

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/374782401>

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

Introduction.....	7
What is DevOps.....	8
Why DevOps	8
Application life cycle with DevOps.....	8
Azure DevOps.....	10
Introduction.....	11
Online and On-Premises.....	11
History of Azure DevOps	12
Keywords	12
Tasks	14
CI/CD Pipeline	13
What is CI/CD pipeline	16
Continuous Integration	17
Continuous Delivery	17
.Net Core and GitHub.....	18
.Net Core	19
GitHub	20
Create .Net Core Application.....	21
Add Unit Test Project.....	25
Add project to the GitHub.....	40
CI/CD Pipeline for .NET with Azure DevOps.....	44
Introduction	45
.Net Core application from the azure portal.....	45
.Net Core application from GitHub	67
Azure DevOps and Visual Studio	84
Clone project from Azure DevOps	84
Clone project using URL	85
Project Details	85
Continuous Integration	87
Create Pipeline.....	88
YAML	88
New Build.....	89
Job	90
Setup Test.....	92
Release	93
Run Test	96

Triggers.....	96
Deployment.....	98
Setup Group.....	99
IIS Deployment Manage	104
IIS Deployment	106

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). 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.

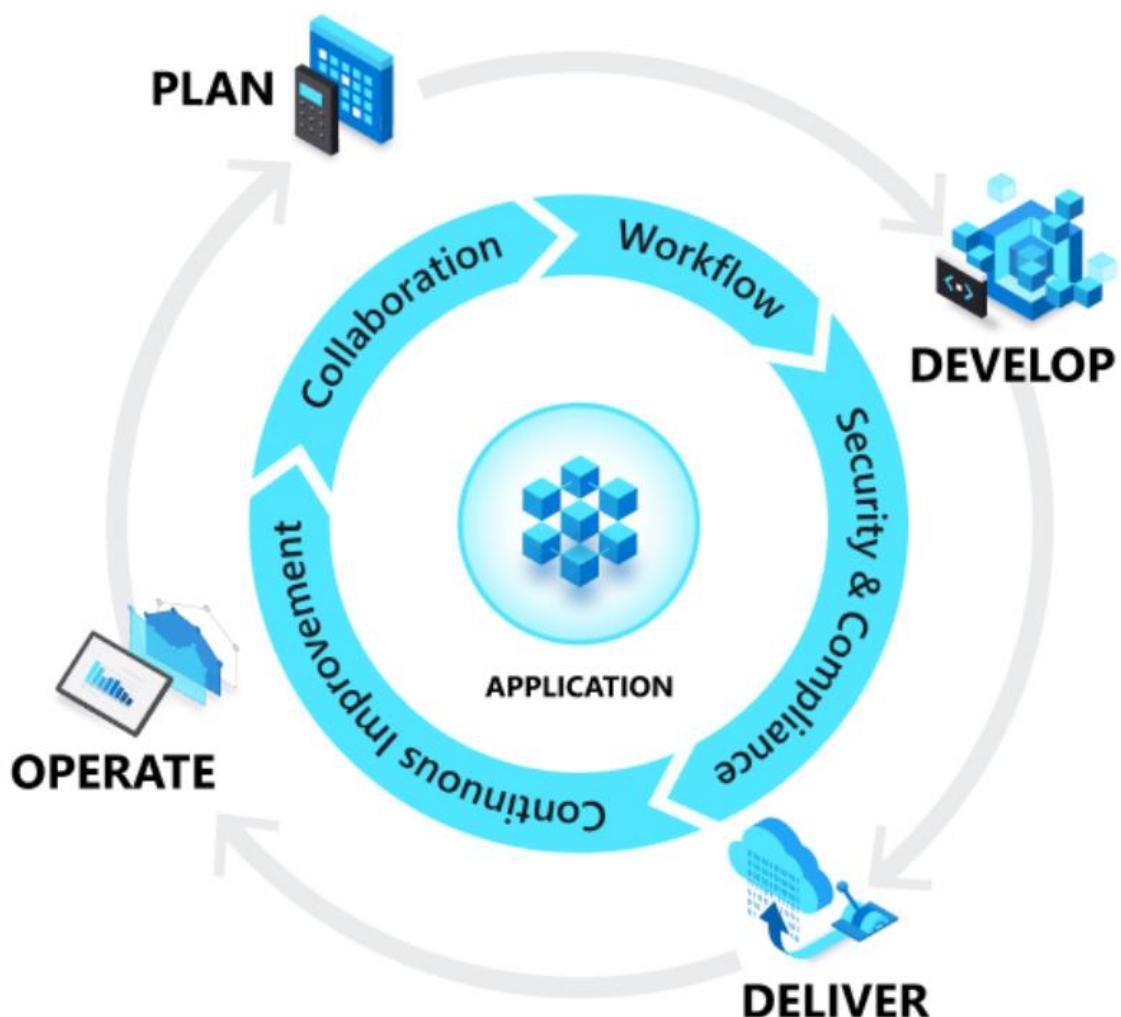
1.2.3 Transparency

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

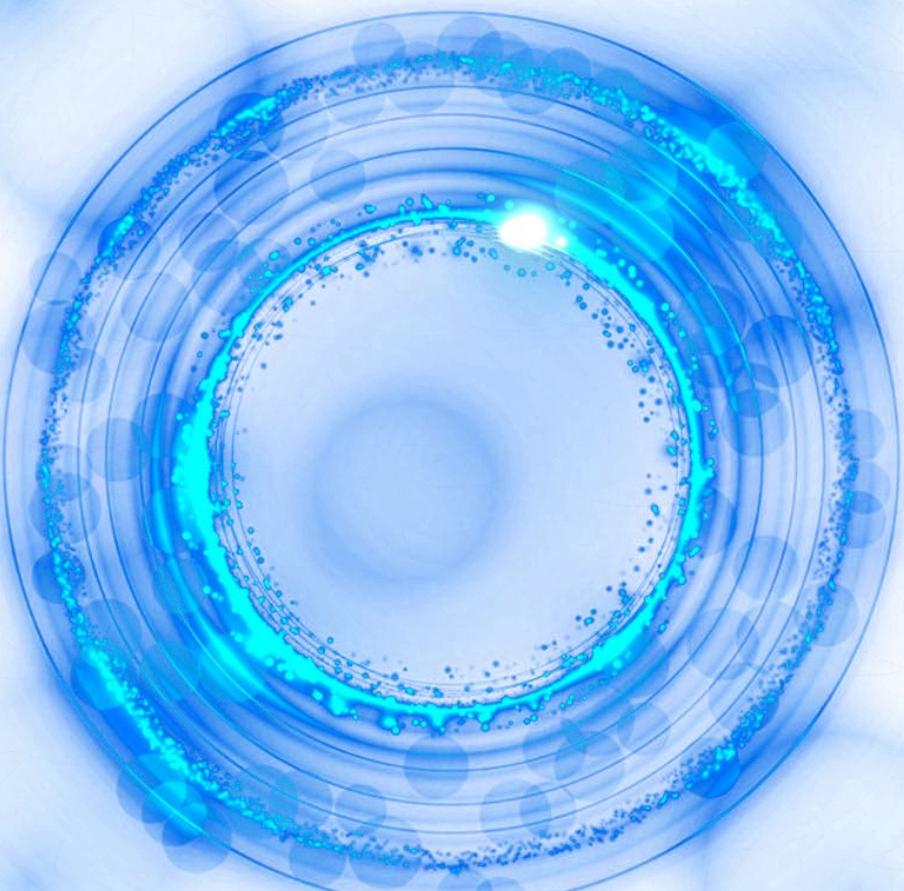
We may reduce the cost of our department's product and management by implementing DevOps practices. All of this is included in the DevOps package, so we don't have to look for separate substitutes for each activity.

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.]

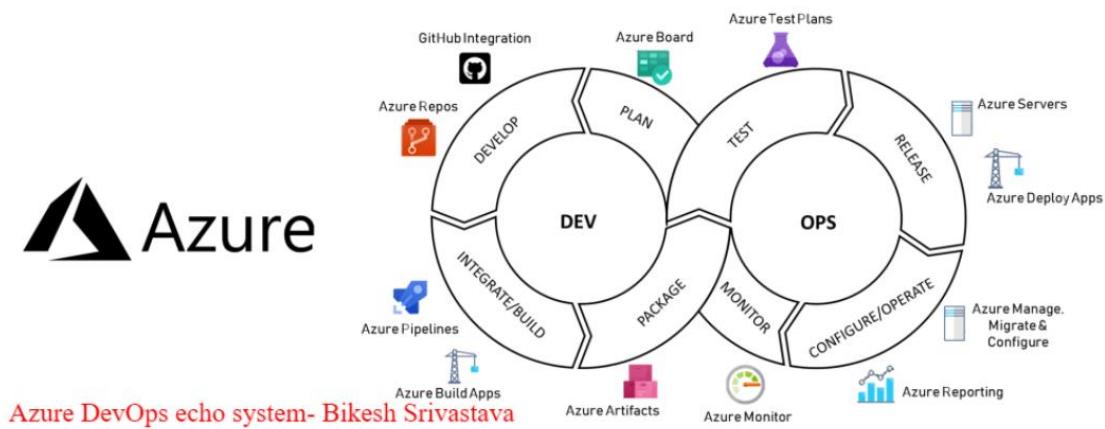
Chapter 02



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.

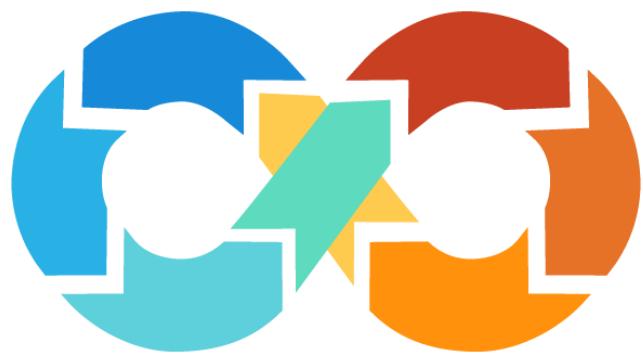
2.5.6 Release

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

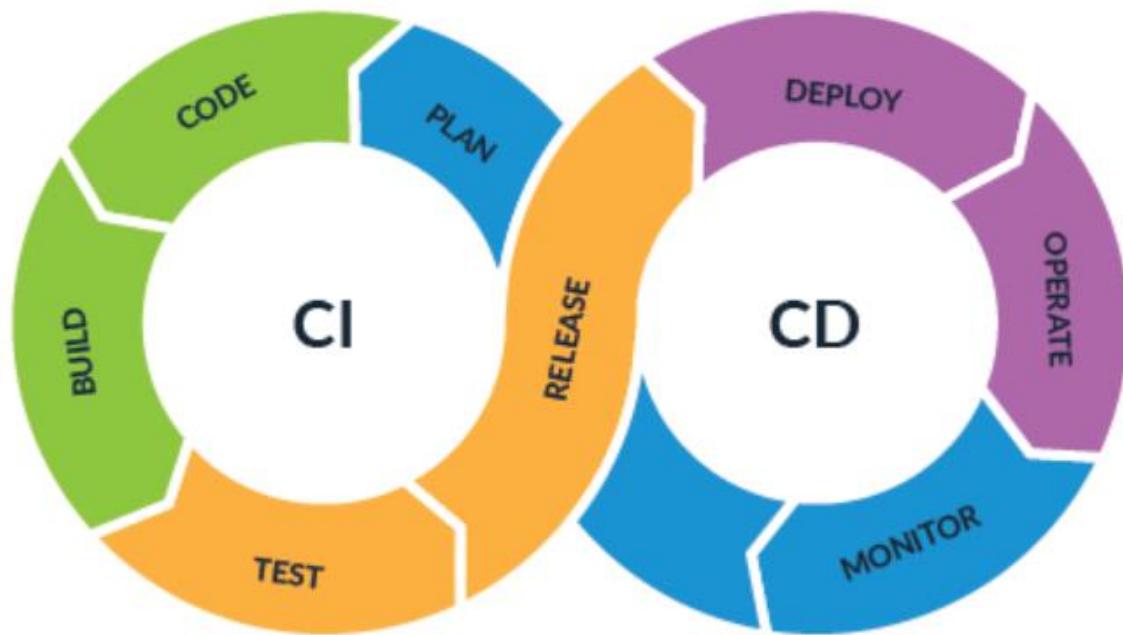


CI/CD Pipeline

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 .NET framework, 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 2019	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.

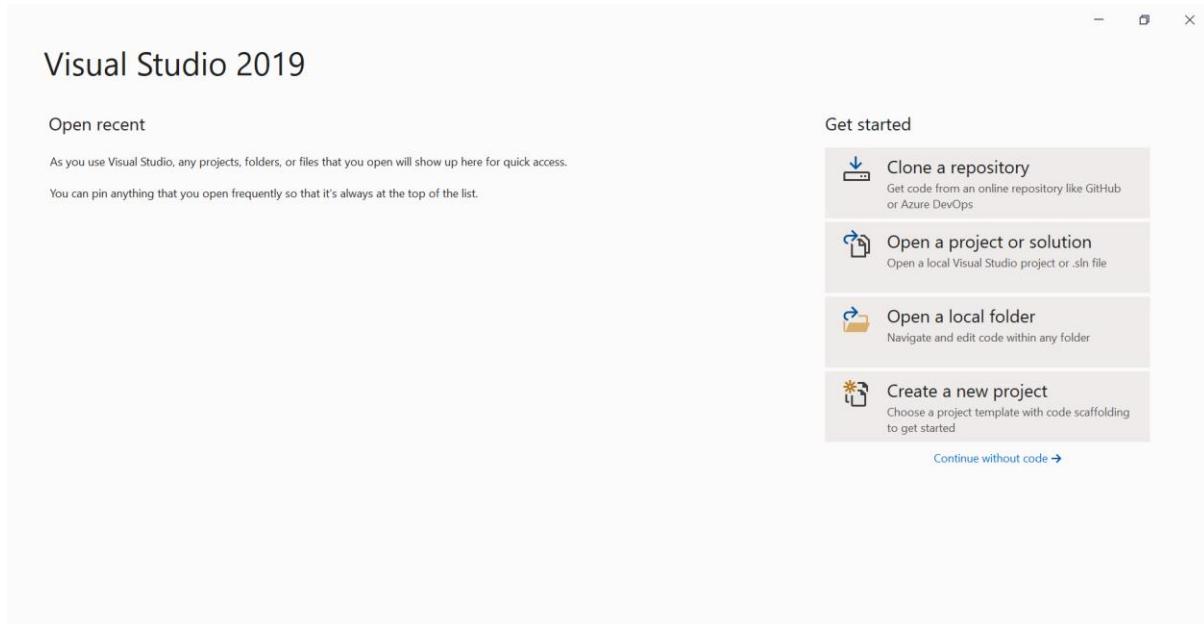
Git

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

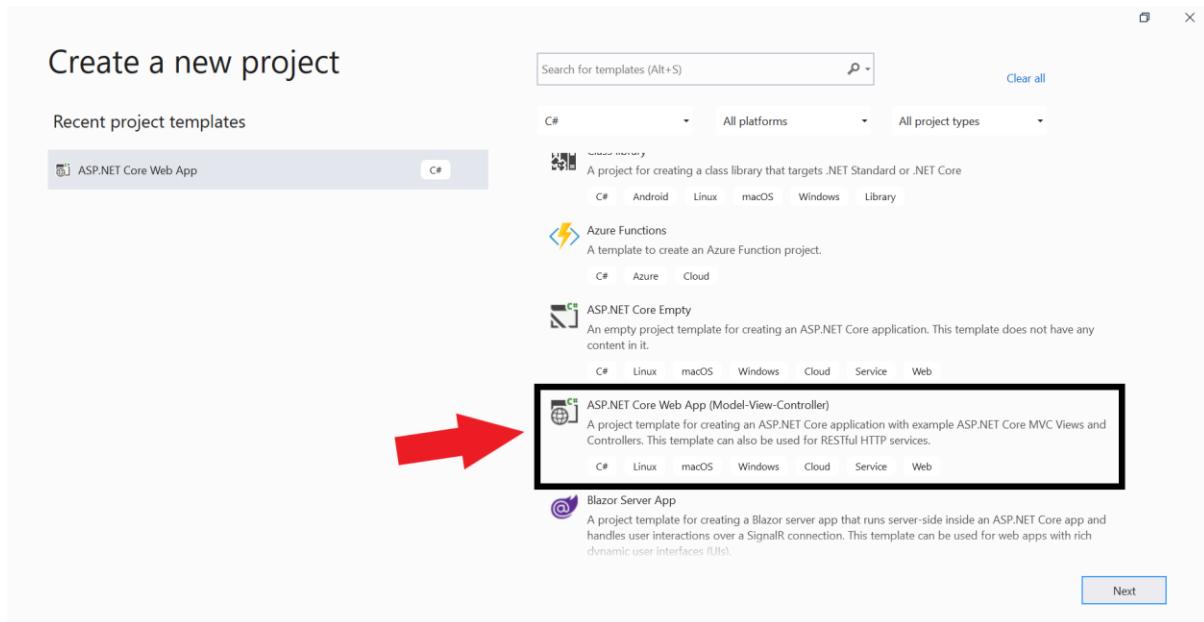
- 1) Select the “Create the new project” option



