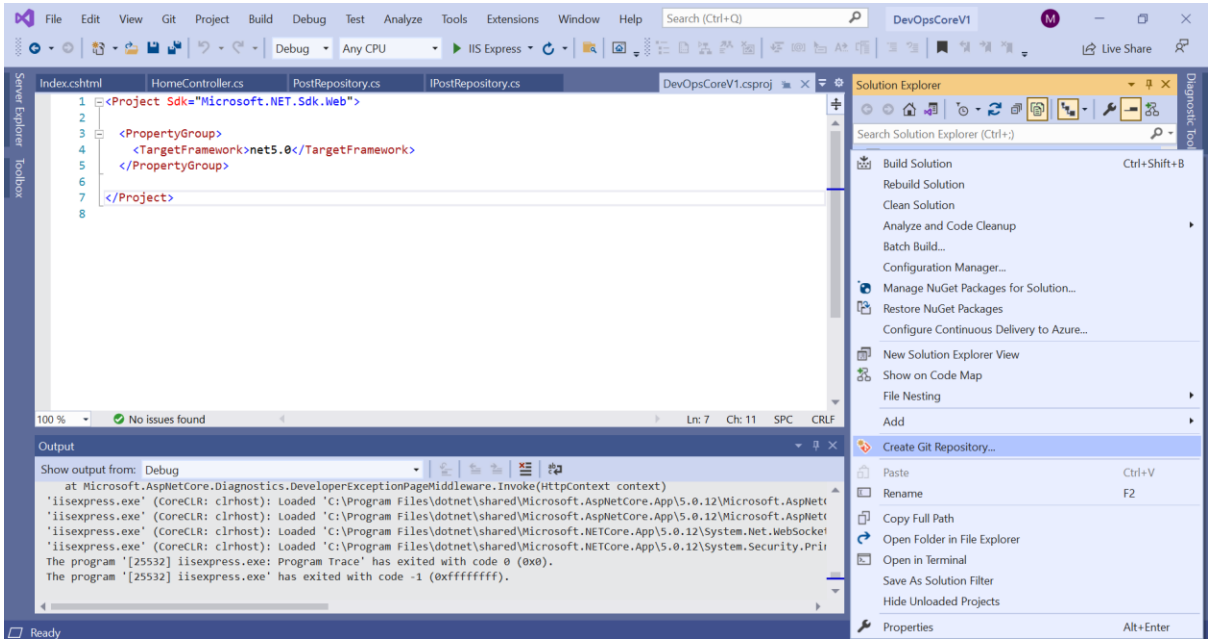
You may upload your project to any repository, such as Bitbucket or GitHub, and you can access it from Azure DevOps at any time. However, for the time being, I'll be using GitHub to upload my project.

First and foremost, if you do not already have a GitHub account, you must establish one right now.

There are a variety of options for submitting your code to GitHub. In this book, I'm going to discuss a few of choices that we may put to action very quickly and simply.

Option 01

You will discover an option named "Create Git Repository" if you right-click on your project solution and choose it from the context menu. Select that item from the drop-down menu.



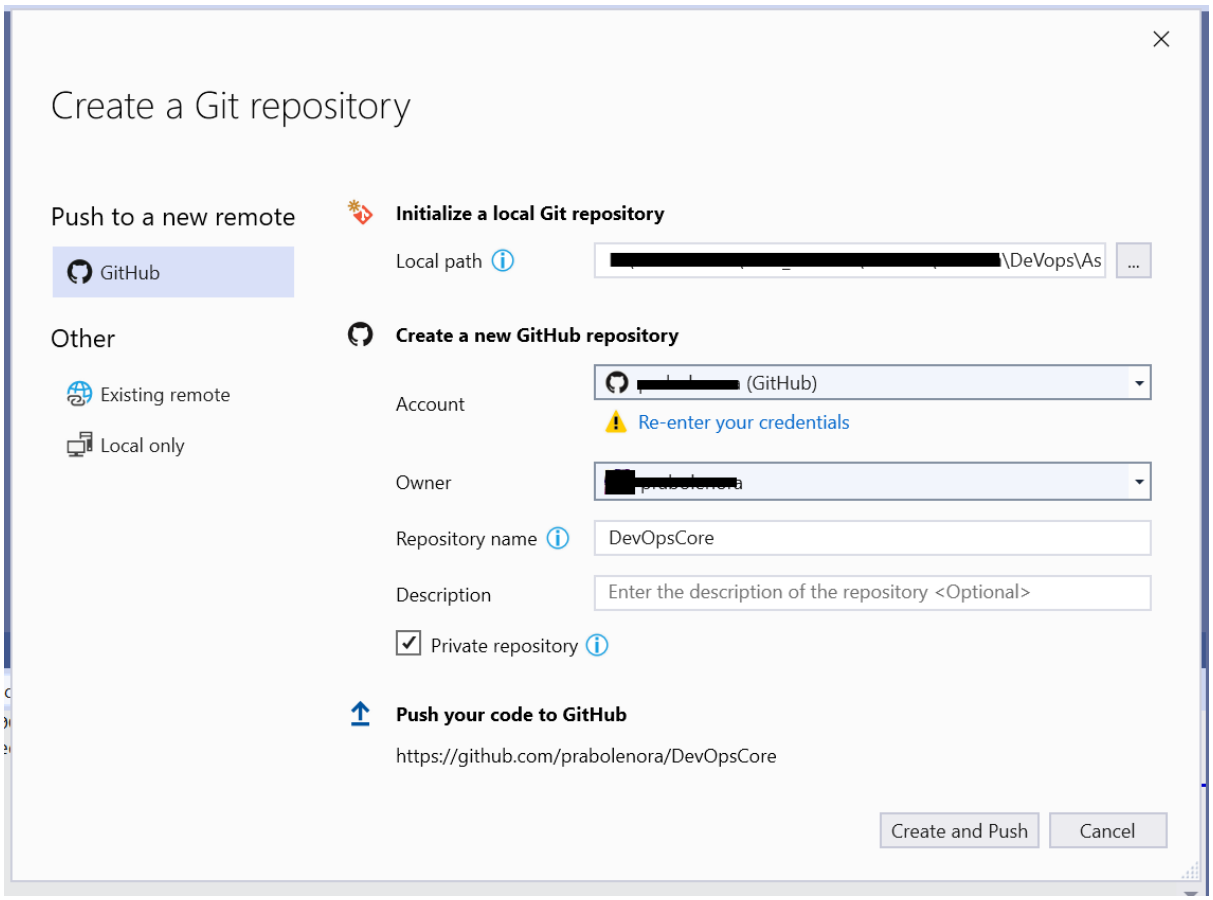
Choose "GitHub" from the drop-down menu in the next new window.

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. 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. 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.

Your project will be made public if you do not pick this option, and anybody may view it using the GitHub platform.

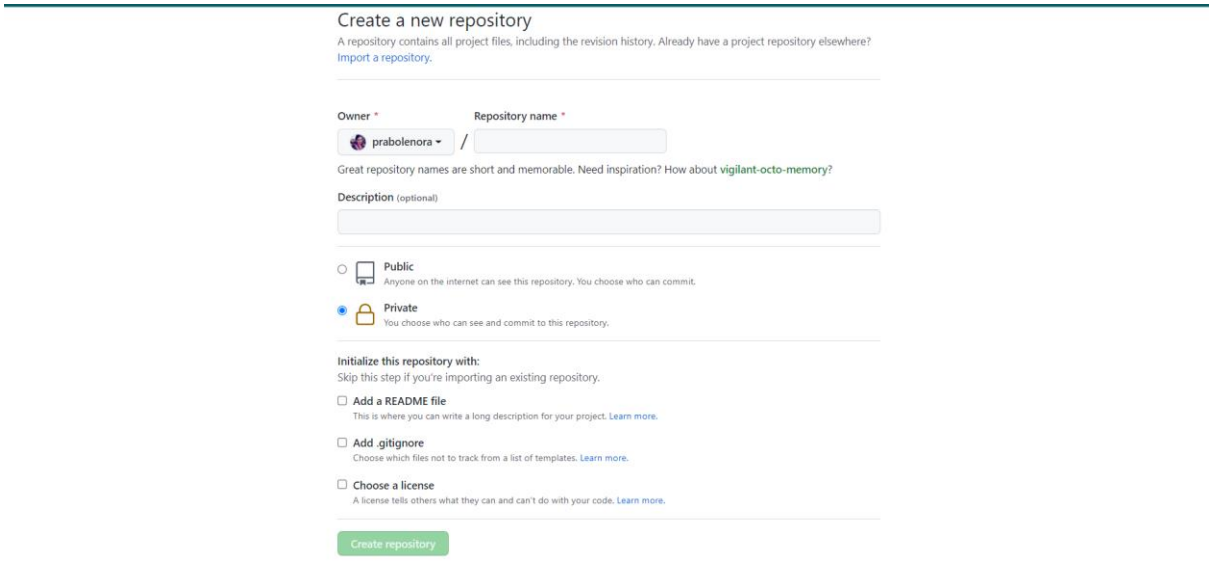


Following a successful commit, you may access your project using your GitHub account.

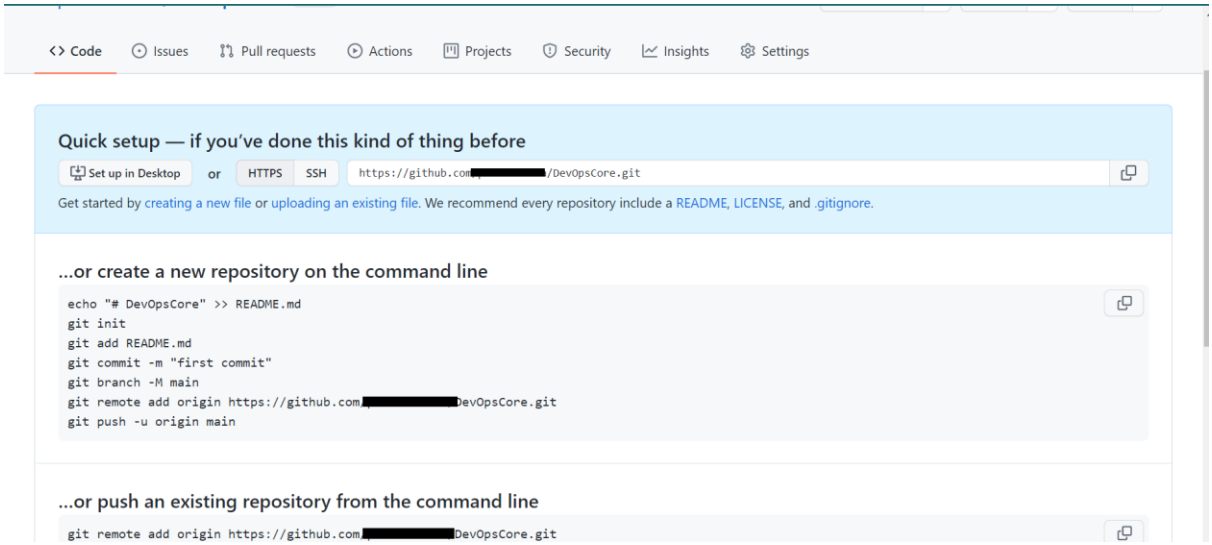
Option 2

You may do this by creating a new repository on your GitHub account.

It includes all of the same choices that we had before when we were using Visual Studio for development. You have the option of making your project public or private. 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.



Your browser will take you to the window below when you have successfully established a repository. From there, you can see the URL for the git clone. Obtain a copy of the URL that will be used to clone your repository to a local folder.



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. 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 "Add" and "Push" choices from the drop-down menu.

Chapter 05

CI/CD Pipeline for .NET with Azure DevOps

5.1 Introduction

In this section, we'll go through two different Azure DevOps choices for .Net Core apps, including CI/CD.

Option 1
.Net Core application from the azure portal

Option 2
.Net Core application from GitHub

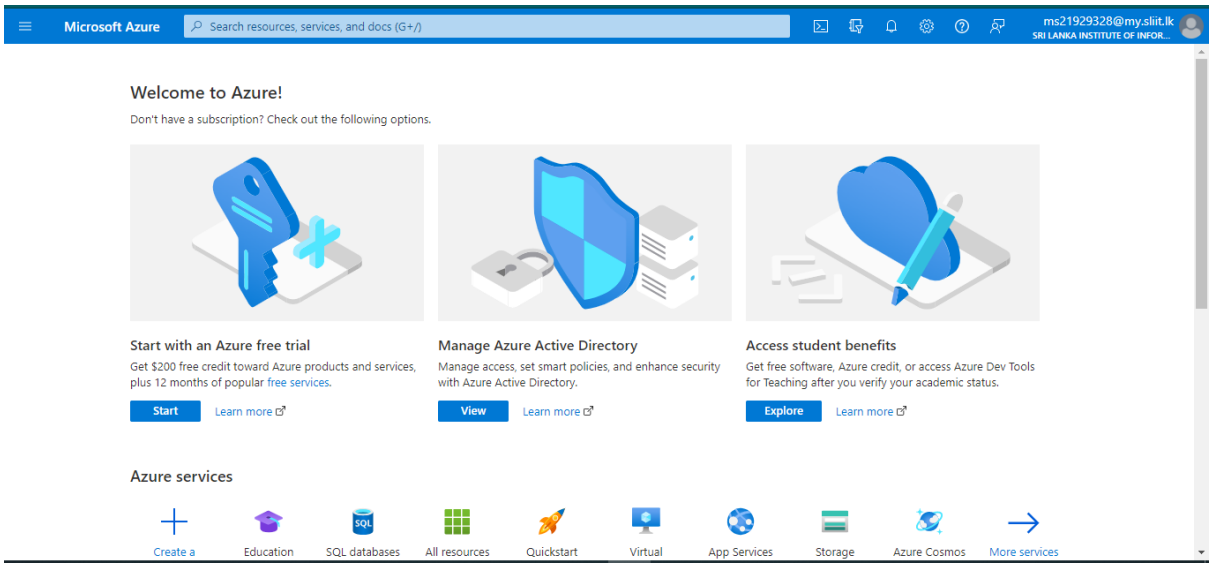
5.2 .Net Core application from the azure portal

- Setting up a .Net Core application using the Azure DevOps
- Take a look at the CI/CD pipelines that the Azure DevOps Project has established.
- Commit the changes to the codebase and run CI/CD.

5.2.1 Setting up a .Net Core application using the Azure DevOps

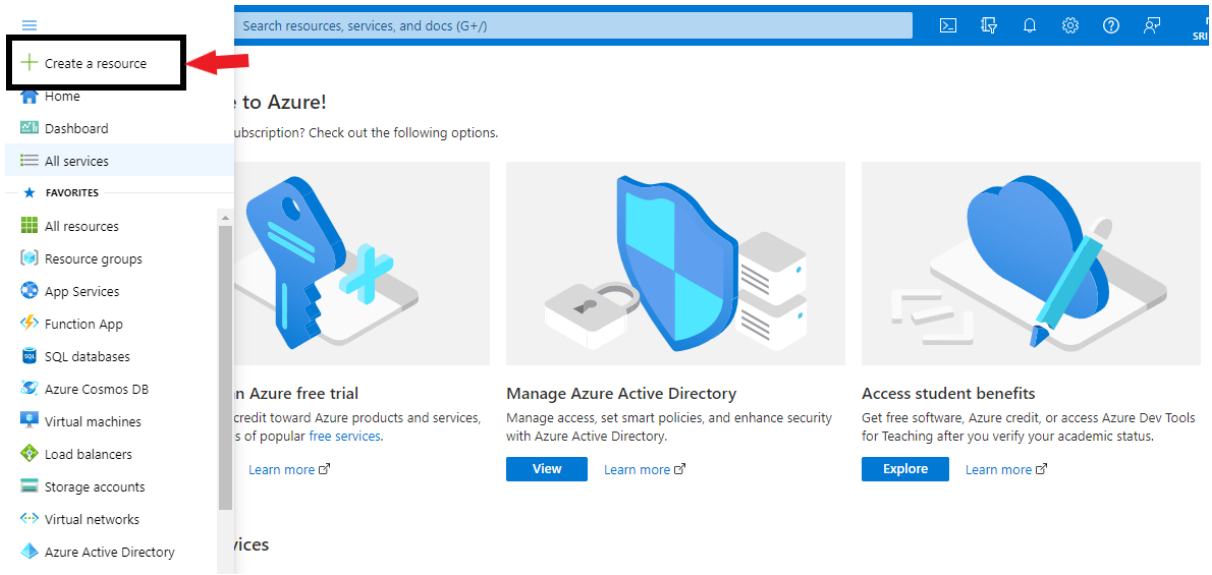
1) Log into your Azure DevOps portal by clicking on the link below.

   https://portal.azure.com/

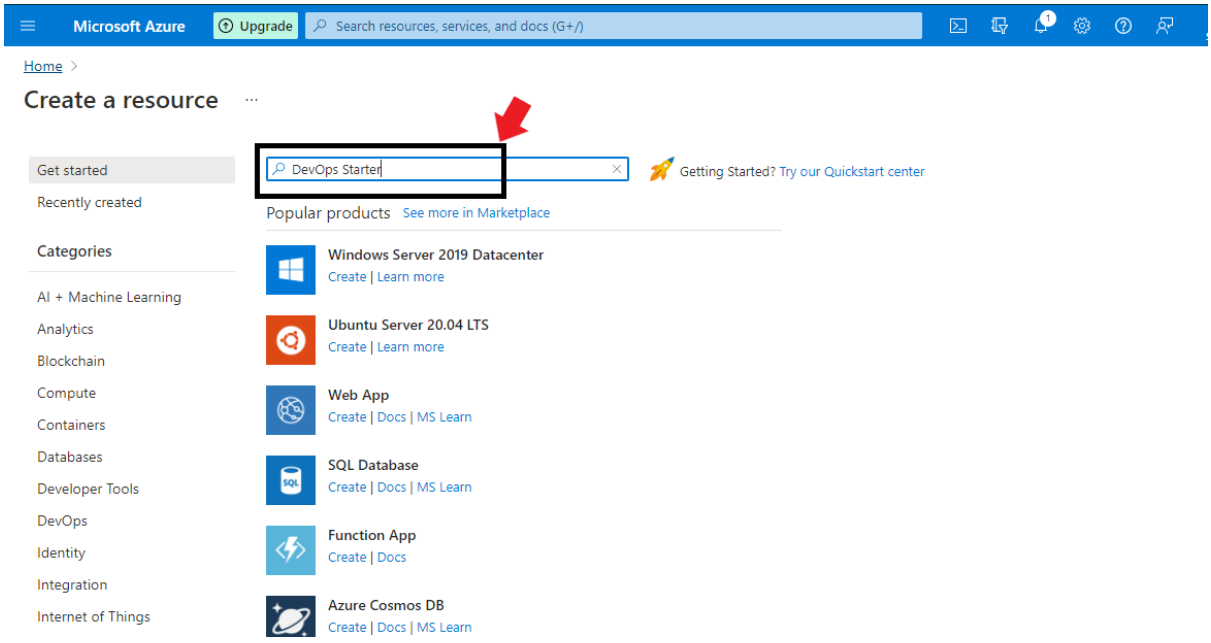


If you don't already have an account, you can sign up for a free new one with a valid email address right now.

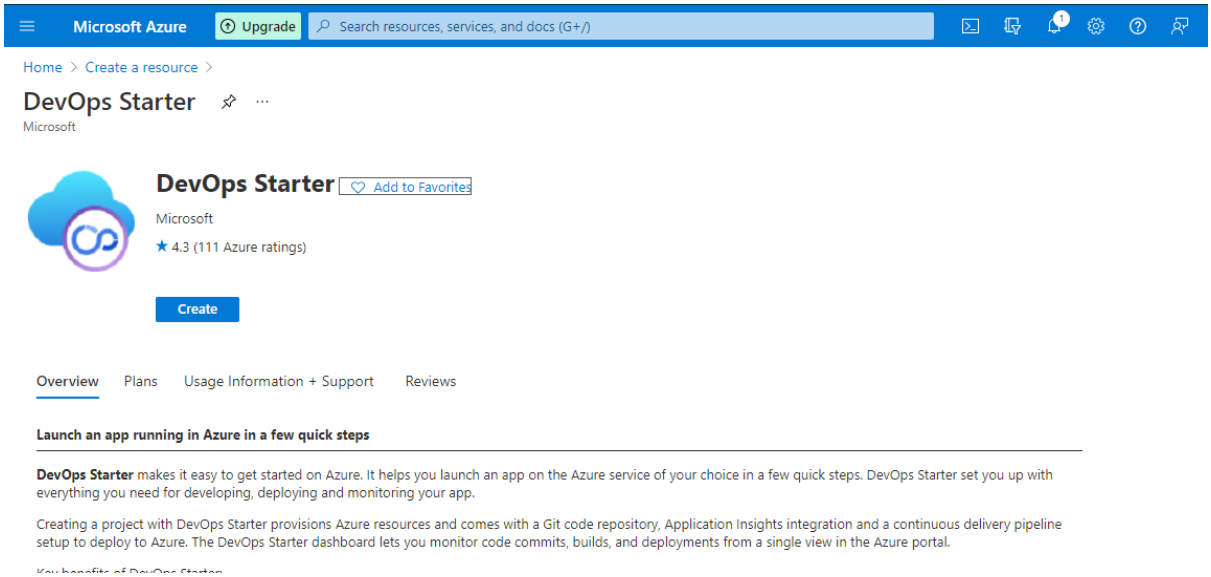
2) Navigate to the navigation bar and click on the '+ Create a resource' button.



3) In the search box, type "DevOps Starter" and press the Enter key to search.

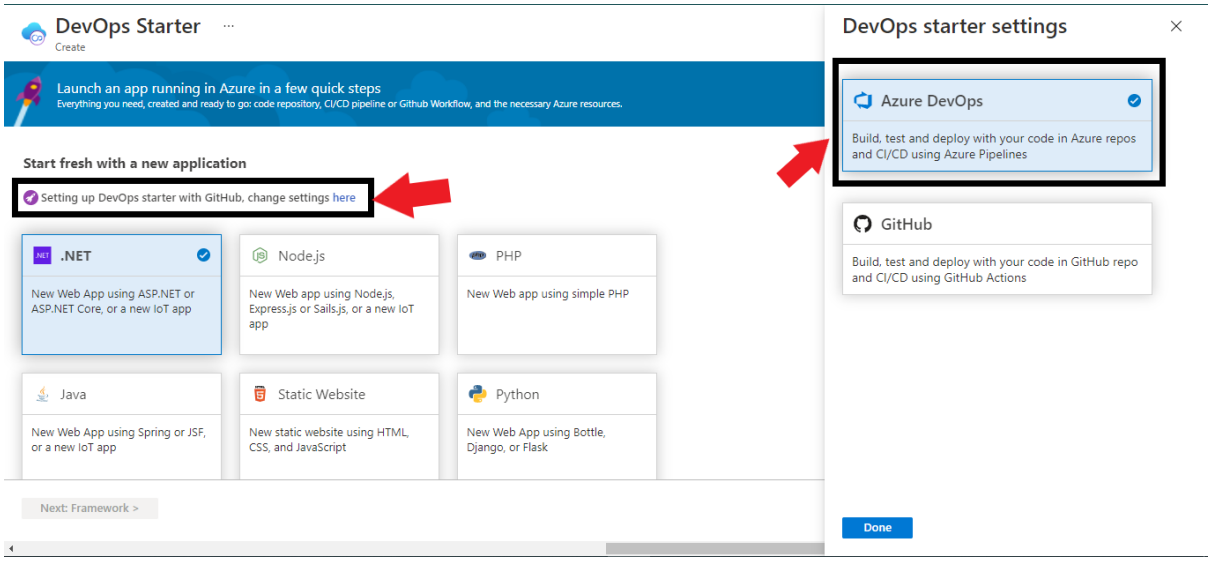


4) You will be sent to a window similar to the one below, where you should click the Create button.

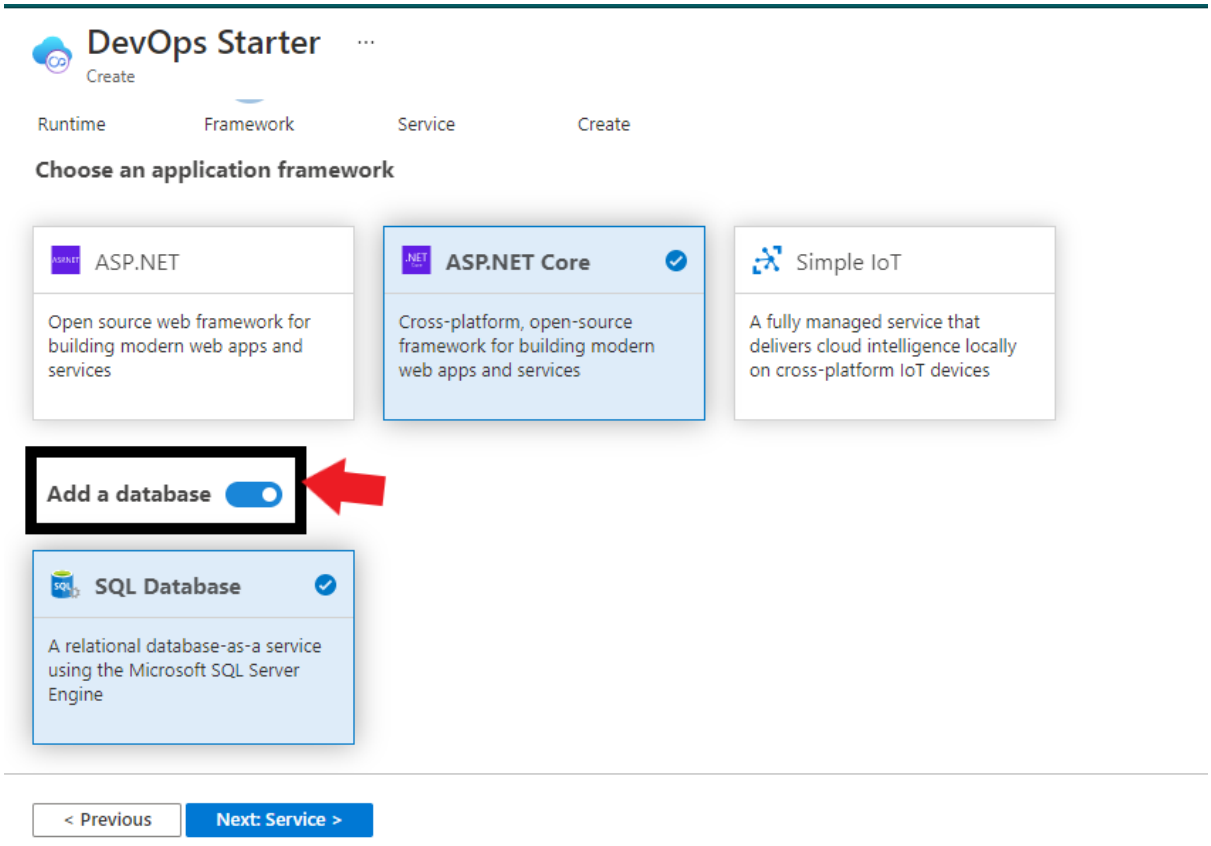


You may be required to register as a free user on occasion. They will ask you a few questions, and you will be required to provide credit or debit card information in order to register. However, we are not required to pay for anything. It's a completely free account. They will ask for your phone number in order to verify your identity.

5) After selecting the.Net example application, press the "Next" button.

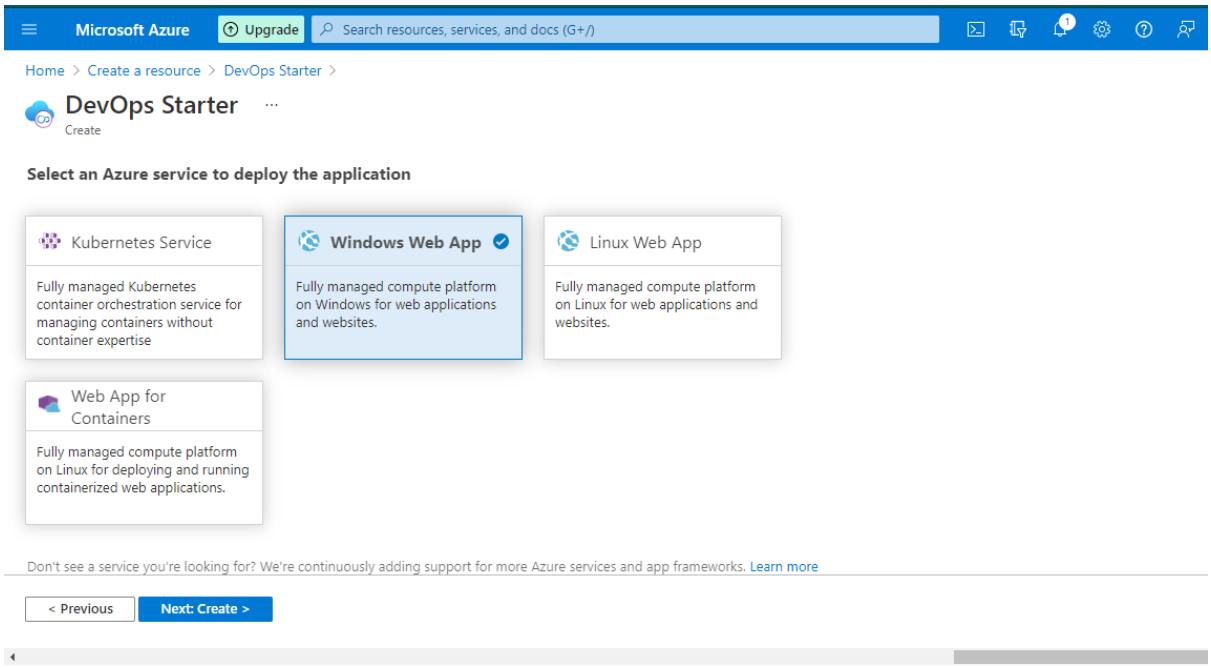


6) Select the.Net Core application and the database from the next window, and then go to the service window to complete the installation.

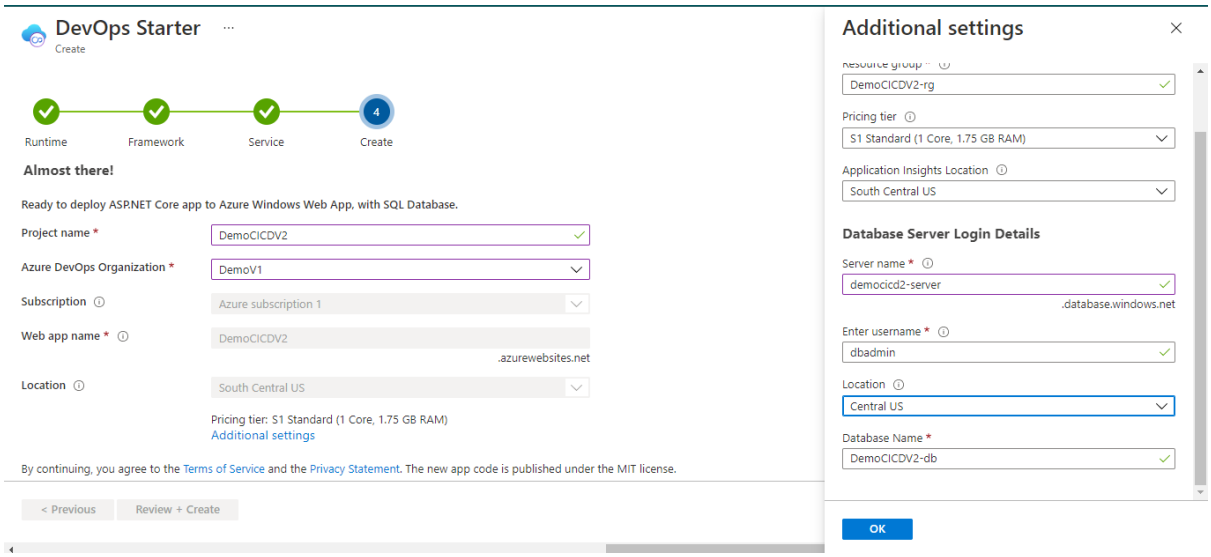


The ASP.Net program is available as free software. ASP.NET Core is a cross-platform Net program and ASP.NET is a cross-platform Net application. We're going to use it in this case. This project is an MVC project that uses a Net Core application.