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

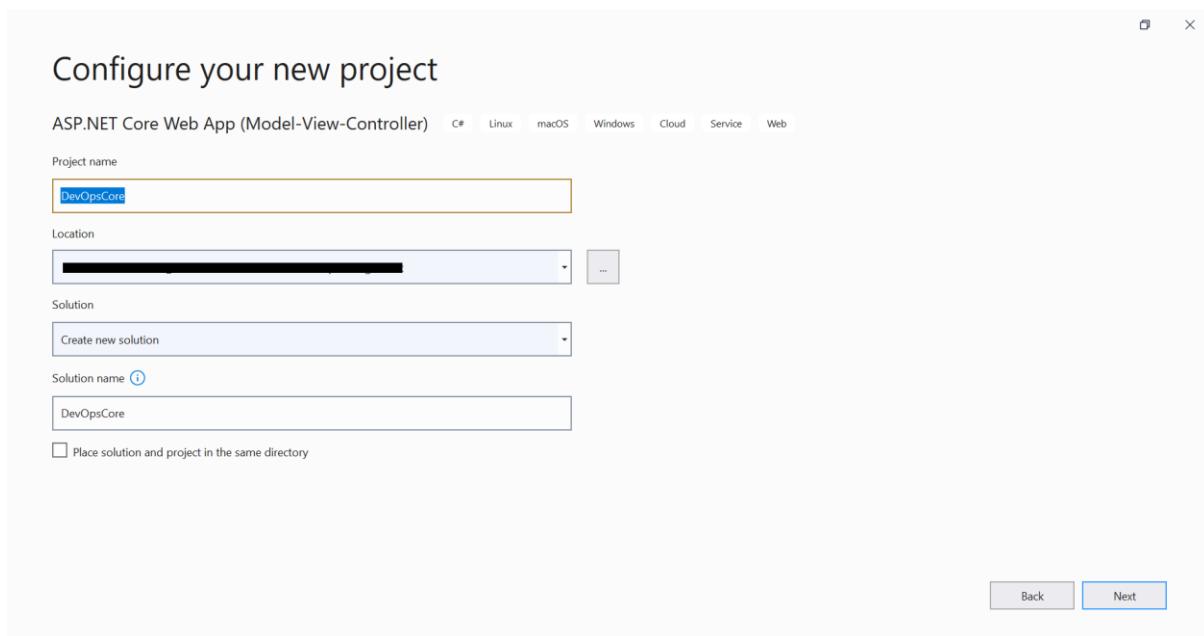
You can search for languages and platforms using drop-down menus.

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

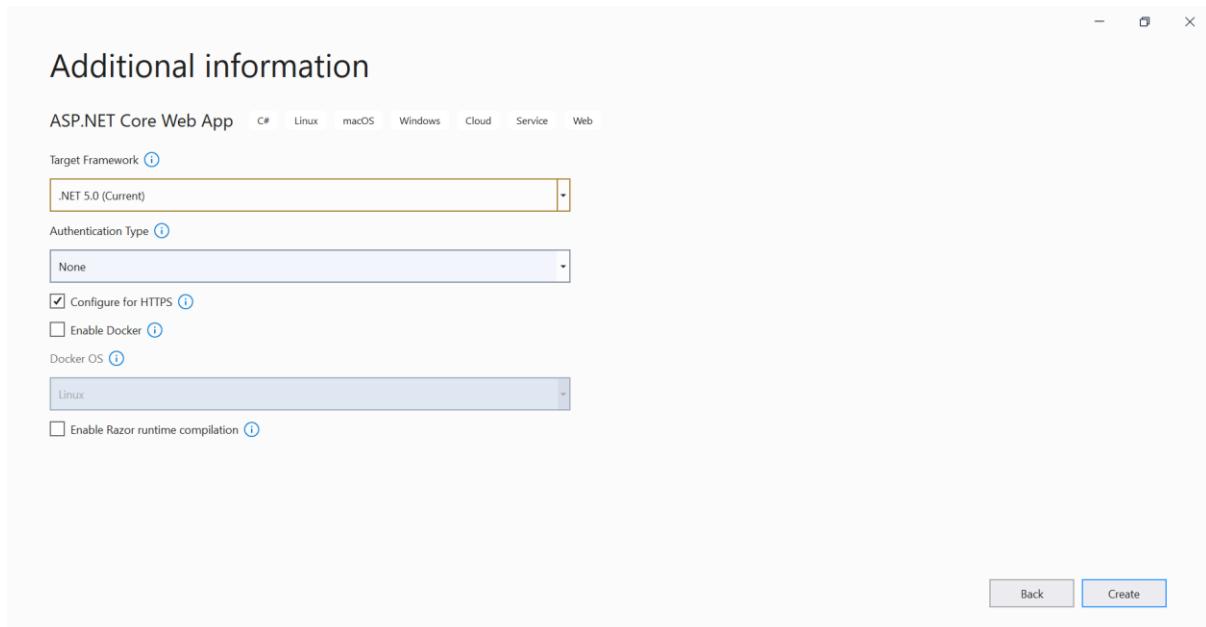
After that click the Next button.



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.

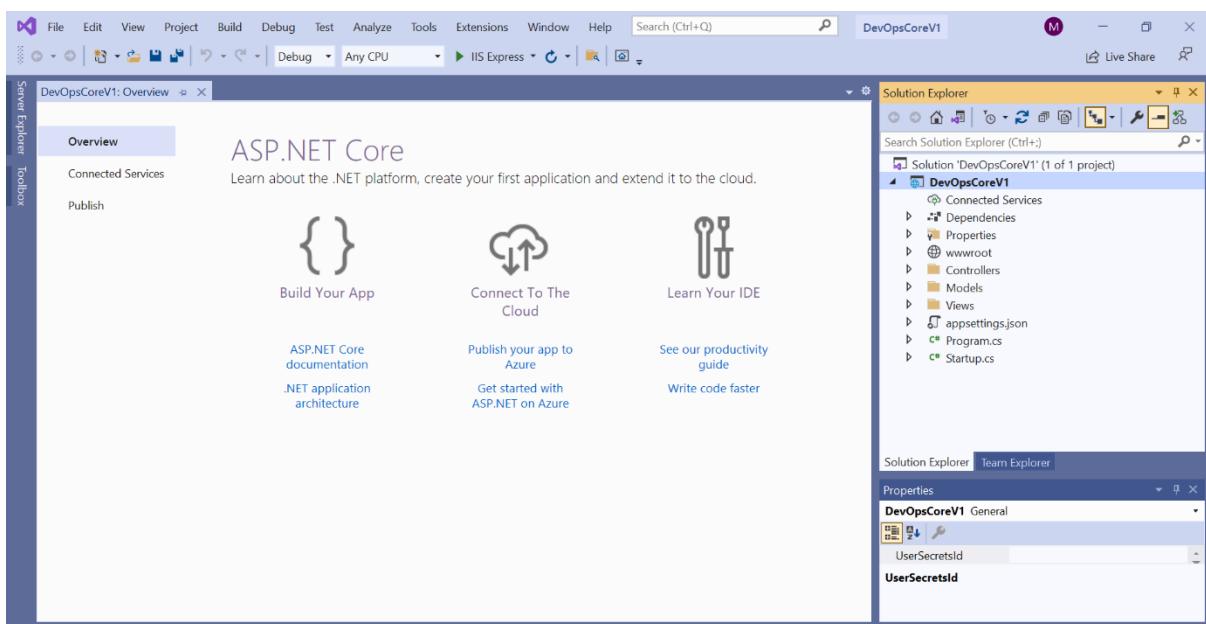
You are free to choose any version from the drop-down menu that you like to work with while creating your document.

Then click the Create button.



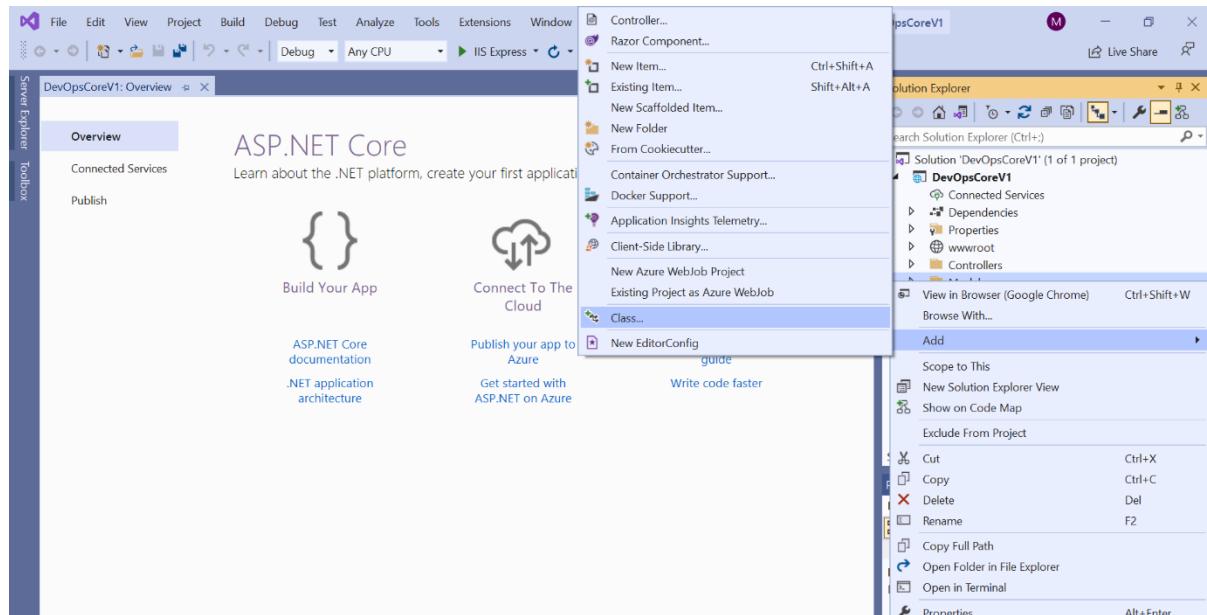
5) Following the preceding step, you will be sent to the Visual Studio IDE.

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



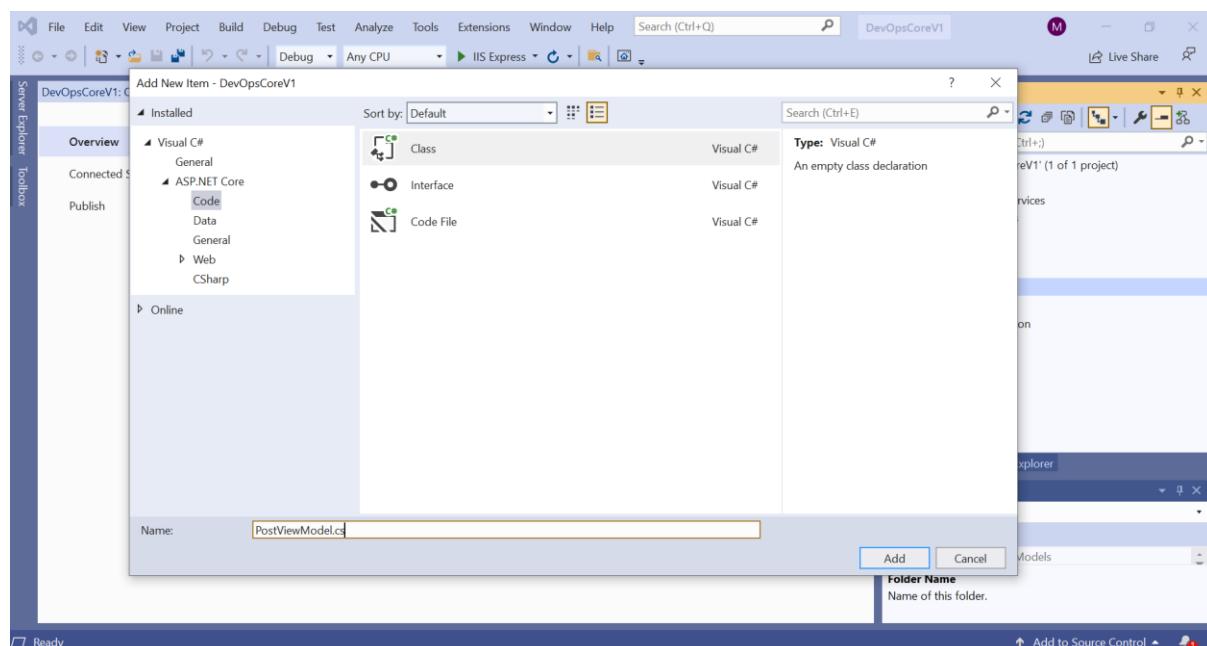
- 6) Let's make a few changes to the Model, View, and Controller to see how they work together.

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.



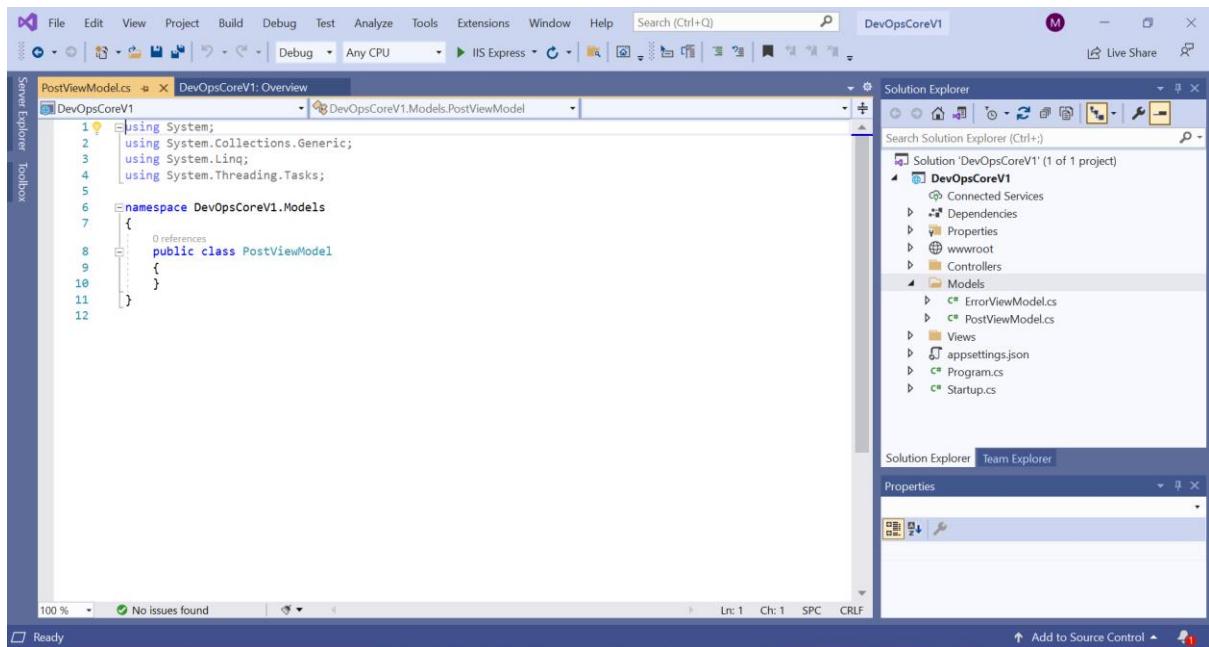
You have the option of giving the model class a name. My model class will be named "PostModel," and it will be implemented in the project

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 Id { get; set; }
        public string Name { get; set; }
    }
}
```

The screenshot shows the Microsoft Visual Studio interface. The code editor displays the following C# code:

```

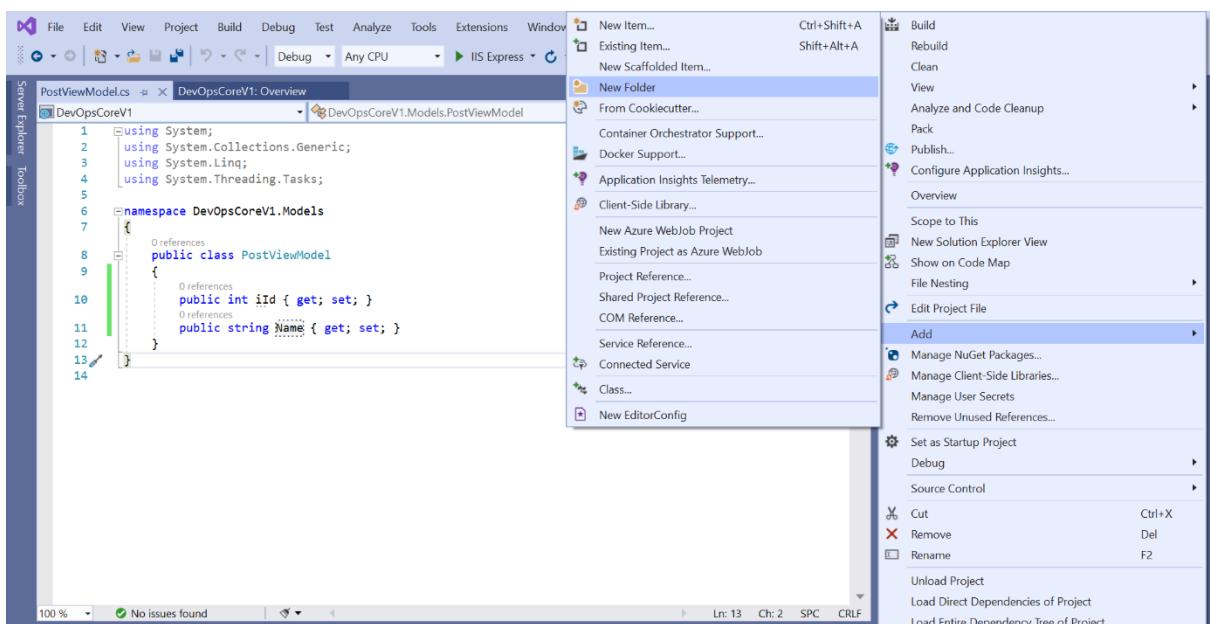
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace DevOpsCoreV1.Models
7  {
8      public class PostViewModel
9      {
10         public int ID { get; set; }
11         public string Name { get; set; }
12     }
13 }
14

```

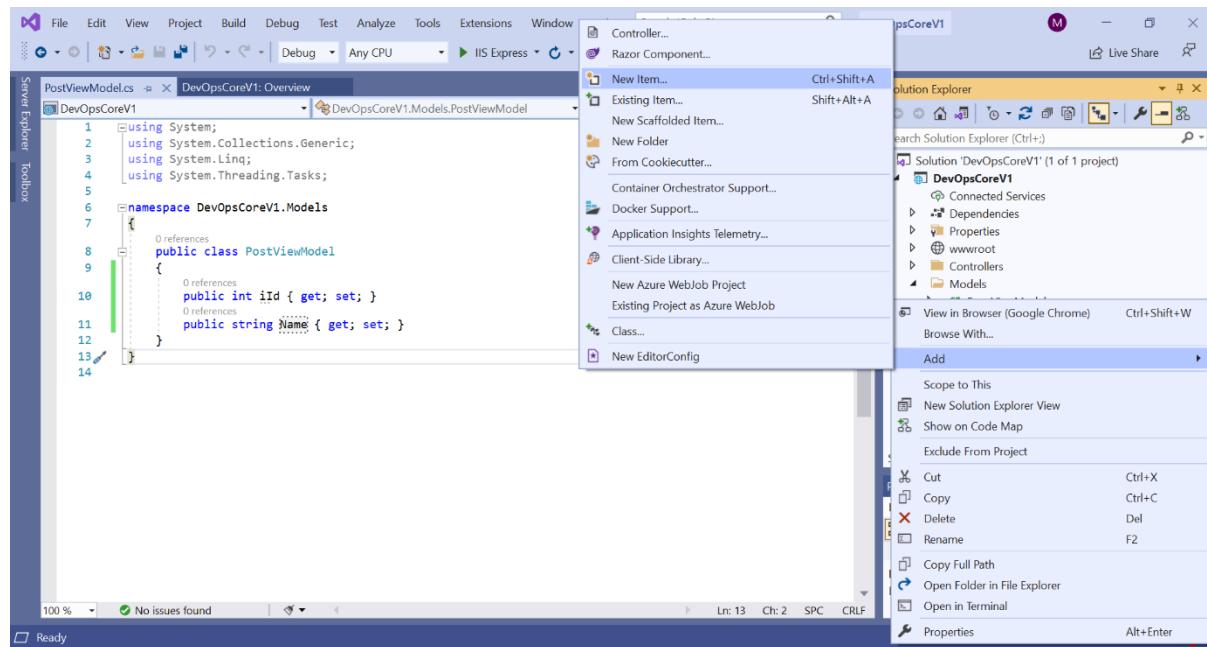
The Solution Explorer window on the right shows the project structure for "DevOpsCoreV1". The "Models" folder contains "ErrorViewModel.cs", "PostViewModel.cs", and "Startup.cs". The Properties window is also visible at the bottom.

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

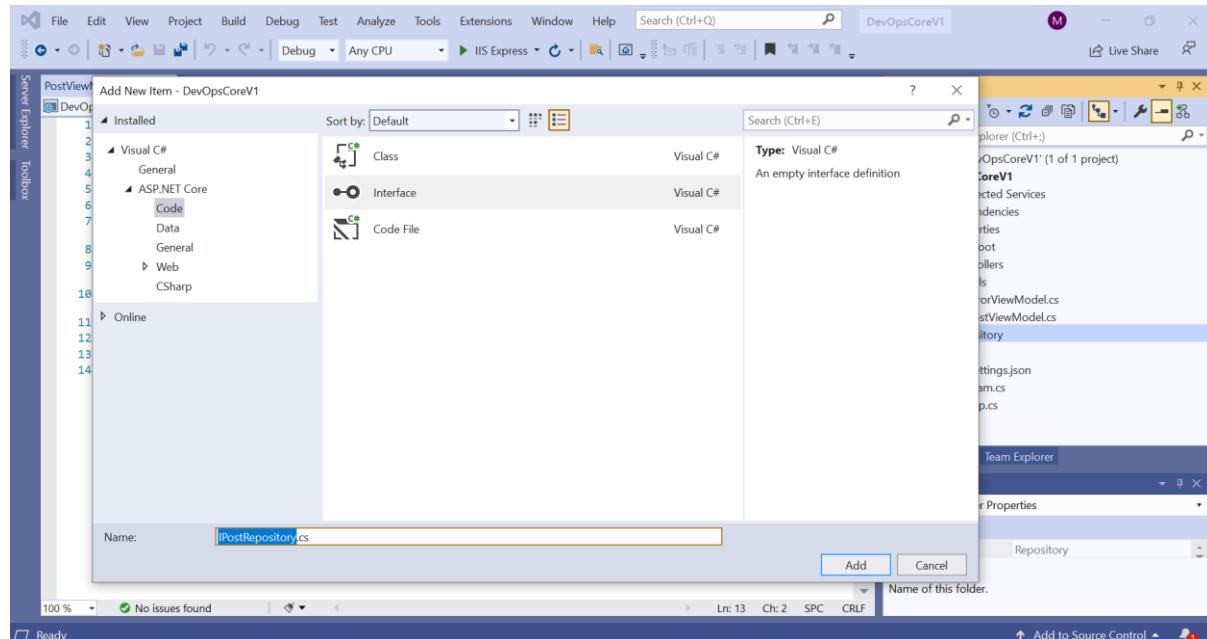
To create a new folder, right click on the folder structure and pick Add->New Folder. After the new window, you have to provide a folder name and I'm going to call it "Repository."



- 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.



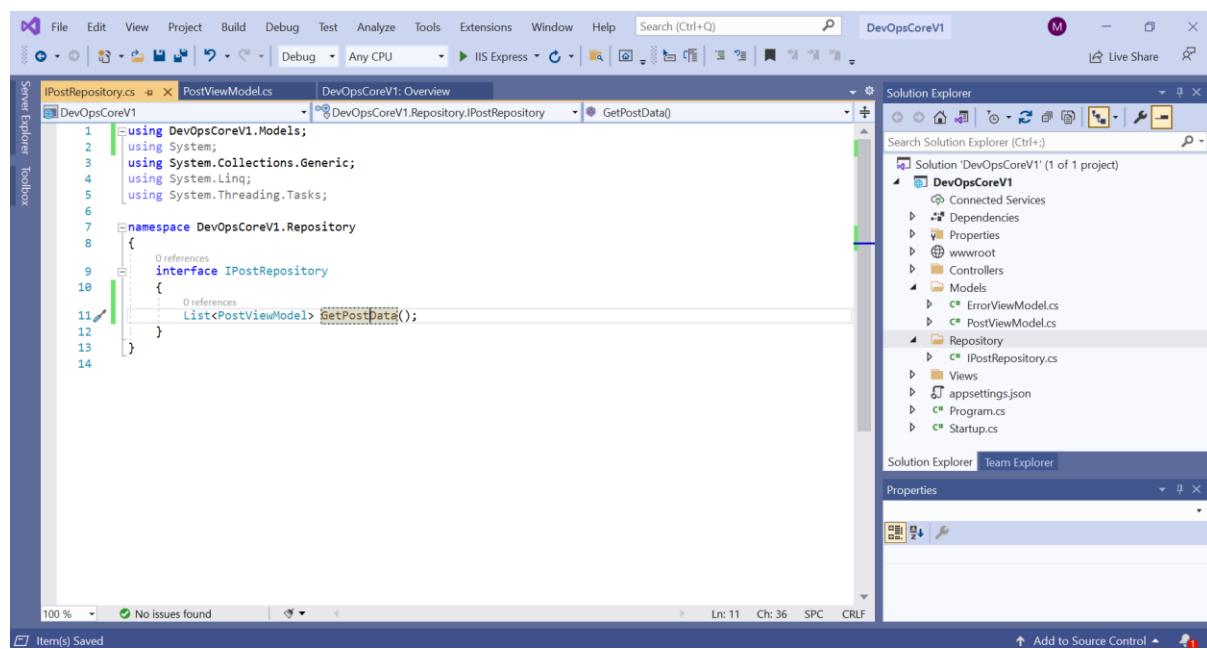
It is necessary to pick an interface and give it a name in the next box. "IPostRepository" is the name I've chosen for my interface.



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.

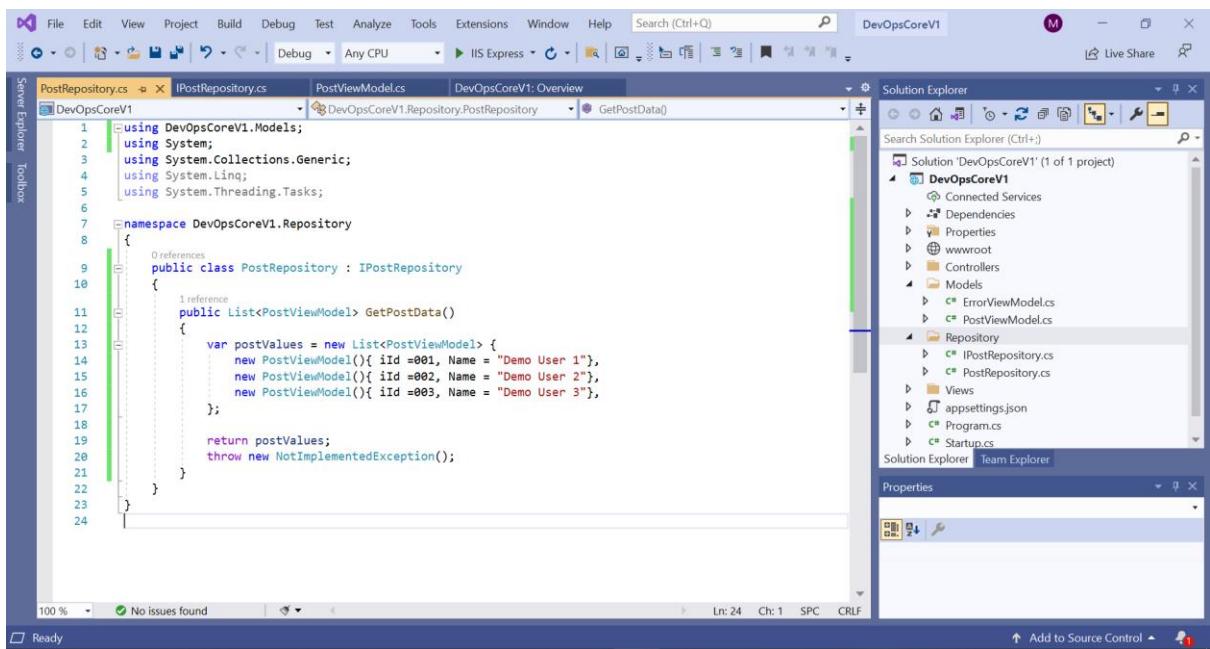
In order to do this, we must first build a new class in the repository folder named "PostRepository."

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(){ id =001, Name = "Demo User 1"},
                new PostViewModel(){ id =002, Name = "Demo User 2"},
                new PostViewModel(){ id =003, Name = "Demo User 3"},
            };

            return postValues;
        }
    }
}
```



The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The toolbar below has icons for file operations like Open, Save, and Print. The main window is divided into several panes: the left pane shows the Server Explorer and Toolbox; the center-left pane displays the code editor with PostRepository.cs selected; the center-right pane shows the Solution Explorer with the project structure; and the bottom right pane shows the Properties window. The code editor contains the following C# code:

```
1  using DevOpsCoreV1.Models;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6
7  namespace DevOpsCoreV1.Repository
8  {
9      public class PostRepository : IPostRepository
10     {
11         public List<PostViewModel> GetpostData()
12         {
13             var postValues = new List<PostViewModel>{
14                 new PostViewModel(){ Id = 001, Name = "Demo User 1" },
15                 new PostViewModel(){ Id = 002, Name = "Demo User 2" },
16                 new PostViewModel(){ Id = 003, Name = "Demo User 3" }
17             };
18
19             return postValues;
20             throw new NotImplementedException();
21         }
22     }
23 }
24
```

- 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;

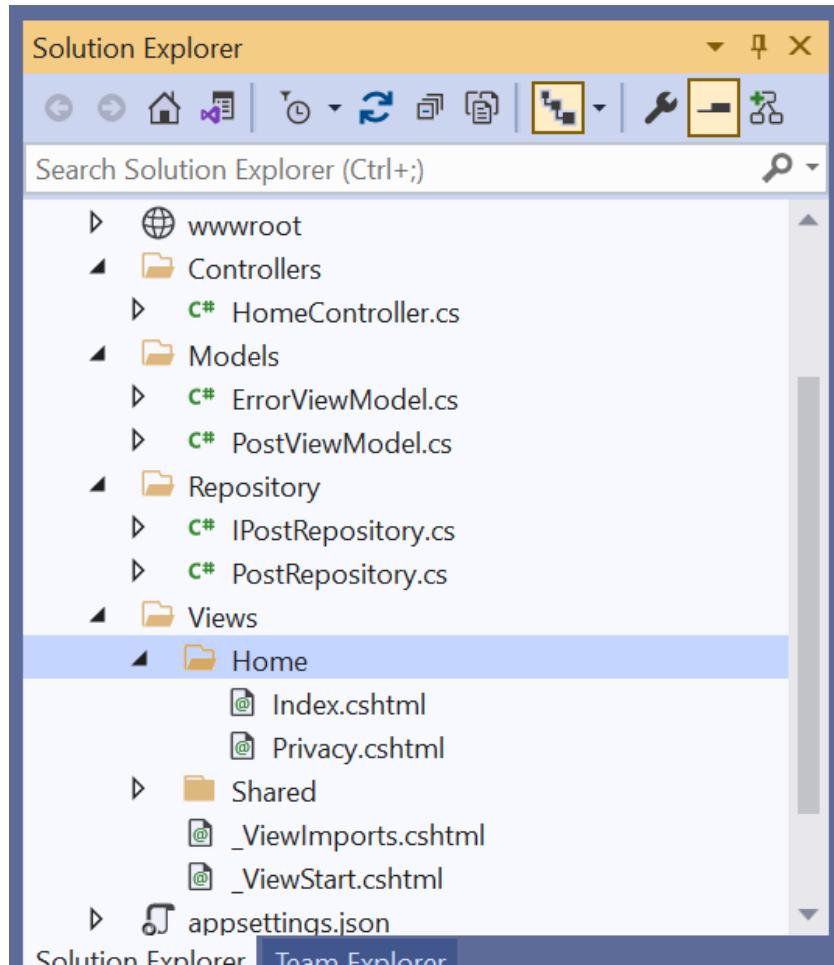
namespace DevOpsCoreV1.Controllers
{
    public class HomeController : Controller
    {
        IPostRepository postRepository;

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

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

The screenshot shows the Microsoft Visual Studio IDE interface. The left pane displays the code editor with `HomeController.cs` open. The code defines a `HomeController` class that implements the `Controller` interface. It injects `ILogger<HomeController>` and `IPostRepository` via constructor injection. The `Index()` action method retrieves data from the repository and returns it to the view. The right pane shows the **Solution Explorer** with the project structure:

- DevOpsCoreV1** (solution)
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - Controllers
 - HomeController.cs
 - Models
 - ErrorViewModel.cs
 - PostViewModel.cs
 - Repository
 - IPostRepository.cs
 - PostRepository.cs
 - Views
 - appsettings.json
 - Program.cs
 - Startup.cs



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.Id)</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.

```
Index.cshtml
1 @model IList<DevOpsCoreV1.Models.PostViewModel>
2 @ ViewData["Title"] = "Home Page";
3
4 <h2>Post List</h2>
5 <table class="table">
6   <thead>
7     <tr>
8       <th>Id</th>
9       <th>Name</th>
10    </tr>
11  </thead>
12  <tbody>
13    <foreach (var item in Model)
14    {
15      <tr>
16        <td>@Html.DisplayFor(modelItem => item.Id)</td>
17        <td>@Html.DisplayFor(modelItem => item.Name)</td>
18      </tr>
19    }
20  </tbody>
21 </table>
```

Your project has now been finished.