7) Select "Windows Web App" from the drop-down menu in the service window and press the next button.



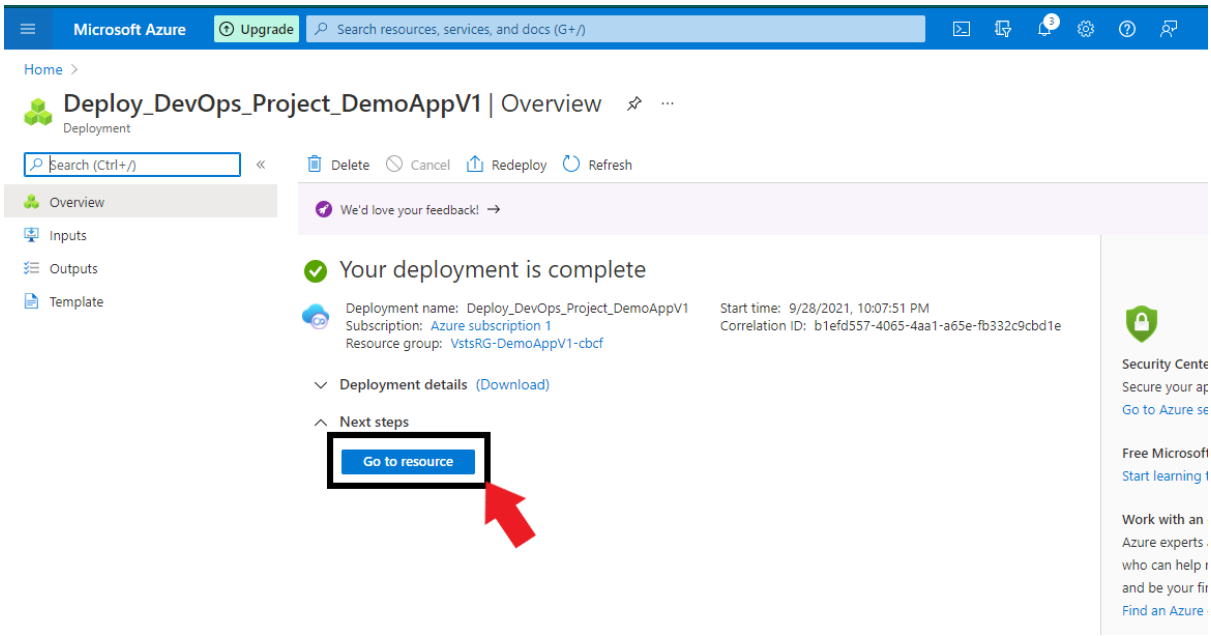
8) Identify your organization and give your project a descriptive name in this box.



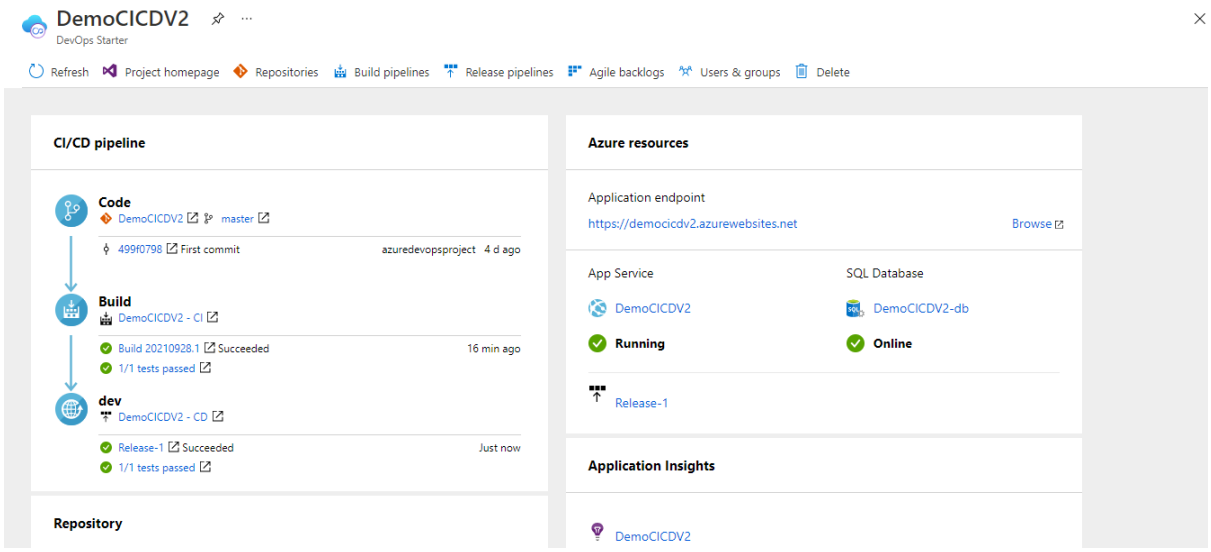
Additional options may be seen by selecting "Additional Settings" from the main menu. You may discover the information about your database in this section. You may make changes to the specifics if you so choose.

If you have finished filling out the text boxes, click on the "Review + Create" button to proceed.

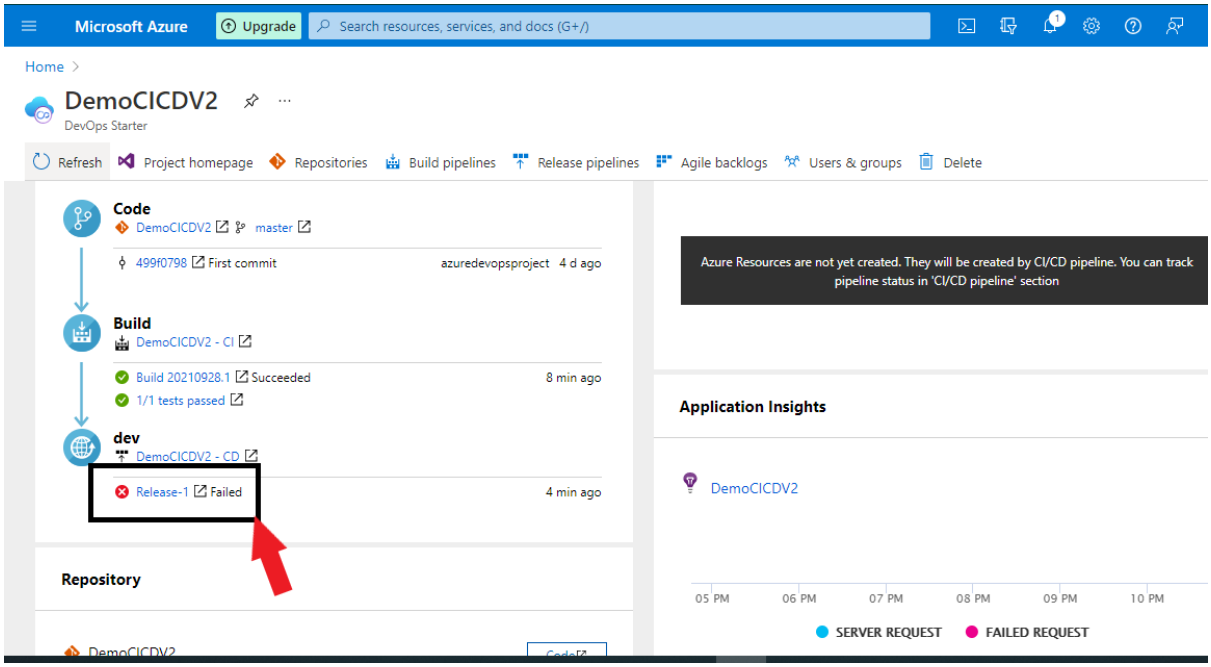
9) After completing the deployment, you will get an email confirming the completion of the deployment and directing you to the resource.



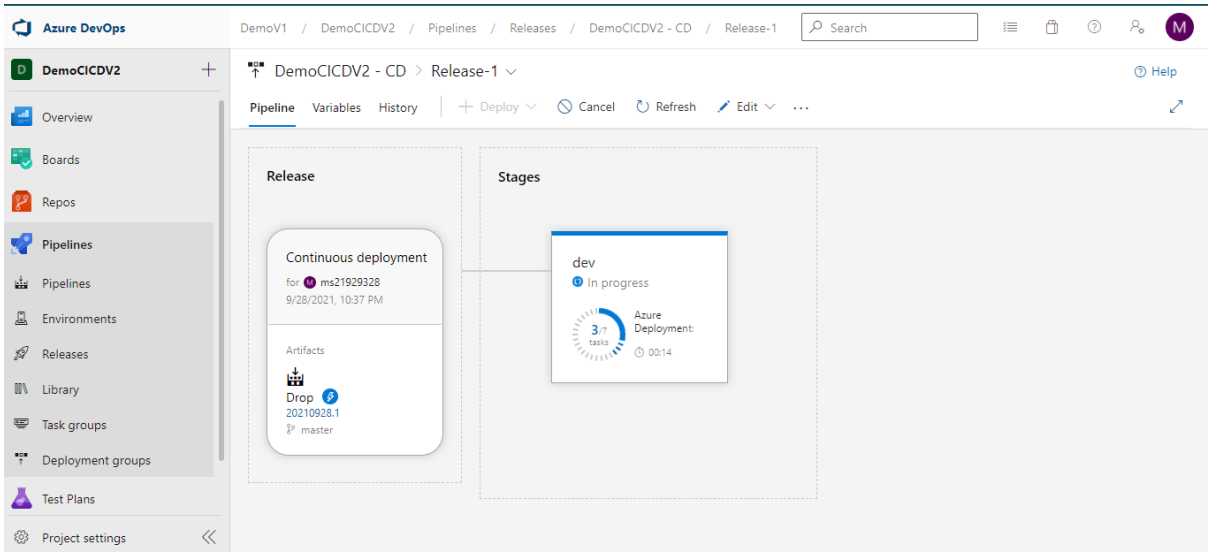
10) Occasionally, they will ask for permission to access your GitHub account via the dashboard.



12) In any event, if you encounter a problem with the procedure, you may retry it as shown below.

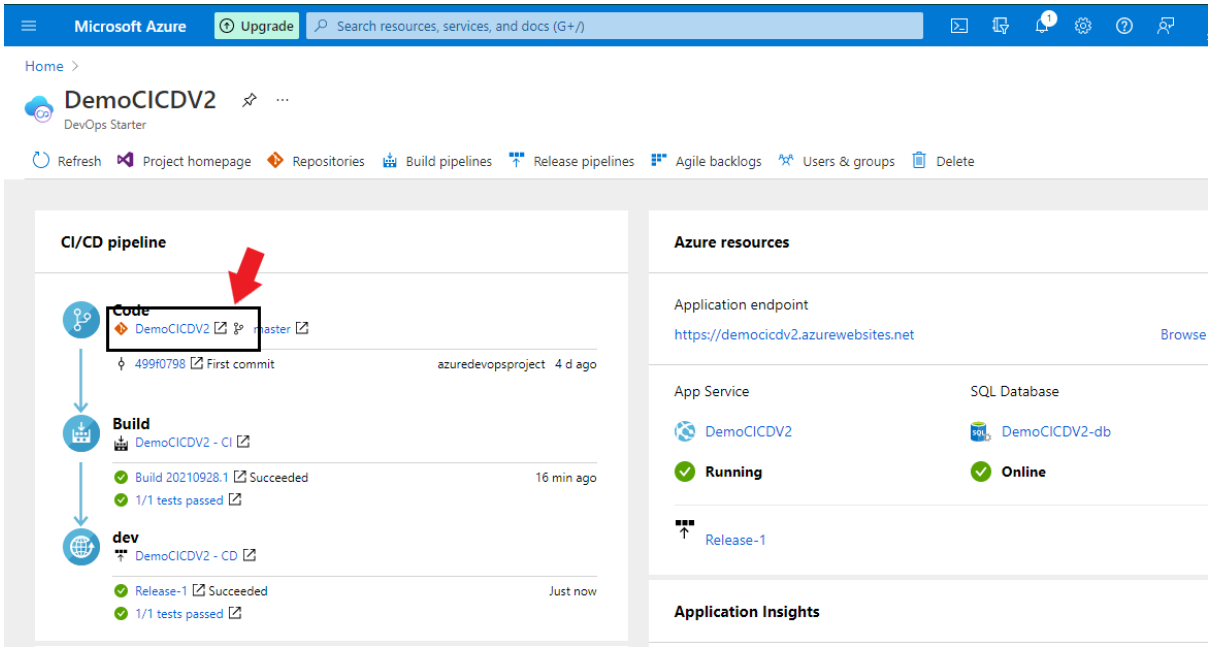


Click on the unsuccessful process, which will take you to the following window, where you may retry the step.

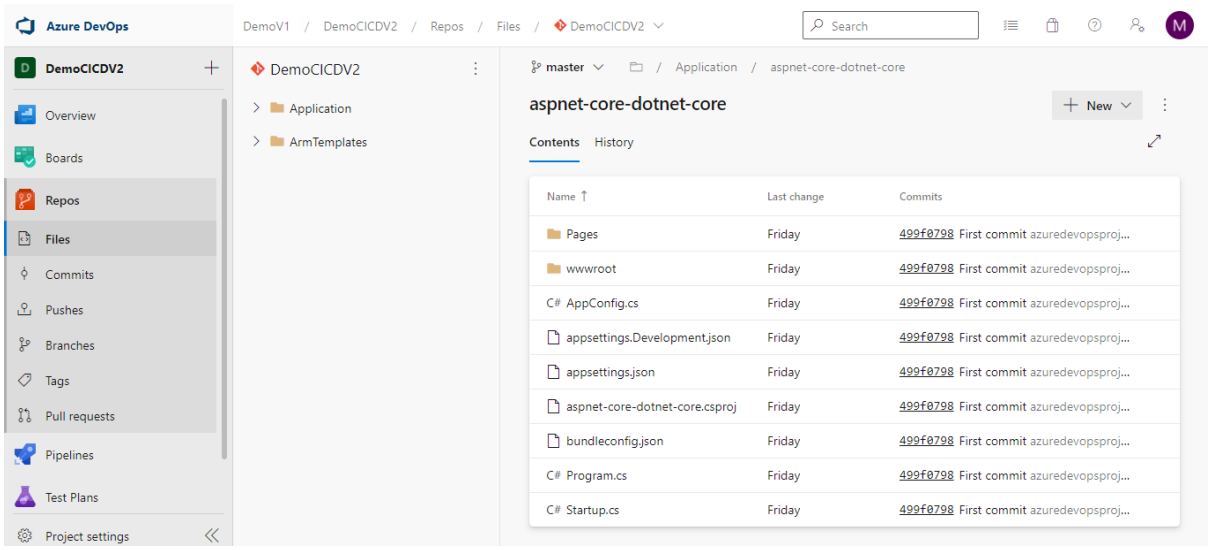


We now have a.Net Core application with a continuous integration/continuous delivery mechanism that automatically deploys the most recent modifications to our website.

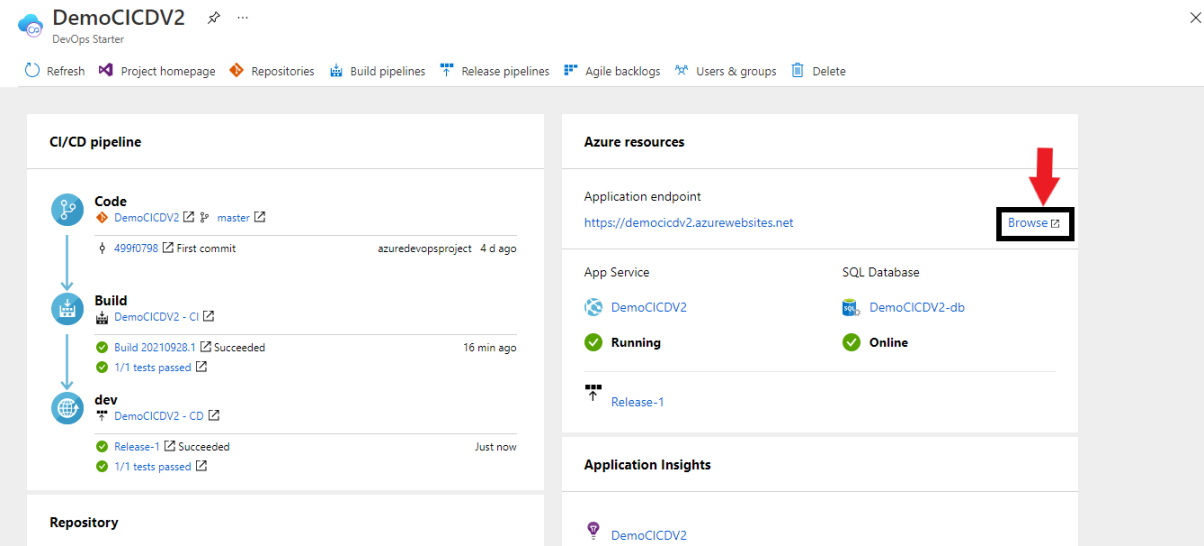
If you wish to look at the source code, you may do so by selecting "Code" from the drop-down menu in the CI/CD pipeline.



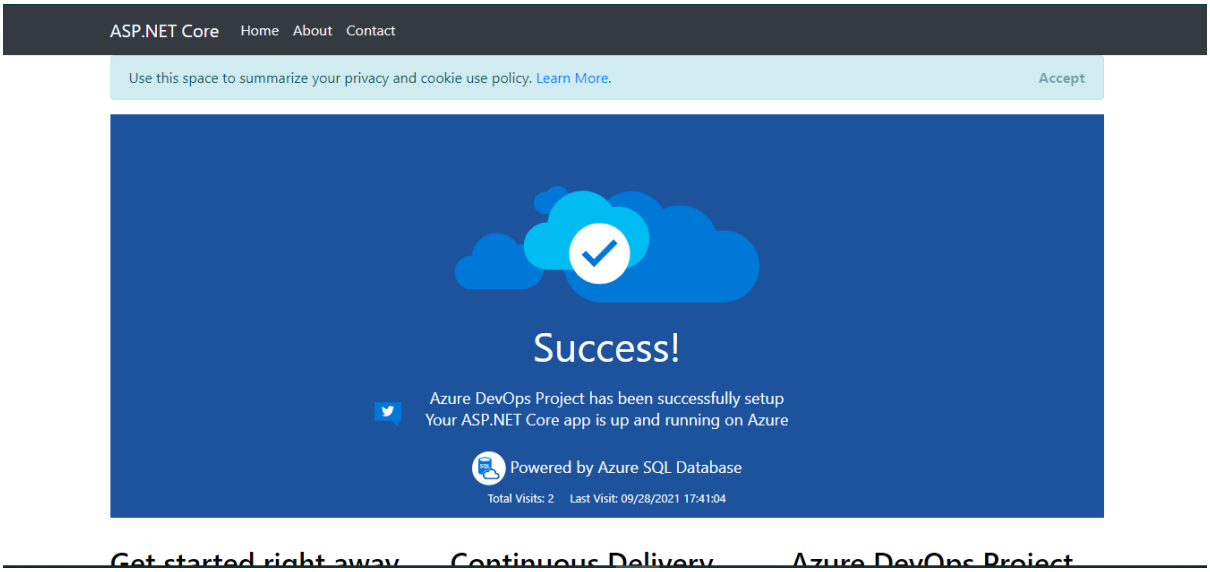
The source code for your project will be opened in your Azure DevOps account.



13) To see your current applications, choose "Browse" from the drop-down menu.



This is our project that has been deployed.

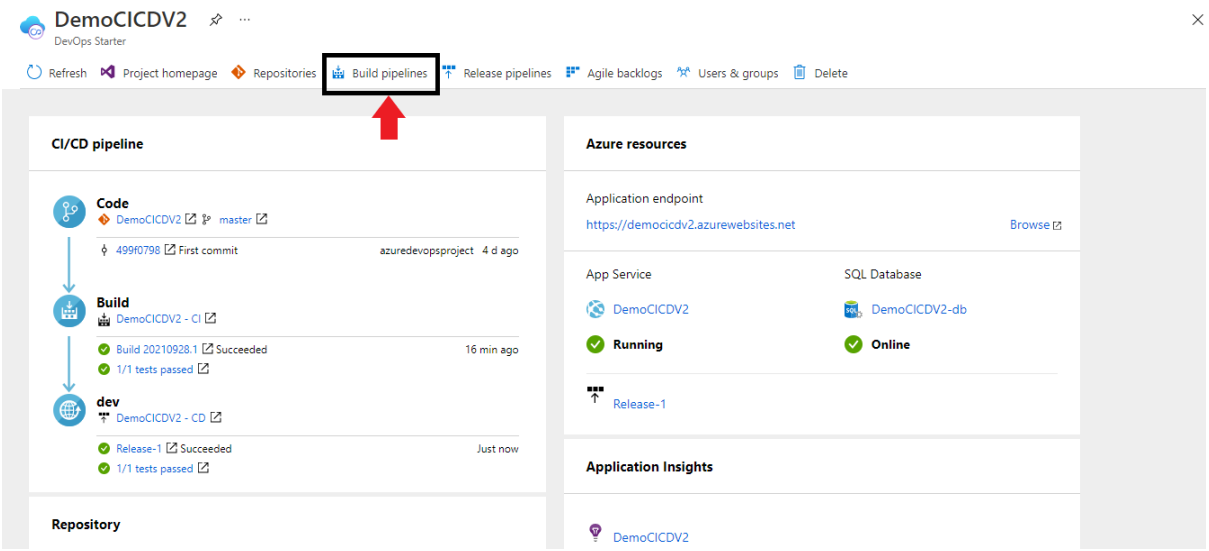


5.2.2 Take a look at the CI/CD pipelines that the Azure DevOps Project has established.

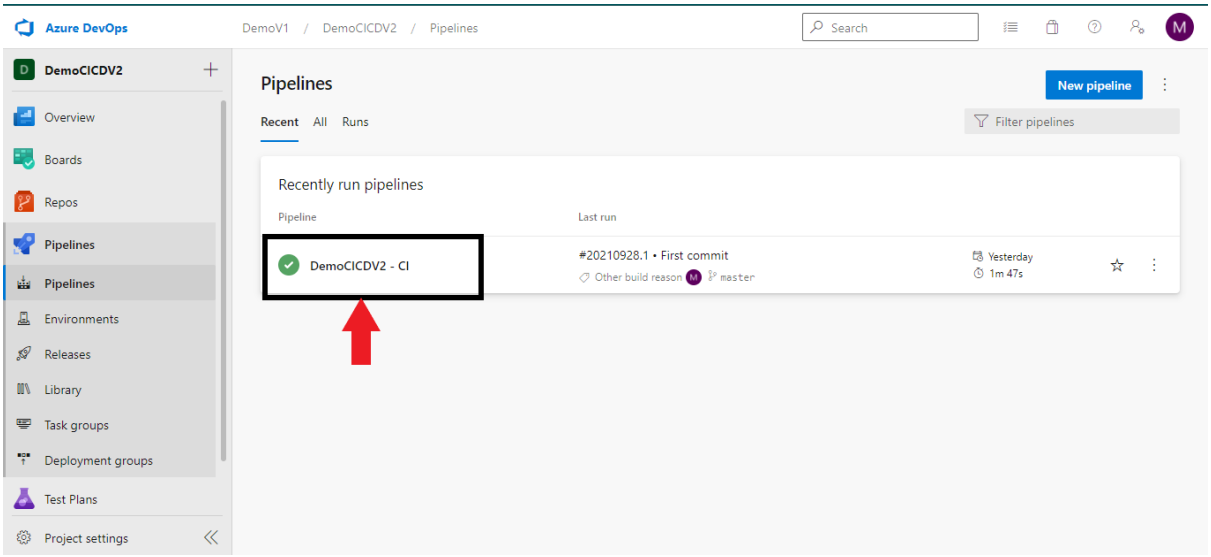
The Azure DevOps project automates the creation of a continuous integration/continuous delivery pipeline for your Azure DevOps company. You have the option of seeing or customizing the pipeline to meet your specific requirements.

Now we're going to construct Azure DevOps and make the pipeline available to the public.

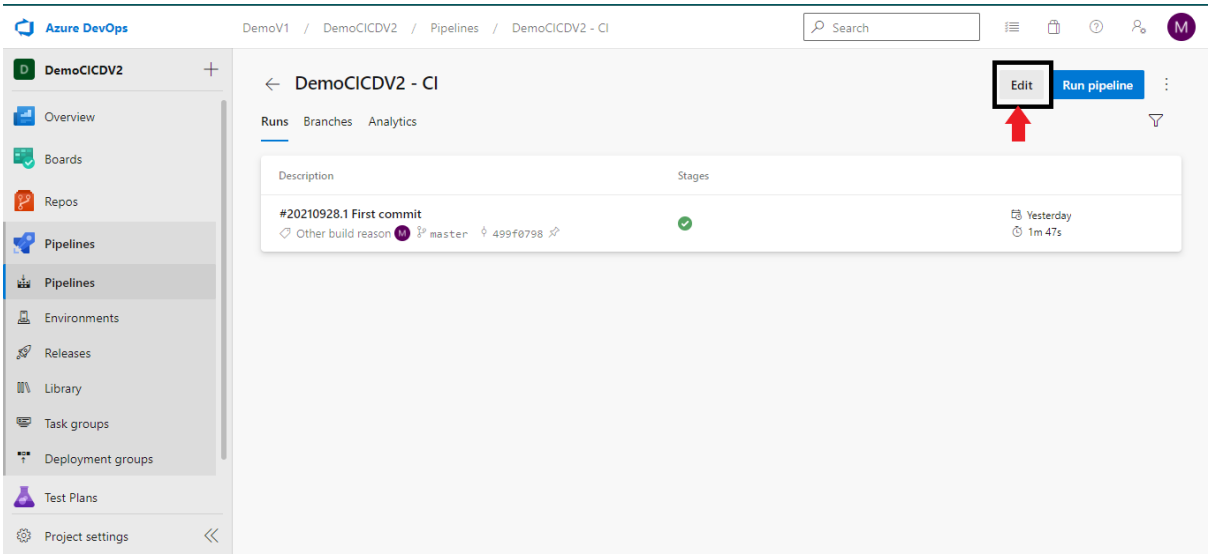
1) In the dashboard, choose "Build Pipelines" from the drop-down menu. It will direct you to your project from inside your Azure DevOps account, and Azure DevOps will construct a pipeline for your project on your behalf.



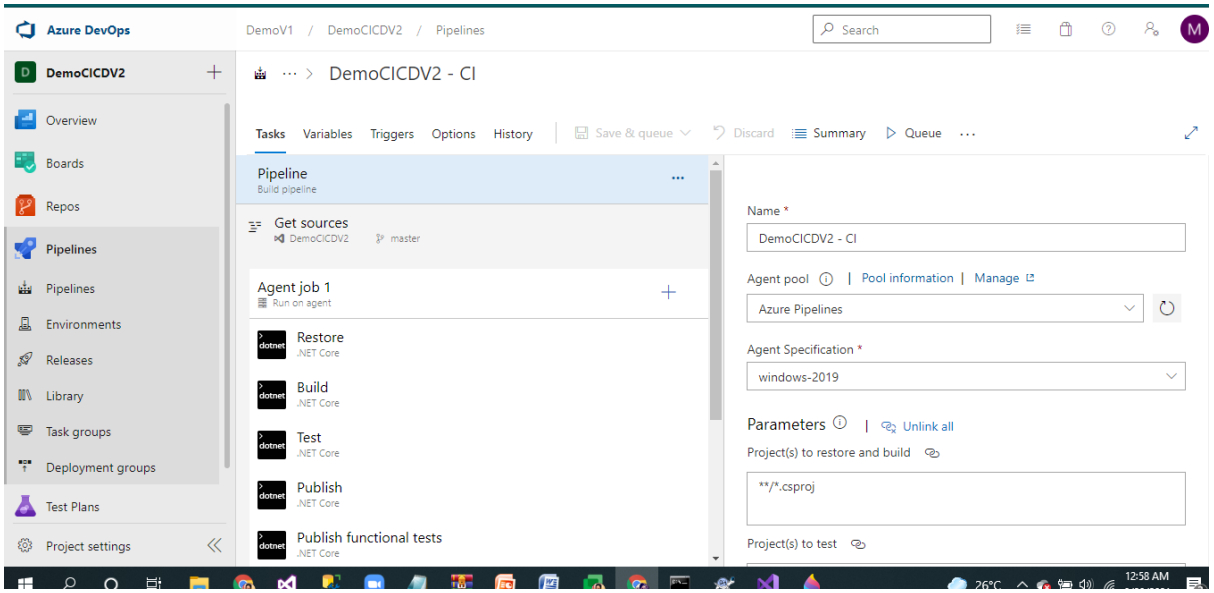
2) Choose a project to work on.



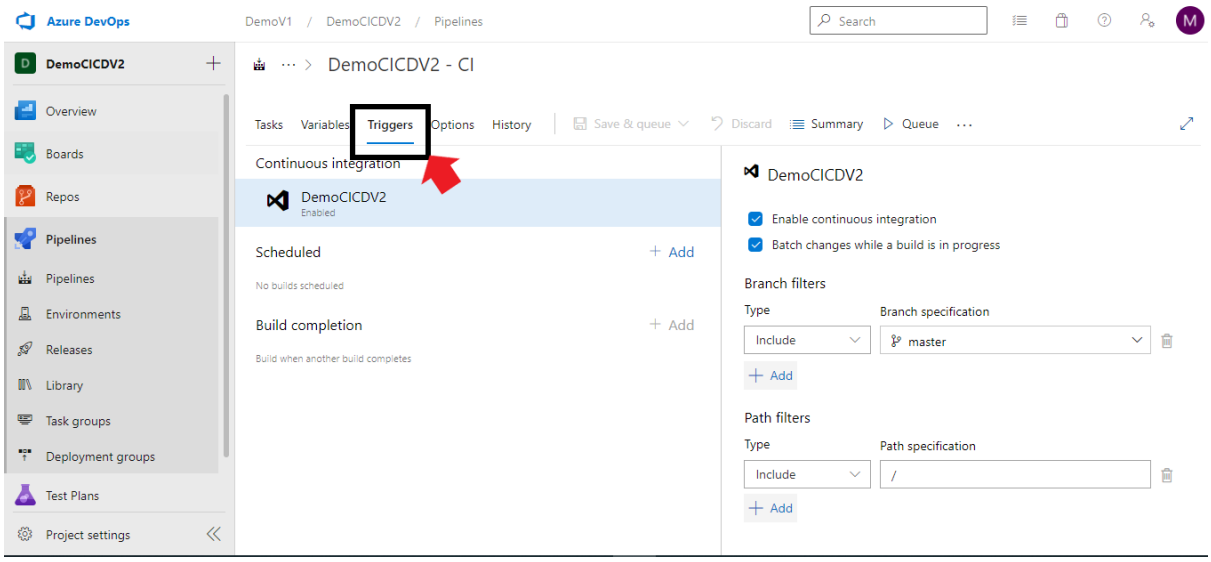
3) Hit on “Edit” button



4) The pipeline overview view allows you to inspect different aspects of your pipeline and do actions like as importing changes from the Git repository, restoring dependencies, compiling, running a test run, and publishing your project.

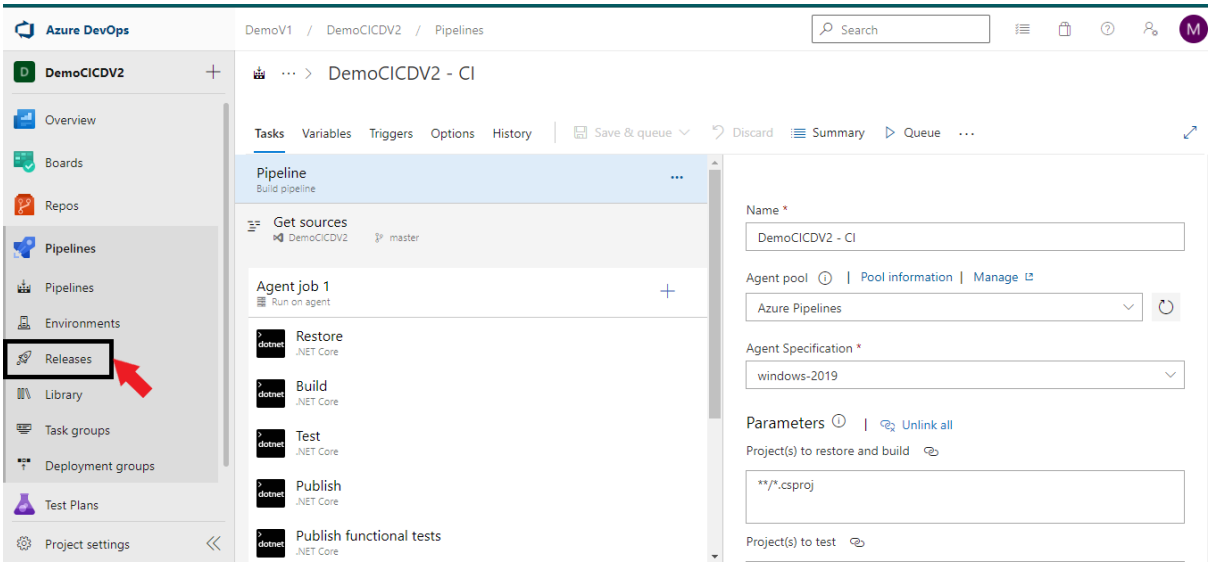


5) Select the "Trigger" tab from the drop-down menu. When we start an Azure DevOps project, the CI trigger will be built for us automatically.



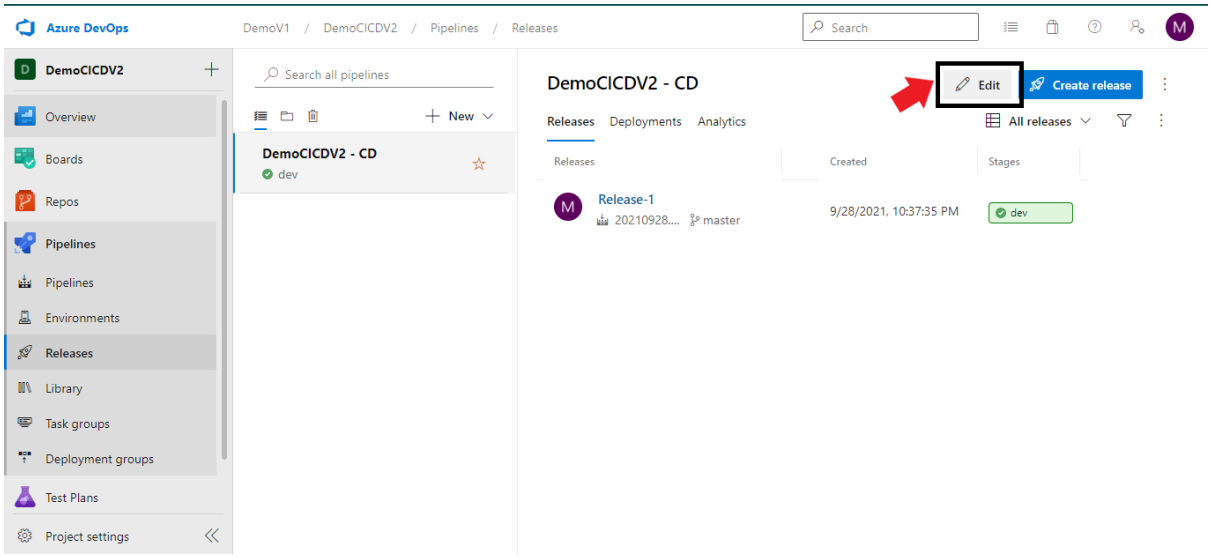
You may choose whatever branch you wish to include in the continuous integration pipeline from this window.

6) Select the release option

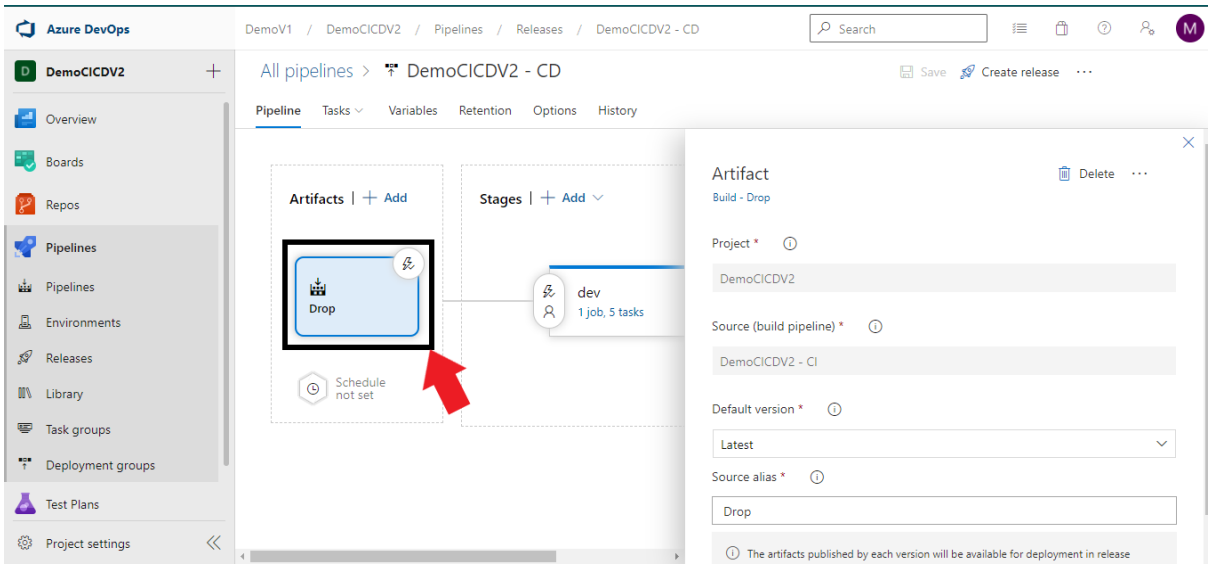


You may manage Azure installations with the help of the release pipeline.

7) In the Releases tab, click on the "Edit" button to the right.

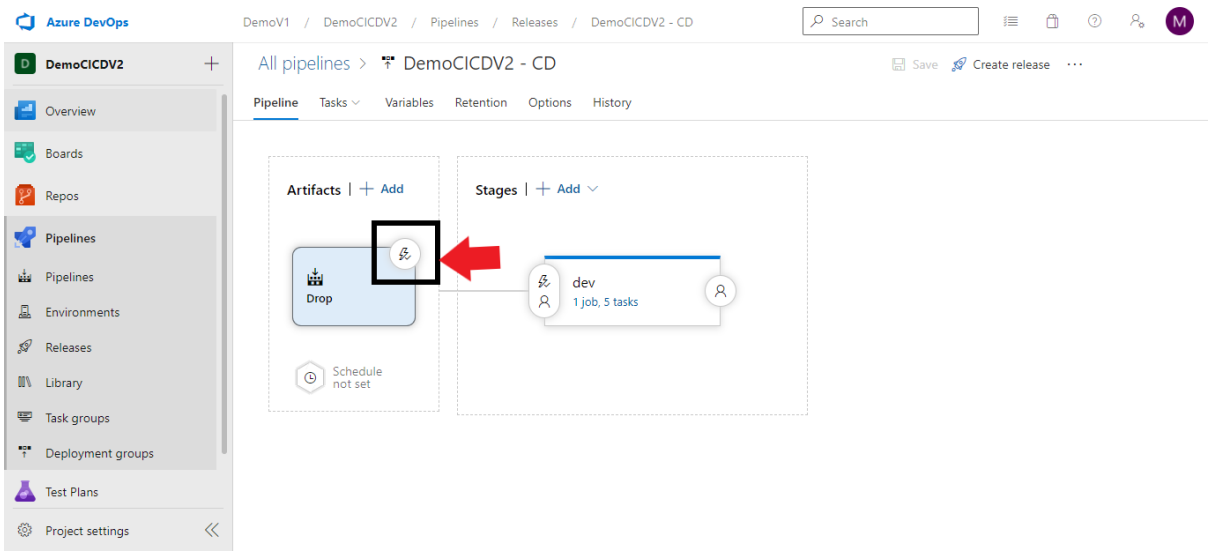


8) Select the “Drop” under “Artifact”.

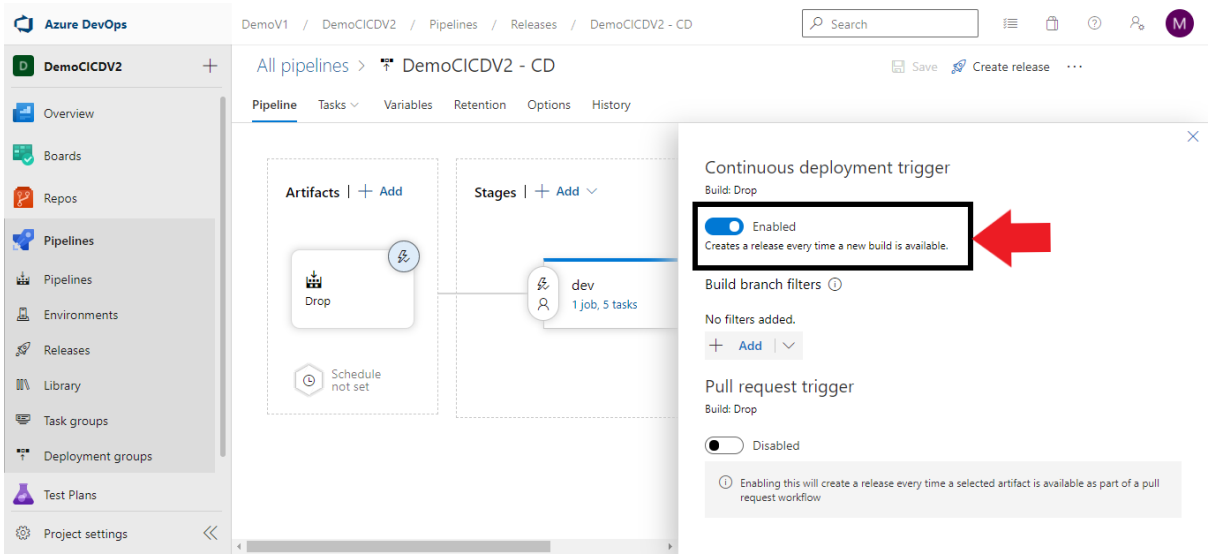


In this case, the output from the preceding step of constructing the pipeline will be shown.

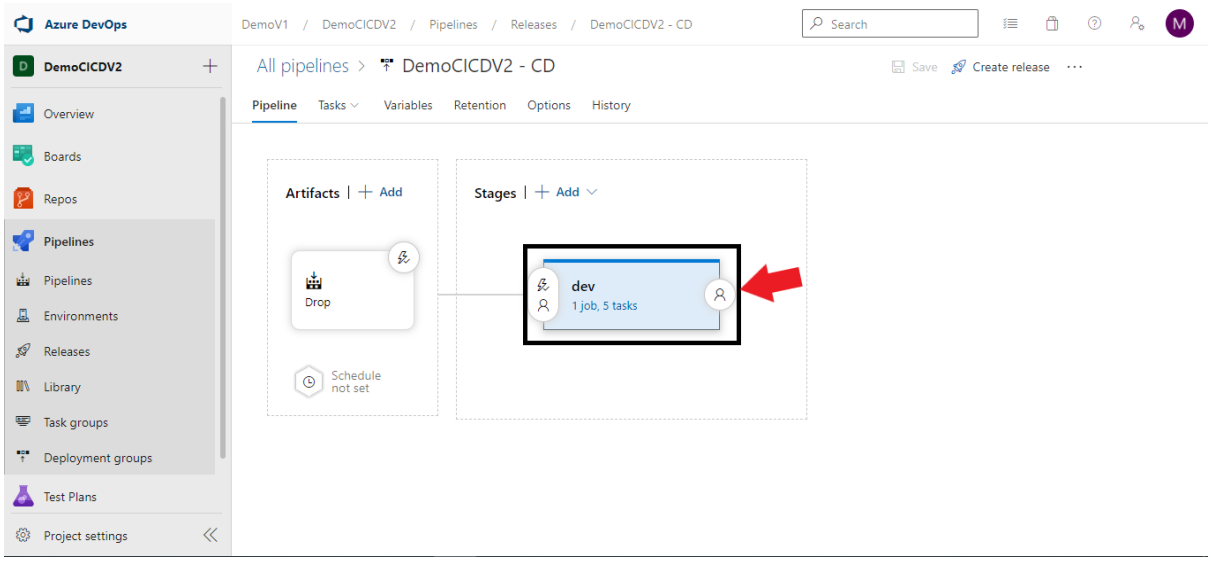
9) To activate the continuous deployment trigger, find it at the top of the "drop" and click on it.



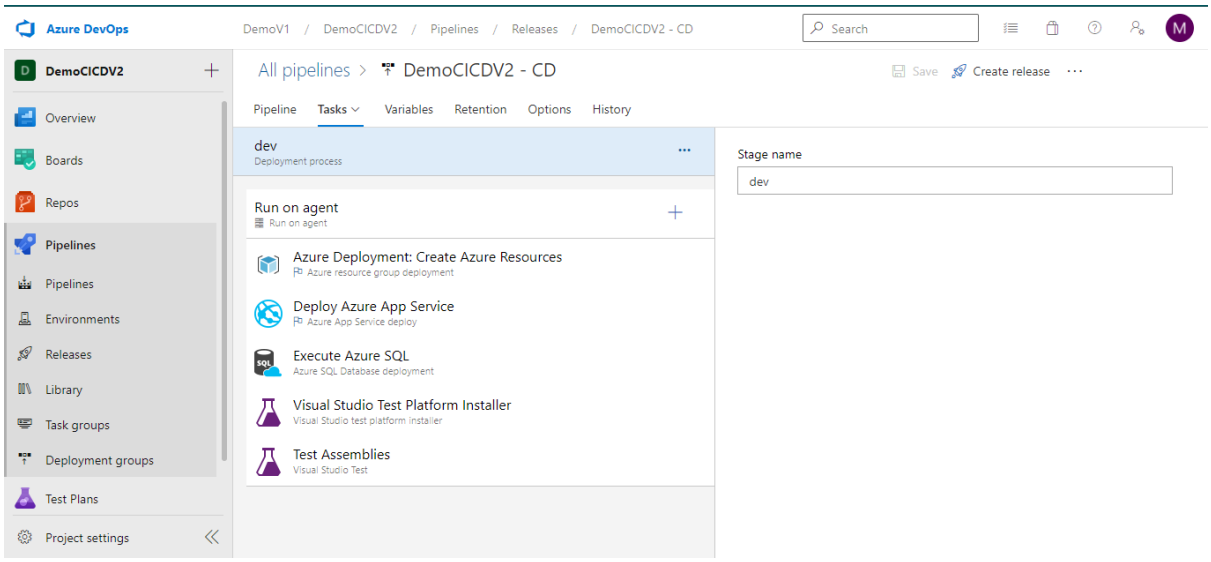
10) In this box, you have the option of enabling or disabling continuous deployment, which is enabled by the release pipeline. In the event that this option is activated, each and every deployment will generate a new artifact. When manual execution is required for your deployment, you may turn off the option to enable it.



11) Now click on the “Task”



12) The task window will show all of the tasks that occurred throughout the deployment process, and they may be accessed from there.



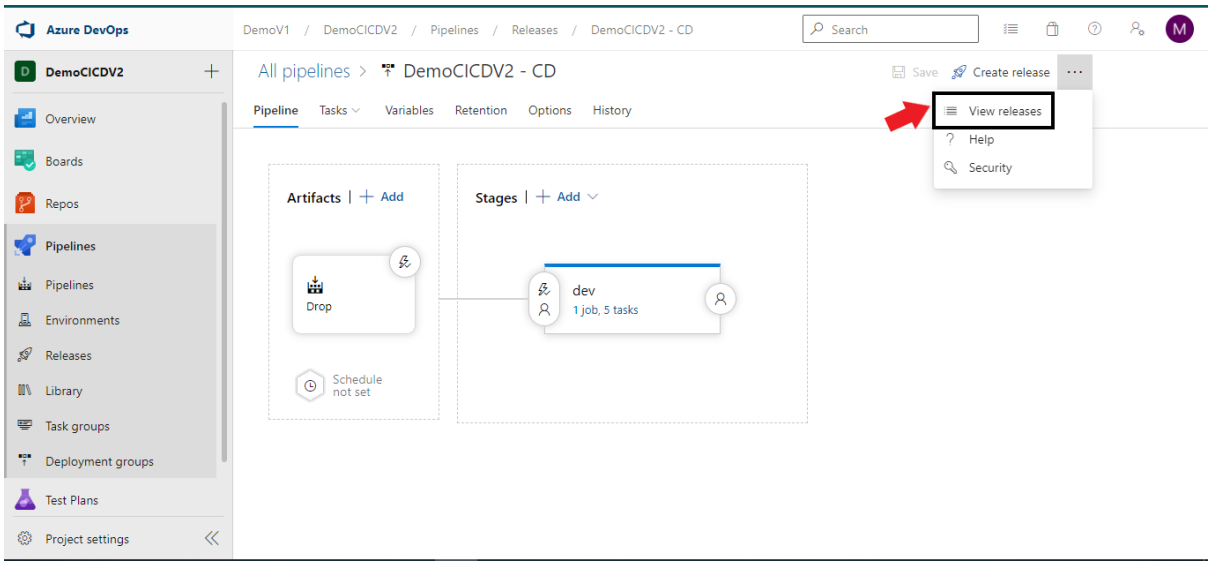
Azure Resource Group Deployment - This will deploy the required resources, such as the SQL database and the Azure Web App, to the appropriate locations.

Azure App Service Deploy - This will cause the application package to be uploaded to the website.

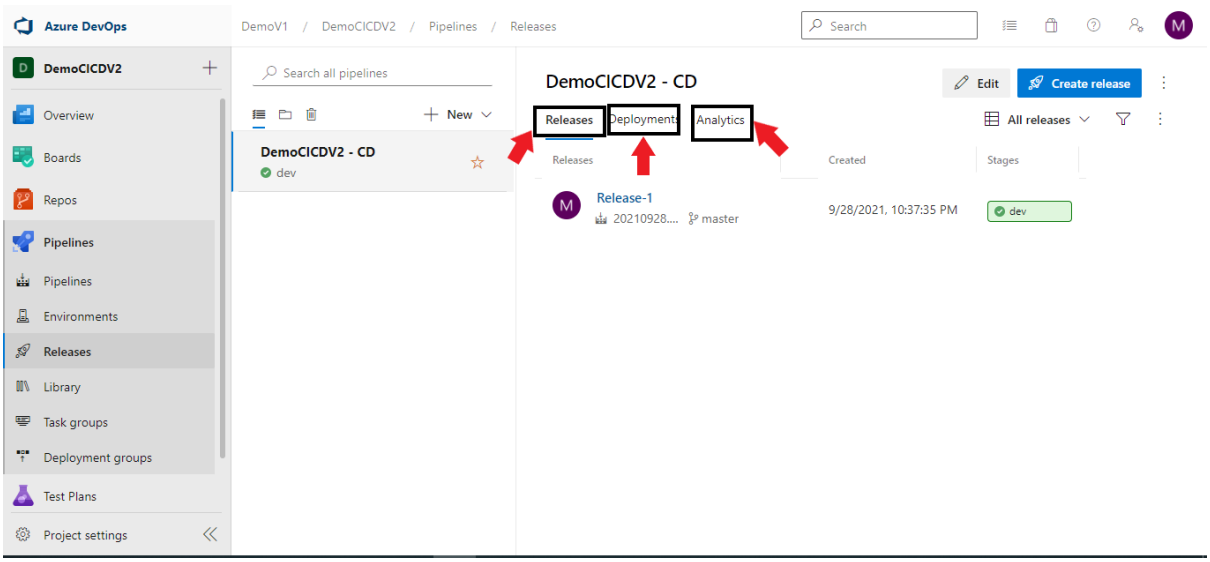
Azure SQL Database deployment - This will make the database updates available to SQL.

Visual Studio Test - After a successful deployment, this will cause the test function to be invoked.

13) Now click on the “Release”



14) As an overview of the releases, you may see the history of the releases from the release window.

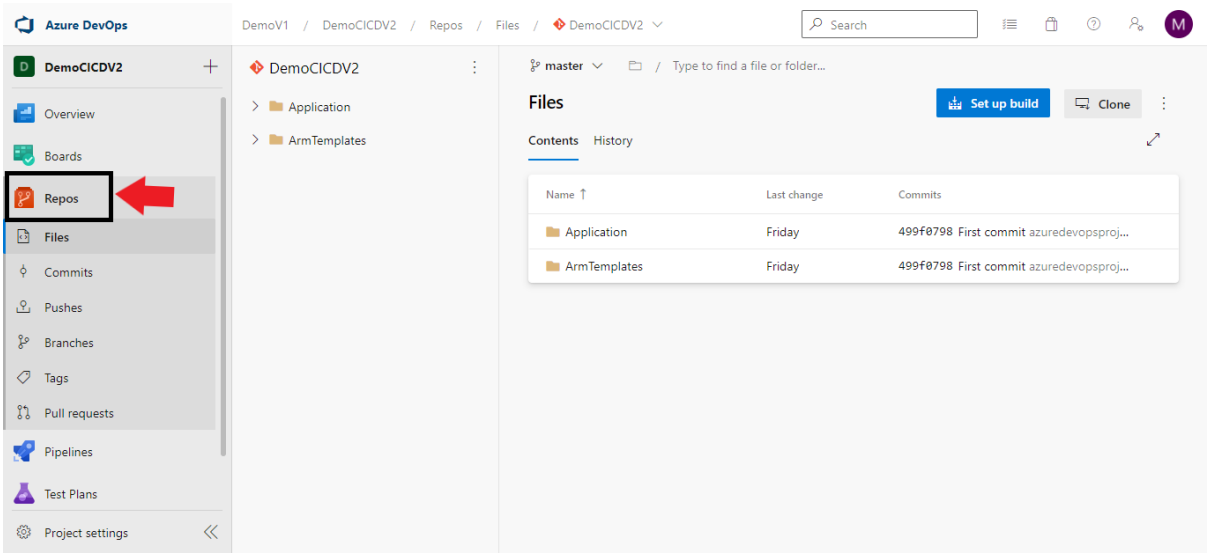


There are a number of tabs accessible from this location. These all tabs provide you with a history of the releases as well as an overview of the releases using graphs.

5.2.3 Commit the changes to the codebase and run CI/CD.

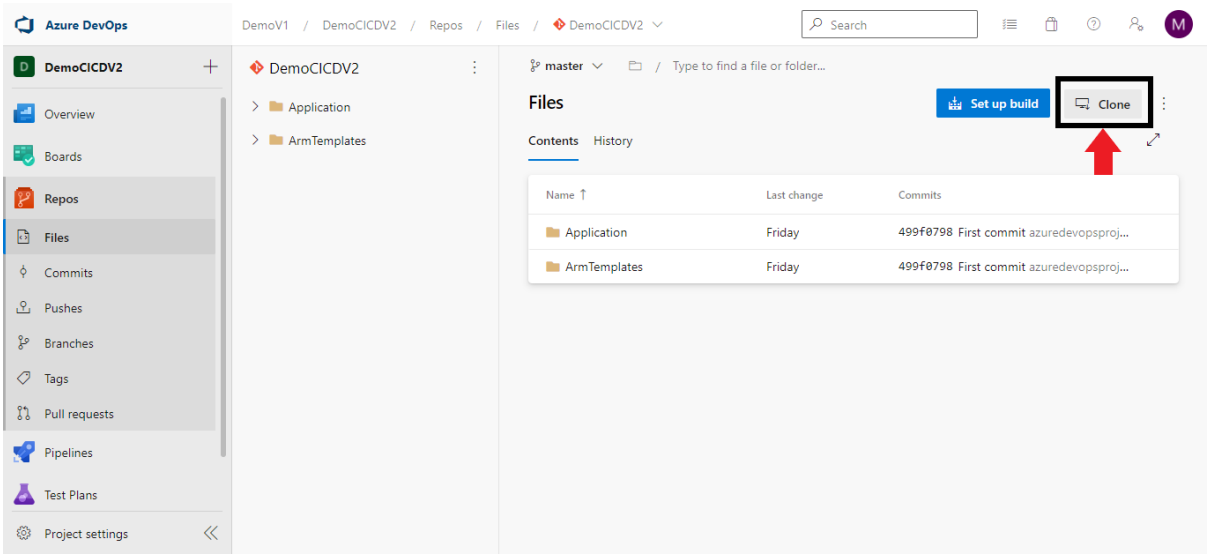
For your business, Azure DevOps will automatically generate a Git repository.

1) Click on the “Repos”

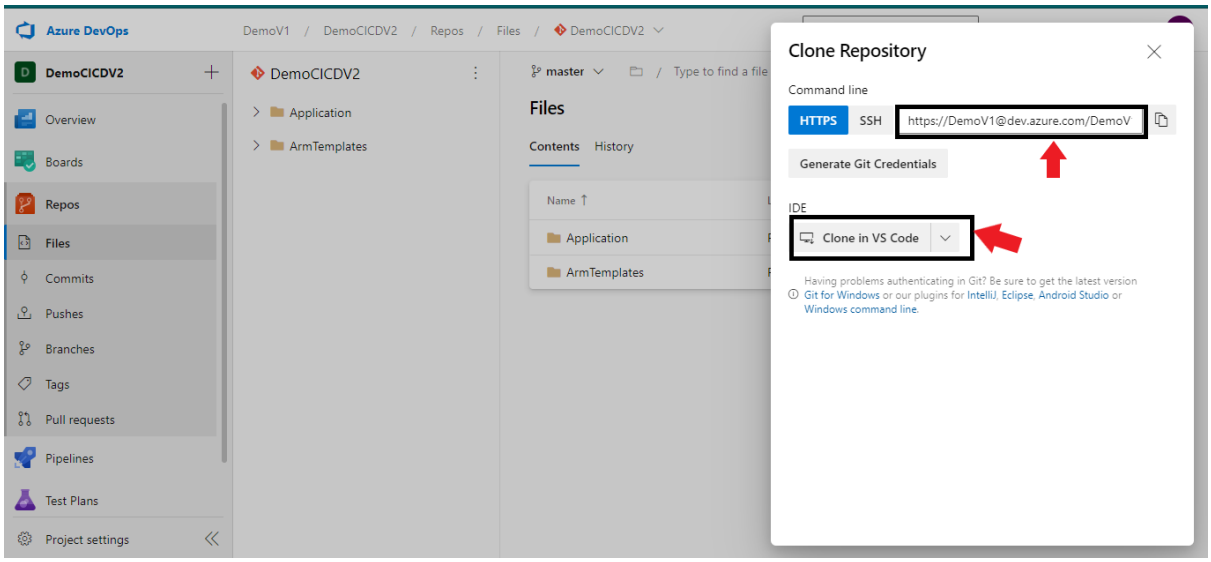


This command will show the Git repositories for the Azure DevOps projects.

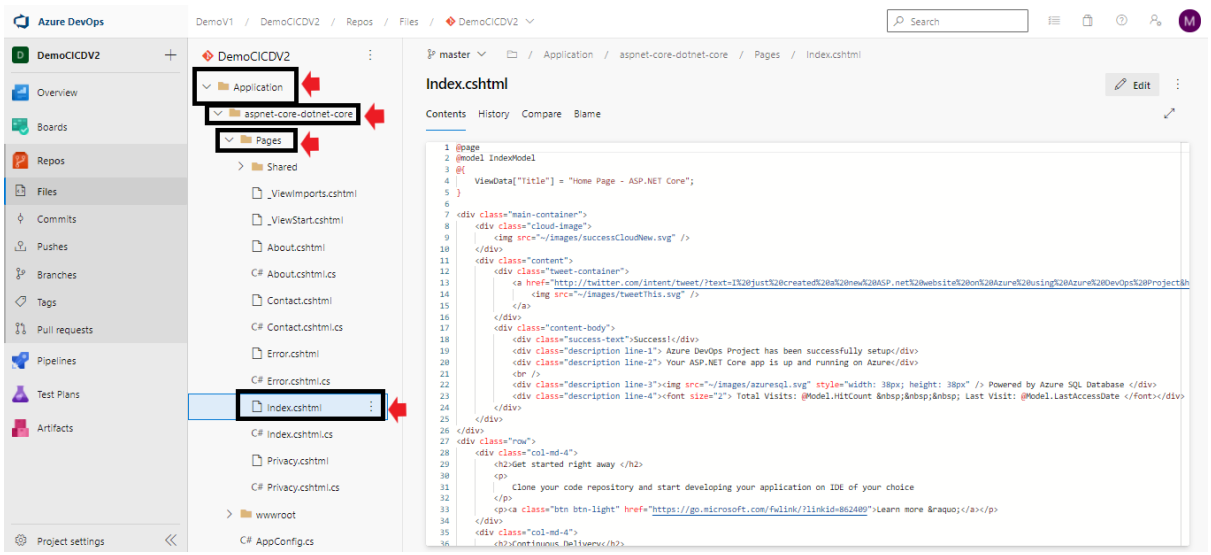
2) Click on the “Clone” button



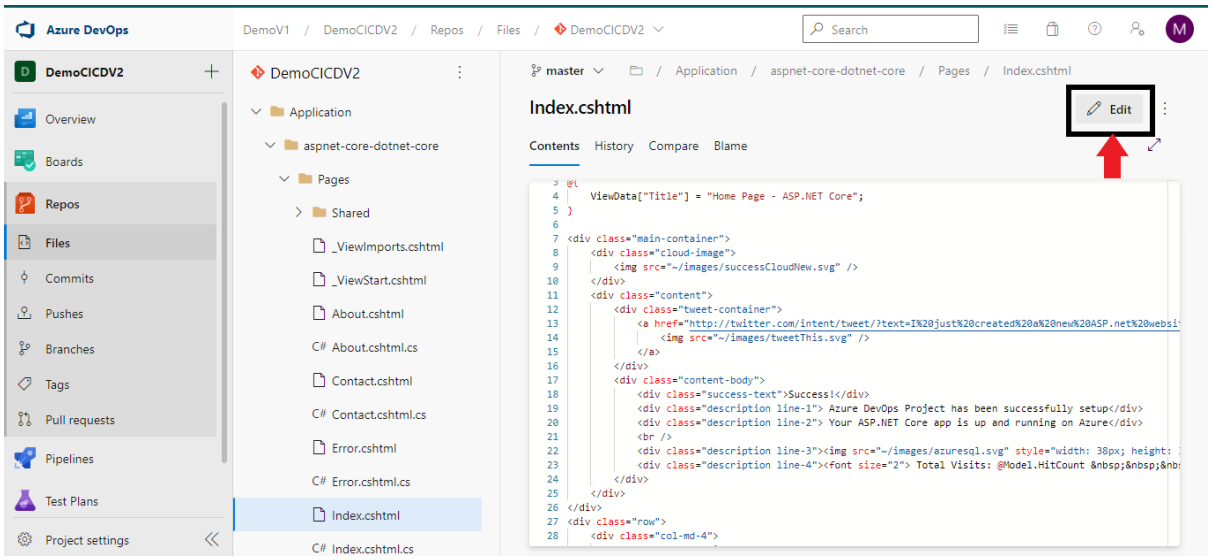
3) It is possible to get your git URL from the window below, and you can also pick an IDE and access your project from the IDE of your choice.