DevOpsCoreV1 Home Privacy

Post List

Id	Name
1	Demo User 1
2	Demo User 2
3	Demo User 3

© 2021 - DevOpsCoreV1 - [Privacy](#)

4.4 Add Unit Test Project

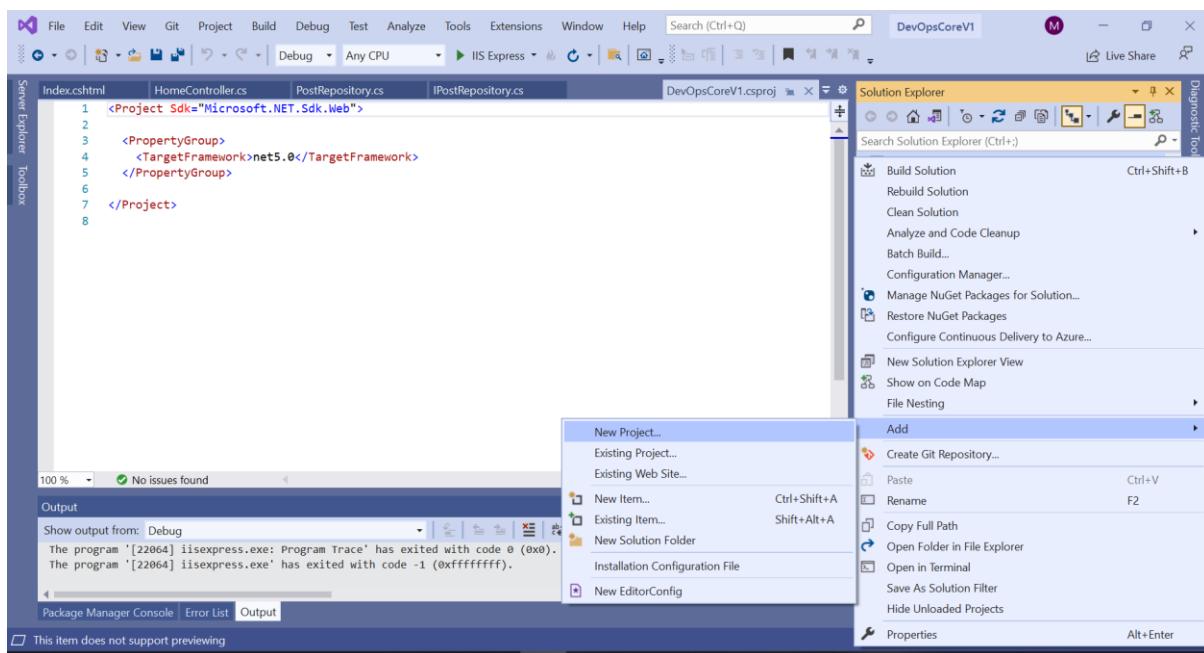
Unit testing is a critical component of the software development process. We can put our system into a production environment without having to test it beforehand.

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.

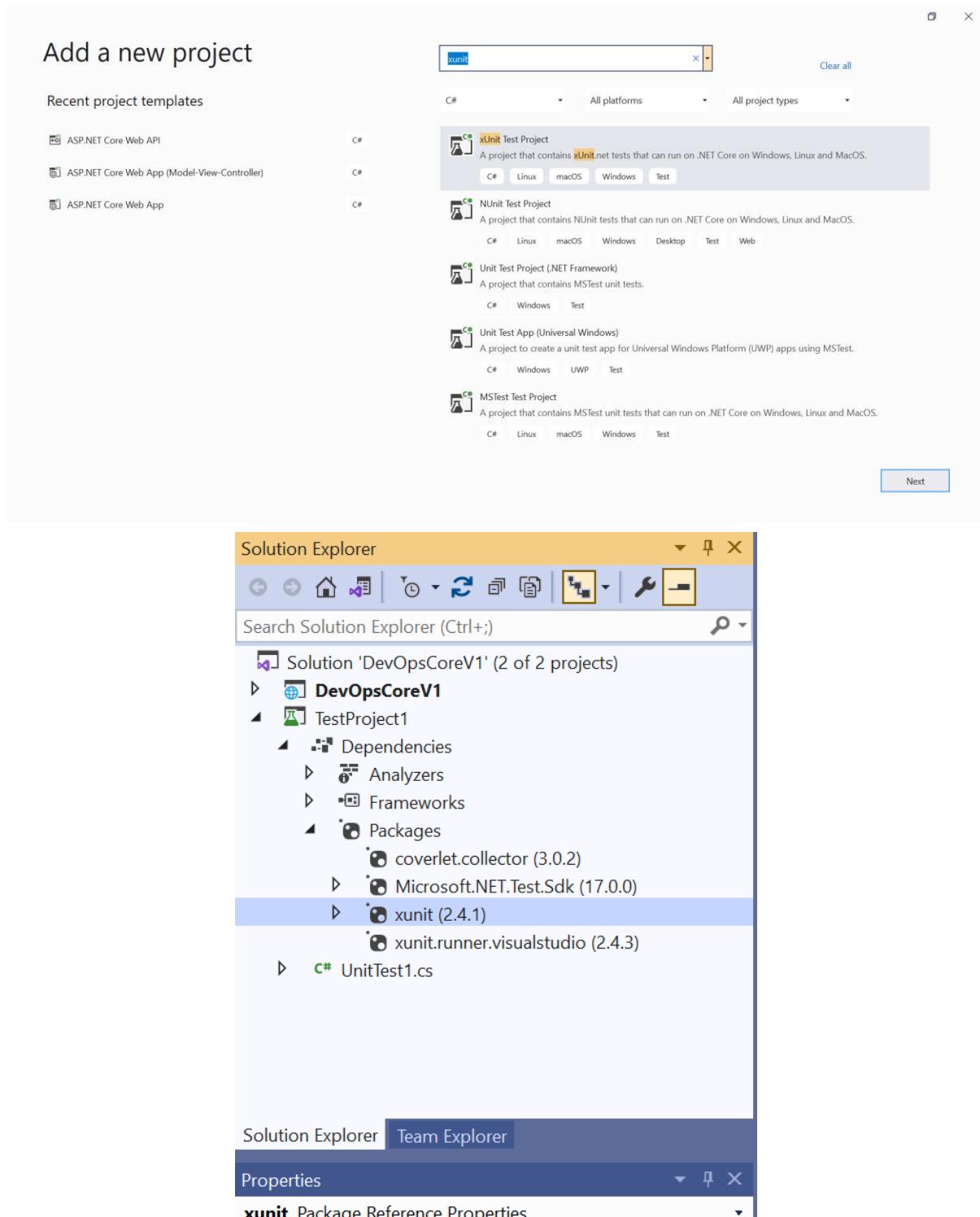
Unit tests allow us to determine whether or not our code is functioning properly. It also aids in the discovery of faults or defects in our code before it is released to the public.

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.

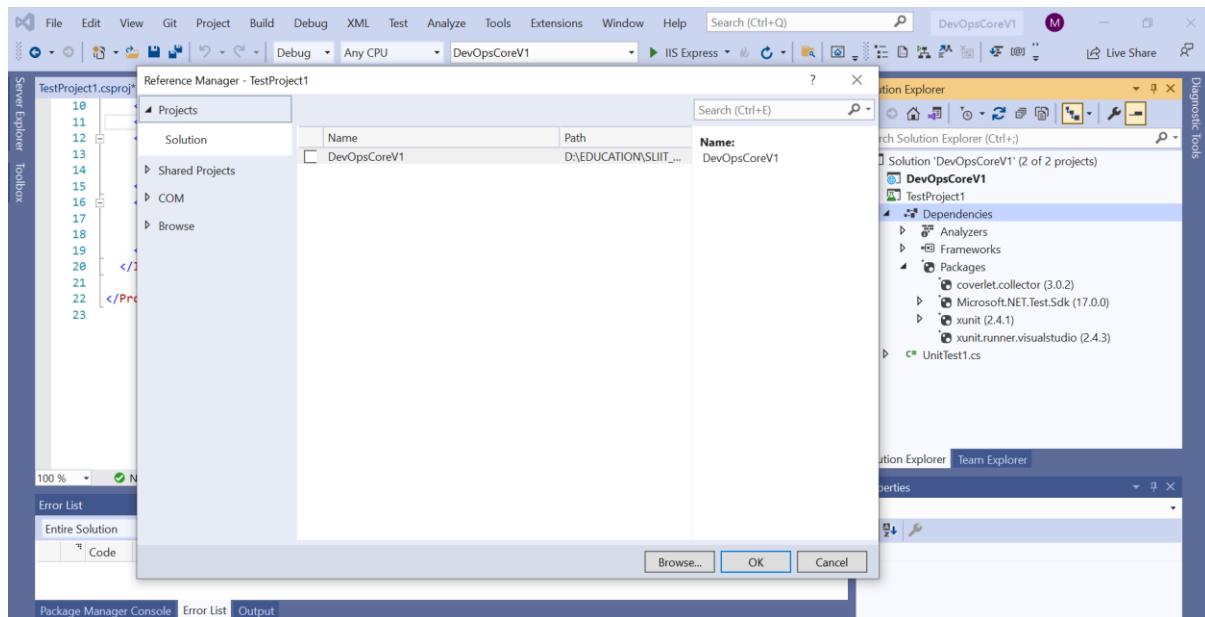
Right-click on your project solution and select "Add"->"New Project".



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);
}

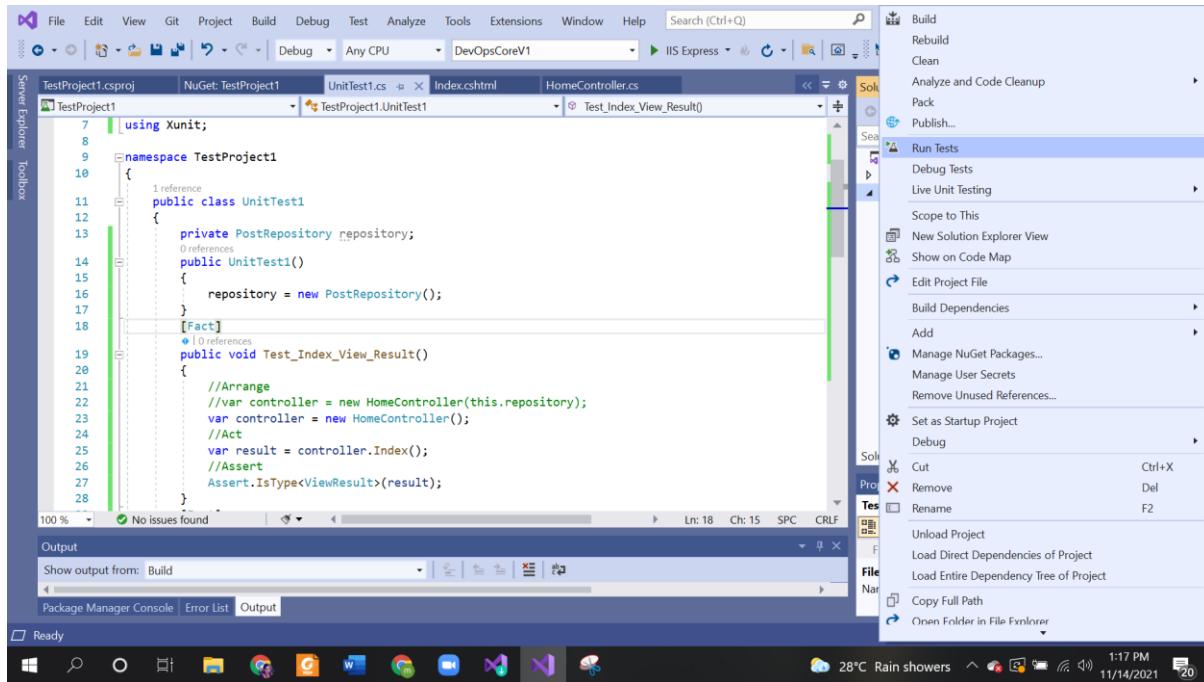
[Fact]
public void Test_Index_GetPosts_MatchData()
{
    //Arrange
    // var controller = new HomeController(this.repository);
    var controller = new HomeController();
    //Act
    var result = controller.Index();
    //Assert
    var viewResult = Assert.IsType<ViewResult>(result);
    var model =
        Assert.IsAssignableFrom<List<PostViewModel>>(viewResult.ViewData.Model);
    Assert.Equal(3, model.Count);
    Assert.Equal(101, model[0].id);
    Assert.Equal("DevOps Demo Title 1", model[0].Name);
}