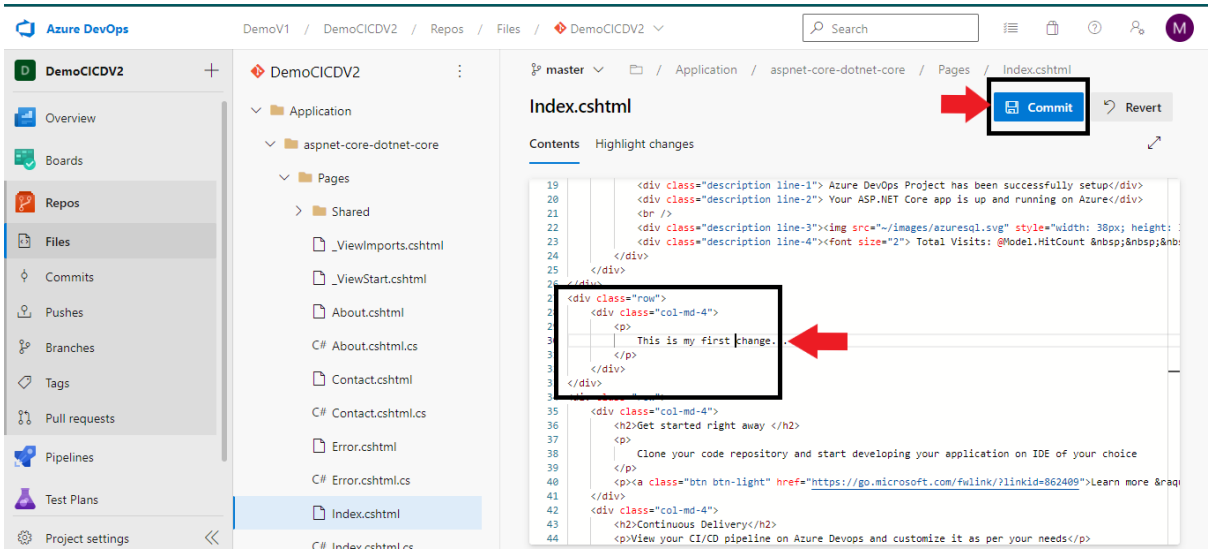
4) Now open the Index file like below



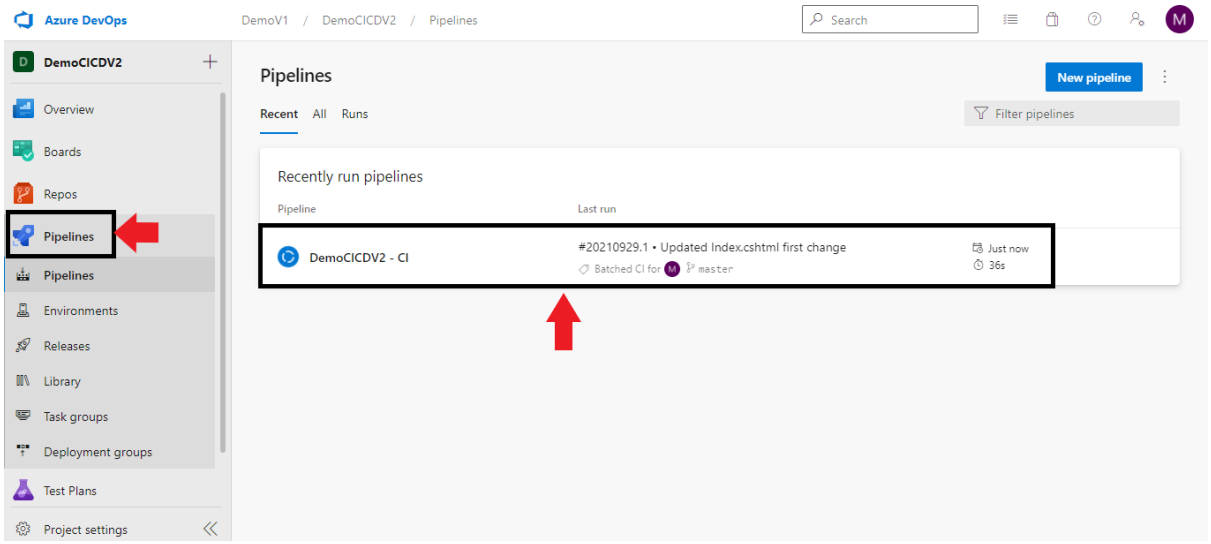
5) Then click on "Edit" and make the necessary changes to the code.



6) After you have completed your modifications, you must now commit the modifications.

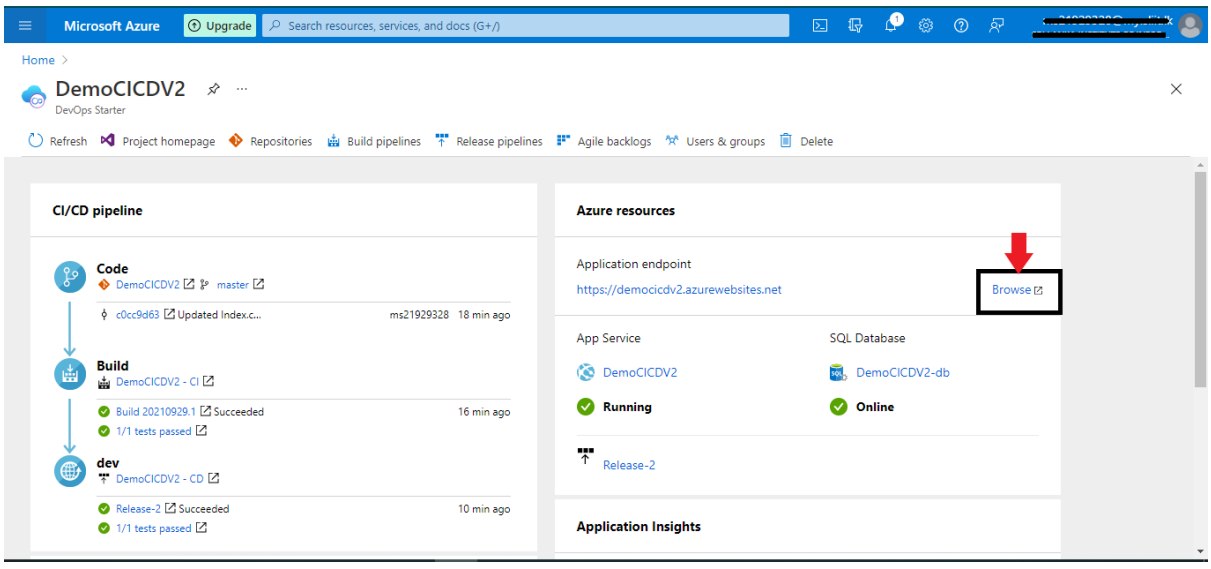


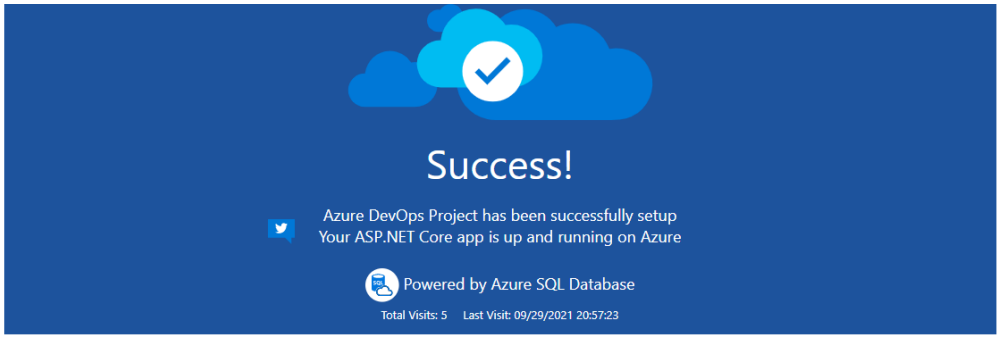
7) Once you have committed, go to the Pipeline and double-check your work.



The building of this structure is presently underway. The Azure DevOps CI/CD pipelines are automatically building and deploying your new modifications, as you can see in the screenshot.

8) Go to your Azure portal (https://portal.azure.com/) when you have finished building and executing your code. You'll notice that your web application has undergone some adjustments.





This is my first change...

Get started right away

Clone your code repository and start developing your application on IDE of your choice

Learn more »

Continuous Delivery

View your CI/CD pipeline on Azure Devops and customize it as per your needs

Learn more »

Azure DevOps Project

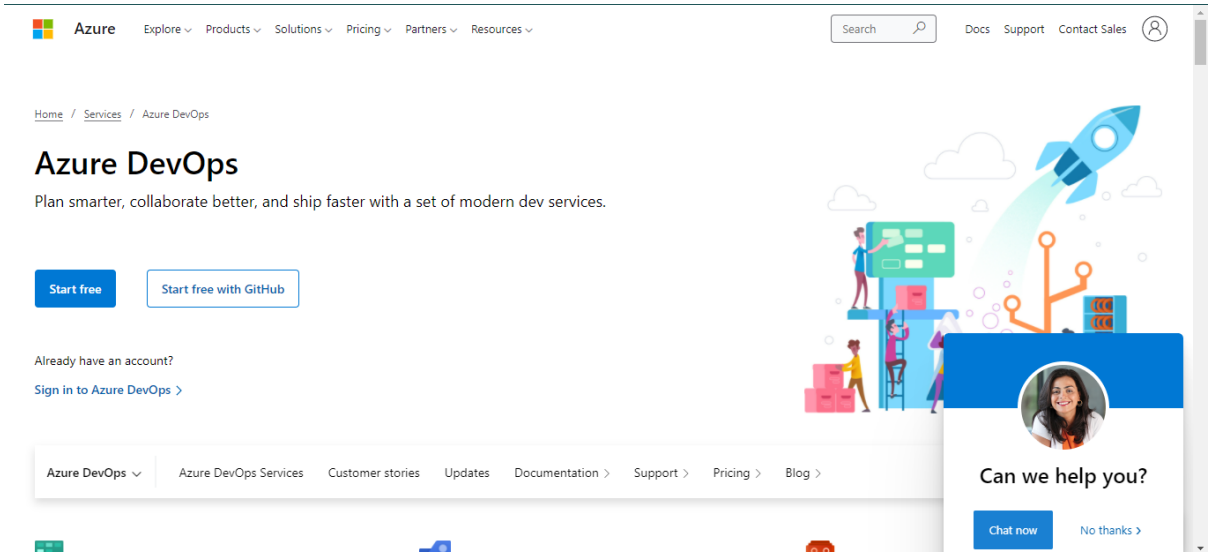
Learn more about all you can do with Azure DevOps project by visiting the documentation

Learn more »

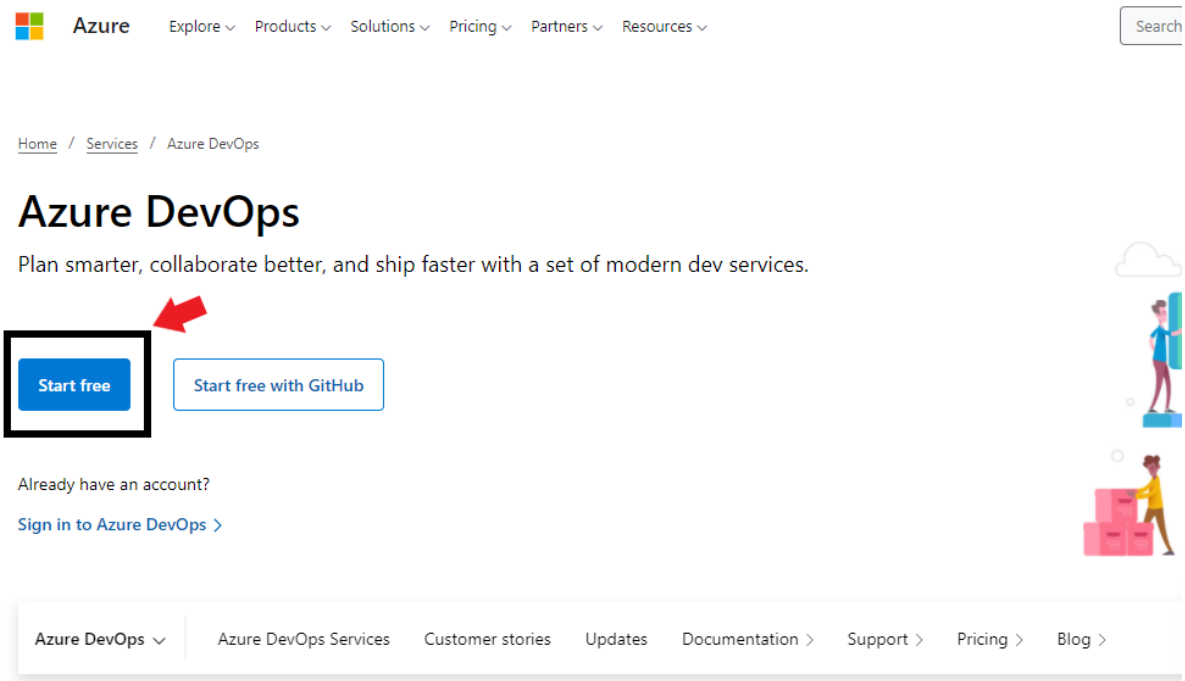
5.3 Net Core application from GitHub

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

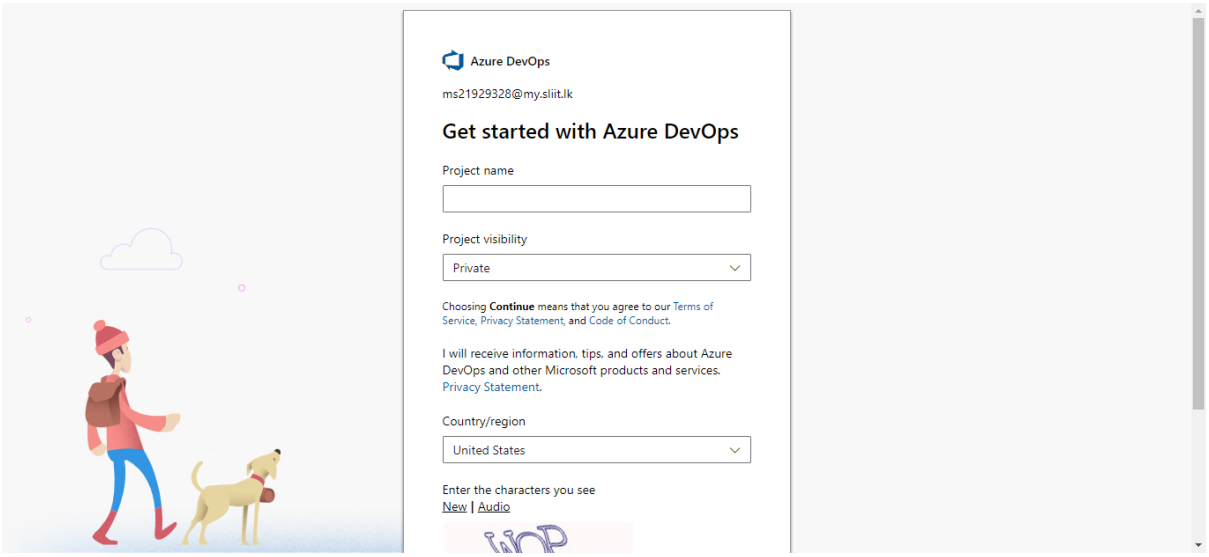
1) Sign into your Azure DevOps account by visiting to devops.azure.com and entering your credentials.

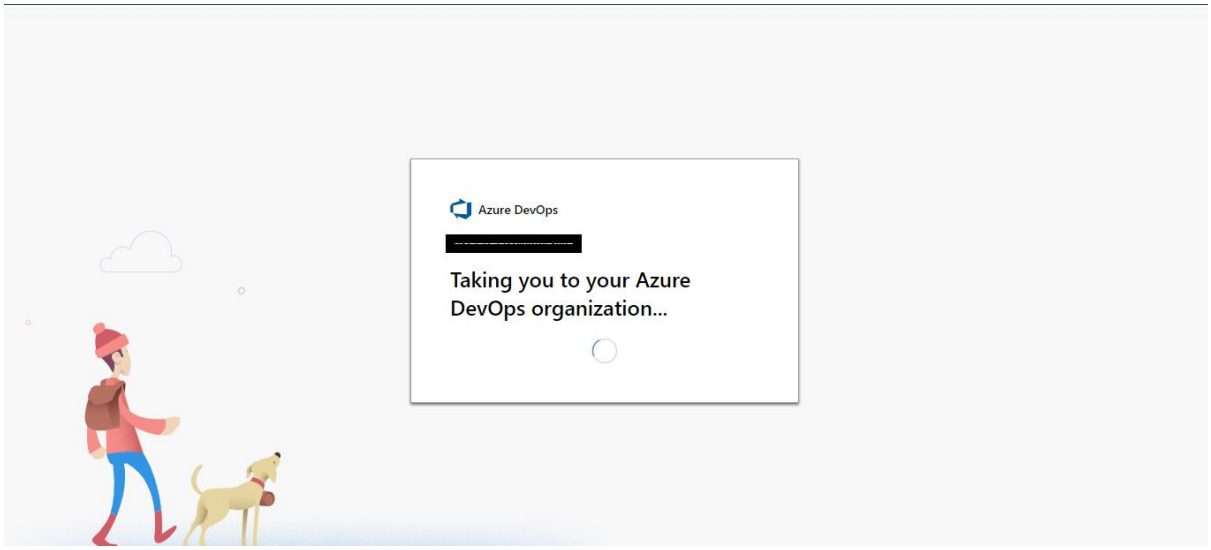


2) Then choose "Start Free" from the window.

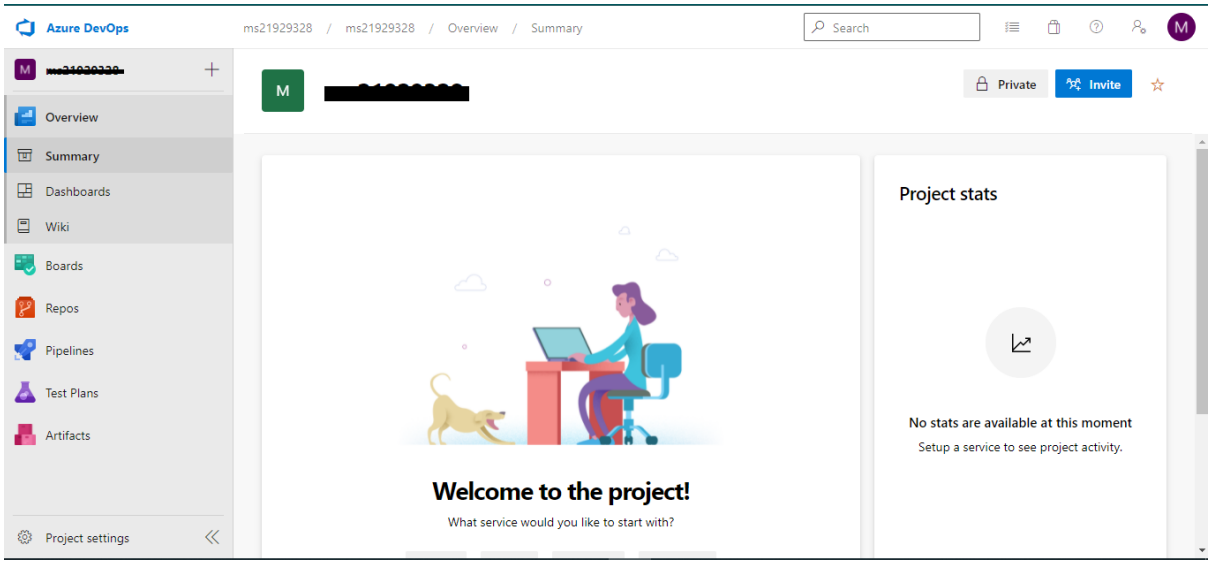


Following that, it will direct you to the next window.

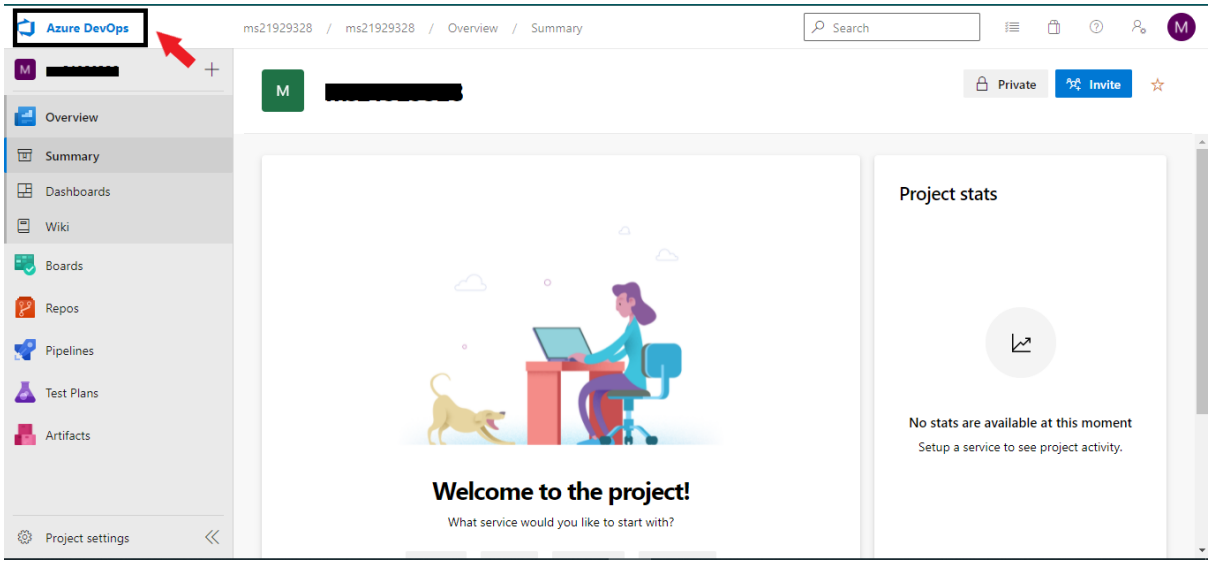




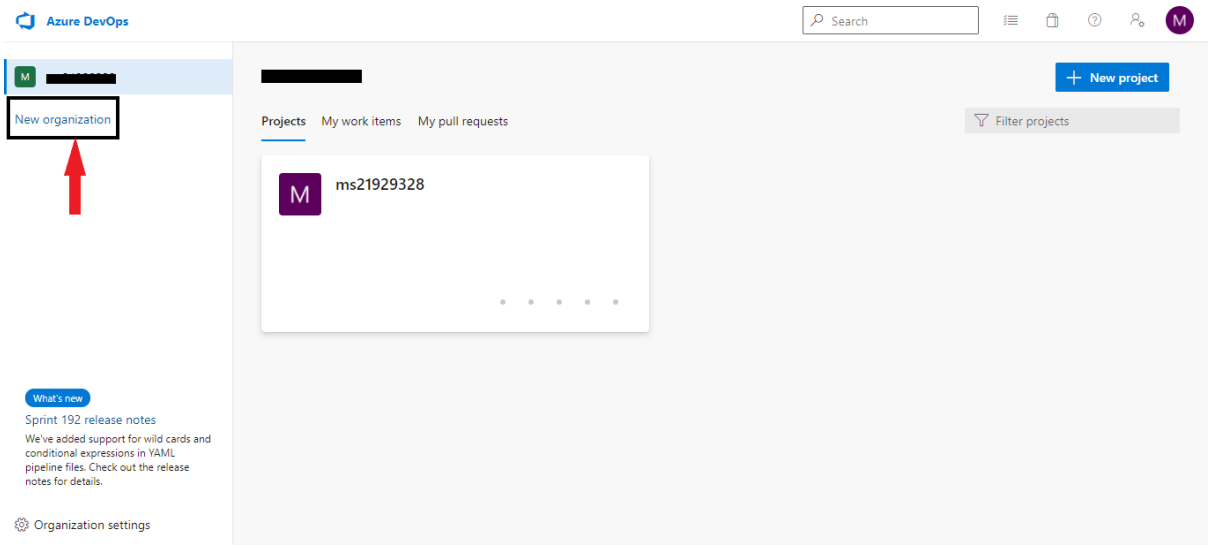
Your Azure DevOps account is now available to use.



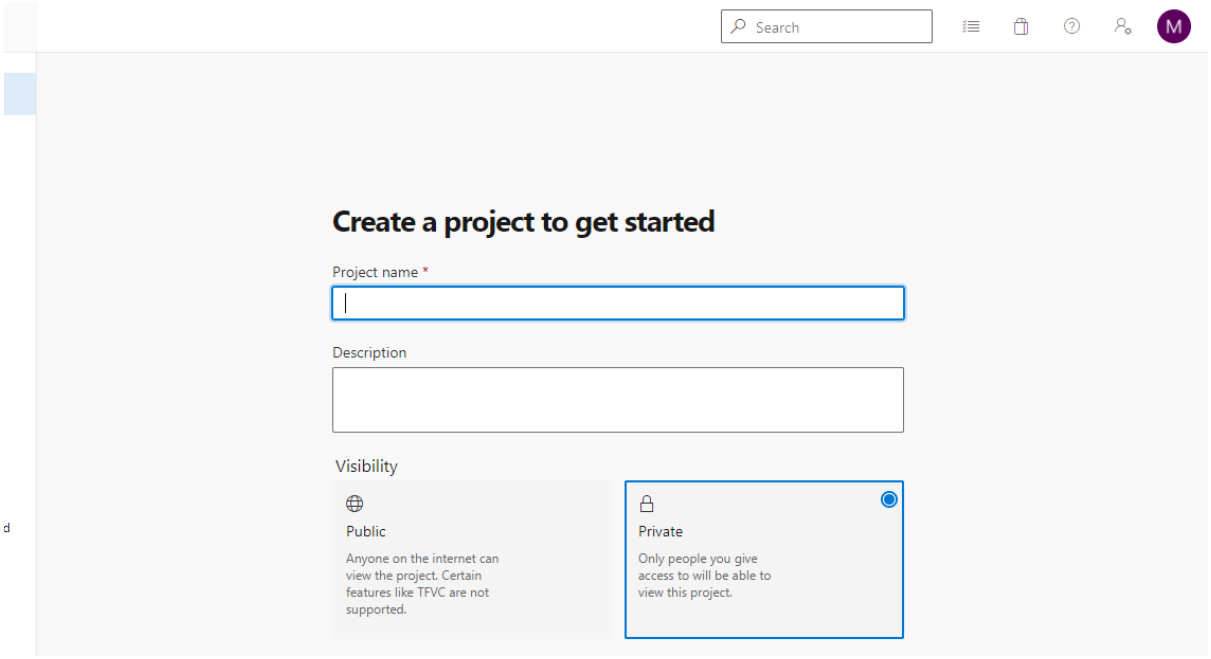
Click on the ‘Azure DevOps' button



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



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.



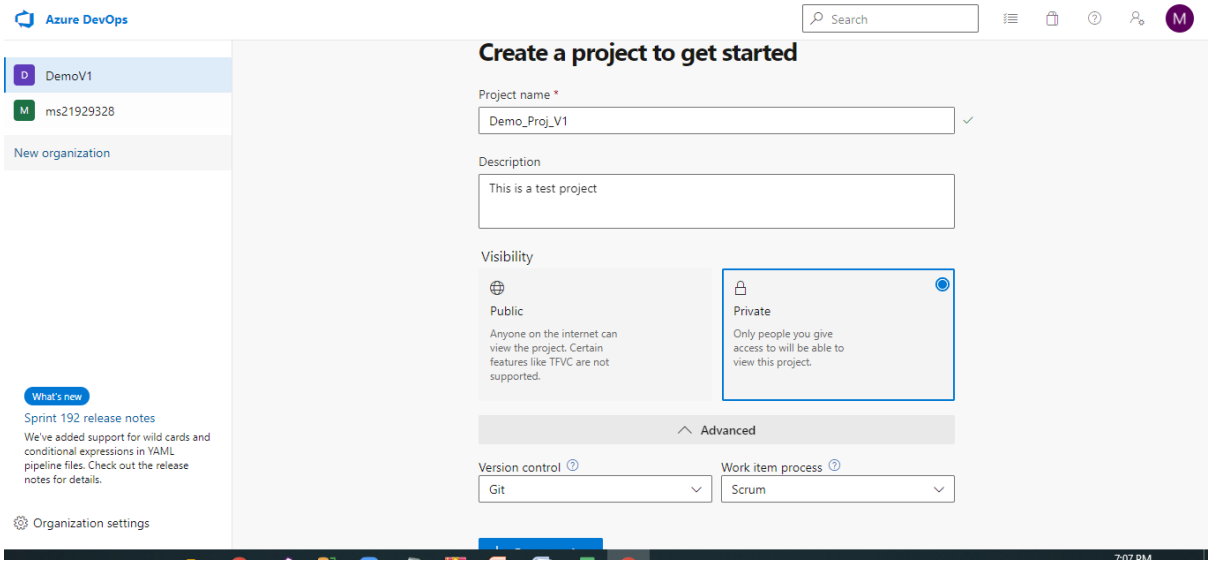
Now you can create a new project

Project name: The project's official name is
Description: A brief overview of the project
Visibility: Your project's exposure is important.

From the Advanced menu, choose
Version control: Git
Work item process: Scrum

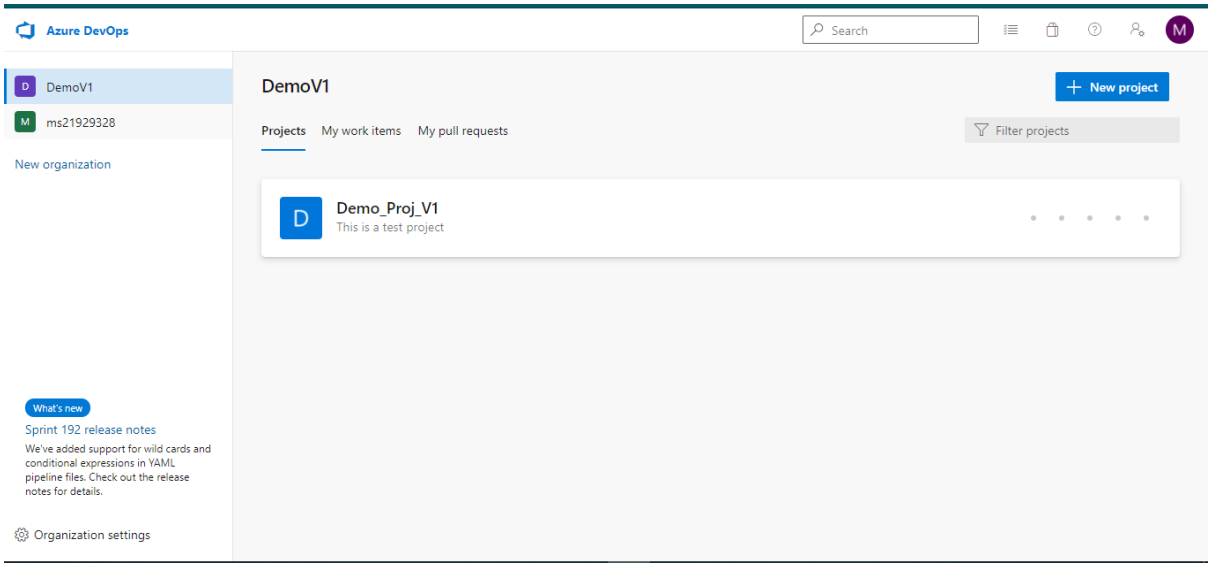
In this form, the project name is a required field, while the other elements are optional to complete.

In this case, I choose to build a secret project, which means that it will not be available to anybody until I share it. 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.



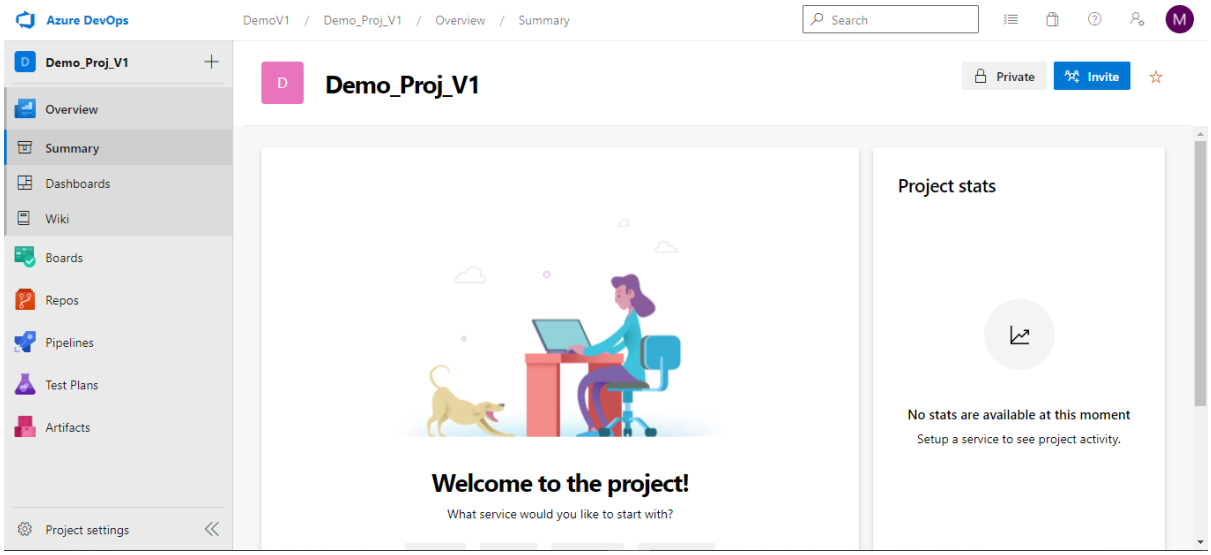
Now click on the **Create Project**

You will be led to the organization's dashboard, which looks like the one shown below.