}
}

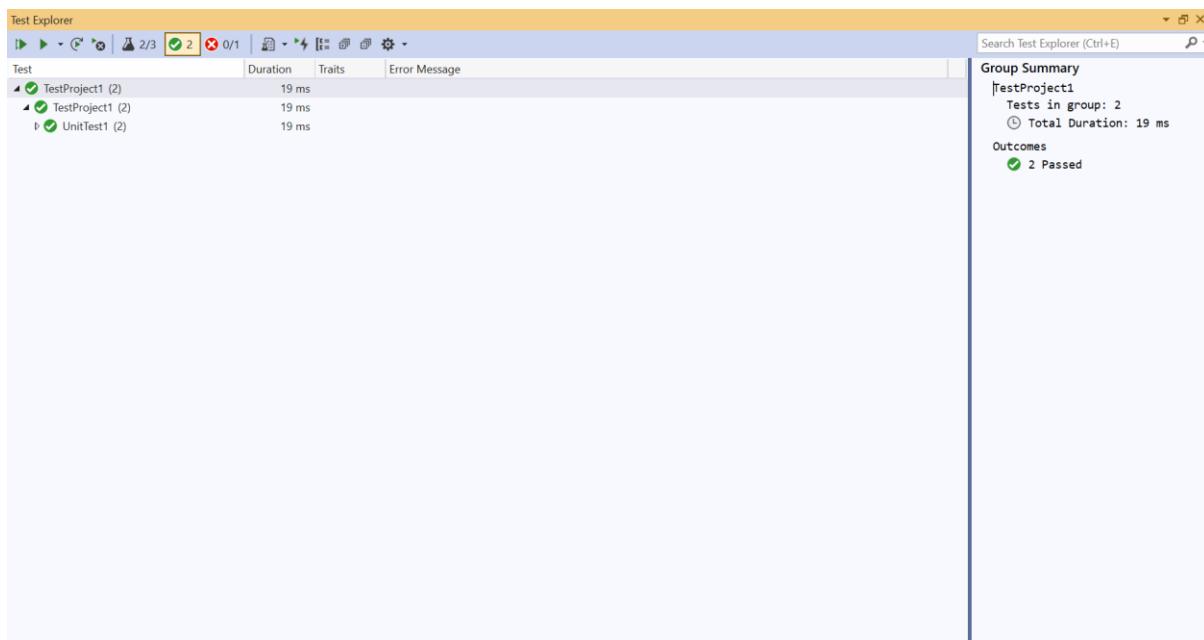
```

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

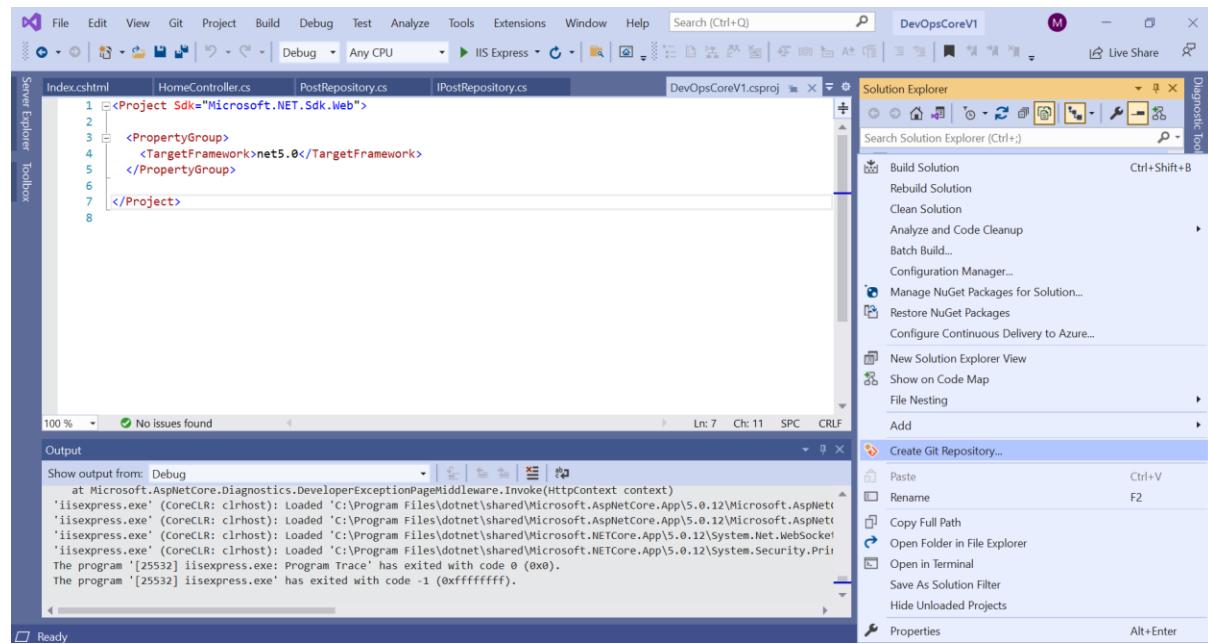
You may upload your project to any repository, such as Bitbucket or GitHub, and you can access it from Azure DevOps at any time. However, for the time being, I'll be using GitHub to upload my project.

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.



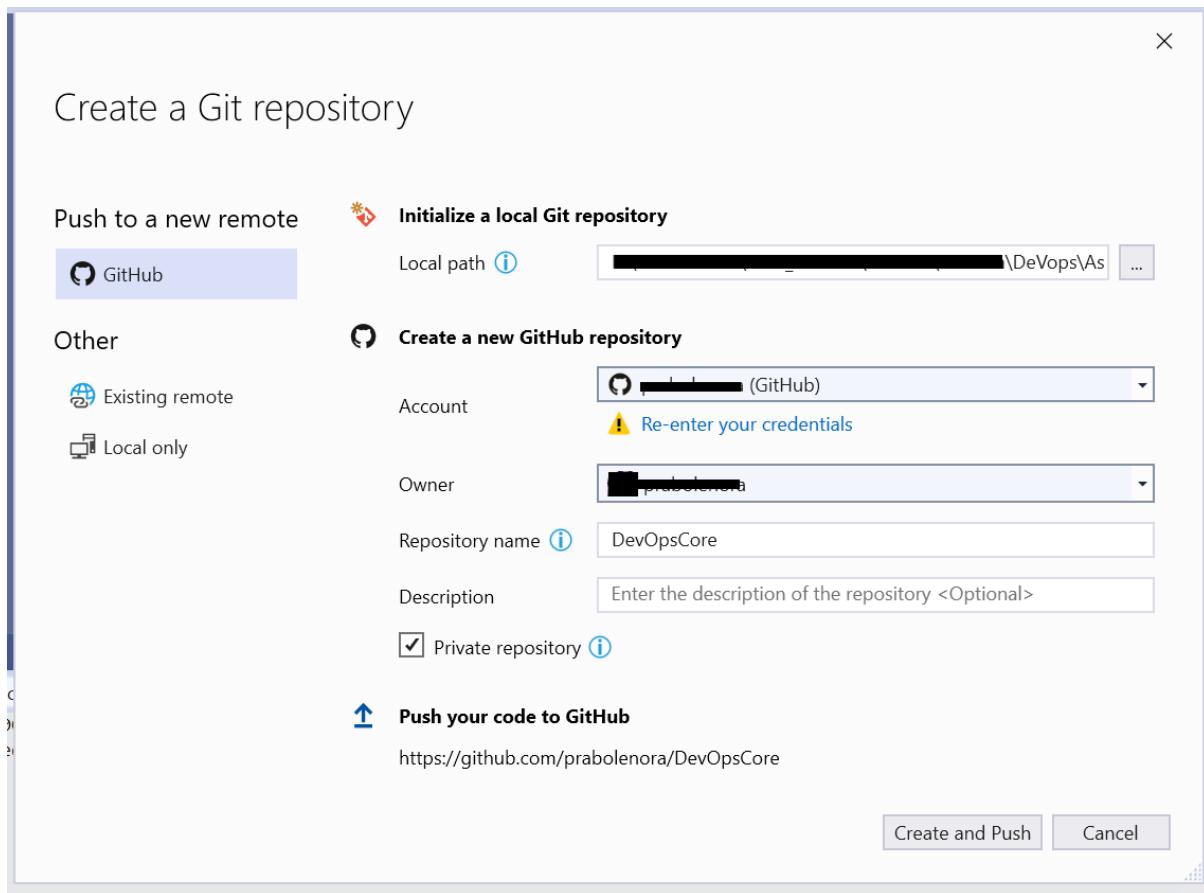
Choose "GitHub" from the drop-down menu in the next new window.

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.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Repository name *

 praboleora /

Great repository names are short and memorable. Need inspiration? How about [vigilant-octo-memory](#)?

Description (optional)

 Public Anyone on the internet can see this repository. You choose who can commit.

 Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file This is where you can write a long description for your project. [Learn more](#).

Add .gitignore Choose which files not to track from a list of templates. [Learn more](#).

Choose a license A license tells others what they can and can't do with your code. [Learn more](#).

Create repository

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.

Code Issues Pull requests Actions Projects Security Insights Settings

Quick setup — if you've done this kind of thing before

 Set up in Desktop or   [https://github.com/\[REDACTED\]/DevOpsCore.git](https://github.com/[REDACTED]/DevOpsCore.git) 

Get started by creating a new file or uploading an existing file. We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# DevOpsCore" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/[REDACTED]/DevOpsCore.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/[REDACTED]/DevOpsCore.git
```

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/>

Welcome to Azure!

Don't have a subscription? Check out the following options.

Start with an Azure free trial
Get \$200 free credit toward Azure products and services, plus 12 months of popular free services.
[Start](#) [Learn more](#)

Manage Azure Active Directory
Manage access, set smart policies, and enhance security with Azure Active Directory.
[View](#) [Learn more](#)

Access student benefits
Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status.
[Explore](#) [Learn more](#)

Azure services

Create a Education SQL databases All resources Quickstart Virtual App Services Storage Azure Cosmos More services

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.

+ Create a resource

Home Dashboard All services

FAVORITES

- All resources
- Resource groups
- App Services
- Function App
- SQL databases
- Azure Cosmos DB
- Virtual machines
- Load balancers
- Storage accounts
- Virtual networks
- Azure Active Directory

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

The screenshot shows the Microsoft Azure portal's 'Create a resource' interface. At the top, there's a search bar with the text 'DevOps Starter'. A red arrow points down to this search bar. Below the search bar, there's a 'Popular products' section with various service icons and names. The 'DevOps Starter' service is listed here.

Icon	Service Name	Actions
Windows Server icon	Windows Server 2019 Datacenter	Create Learn more
Ubuntu Server icon	Ubuntu Server 20.04 LTS	Create Learn more
Web App icon	Web App	Create Docs MS Learn
SQL Database icon	SQL Database	Create Docs MS Learn
Function App icon	Function App	Create Docs
Azure Cosmos DB icon	Azure Cosmos DB	Create Docs MS Learn

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

The screenshot shows the 'DevOps Starter' product page. At the top, there's a 'Create' button highlighted with a red box and a red arrow. Below the button, there's a brief description of what DevOps Starter does and some user reviews.

DevOps Starter [Add to Favorites](#)
Microsoft
★ 4.3 (111 Azure ratings)

[Create](#)

Overview Plans Usage Information + Support Reviews

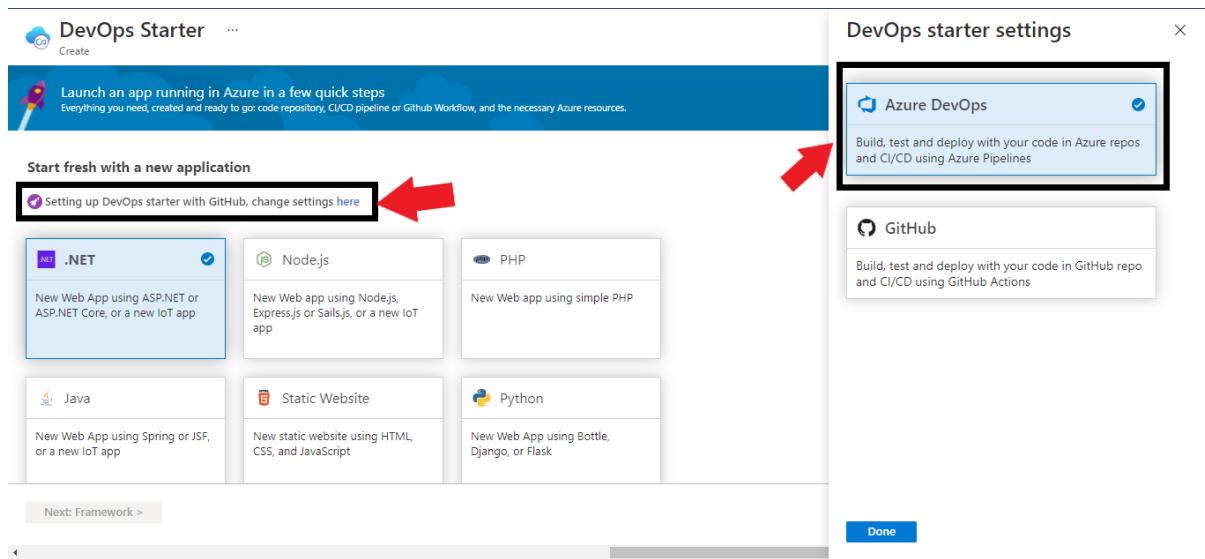
Launch an app running in Azure in a few quick steps

DevOps Starter makes it easy to get started on Azure. It helps you launch an app on the Azure service of your choice in a few quick steps. DevOps Starter set you up with everything you need for developing, deploying and monitoring your app.

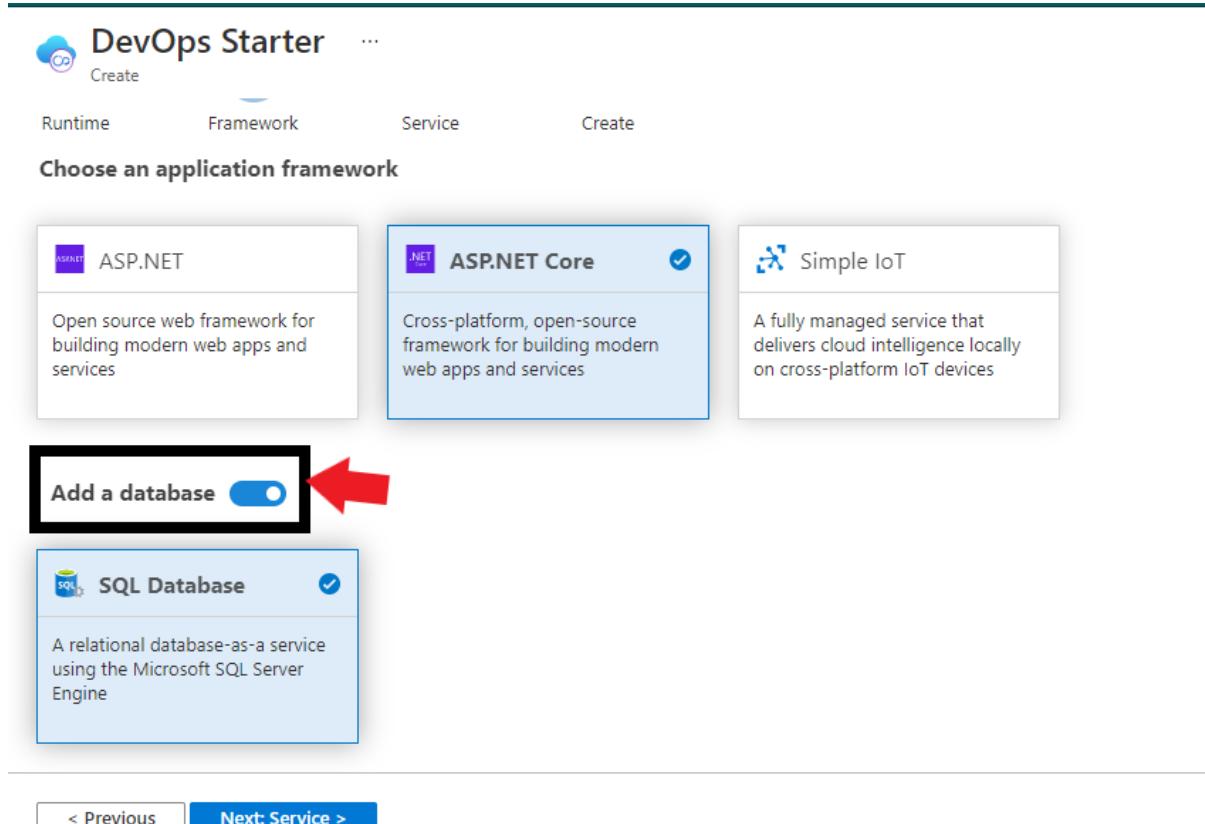
Creating a project with DevOps Starter provisions Azure resources and comes with a Git code repository, Application Insights integration and a continuous delivery pipeline setup to deploy to Azure. The DevOps Starter dashboard lets you monitor code commits, builds, and deployments from a single view in the Azure portal.

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.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a blue header bar with the Microsoft Azure logo, an 'Upgrade' button, a search bar, and several icons. Below the header, the URL 'Home > Create a resource > DevOps Starter >' is visible. The main content area is titled 'DevOps Starter' with a 'Create' button. A sub-section titled 'Select an Azure service to deploy the application' contains four options:

- Kubernetes Service: Fully managed Kubernetes container orchestration service for managing containers without container expertise.
- Windows Web App**: Fully managed compute platform on Windows for web applications and websites. This option is highlighted with a blue border and a checkmark icon.
- Linux Web App: Fully managed compute platform on Linux for web applications and websites.
- Web App for Containers: Fully managed compute platform on Linux for deploying and running containerized web applications.

Below these options, a note says 'Don't see a service you're looking for? We're continuously adding support for more Azure services and app frameworks. [Learn more](#)'. At the bottom, there are navigation buttons: '< Previous' and 'Next: Create >'. A progress bar at the bottom indicates the user is on step 4 of 4.

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

The screenshot shows the 'Create' step of the DevOps Starter wizard. On the left, a progress bar shows steps: Runtime (checkmark), Framework (checkmark), Service (checkmark), and Create (step 4). The main area displays the following configuration:

- Project name ***: DemoCICDV2
- Azure DevOps Organization ***: DemoV1
- Subscription**: Azure subscription 1
- Web app name ***: DemoCICDV2.azurewebsites.net
- Location**: South Central US
- Pricing tier**: S1 Standard (1 Core, 1.75 GB RAM)

Below these fields, a note says 'Ready to deploy ASP.NET Core app to Azure Windows Web App, with SQL Database.' and 'By continuing, you agree to the [Terms of Service](#) and the [Privacy Statement](#). The new app code is published under the MIT license.'

On the right, an 'Additional settings' panel is open, containing the following fields:

- Resource group**: DemoCICDV2-rg
- Pricing tier**: S1 Standard (1 Core, 1.75 GB RAM)
- Application Insights Location**: South Central US
- Database Server Login Details**
 - Server name**: democicdv2-server.database.windows.net
 - Enter username**: dbadmin
 - Location**: Central US
 - Database Name**: DemoCICDV2-db

At the bottom of the 'Additional settings' panel, there are 'OK' and 'Cancel' buttons.

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.

The screenshot shows the Microsoft Azure Deployment Overview page for a deployment named 'Deploy_DevOps_Project_DemoAppV1'. The main message is 'Your deployment is complete'. It provides deployment details: name, subscription, resource group, start time, and correlation ID. Below this, there are sections for 'Deployment details' and 'Next steps'. A prominent blue button labeled 'Go to resource' is highlighted with a red arrow pointing to it. To the right, there's a sidebar with links to Security Center, Free Microsoft Start learning!, and Work with an Azure expert.

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

The screenshot shows the Azure DevOps Starter dashboard for a project named 'DemoCICDV2'. On the left, the 'CI/CD pipeline' section displays a flow from 'Code' (GitHub commit) through 'Build' (DemoCICDV2 - CI) to 'dev' (DemoCICDV2 - CD). The 'Build' step shows a successful run. On the right, the 'Azure resources' section lists an Application endpoint (https://democicdv2.azurewebsites.net), an App Service (DemoCICDV2, Running), a SQL Database (DemoCICDV2-db, Online), and an Application Insights instance (DemoCICDV2).