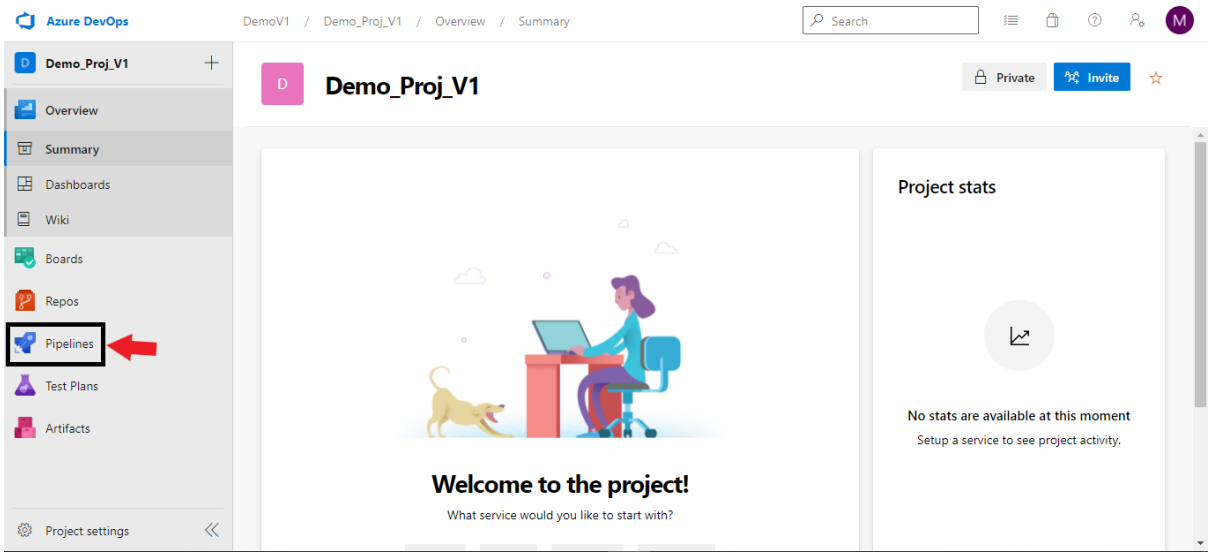


We will now develop a.Net Core continuous integration/continuous delivery pipeline.

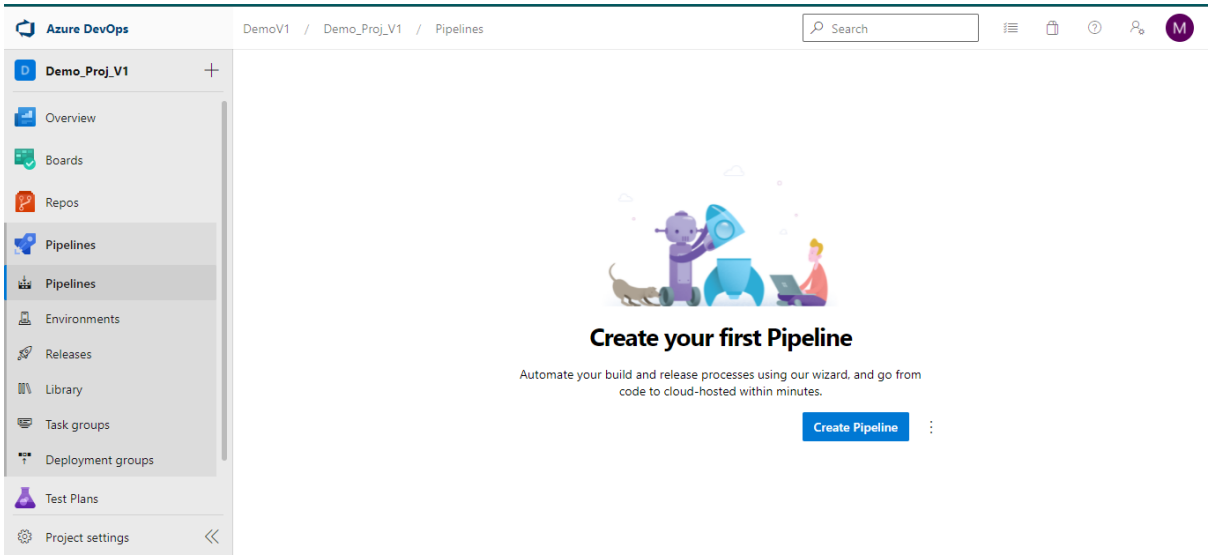
On the dashboard, choose the project you want to work on. In my instance, the project name is "Demo Proj V1."



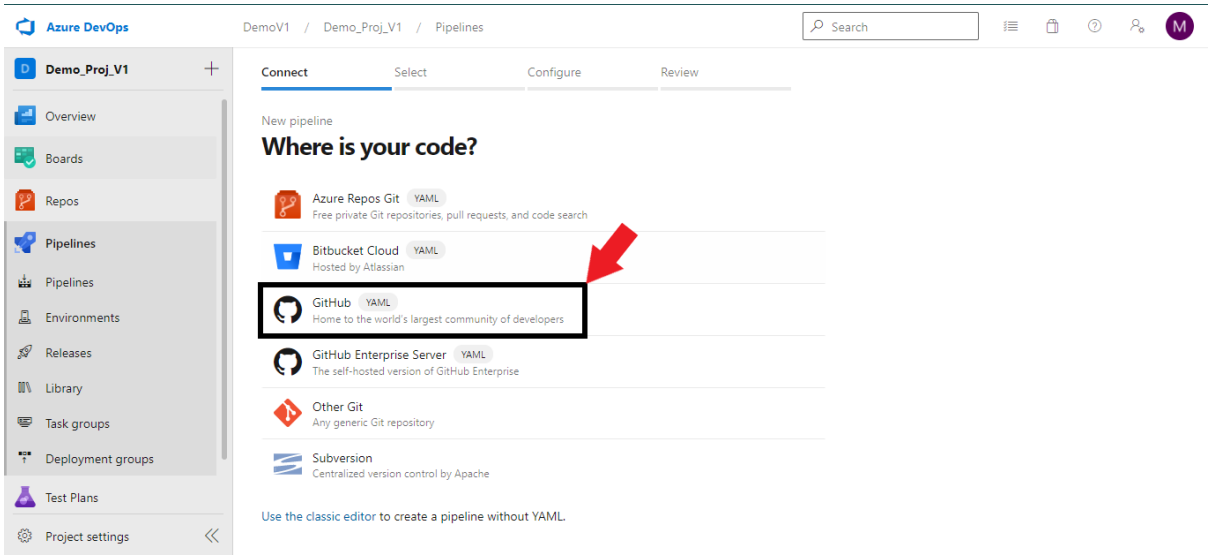
To construct a new pipeline, choose "Pipeline" from the drop-down menu in the window.



Click on the “Create Pipeline”

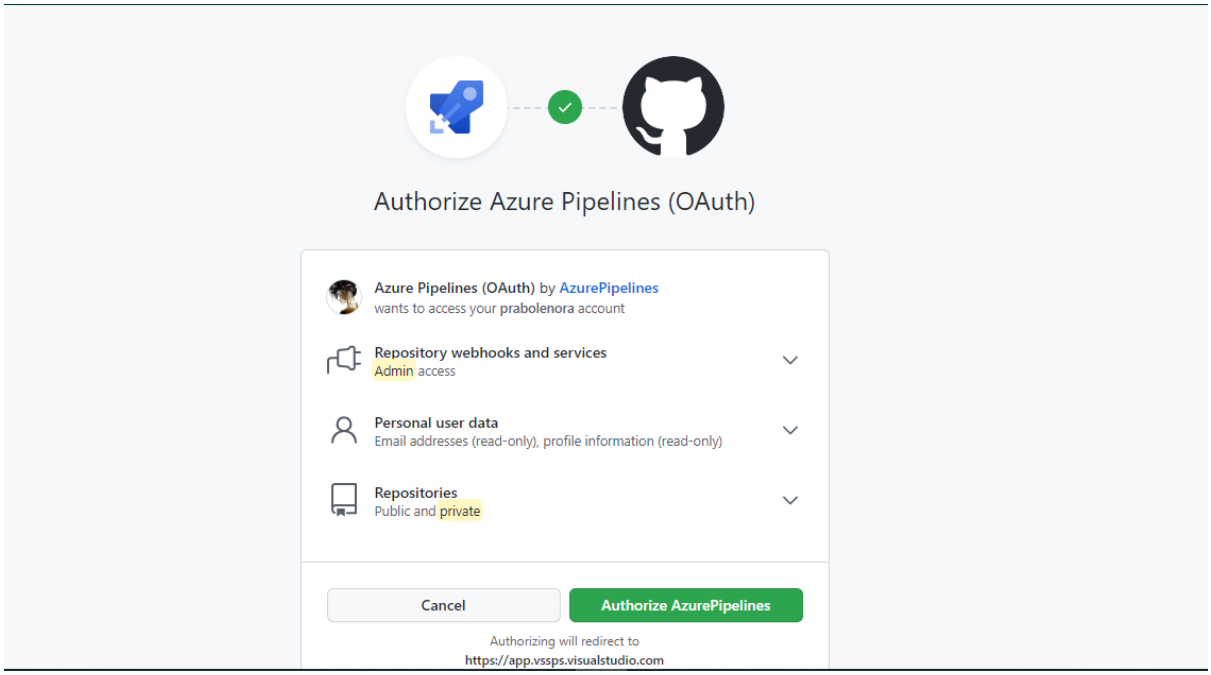


Choose GitHub as the repository for your source code.

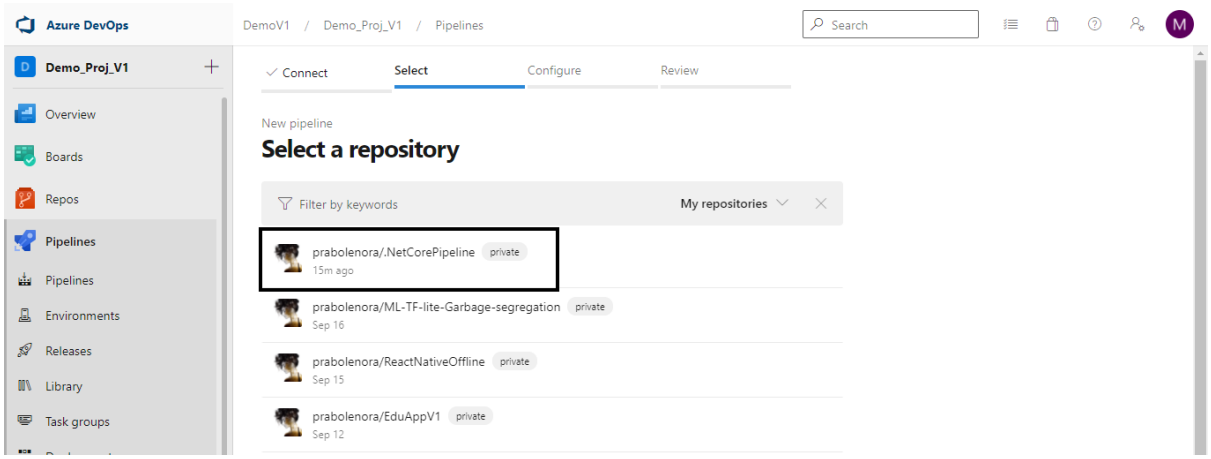


It is necessary to upload your .netCore project to your GitHub account before proceeding. Next that, proceed with the following steps.

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. By using the Azure portal, you may input your GitHub credentials and connect to your account.



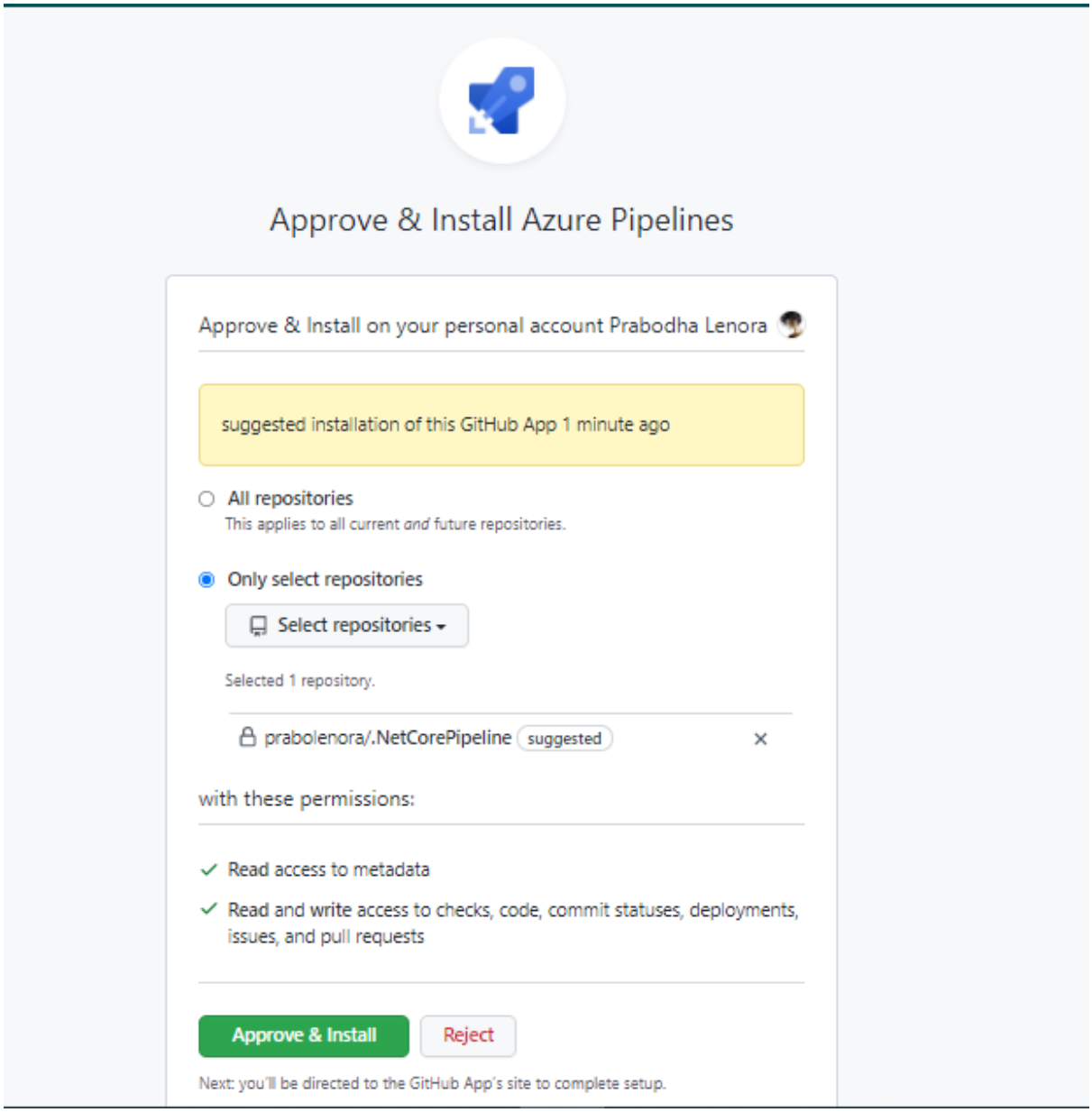
Click **Authorize AzurePipelines**



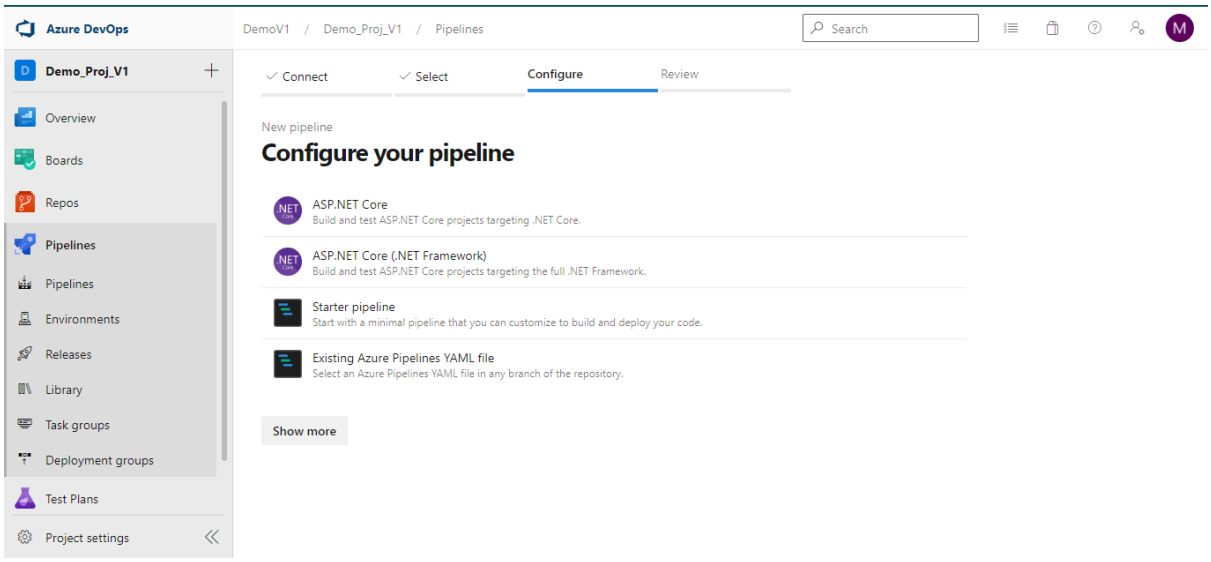
Select your project from a list of all of your GitHub repositories, which will appear here.

Then you'll be sent to a new window where you can install the Azure pipeline application.

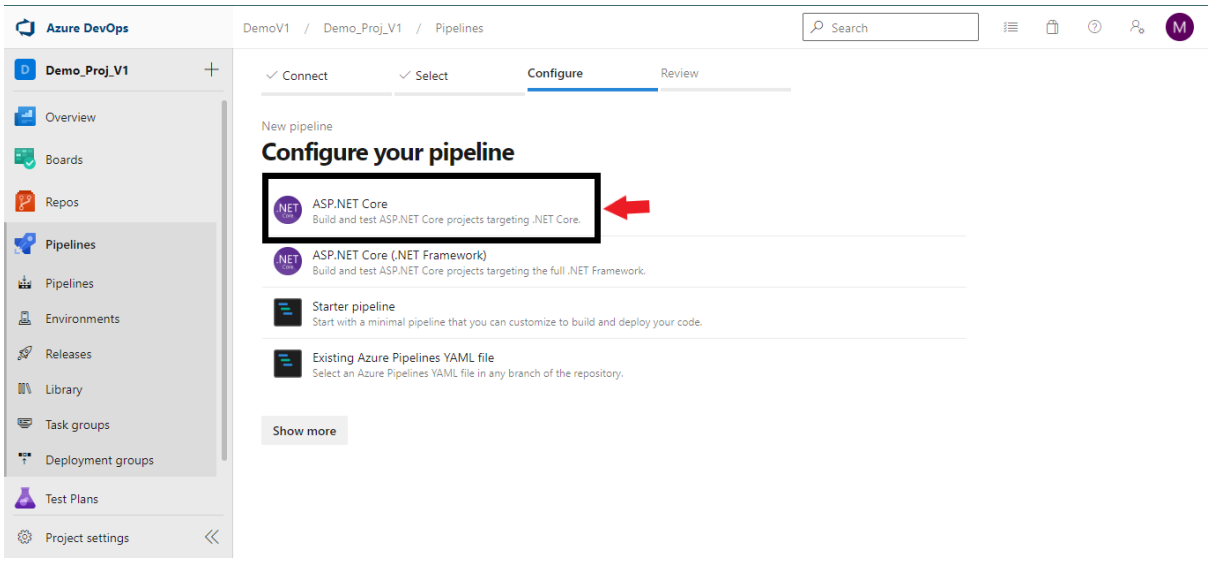
Select "Choose and Approve" from the drop-down menu.



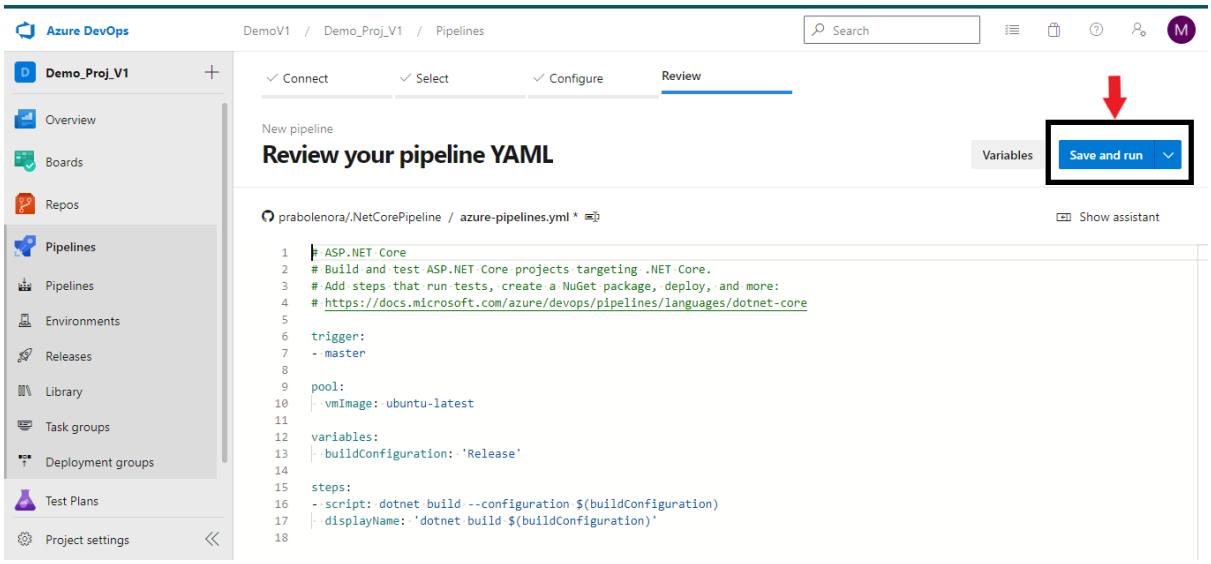
The configure window will appear when the installation has been completed successfully.



From the configure window choose the ASP .Net Core

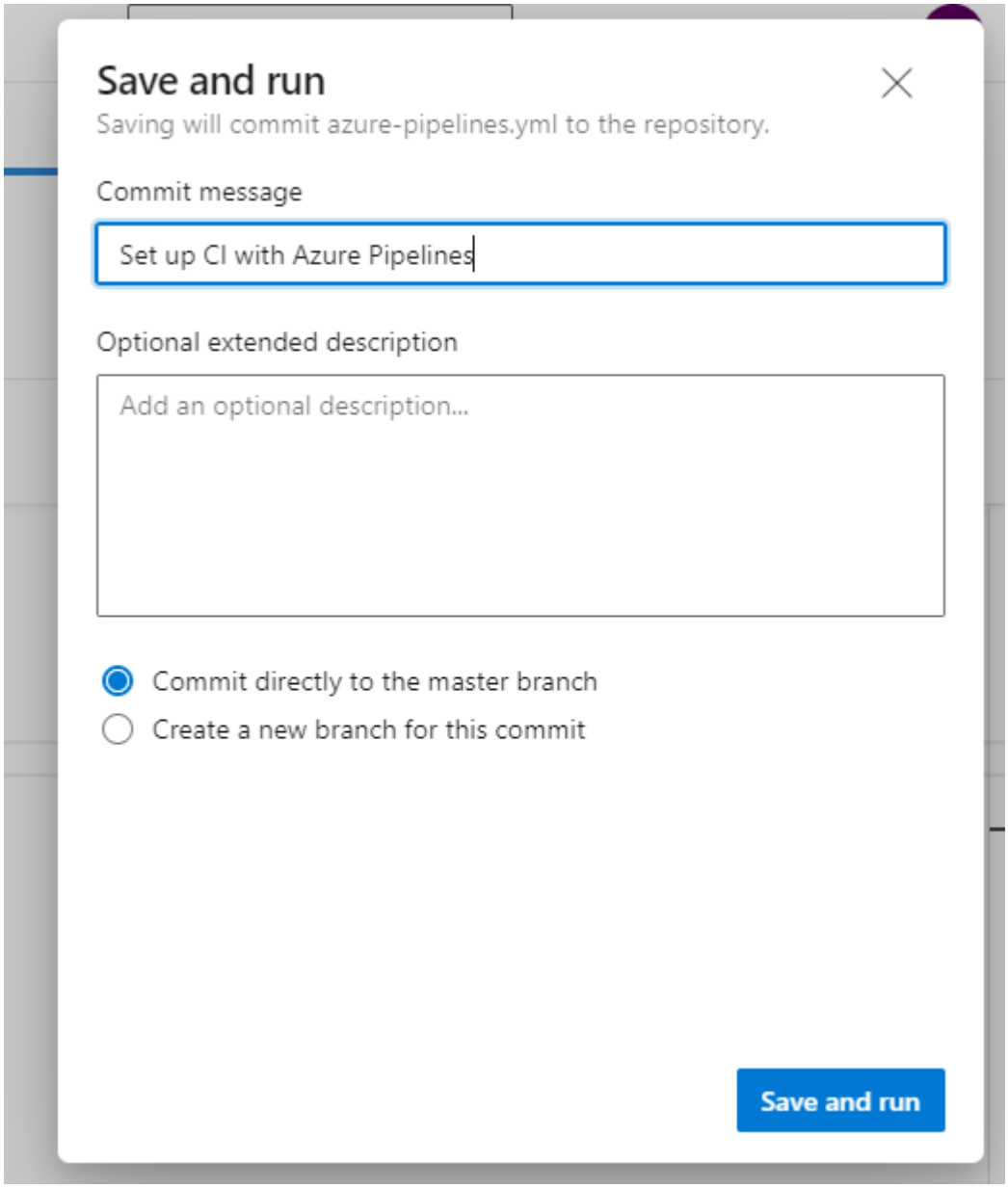


Your pipeline file will now display as a YAML file on the review tab of the application. To make the YAML file executable, choose "Save & Run."



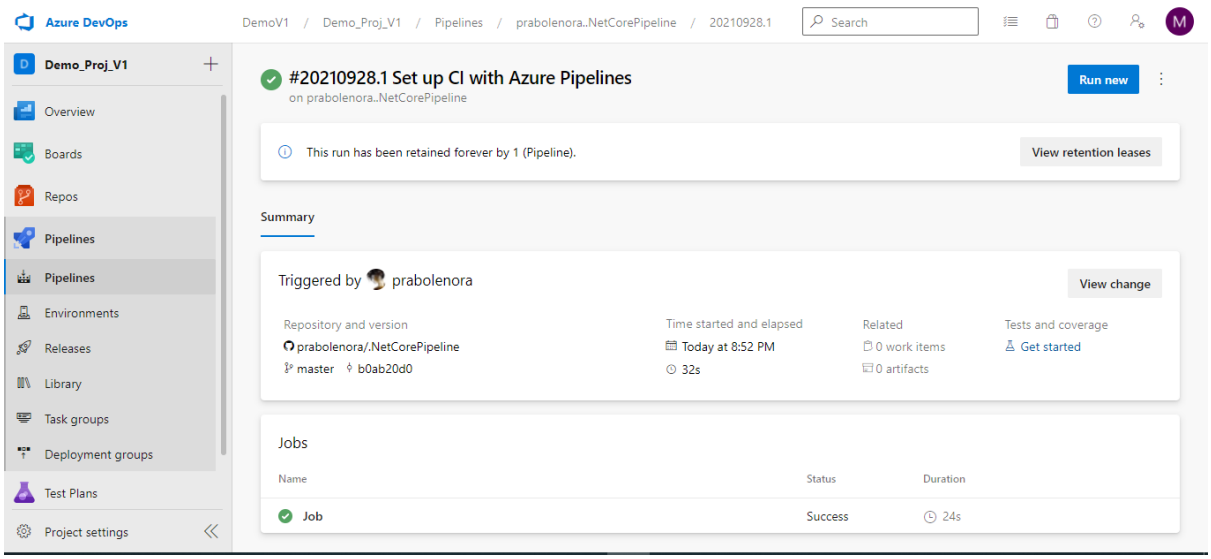
A new popup window will open before the program can be launched. Simply choose "Commit straight to the master branch" from the drop-down menu and hit "Save and Run."

Otherwise, your project will be saved in a new GitHub branch if you choose any of the other choices available.



The fundamentals of pipeline development are now complete, and the pipeline may be used to execute the program and create the program.

Following the deployment, you will be presented with the window shown below.



In certain situations, you may see an error message similar to the one shown below.

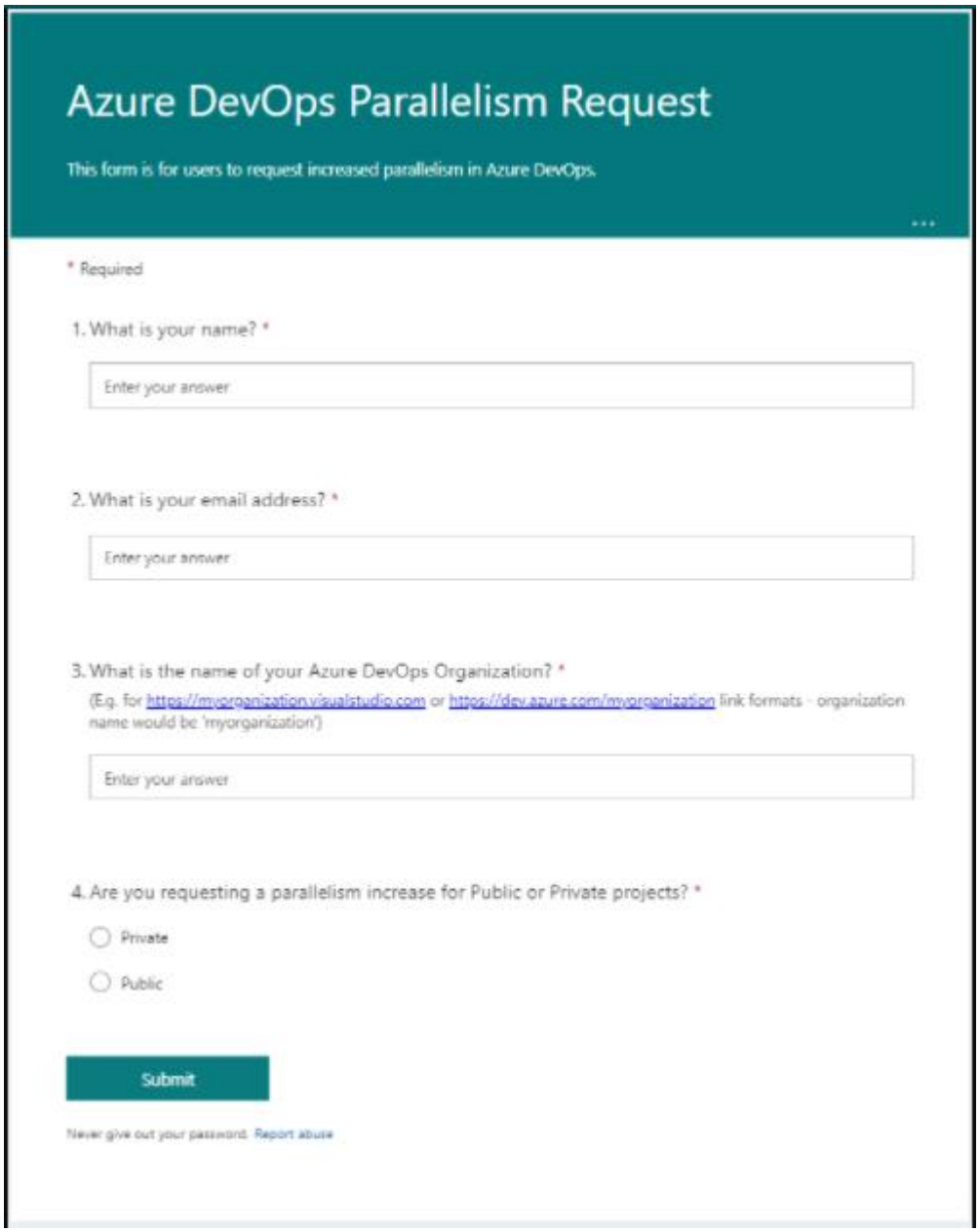
"*No hosted parallelism has been purchased or granted. To request a free parallelism grant, please fill out the following form https://aka.ms/azpipelines-parallelism-request*"

This is due to a change in Microsoft policy about the hosted agent pool, which has caused the problem.

For help with the issue, go to the URL given in the error message and complete the questions to have Microsoft approve your solution.

If you want more information on this, the following link will be of assistance: https://devblogs.microsoft.com/devops/change-in-azure-pipelines-grant-for-private-projects/.

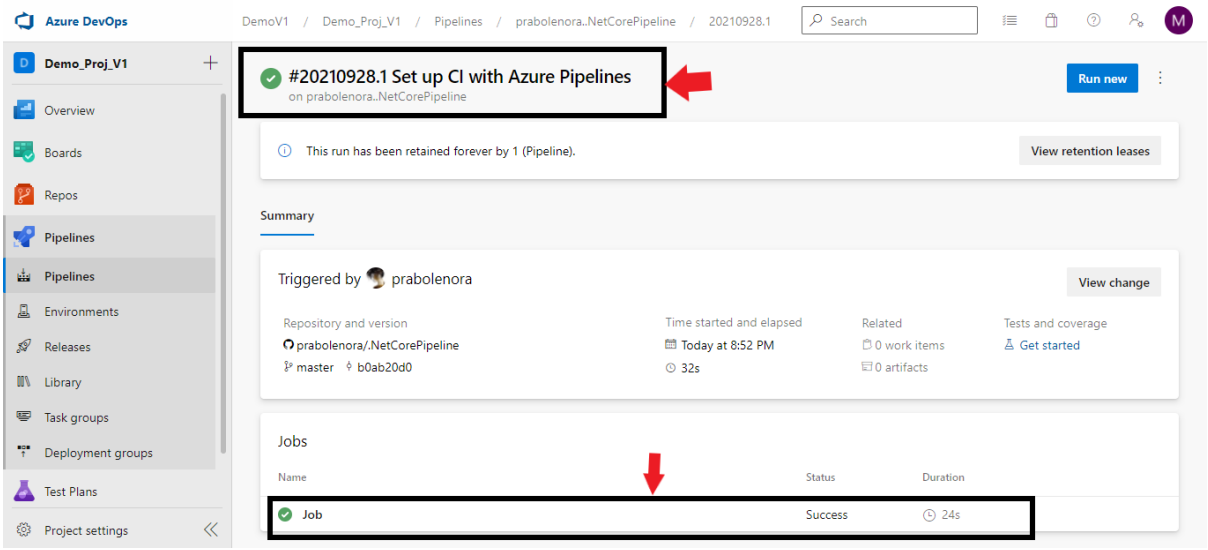
This is the window that appears when you click on the hyperlink.



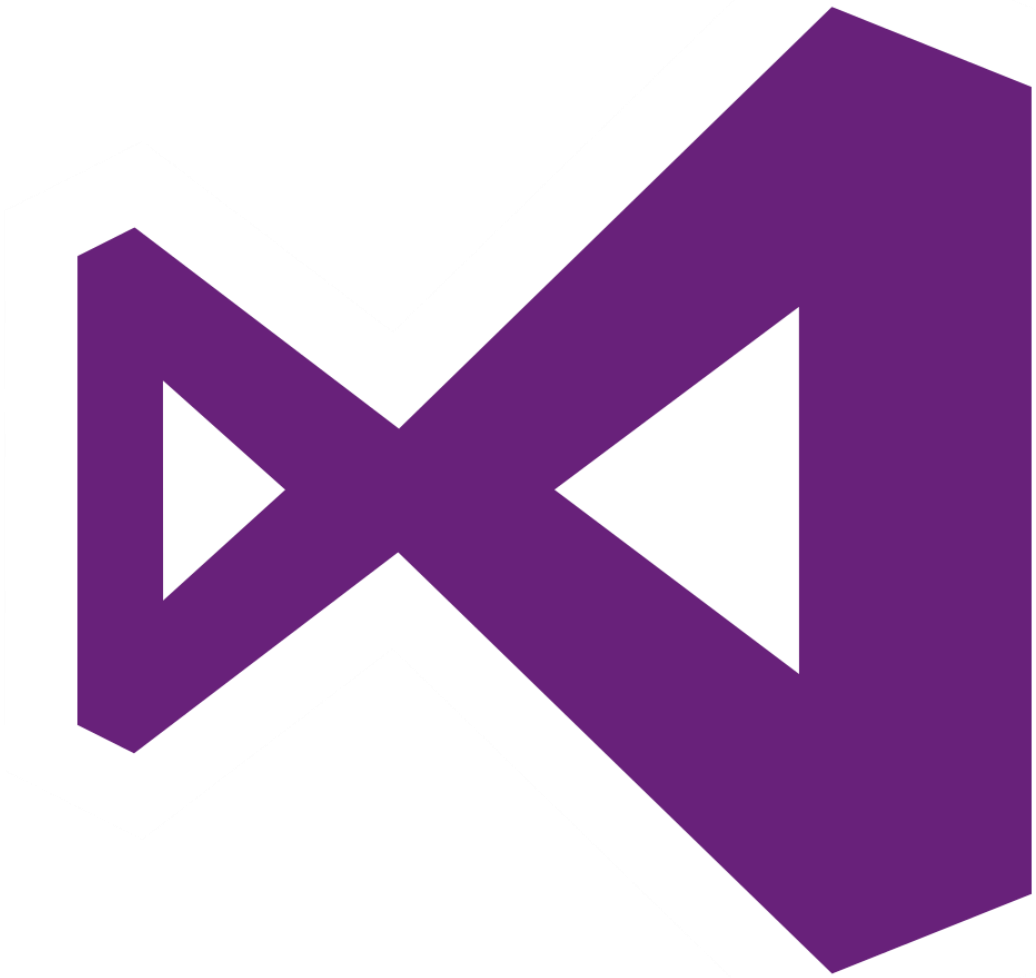
After Microsoft has approved your request, you will get an email confirming your permission.

After then, you may go back and re-do the run.

Following the conclusion of the run, you will see a success or error notice in your window, and the administrator of your company will get an email with the job's information.



Chapter 06

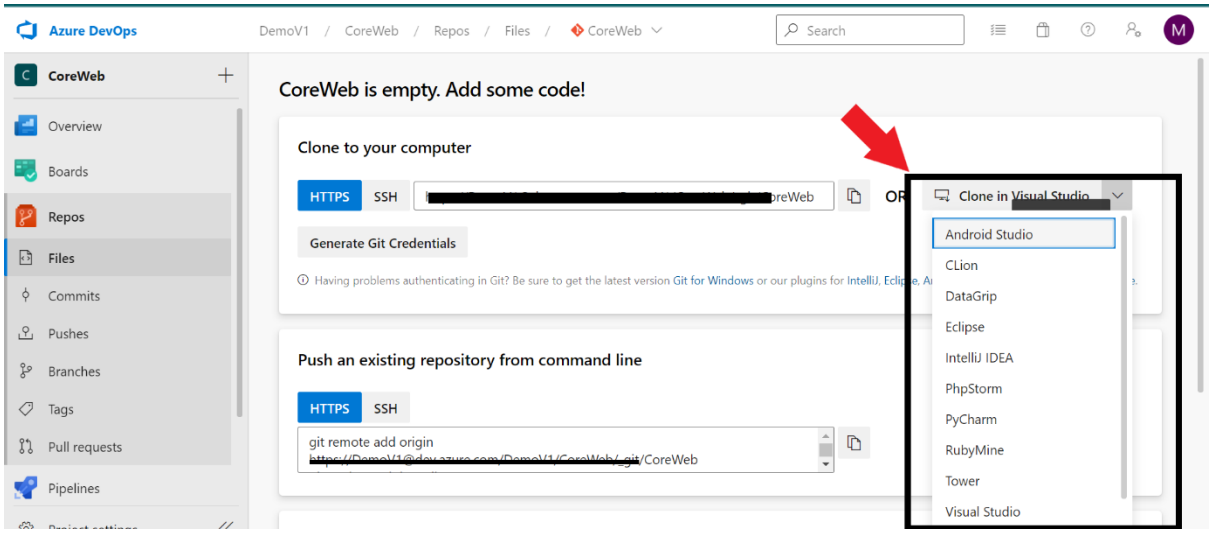


Azure DevOps and Visual Studio

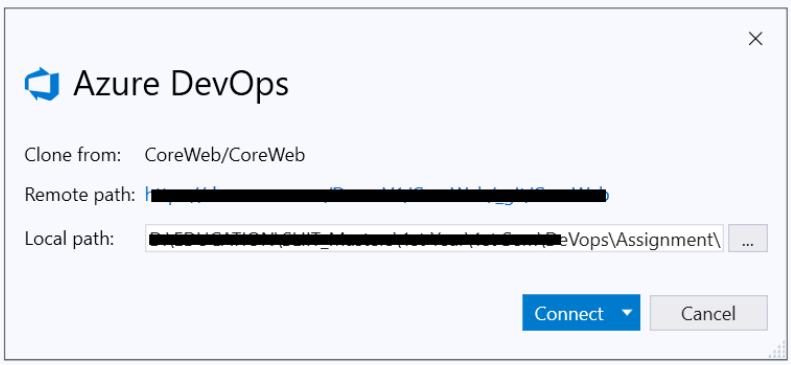
6.1 Clone project from Azure DevOps

We are now collaborating with them. We must utilize Visual Studio as an integrated development environment (IDE) for NetCore. Your code may be cloned into Visual Studio using the Azure DevOps interface. It provides a diverse range of apps with which to collaborate.

It is quite simple to manage your code in conjunction with your local repository when using Azure DevOps.



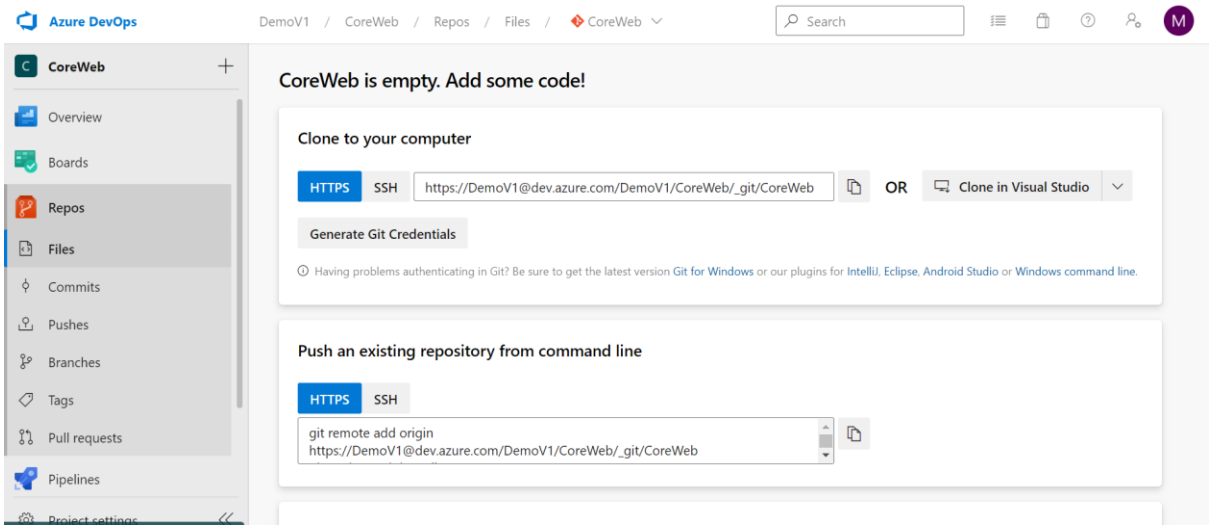
Following your selection of a concept, you will be routed to that idea. Your suggestion will be immediately accepted. You may either choose a local path on your computer to clone your project or it will clone your project to a default directory on your computer from there. The Azure DevOps clone URL will be shown after that.



Following the click of the link button, your project from the Azure Portal will be cloned into the location you choose, and it will instantly open in your visual studio.

6.2 Clone project using URL

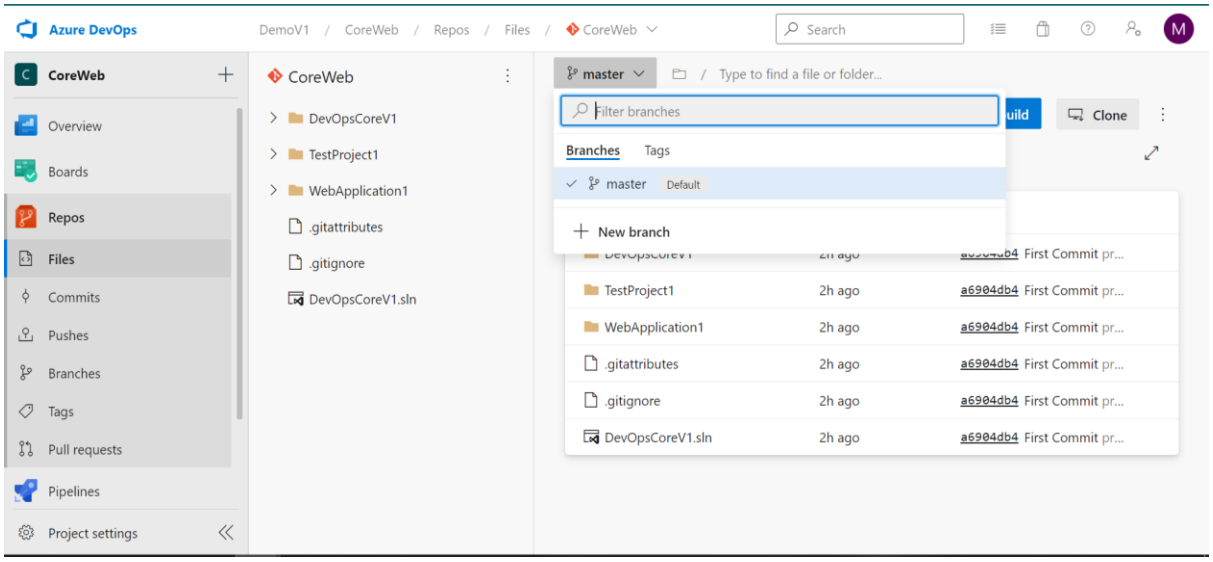
All of the project information, including the clone URL, will be available under the repository -> files section. Manually cloning your project to a local system is possible if you copy the clone URL from the server and then use GIT to clone it to your PC.

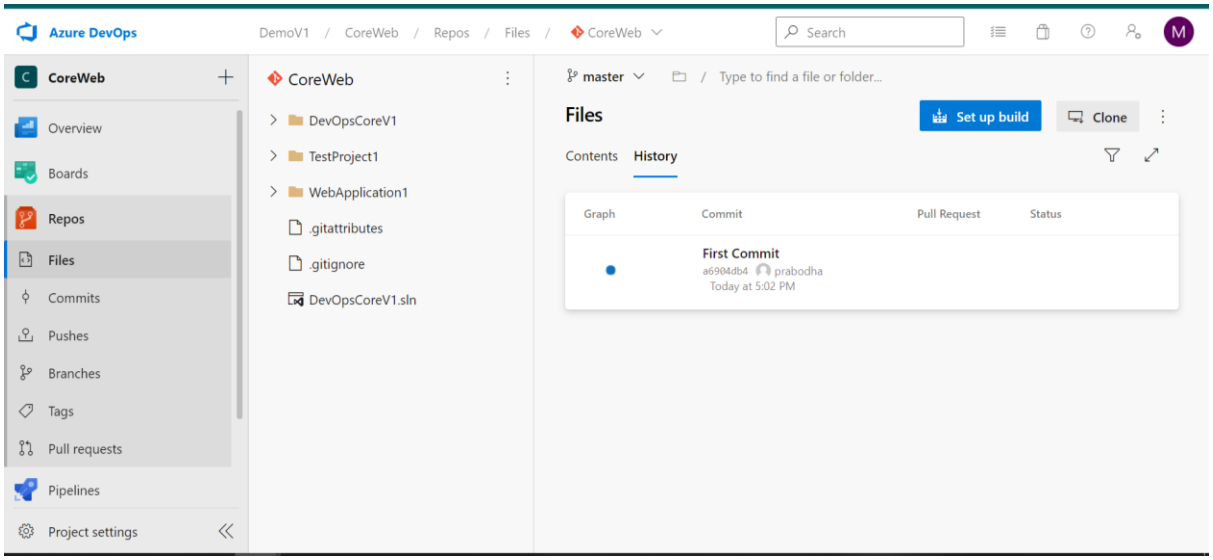


6.3 Project Details

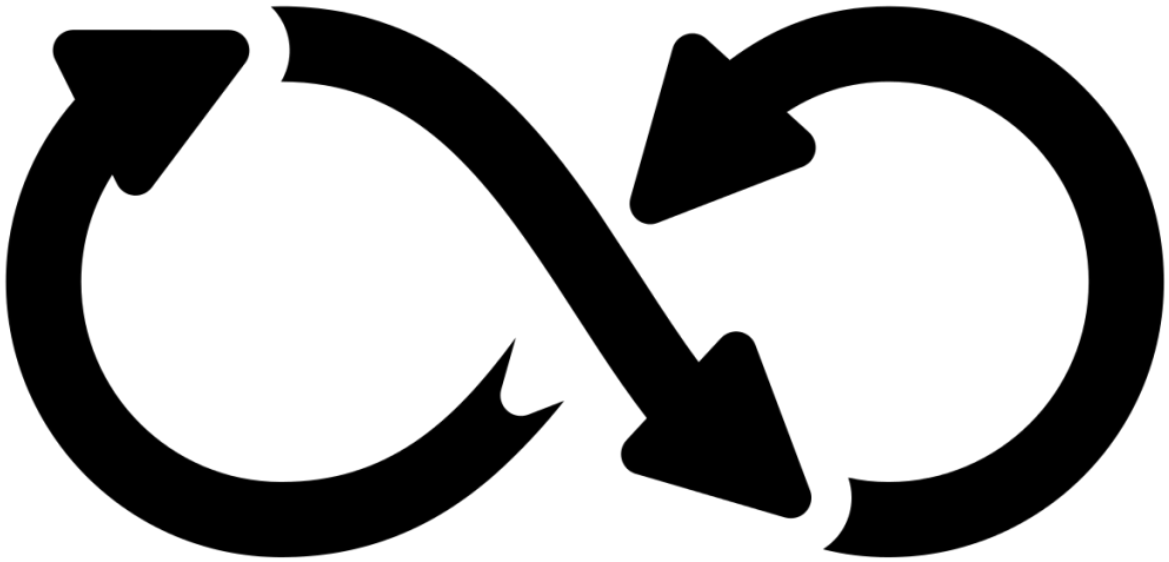
The files associated with your project will be visible under repo->files once you have cloned it using Azure DevOps.

You may use this page to determine which branch your code is now in, and you can examine the committed changes and files based on which branch you are currently in. and it will show all of the committed data, including the branch, if you choose it from the history tab.


Azure DevOps
DemoV1 / CoreWeb / Repos / Files / CoreWeb
Search
CoreWeb
Overview
Boards
Repos
Files
Commits
Pushes
Branches
Tags
Pull requests
Pipelines
Project settings
CoreWeb
DevOpsCoreV1
TestProject1
WebApplication1
.gitattributes
.gitignore
DevOpsCoreV1.sln
master
Type to find a file or folder...
Filter branches
Branches
Tags
master
Default
New branch
Clone
TestProject1 2h ago a6904db4 First Commit pr...
WebApplication1 2h ago a6904db4 First Commit pr...
.gitattributes 2h ago a6904db4 First Commit pr...
.gitignore 2h ago a6904db4 First Commit pr...
DevOpsCoreV1.sln 2h ago a6904db4 First Commit pr...


Azure DevOps
DemoV1 / CoreWeb / Repos / Files / CoreWeb
Search
CoreWeb
Overview
Boards
Repos
Files
Commits
Pushes
Branches
Tags
Pull requests
Pipelines
Project settings
CoreWeb
DevOpsCoreV1
TestProject1
WebApplication1
.gitattributes
.gitignore
DevOpsCoreV1.sln
master
Type to find a file or folder...
Files
Set up build
Clone
Contents
History
Graph
Commit
Pull Request
Status
First Commit
a6904db4 prabodha
Today at 5:02 PM

Chapter 07

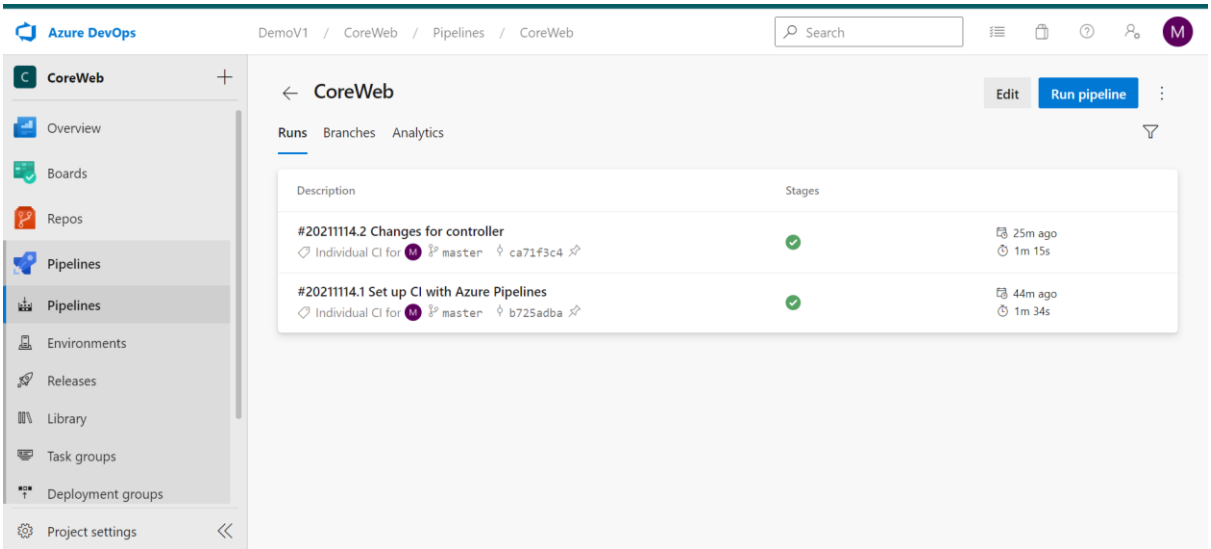


Continuous Integration

7.1 Create Pipeline

First and foremost, you must establish a pipeline for your new project.

Select the "Pipeline" option from the drop-down menu.
After that, you may start working on a new pipeline for the project. An email will be sent to the address associated with your organization's email address when a successful program is created.

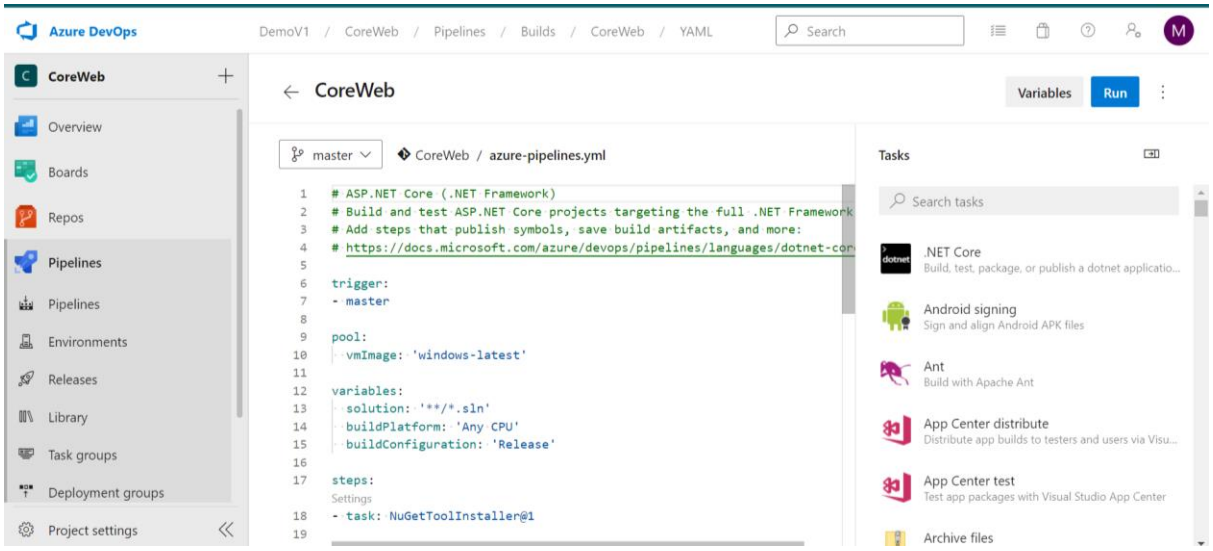


7.2 YAML

To make changes, click the edit button, which is situated at the top of the right-hand corner of the screen.

You will be able to see the YAML file from there. It provides all of the pertinent information about the project. This is how it operates, and this is the branch that is required to operate in this manner.

Make careful not to make any changes to this file.



7.3 New Build

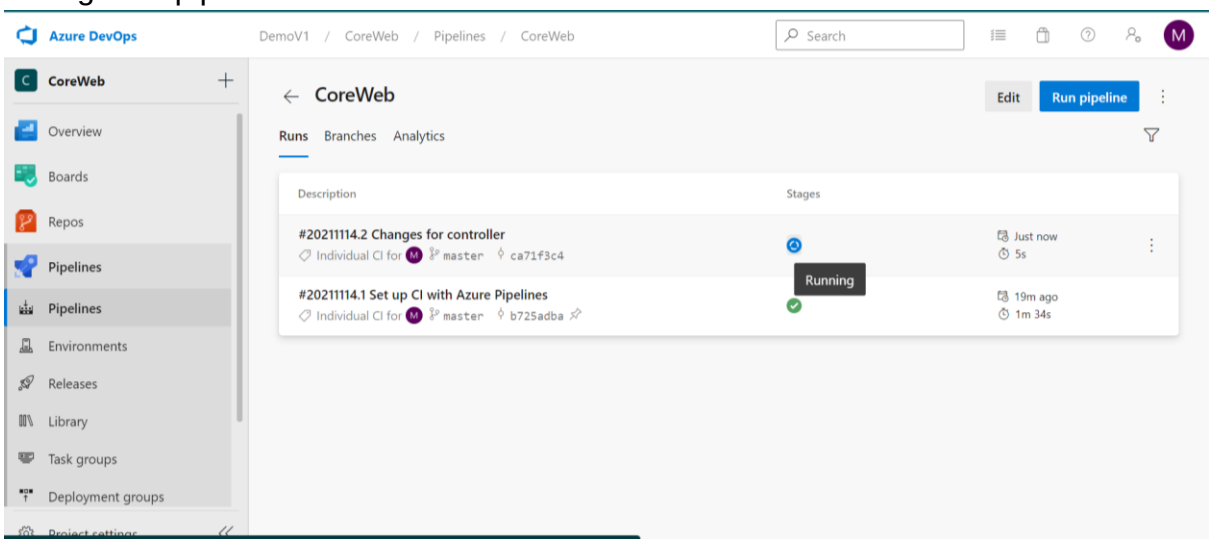
Using the Visual Studio integrated development environment, I made some modifications to the code. and I commit it to the GIT repository.

Now it's time to update your Azure DevOps.

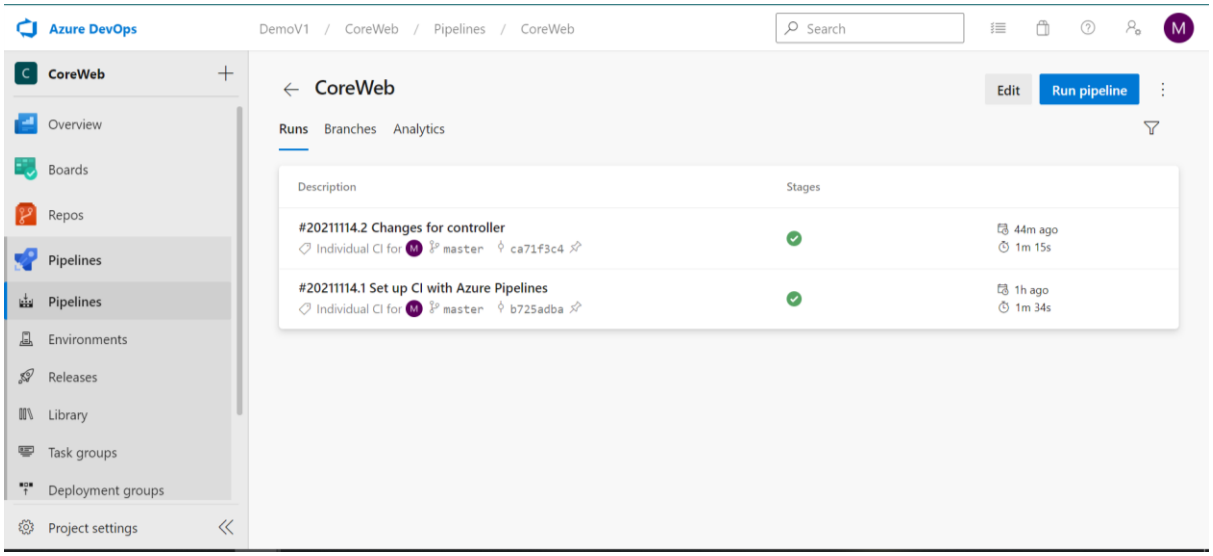
You can see that your new modifications will be visible under the "Run" tab after they have been implemented in the pipeline.

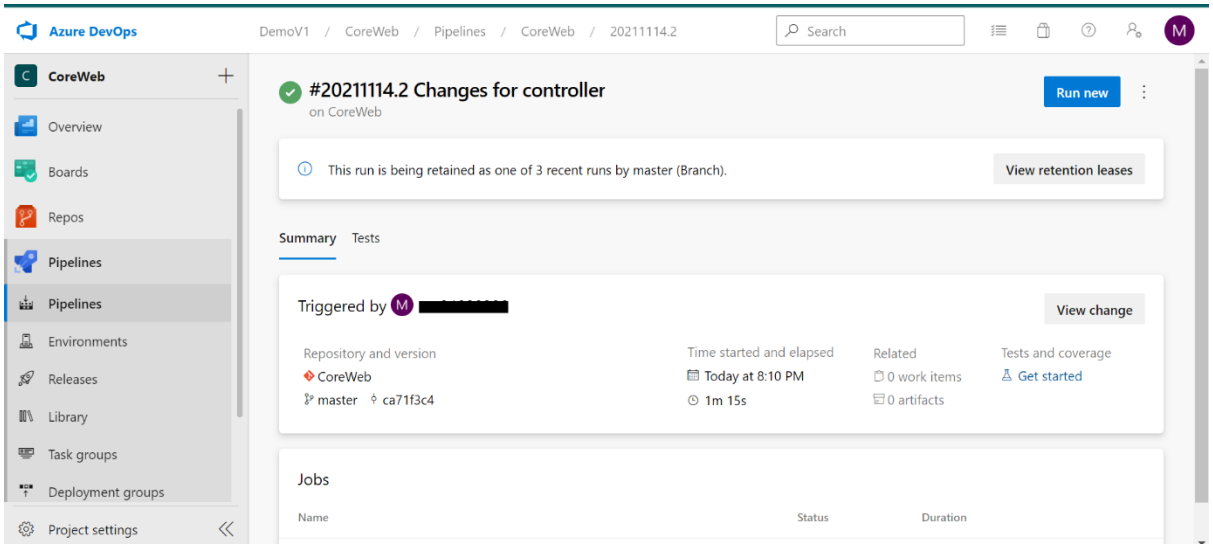
Your new modifications are listed as "Running" in the stage column, which indicates that they have been implemented.