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 screenshot shows the Azure DevOps interface for the project 'DemoCICDV2'. On the left, the CI/CD pipeline is displayed with three stages: Code, Build, and dev. A red arrow points from the 'Code' stage to the 'master' branch under the 'Code' section. The 'Build' stage shows a successful build (Build 20210928.1) and a release (Release-1). The 'dev' stage shows a successful release (Release-1). On the right, the 'Azure resources' section lists an Application endpoint at <https://democicdv2.azurewebsites.net>, an App Service named 'DemoCICDV2' which is Running, and an SQL Database named 'DemoCICDV2-db' which is Online. There is also a 'Release-1' entry.

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

The screenshot shows the Azure DevOps repository 'DemoCICDV2'. The left sidebar shows navigation options like Overview, Boards, Repos, Files, Commits, Pushes, Branches, Tags, Pull requests, Pipelines, Test Plans, and Project settings. The 'Files' option is selected. The main area displays the file structure of the 'aspnet-core-dotnet-core' folder, which contains 'Application' and 'ArmTemplates' subfolders. The 'Contents' tab is selected, showing a list of files with their names, last change dates, and commit details. The commits all point to the commit ID [499f0798](#) and the message 'First commit azureddevopsproj...'. The files listed include Pages, wwwroot, AppConfig.cs, appsettings.Development.json, appsettings.json, aspnet-core-dotnet-core.csproj, bundleconfig.json, Program.cs, and Startup.cs.

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

The screenshot shows the Azure DevOps CI/CD pipeline page for the project 'DemoCICDV2'. On the left, the 'CI/CD pipeline' section displays a flow from 'Code' (GitHub) through 'Build' (DemoCICDV2 - CI) and 'dev' (DemoCICDV2 - CD) to 'Repository'. The 'Build' step shows a successful build named 'Build 20210928.1' from 16 min ago. The 'dev' step shows a successful release named 'Release-1' from just now. On the right, the 'Azure resources' section lists an 'Application endpoint' at <https://democicd2.azurewebsites.net>, which is highlighted with a red arrow. Below it, there are entries for 'App Service' (DemoCICDV2, Running) and 'SQL Database' (DemoCICDV2-db, Online). A 'Release-1' entry is also shown under 'Azure resources'. At the bottom, there is a 'Application Insights' section.

This is our project that has been deployed.

The screenshot shows the deployed ASP.NET Core application's homepage. The top navigation bar includes 'ASP.NET Core' and links for 'Home', 'About', and 'Contact'. A message box at the top says 'Use this space to summarize your privacy and cookie use policy. [Learn More.](#)' with an 'Accept' button. The main content area features a large blue cloud icon with a checkmark and the word 'Success!'. Below it, a Twitter icon and the text 'Azure DevOps Project has been successfully setup Your ASP.NET Core app is up and running on Azure'. At the bottom, there is a footer with icons for GitHub, LinkedIn, and YouTube, followed by the text 'Powered by Azure SQL Database', 'Total Visits: 2 Last Visit: 09/28/2021 17:41:04', and navigation links for 'Get started right away', 'Continuous Delivery', and 'Azure DevOps Project'.

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.

The screenshot shows the Azure DevOps CI/CD pipeline dashboard for the project 'DemoCICDV2'. At the top, there's a navigation bar with links for Refresh, Project homepage, Repositories, Build pipelines (which is highlighted with a red box and a red arrow pointing to it), Release pipelines, Agile backlog, Users & groups, and Delete. The main area is divided into two sections: 'CI/CD pipeline' on the left and 'Azure resources' on the right. The 'CI/CD pipeline' section shows a flow from 'Code' (with a commit history) through 'Build' (with a successful build step) to 'dev' (with a successful release step). The 'Azure resources' section lists an Application endpoint (https://democicdv2.azurewebsites.net), an App Service named 'DemoCICDV2' (status: Running), and a SQL Database named 'DemoCICDV2-db' (status: Online). There's also a 'Release-1' entry under the Azure resources. The bottom section is labeled 'Application Insights'.

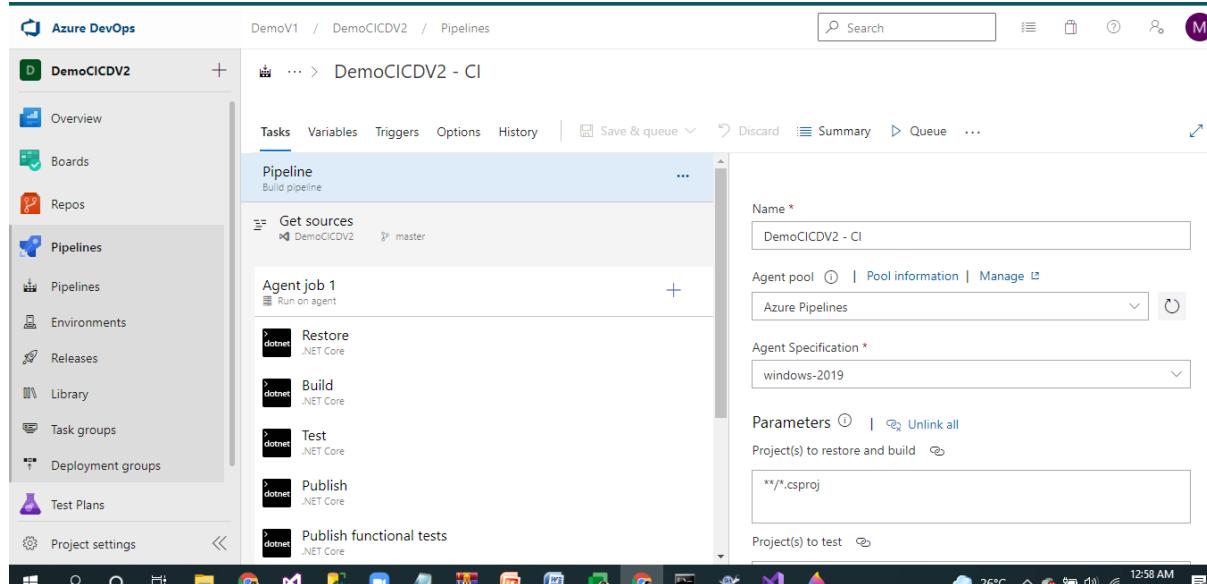
2) Choose a project to work on.

The screenshot shows the Azure DevOps Pipelines interface. On the left, a sidebar menu is open with several options: Overview, Boards, Repos, Pipelines (selected), Pipelines (disabled), Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Project settings. The main area is titled 'Pipelines' and shows a section for 'Recently run pipelines'. A single pipeline run is listed: 'DemoCICDV2 - CI' (Last run: #20210928.1 • First commit, Other build reason: master, Yesterday, 1m 47s). A red arrow points upwards from the pipeline run to the pipeline name. At the top right of the main area, there is a 'New pipeline' button and a 'Filter pipelines' search bar.

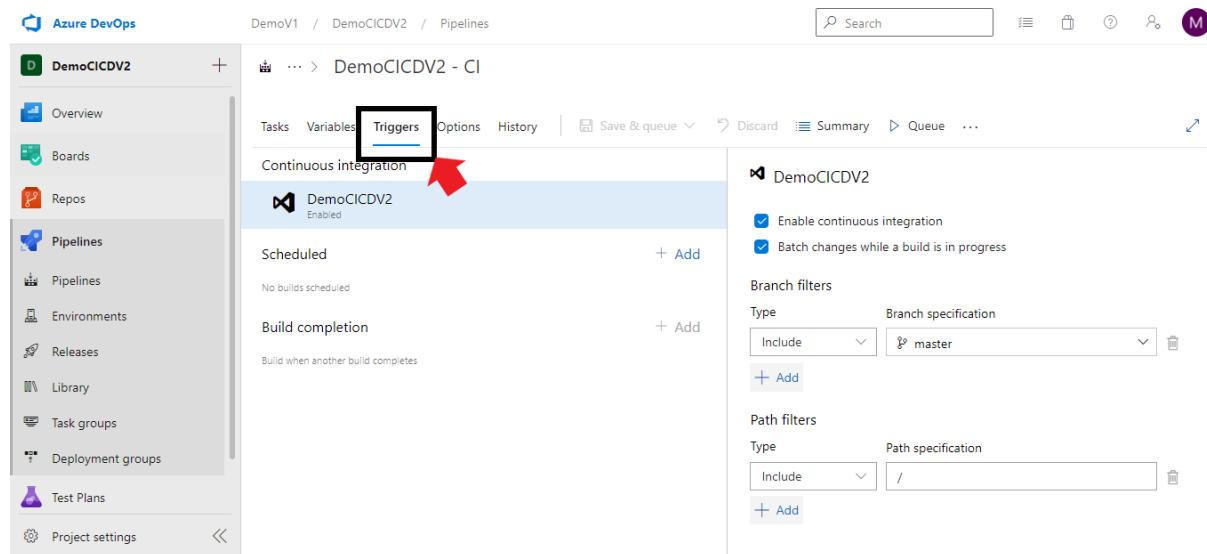
3) Hit on “Edit” button

The screenshot shows the Azure DevOps Pipeline details page for 'DemoCICDV2 - CI'. The left sidebar is identical to the previous screenshot. The main area shows the pipeline run details: Description: '#20210928.1 First commit', Stages: 'master' (green checkmark), Last run: #20210928.1, Other build reason: master, 499f0798, Yesterday, 1m 47s. At the top right, there are three buttons: 'Edit' (highlighted with a red box and arrow), 'Run pipeline', and a more options menu. The 'Edit' button is the target of the red arrow.

- 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.

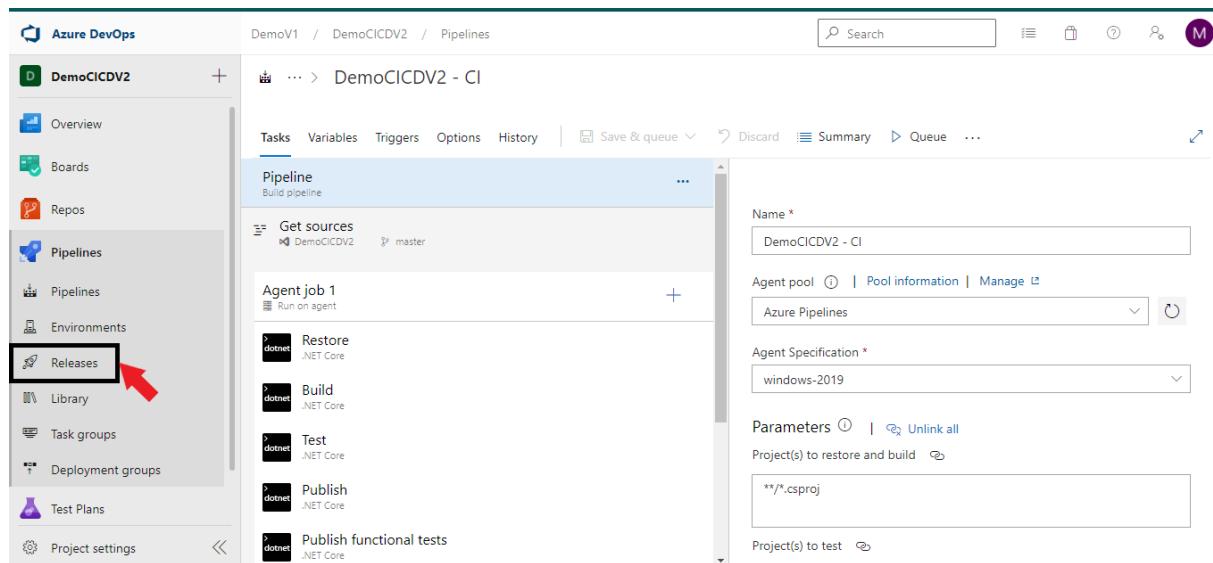


- 5) Select the "Trigger" tab from the drop-down menu. When we start an Azure DevOps project, the CI trigger will be built for us automatically.



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

6) Select the release option

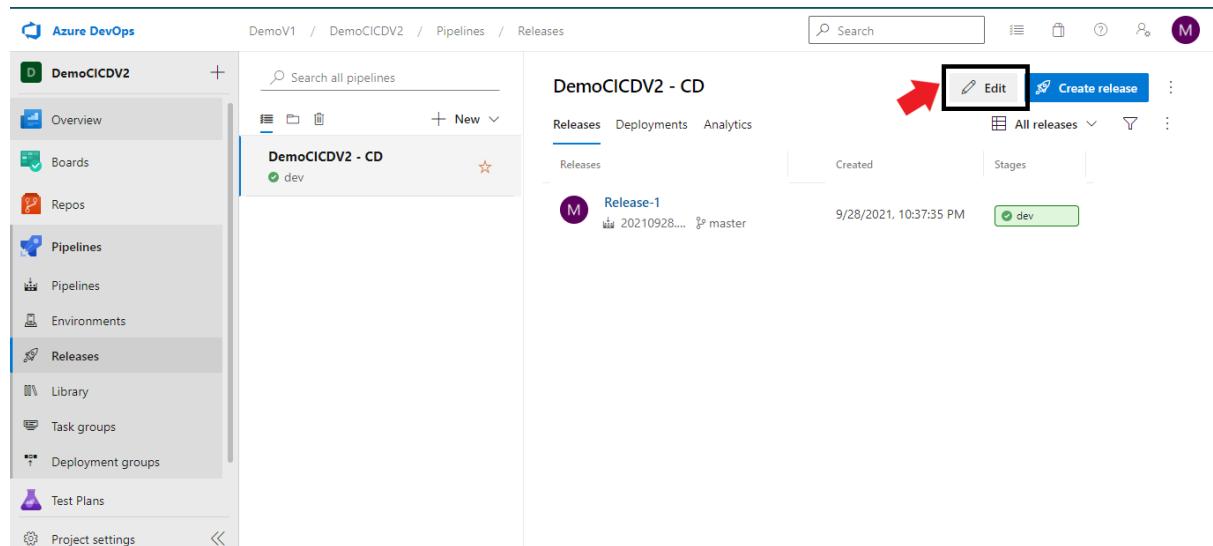


The screenshot shows the Azure DevOps Pipelines interface. On the left sidebar, under the 'Pipelines' section, the 'Releases' option is highlighted with a red arrow pointing to it. The main area displays a 'Pipeline' build pipeline with several tasks: 'Get sources', 'Agent job 1' (which includes 'Restore', 'Build', 'Test', 'Publish', and 'Publish functional tests'), and 'Publish functional tests'. To the right, there is a configuration panel for a new release definition:

- Name:** DemoCICDV2 - CI
- Agent pool:** Azure Pipelines
- Agent Specification:** windows-2019
- Parameters:** **/*.csproj

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

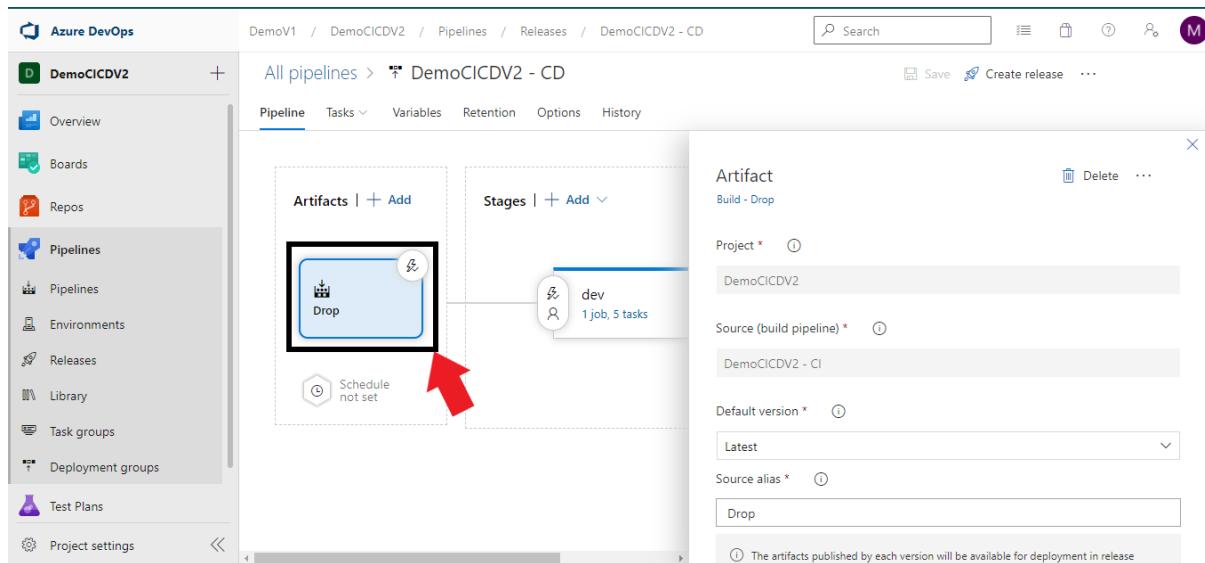
7) In the Releases tab, click on the "Edit" button to the right.