That indicates that your project has not yet been completed. This is an example of continuous integration. We are not required to assume any responsibility for the development of the application. It will automatically construct the project as it moves through the pipeline.



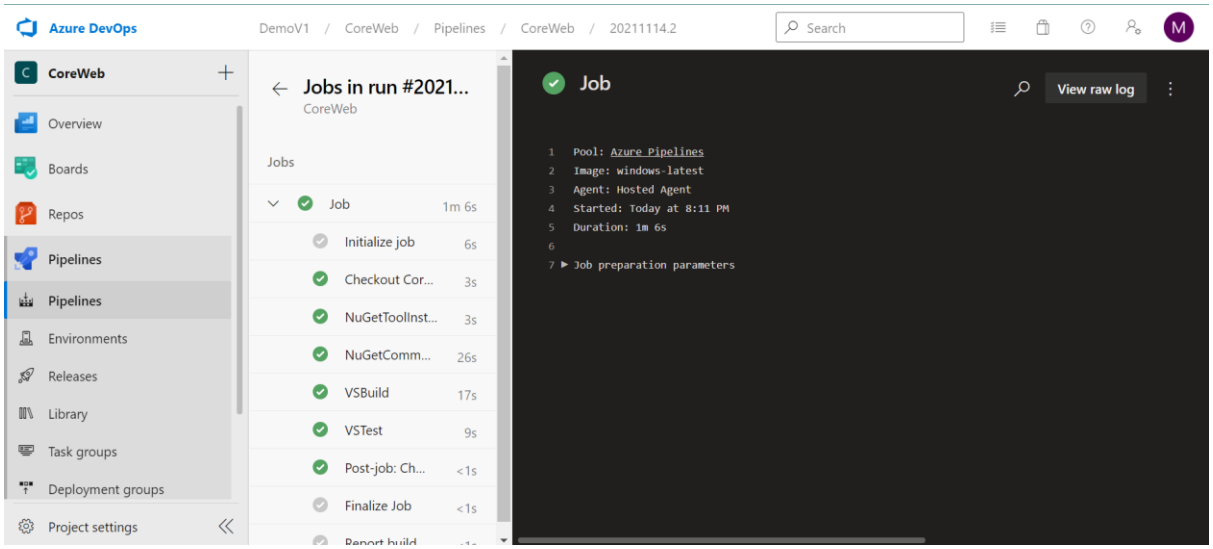
After successfully producing the blue symbol seen above, it was turned into the green icon shown above.



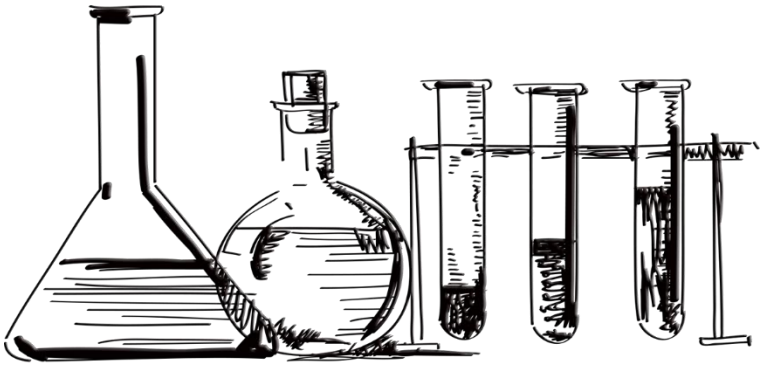


7.3 Job

The construction details of your project may be found in the task window, which is accessible from any computer. It is also protected by a code. You may then go through your project files to see what failed, what is still in progress, and what was a successful outcome.


Azure DevOps
DemoV1 / CoreWeb / Pipelines / CoreWeb / 20211114.2
Search
M
CoreWeb
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Releases
Library
Task groups
Deployment groups
Project settings
Jobs in run #2021...
CoreWeb
Jobs
Job 1m 6s
Initialize job 6s
Checkout Cor... 3s
NuGetToolInst... 3s
NuGetComm... 26s
VSBuild 17s
VSTest 9s
Post-job: Ch... <1s
Finalize Job <1s
Job
View raw log
1 Pool: Azure Pipelines
2 Image: windows-latest
3 Agent: Hosted Agent
4 Started: Today at 8:11 PM
5 Duration: 1m 6s
6
7 ▶ Job preparation parameters

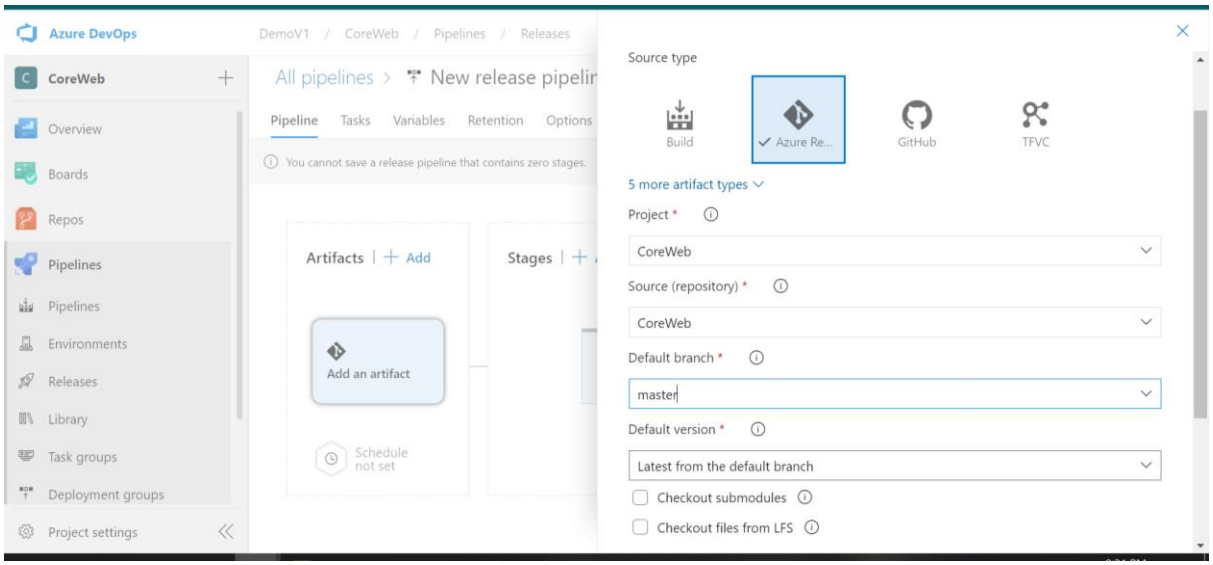
Chapter 08

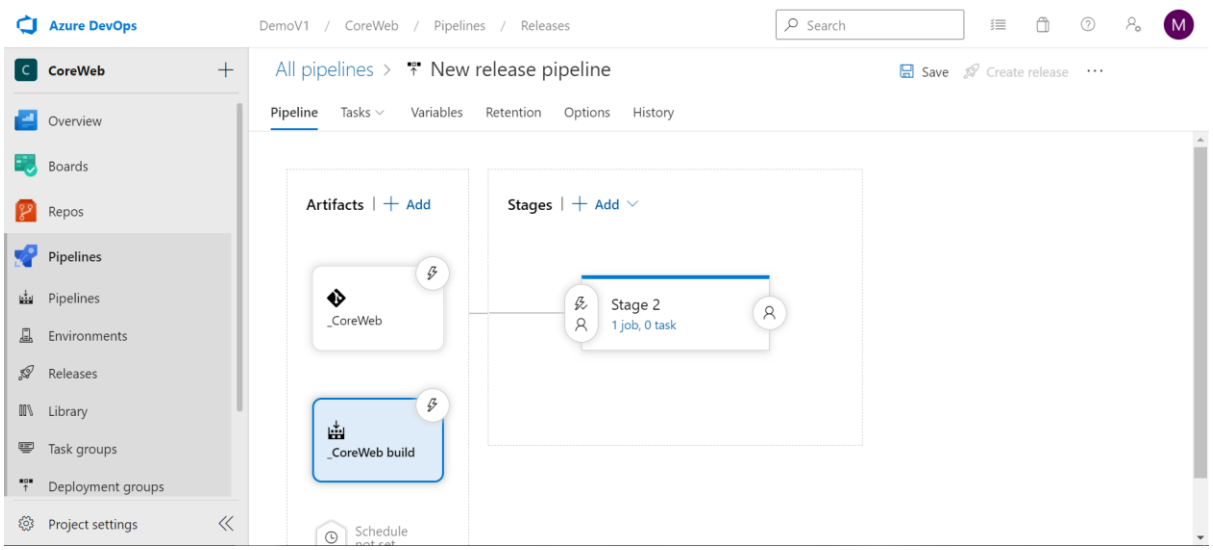


Setup Test

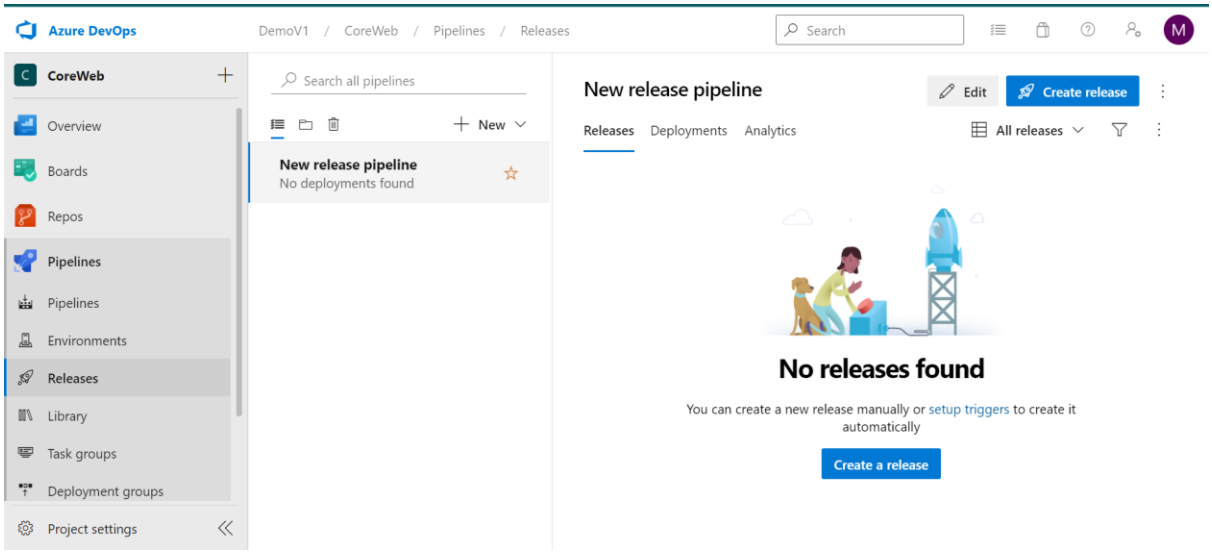
8.1 Release

Creating a release pipeline for our project is the next step in the process.
Under the pipeline, choose the "Release" option from the drop-down menu. By choosing "create new pipeline," you may begin the process of creating a new pipeline.
We may utilize either our most recent build or the most recent code from the repository for the release. I'm going to use the most recent build available from Artifact and design a stage as well.

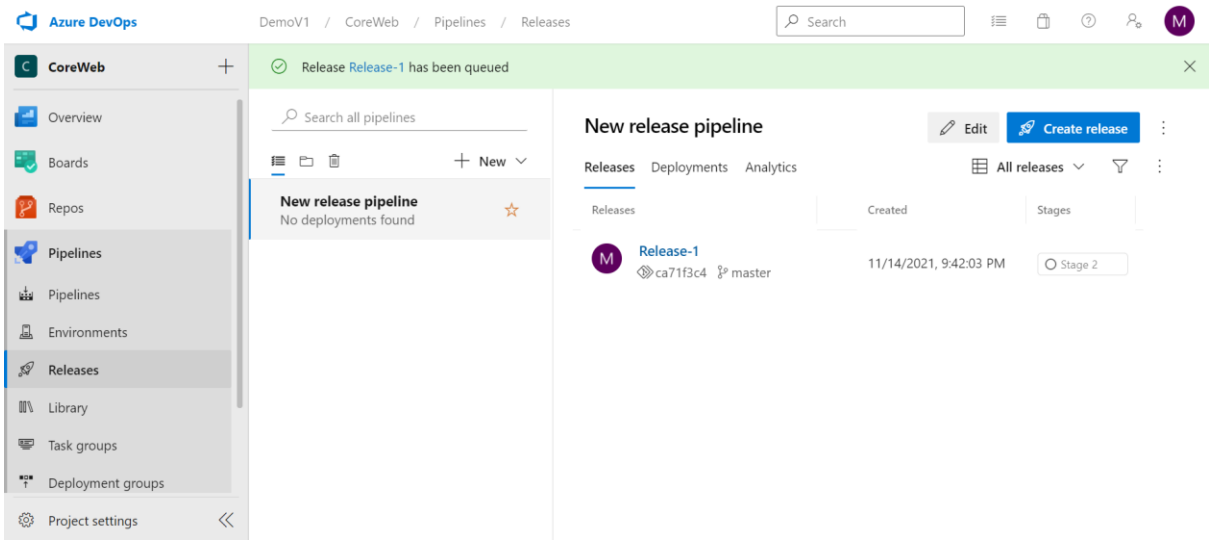




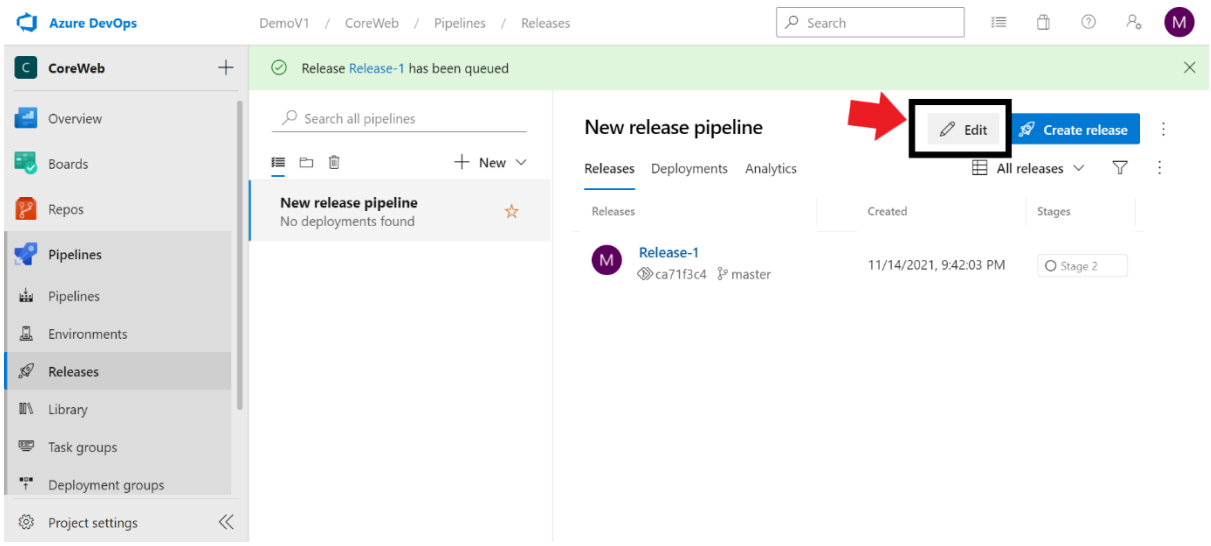
As soon as your release pipeline is successfully created, it will be shown in the release window for your review.



Create a release for the pipeline at this point.

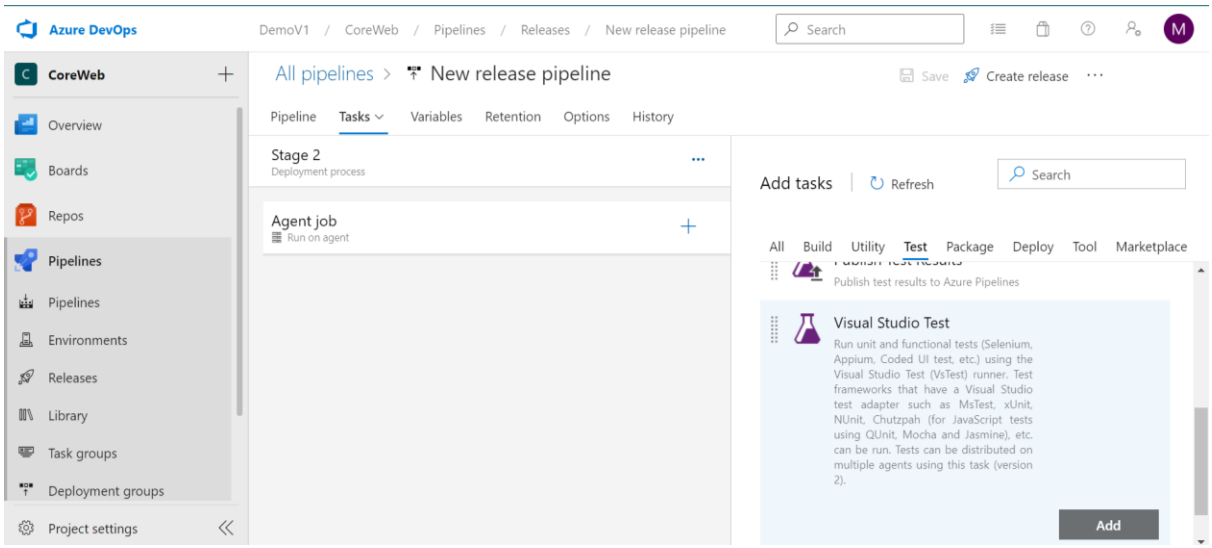


Lest edit the release pipeline. Click on the edit button.



"Visual studio test" should be included to the list of jobs for the new release process.

For our project, we made use of xunit. This job will be utilized for the pipeline in Azure DevOps, according to the documentation. The specifics may be found in the description that is shown in the window.



8.2 Run Test

By selecting the job under release stages and clicking on it, you will be able to access the visual studio test from the task window.

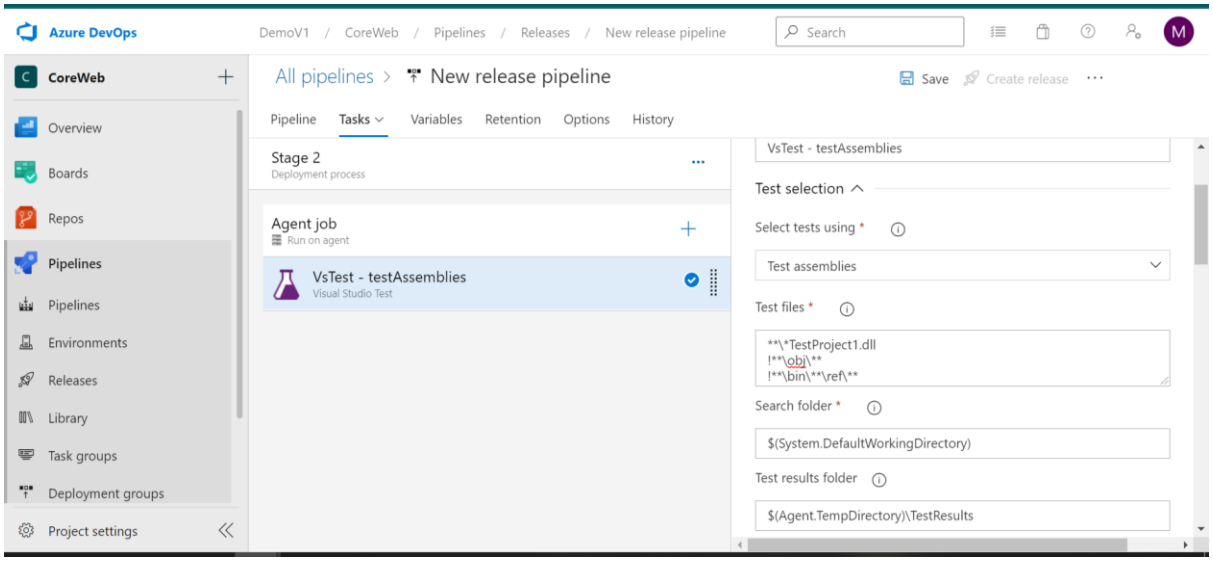
When you choose that Visual Studio test, the details of the test will be presented on the right side of the screen.

Always ensure that you are using the most recent version of everything. These are already configured by default, so you won't have to do anything to get started.

Now, in the text box of the test file, replace the text given below with the text shown below.

You should provide the name of your test project in the description.

****TestProject1.dll*
*!**\obj***
*!**\bin**\ref***

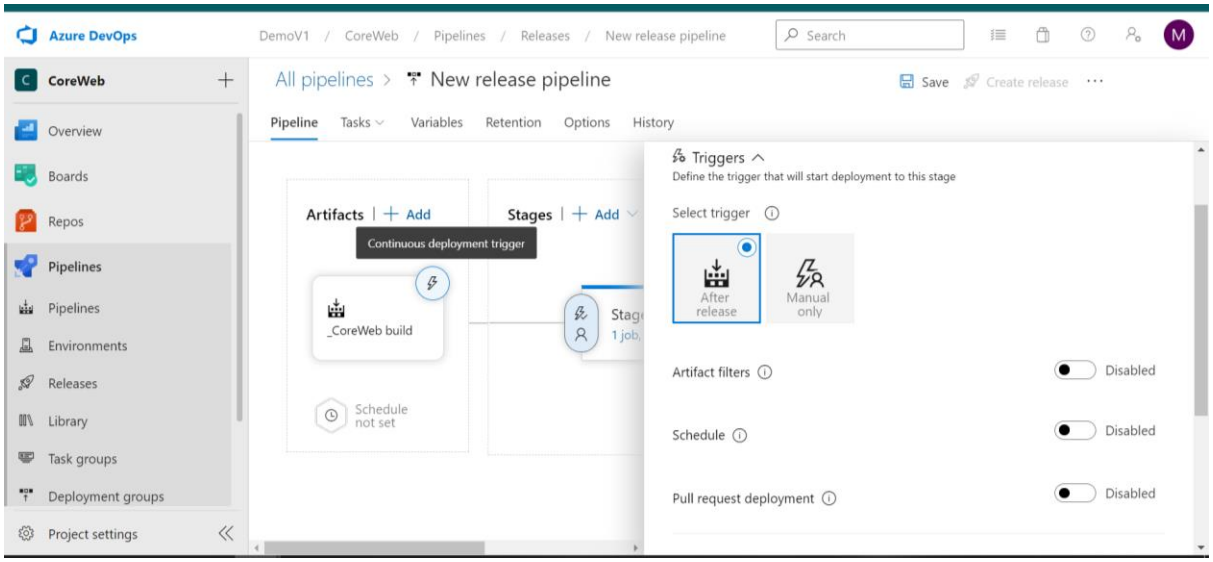


8.3 Triggers

You can discover the trigger for the pipeline by selecting it from the thunderbolt symbol. You have the option of setting up these triggers later or before the release of the product.

For the stage one filter, you can choose it after the release option.

There are a plethora of alternatives accessible when it comes to triggers.



Chapter 09

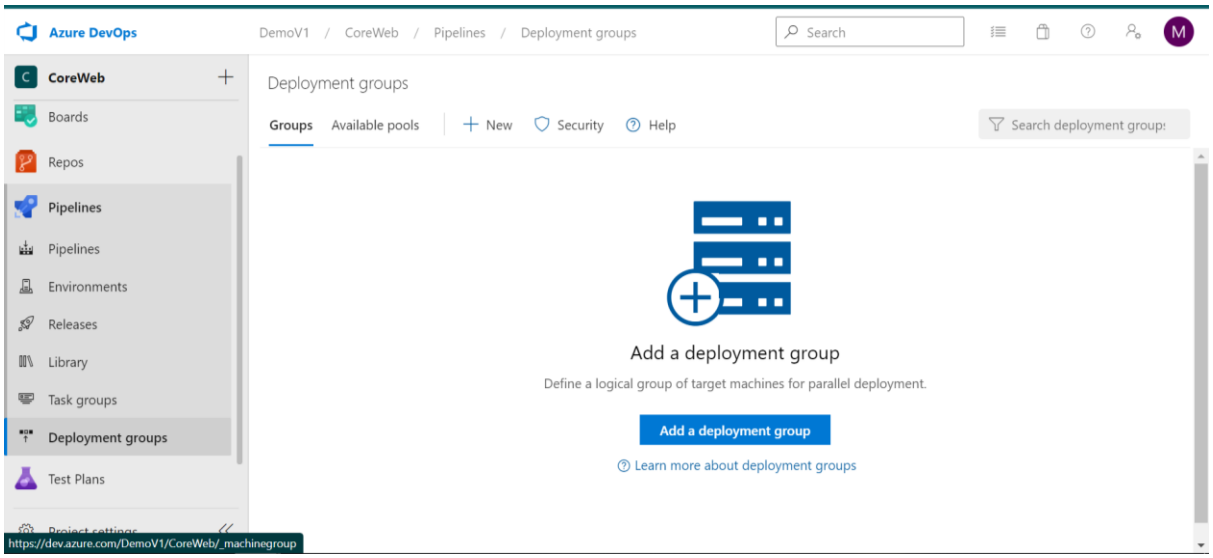
Deployment

9.1 Setup Group

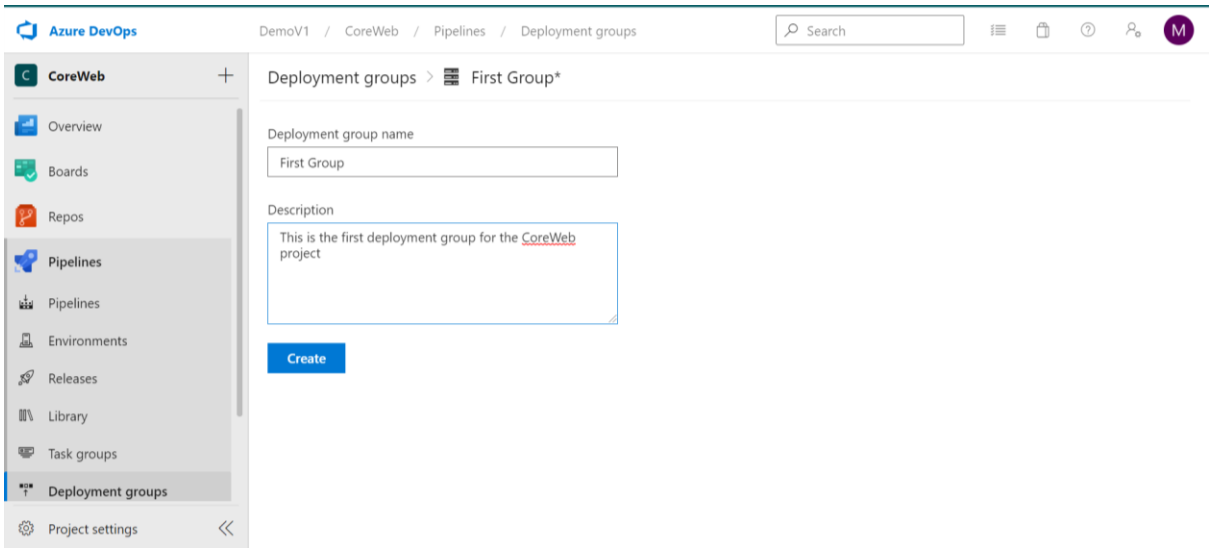
This implies that it is essentially a connection to the server, and in order to establish this connection, we must first get an access token.

To begin, you must first establish a group.

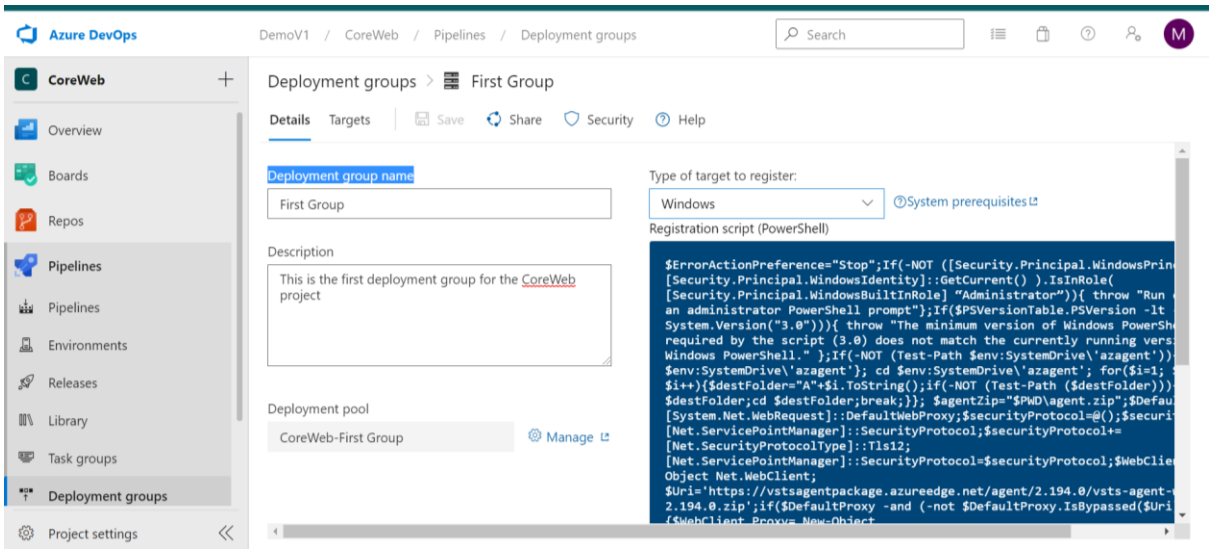
"Deployment Groups" is an option that can be found under the "Pipeline" menu. Select that choice, and a new window will appear, allowing you to form a "Deployment Group."



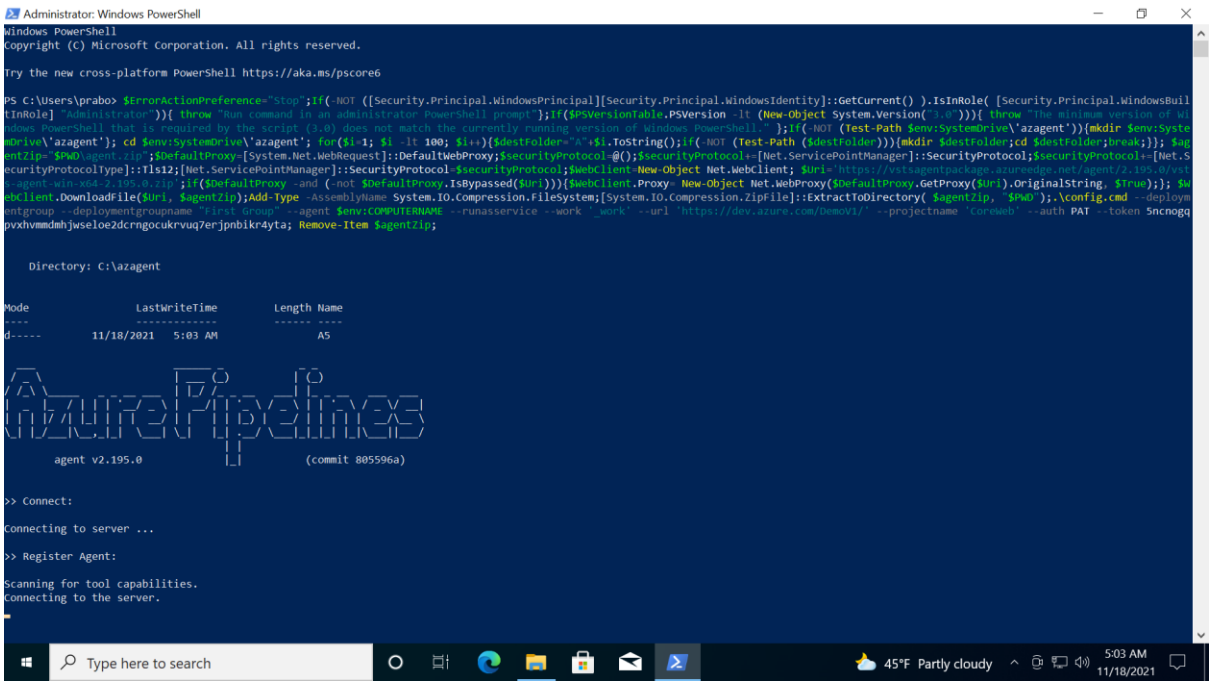
Save it by clicking on "Add a deployment group" and providing it with a name, a description, and the option to add other deployment groups.



Your registration script will be presented in a new window when you have completed the registration process.

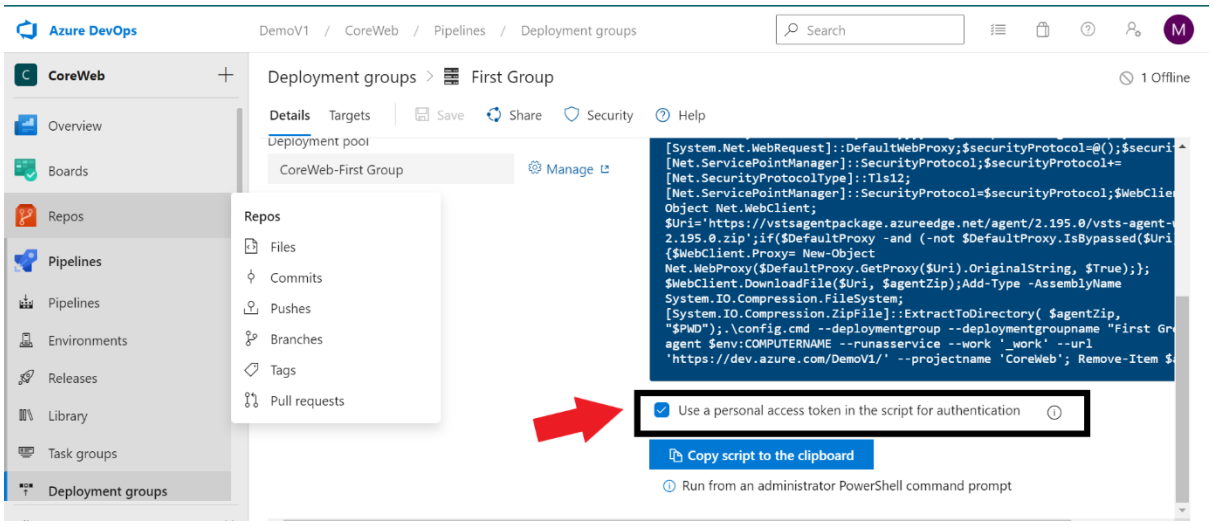


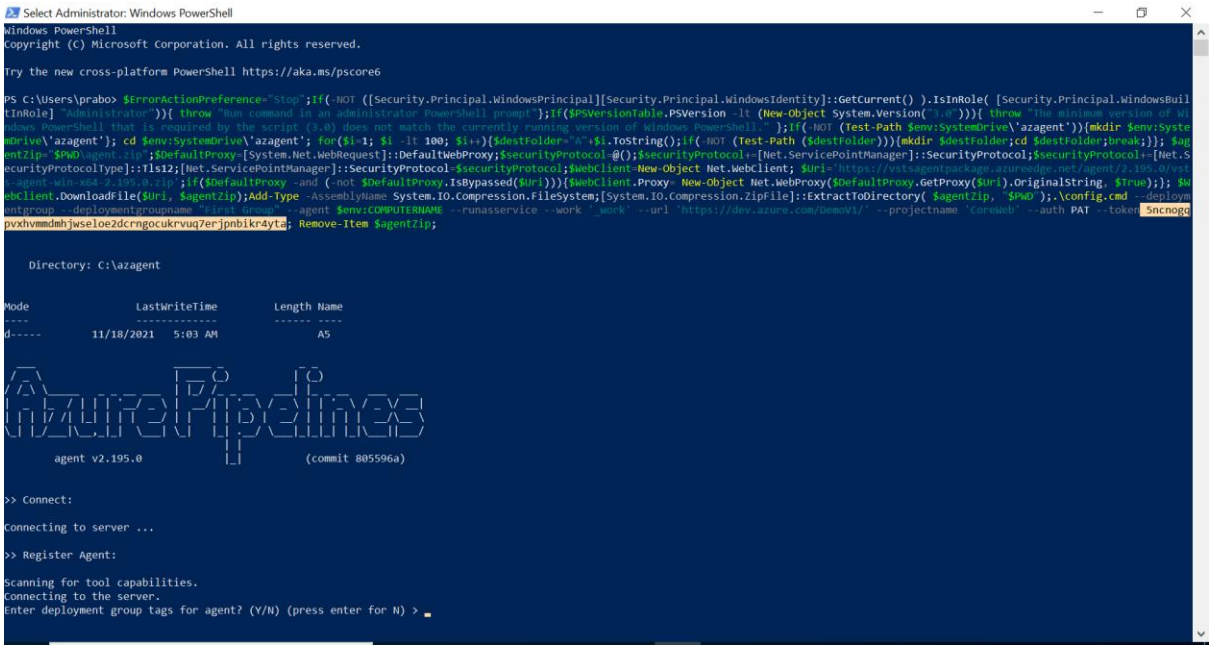
The script should be copied from the window and pasted into the power shell of the server where you want your project to be executed. Paste your script into power shell (this should be PowerShell, not command prompt); otherwise, it will not work.



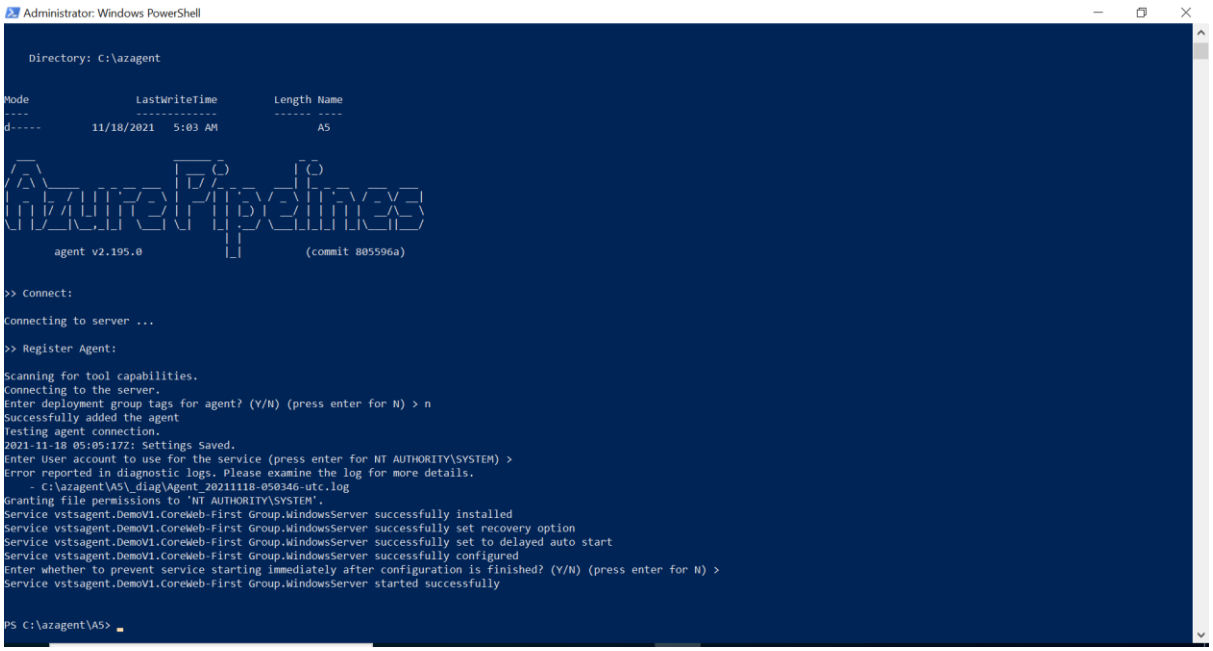
Check the option "Use a personal access token in the script for authentication" before downloading the deployment registration script for your deployment group so that you may authenticate using your personal access token. This will add a PAT (Personal Access Token) to the script you are now running. Otherwise, when you

execute your script on the server, it will request your PAT in order to create a connection with the remote computer.

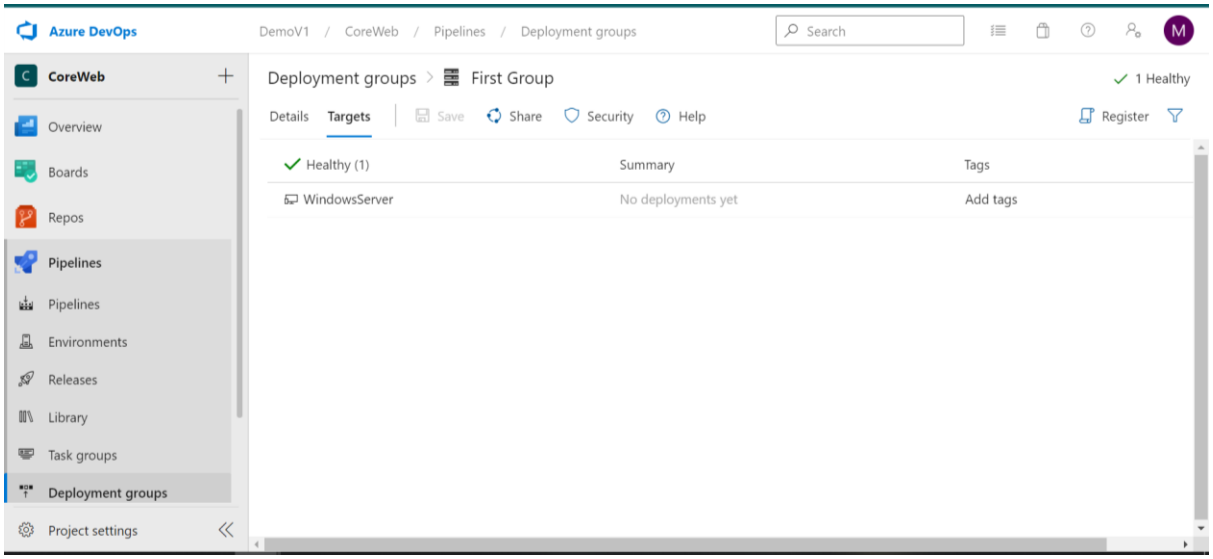




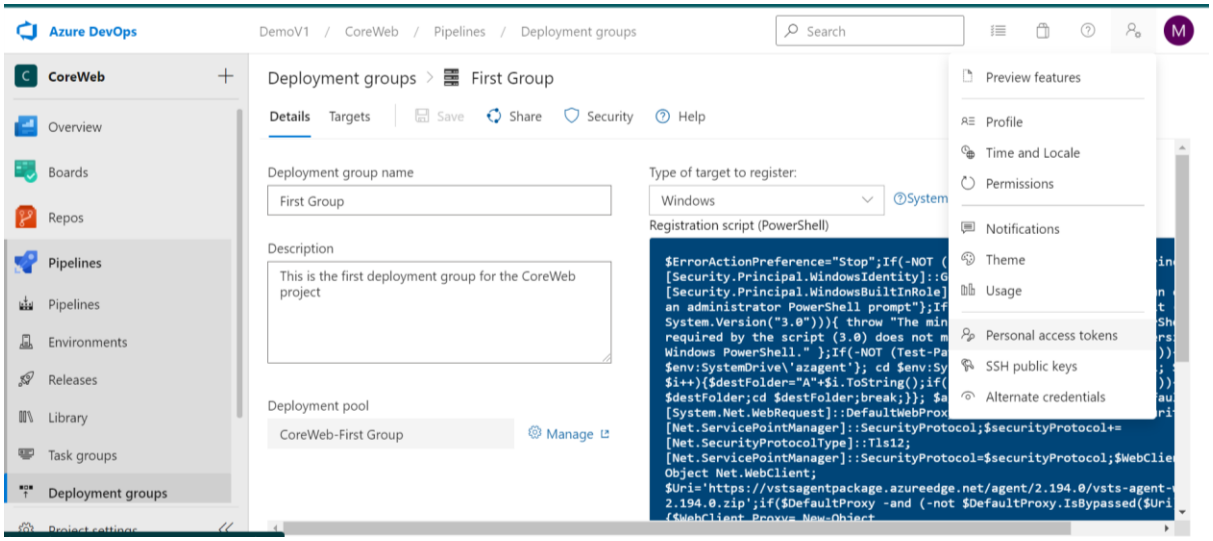
When the setup is complete, you will be able to view the power shell as seen below.



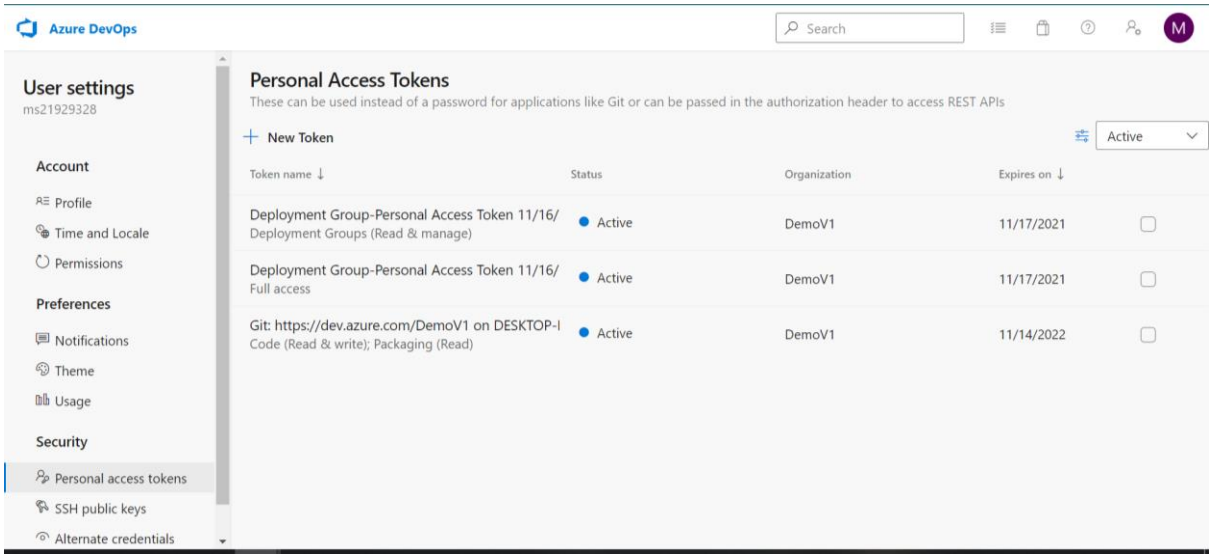
Navigate to your Azure DevOps portal at this point. Go to your deployment group, and under your deployment group, there is a target tab that you may use to find your target. If you go to that page, you will be able to see that your server will be available. As a result, your connection was established successfully.



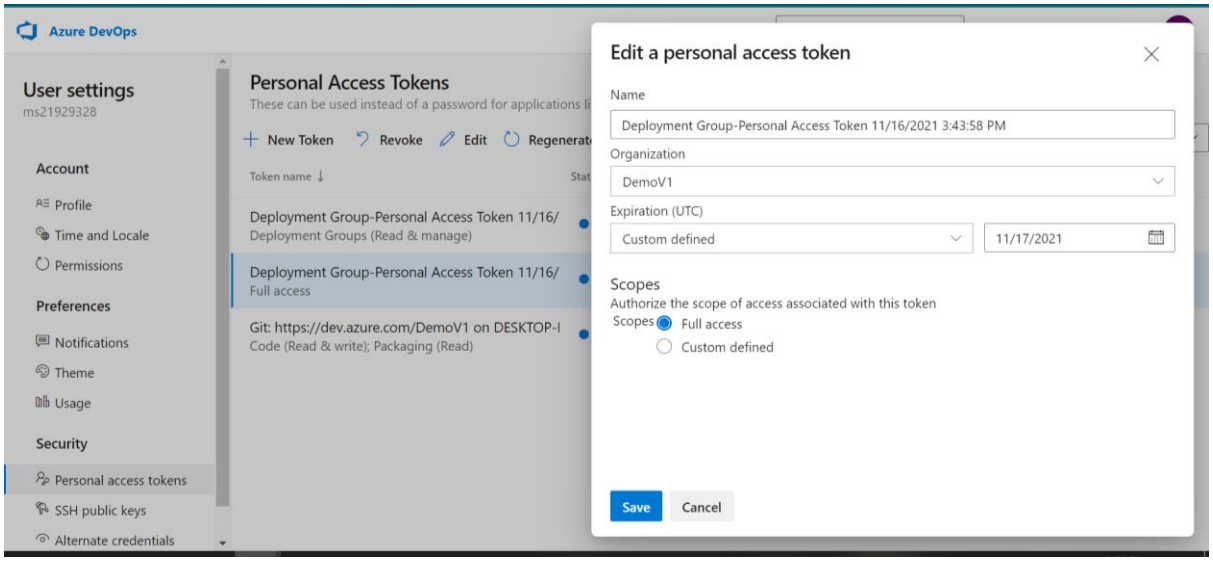
You have the option of adding a personal access token for your organization if you so want. That option is accessible by clicking on your profile picture. the option should be chosen



The access token pane for your personal information will reveal the access token that is presently in use for you. Without a doubt, there should be a number of different personal access tokens that are available.

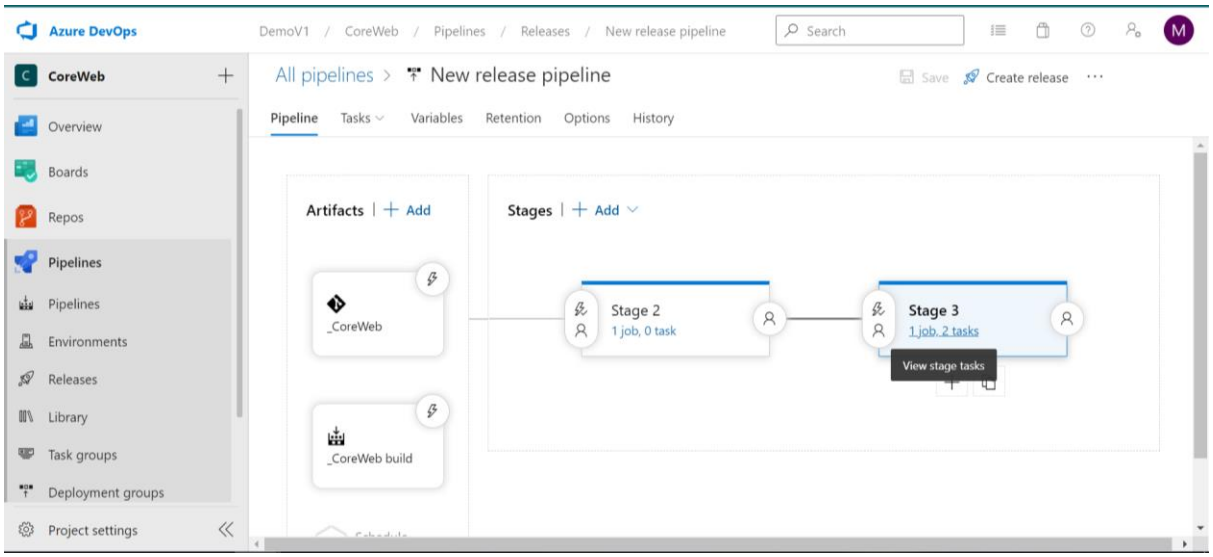


You have the ability to modify those access tokens. For your access tokens, you have a number of different alternatives to consider. However, it is preferable if you can provide complete access to your personal tokens since something will forget you in the future, causing you a lot of trouble. However, if you provide complete access, there will be no problem.

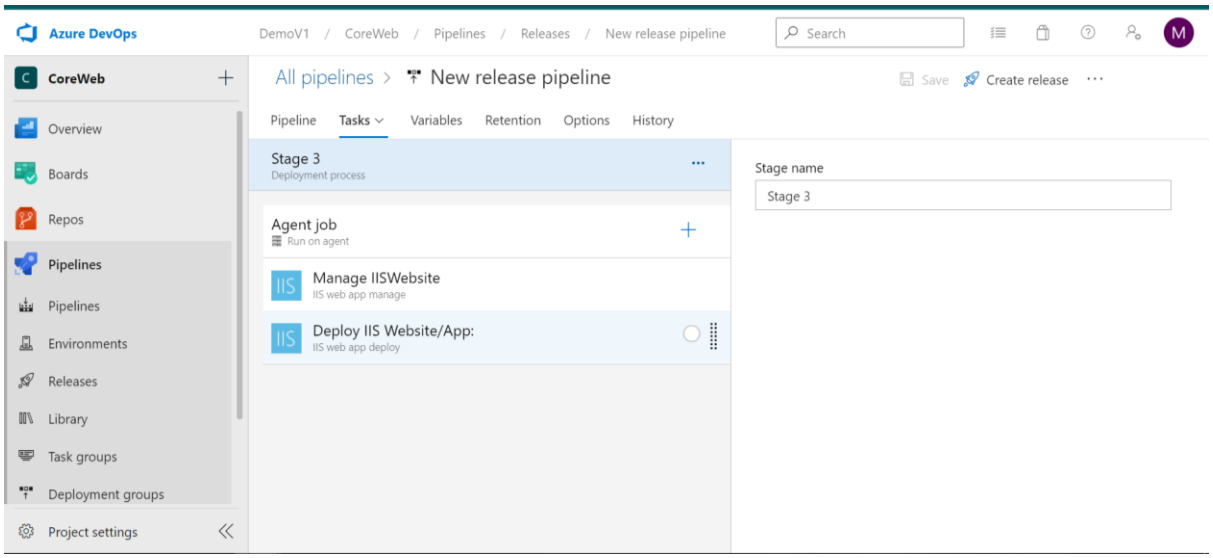


9.2 IIS Deployment Manage

I've introduced a new step to our pipeline in order to streamline the deployment process. "Stage 3" is what I called it.



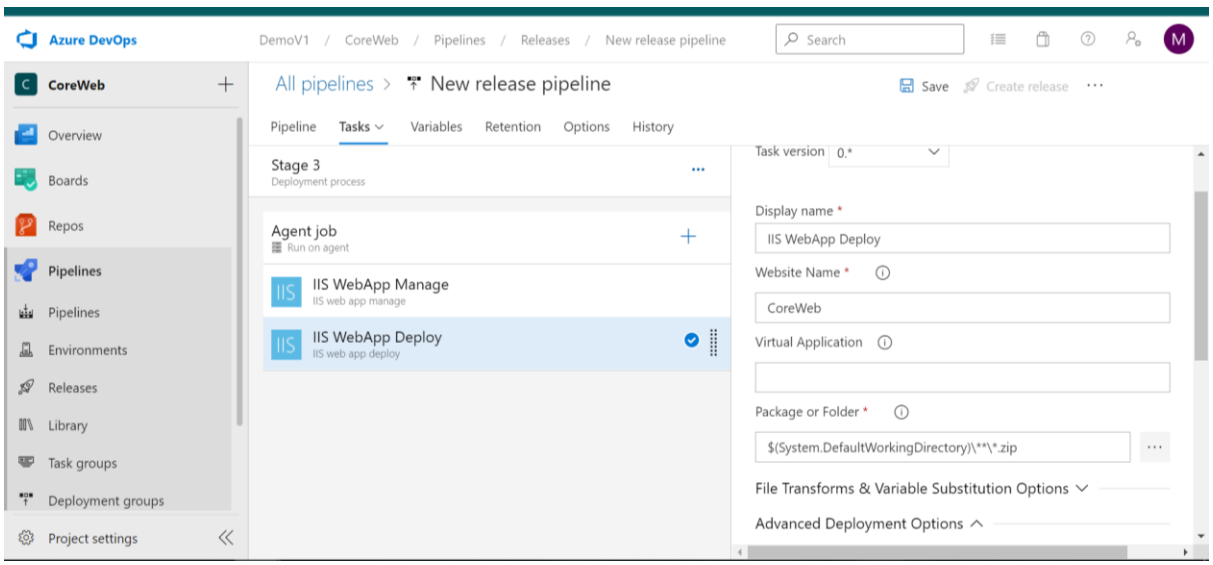
if we review the stage 3 it has "IIS Web app manage" and "IIS web app deploy". so this stage is work for iis management and deployment.



I renamed the two tasks "IIS WebApp Manage" and "IIS WebApp Deploy" in order to make them simpler to comprehend and distinguish between them.

The "IIS WebApp Manage" command is used to configure the website. This includes both the IIS manager and the website, as well as the fundamentals of administering a web site.

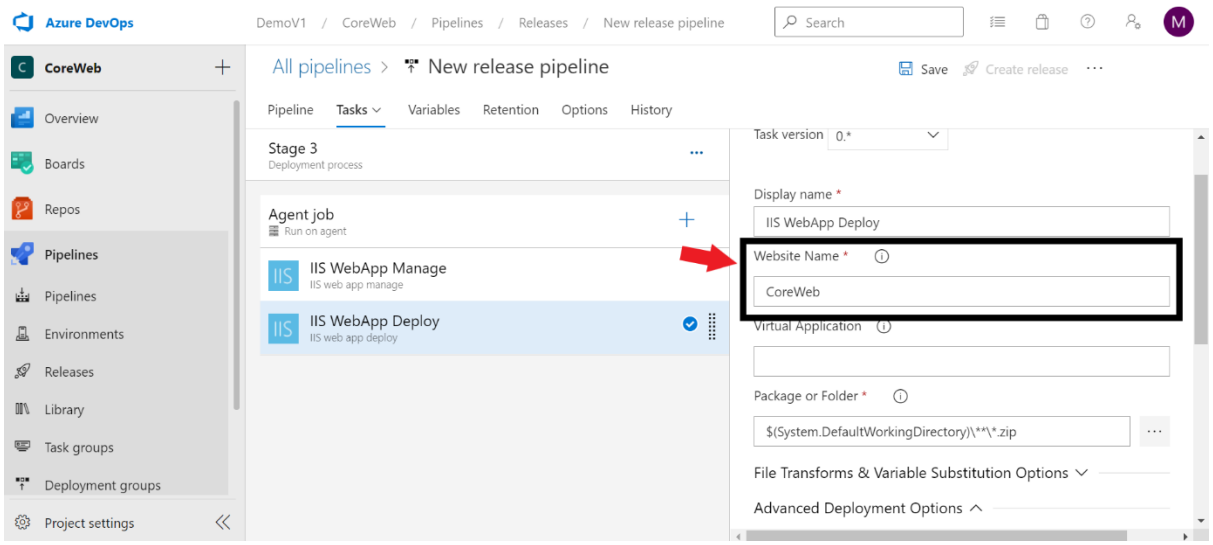
The "IIS WebApp Deploy" command is used to initiate the deployment of a web application.



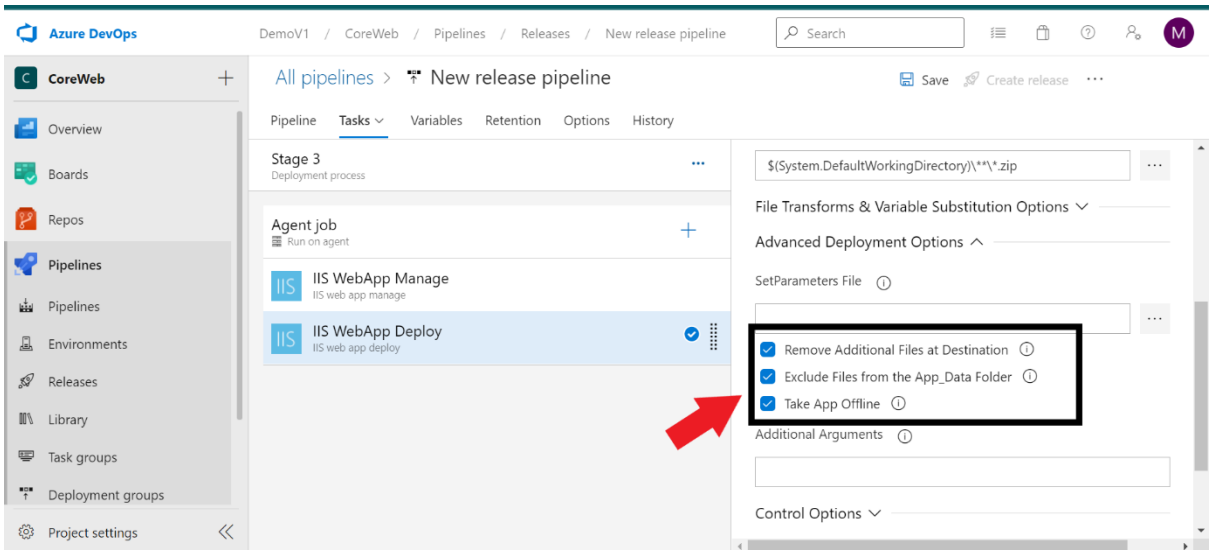
9.3 IIS Deployment

Let's have a look at how we can put the idea into production.

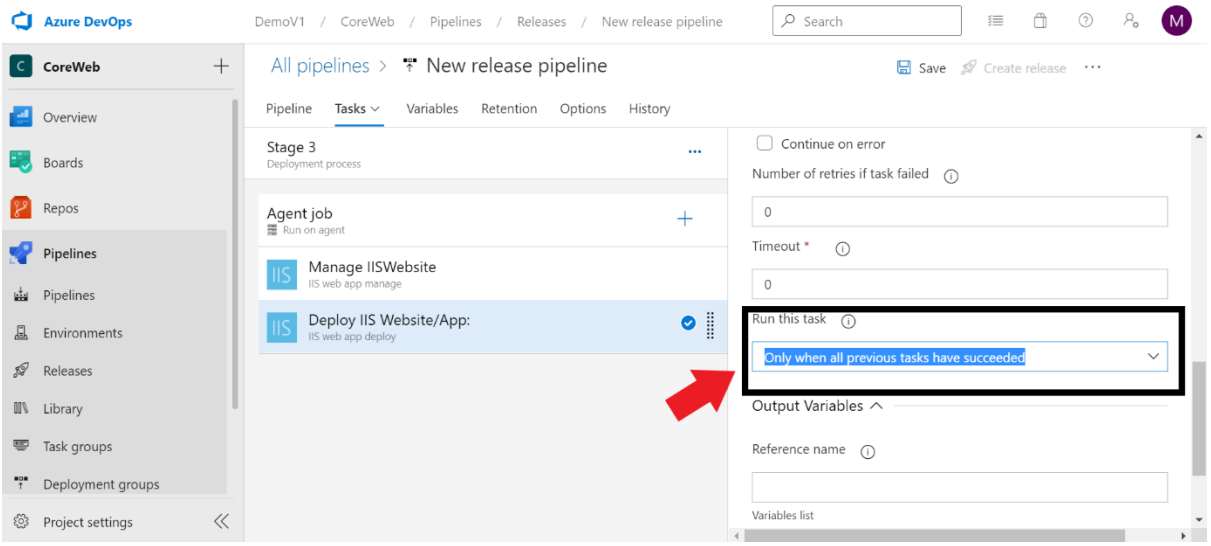
You may locate your name on the website that you are going to construct based on the domain name that you have chosen. You may make changes to it right here if you like.



I'm going to choose the following parameters from the advanced options since I'll require these files for my web application that has already been launched.



We must deploy this application "only after all preceding tasks have been completed successfully." Otherwise, it contains problems, and faults necessitate the necessity for a deployment. Check to determine whether the previous build was successful before deploying the new version. If it is successful, the deployment will take place.



References

https://www.tutorialsteacher.com/core/dotnet-core

https://www.redhat.com/en/topics/devops/what-is-ci-cd

https://www.redhat.com/en/topics/devops/what-cicd-pipeline

https://semaphoreci.com/blog/cicd-pipeline

https://www.atlassian.com/devops/what-is-devops/benefits-of-devops

https://www.c-sharpcorner.com/article/what-is-azure-devops/

https://azure.microsoft.com/en-us/services/devops/#overview

https://kinsta.com/knowledgebase/what-is-github/