The screenshot shows the Azure DevOps Releases interface. On the left sidebar, the 'Releases' option is selected. The main area shows a release named 'DemoCICDV2 - CD' with a status of 'dev'. At the top right, there are two buttons: 'Edit' (highlighted with a red arrow) and 'Create release'. Below the release name, there are tabs for 'Releases', 'Deployments', and 'Analytics'. A single release item is listed:

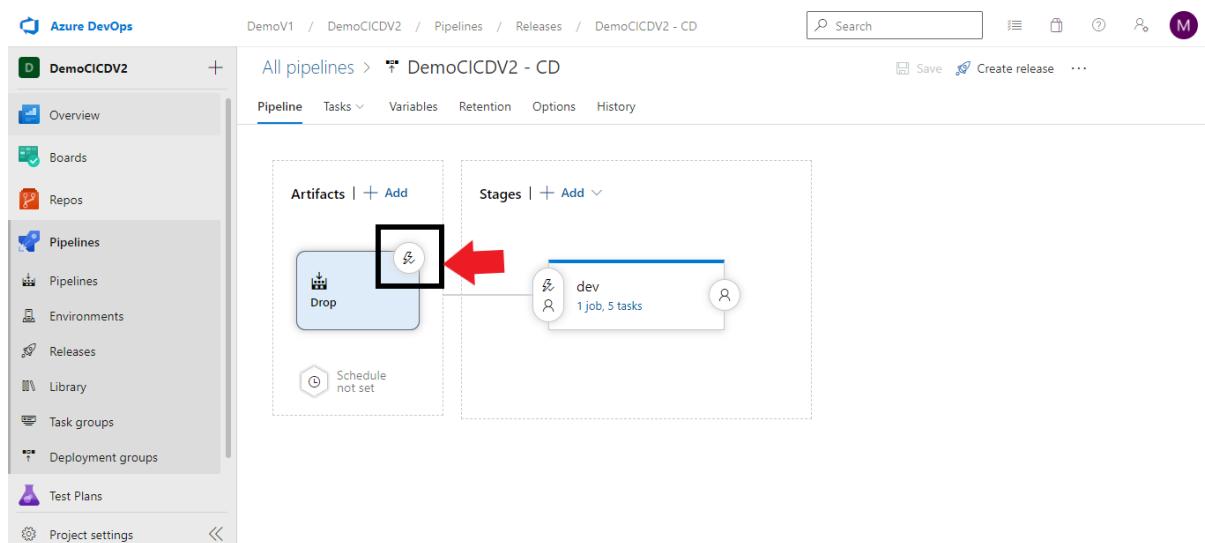
- Release-1**: Created 9/28/2021, 10:37:35 PM, Stages dev

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

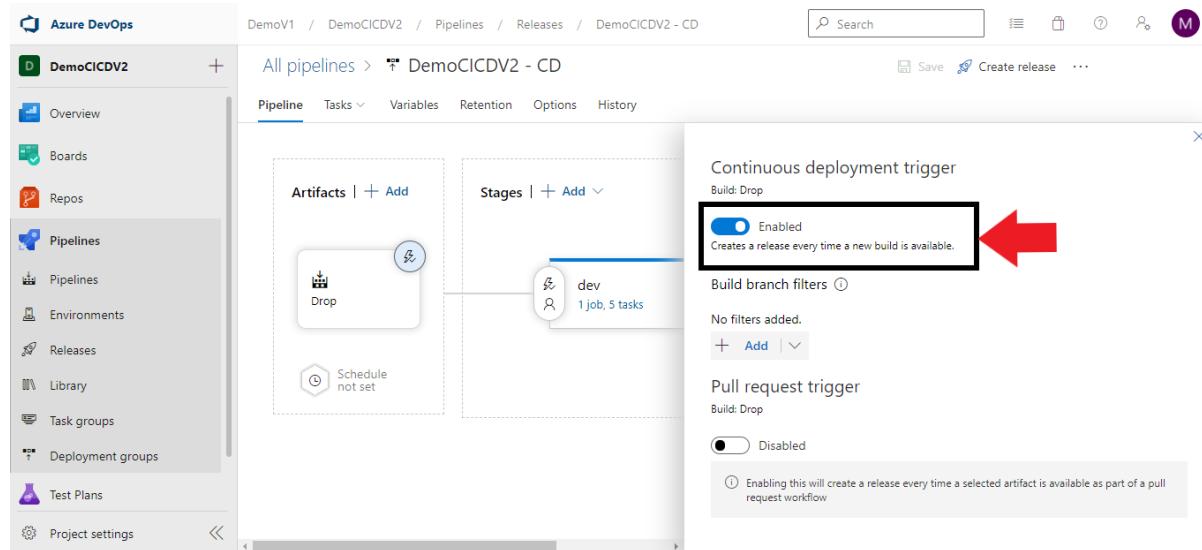


In this case, the output from the preceding step of constructing the pipeline will be shown.

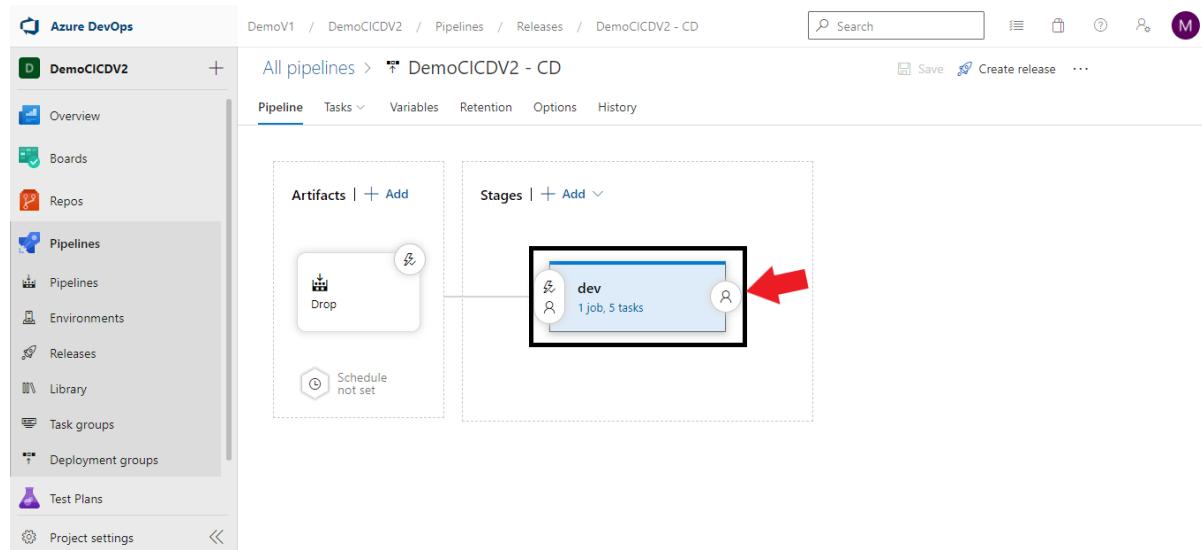
9) To activate the continuous deployment trigger, find it at the top of the "drop" and click on it.



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.

The screenshot shows the Azure DevOps interface for a pipeline named "DemoCICDV2 - CD". The left sidebar is open, showing options like Overview, Boards, Repos, Pipelines (which is selected), Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Project settings. The main area displays the "Tasks" tab for the "dev" stage. The stage name is "dev". The tasks listed are:

- Run on agent
- Azure Deployment: Create Azure Resources
- Deploy Azure App Service
- Execute Azure SQL
- Visual Studio Test Platform Installer
- Test Assemblies

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”

The screenshot shows the Azure DevOps interface for a project named 'DemoV1' under 'DemoCICDV2'. The left sidebar is open with the 'Pipelines' section selected. The main area displays a pipeline named 'DemoCICDV2 - CD'. This pipeline consists of a single stage called 'dev', which contains one job and five tasks. In the top right corner of the pipeline details, there is a dropdown menu with options: 'Save', 'Create release', '...', 'View releases' (which is highlighted with a red box and a red arrow), 'Help', and 'Security'.

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

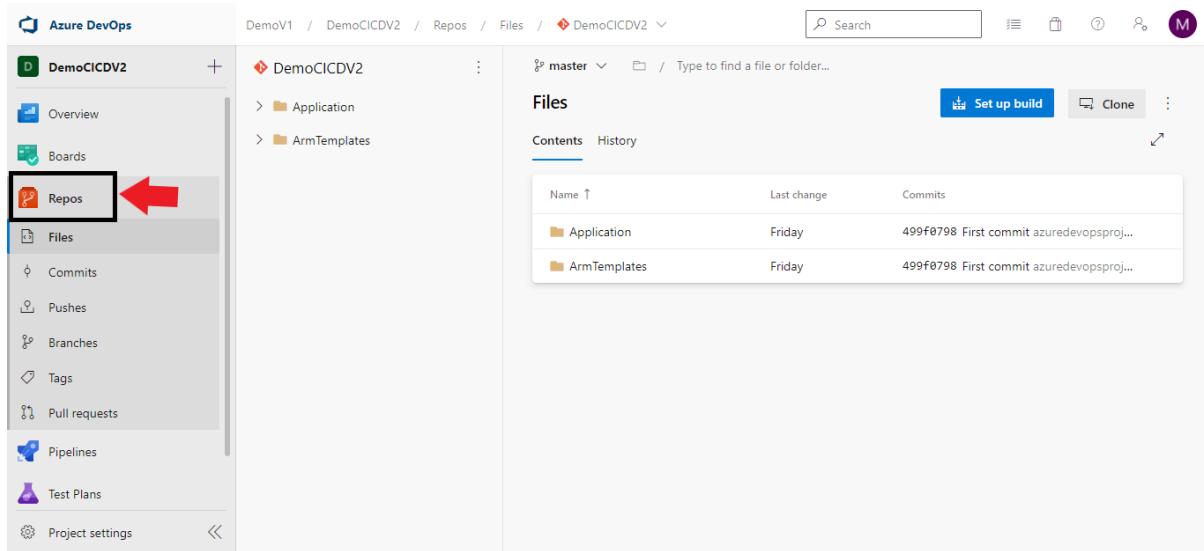
The screenshot shows the Azure DevOps interface for the 'Releases' section of the 'DemoCICDV2 - CD' pipeline. The left sidebar is open with the 'Releases' section selected. The main area shows a single release named 'Release-1' created on 9/28/2021 at 10:37:35 PM. Below the release, there are three tabs: 'Releases' (highlighted with a red box and an upward arrow), 'Deployment' (highlighted with a red box), and 'Analytics' (highlighted with a red box). There is also a 'Edit' button and a 'Create release' button.

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.

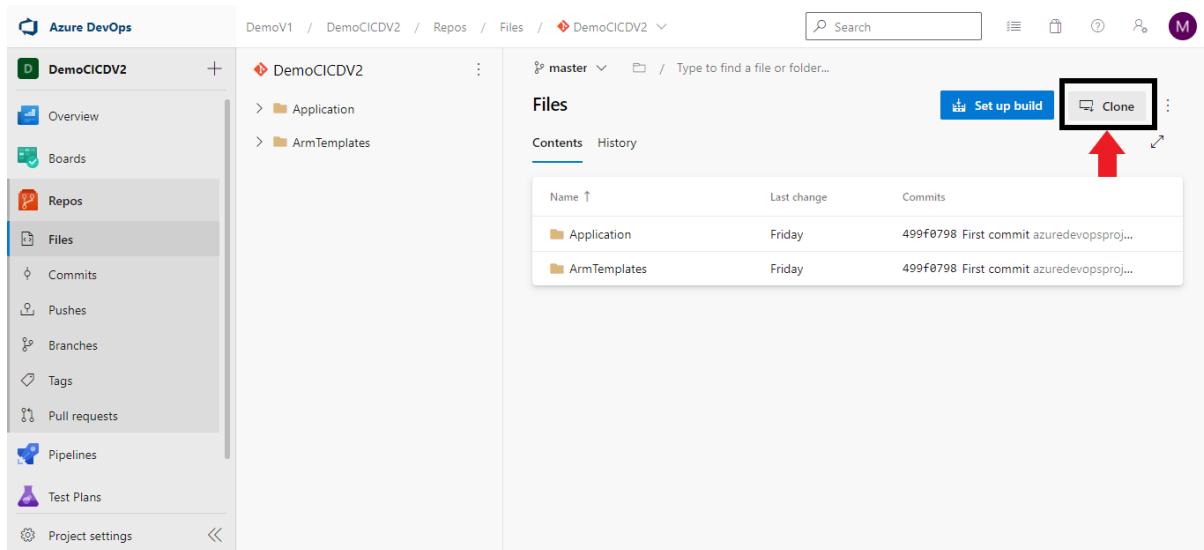
- 1) Click on the “Repos”



The screenshot shows the Azure DevOps interface for a project named 'DemoCICDV2'. The left sidebar has a 'Repos' icon highlighted with a red arrow. The main area shows a 'Files' view for the 'master' branch. It lists two folders: 'Application' and 'ArmTemplates'. The 'Application' folder has a commit history entry: '499f0798 First commit azureddevopsproj...'. The 'Clone' button is visible at the top right of the file list.

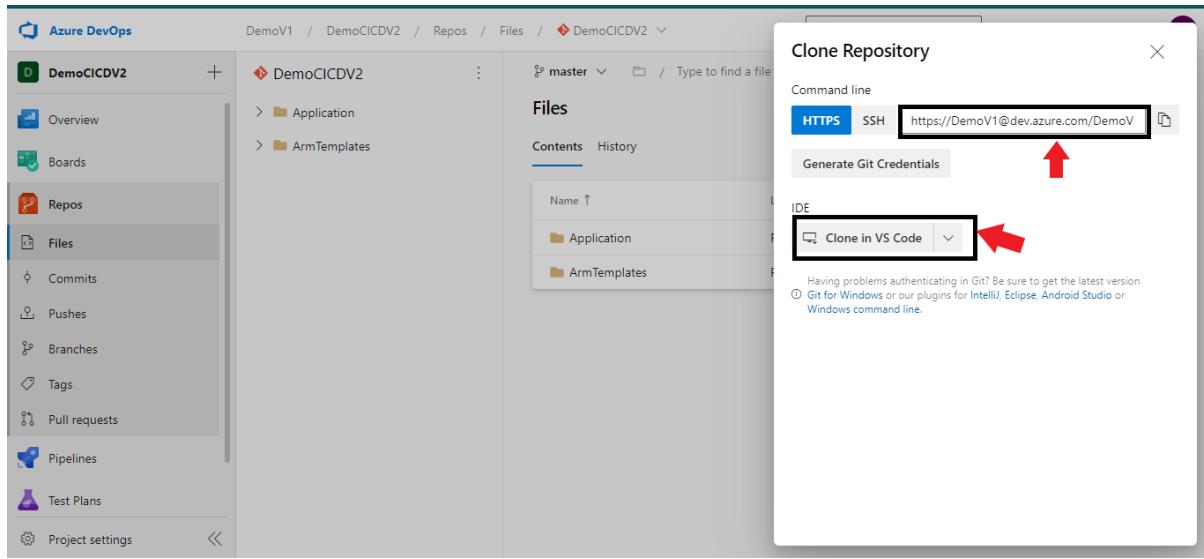
This command will show the Git repositories for the Azure DevOps projects.

- 2) Click on the “Clone” button

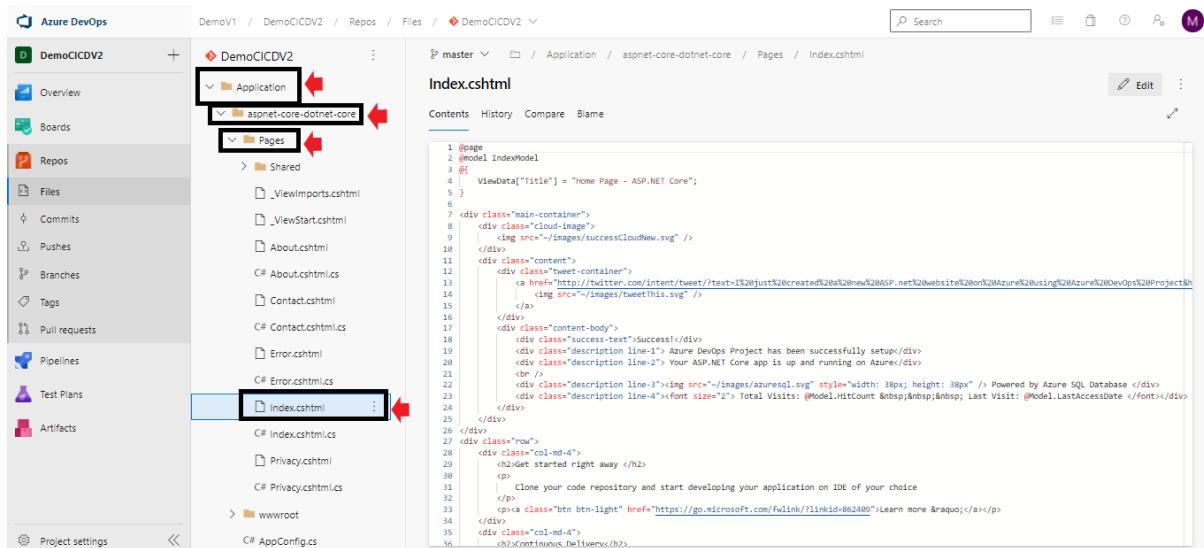


The screenshot shows the same Azure DevOps interface as the previous one, but with a red arrow pointing to the 'Clone' button in the top right corner of the file list area. This indicates the step to clone the repository.

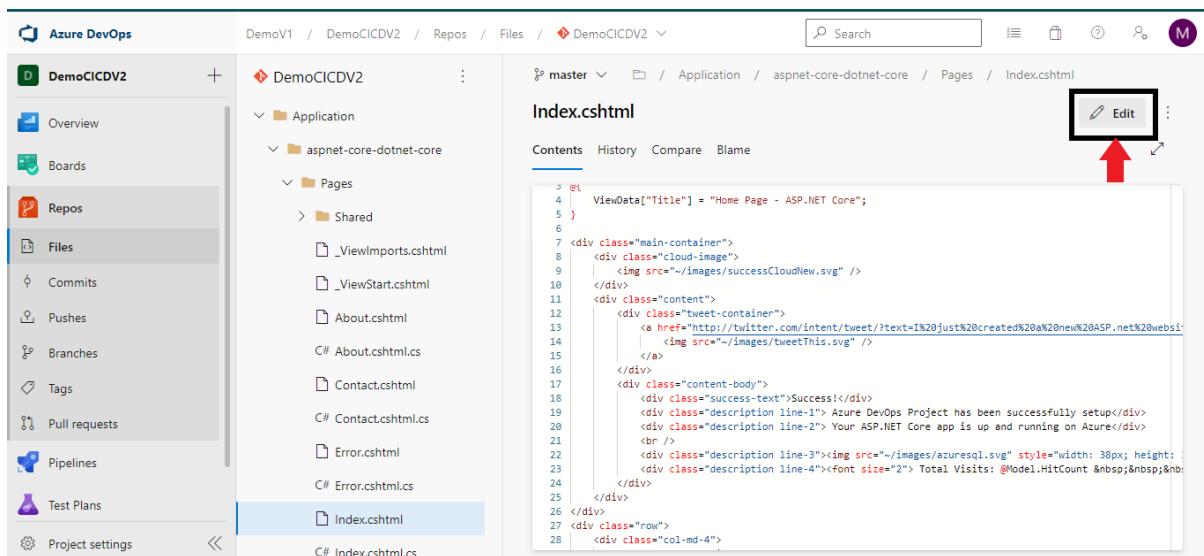
- 3) It is possible to get your git URL from the window below, and you can also pick an IDE and access your project from the IDE of your choice.



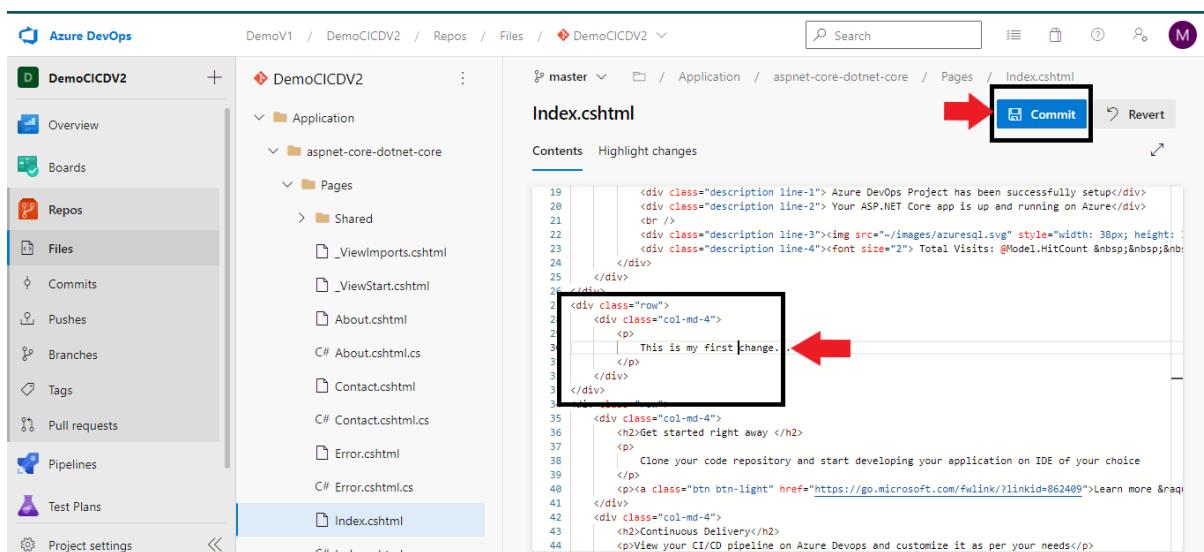
- 4) Now open the Index file like below



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.



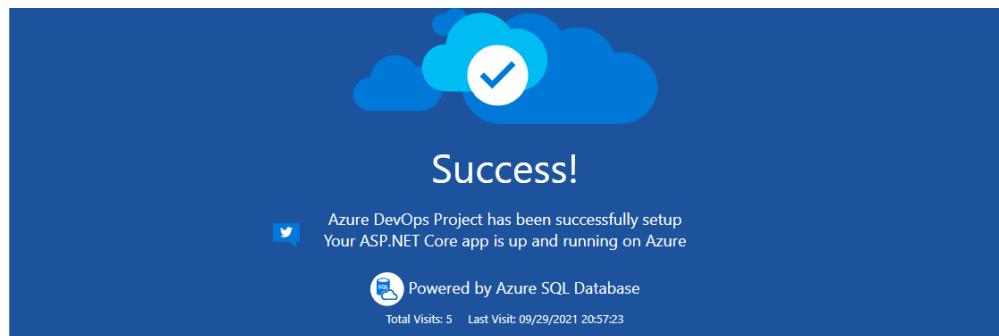
- 7) Once you have committed, go to the Pipeline and double-check your work.

The screenshot shows the Azure DevOps Pipelines interface. On the left, a sidebar menu has 'Pipelines' selected, indicated by a red arrow. The main area displays a table titled 'Recently run pipelines'. A specific row is highlighted with a black box and a red arrow pointing to it. The row details a pipeline named 'DemoCICDV2 - CI' with the run ID '#20210929.1'. It notes an update to 'Index.cshtml' and a 'Batched CI for master' run. The status is 'Just now' with a duration of '36s'.

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.

The screenshot shows the Microsoft Azure portal. In the top navigation bar, 'Microsoft Azure' and 'Upgrade' are visible. Below the bar, the project 'DemoCICDV2' is selected. The main content area is divided into two sections: 'CI/CD pipeline' on the left and 'Azure resources' on the right. The CI/CD pipeline section shows a sequence of steps: 'Code' (repository), 'Build' (pipeline 'DemoCICDV2 - CI'), and 'dev' (environment). The build step is shown as completed ('Succeeded'). The Azure resources section lists an 'Application endpoint' at 'https://democicdv2.azurewebsites.net' with a 'Browse' button highlighted by a red arrow. Other listed resources include an 'App Service' named 'DemoCICDV2' (status 'Running') and an 'SQL Database' named 'DemoCICDV2-db' (status 'Online').



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

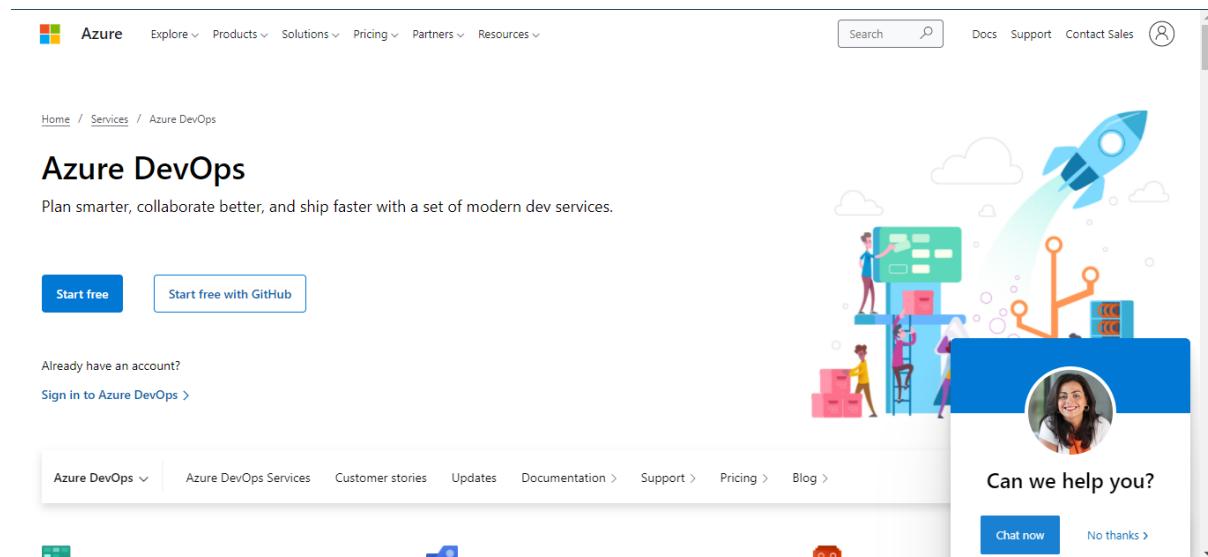
Learn more about all you can do with Azure DevOps project by visiting the documentation

[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

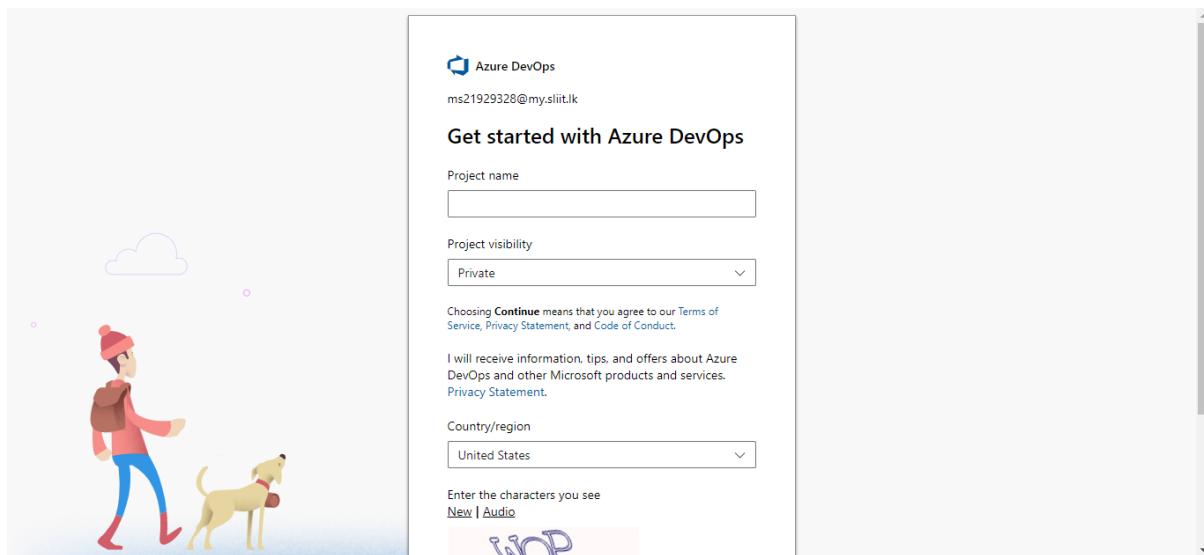
- 1) Sign into your Azure DevOps account by visiting to devops.azure.com and entering your credentials.

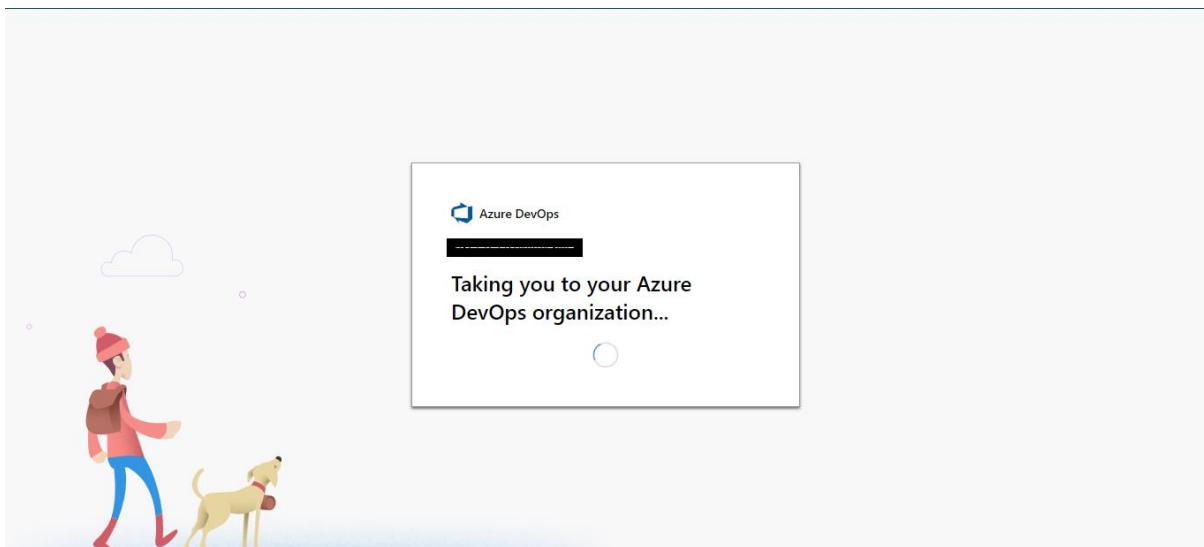


2) Then choose "Start Free" from the window.

The screenshot shows the Azure DevOps homepage. At the top, there is a navigation bar with the Azure logo, a search bar, and links for Explore, Products, Solutions, Pricing, Partners, and Resources. Below the navigation bar, the URL is shown as Home / Services / Azure DevOps. The main heading is "Azure DevOps" with the subtext "Plan smarter, collaborate better, and ship faster with a set of modern dev services." Two buttons are visible: a blue "Start free" button and a white "Start free with GitHub" button. A red arrow points to the "Start free" button. To the right of the buttons, there are two cartoon illustrations: one of a person standing next to a cloud icon, and another of a person pushing a stack of boxes. Below the buttons, there is a link "Already have an account?" and a "Sign in to Azure DevOps >" link. At the bottom of the page, there is a navigation bar with links for Azure DevOps Services, Customer stories, Updates, Documentation, Support, Pricing, and Blog.

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

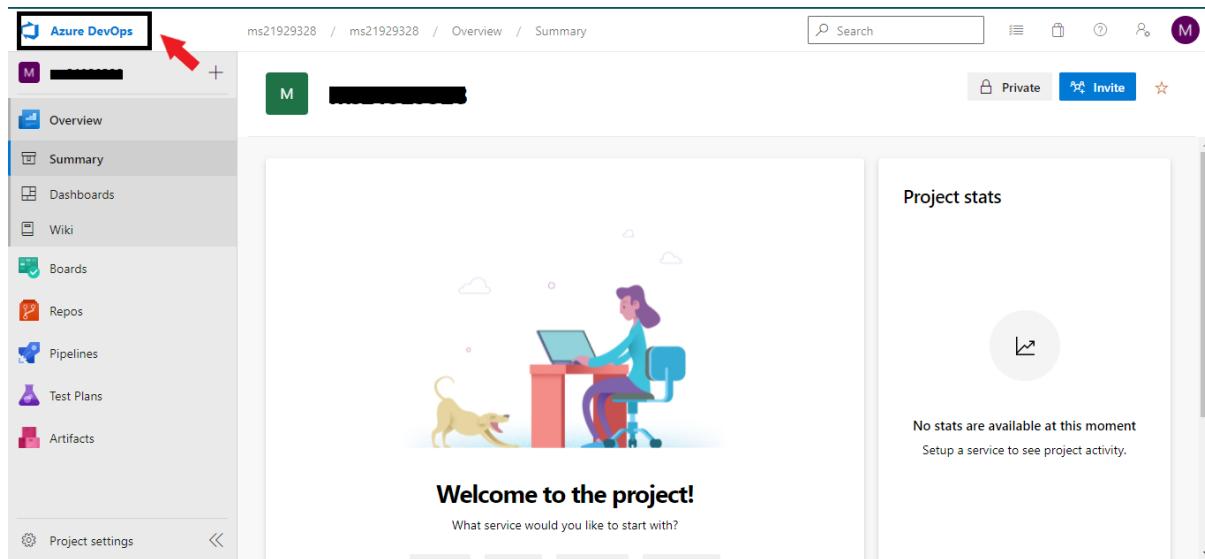




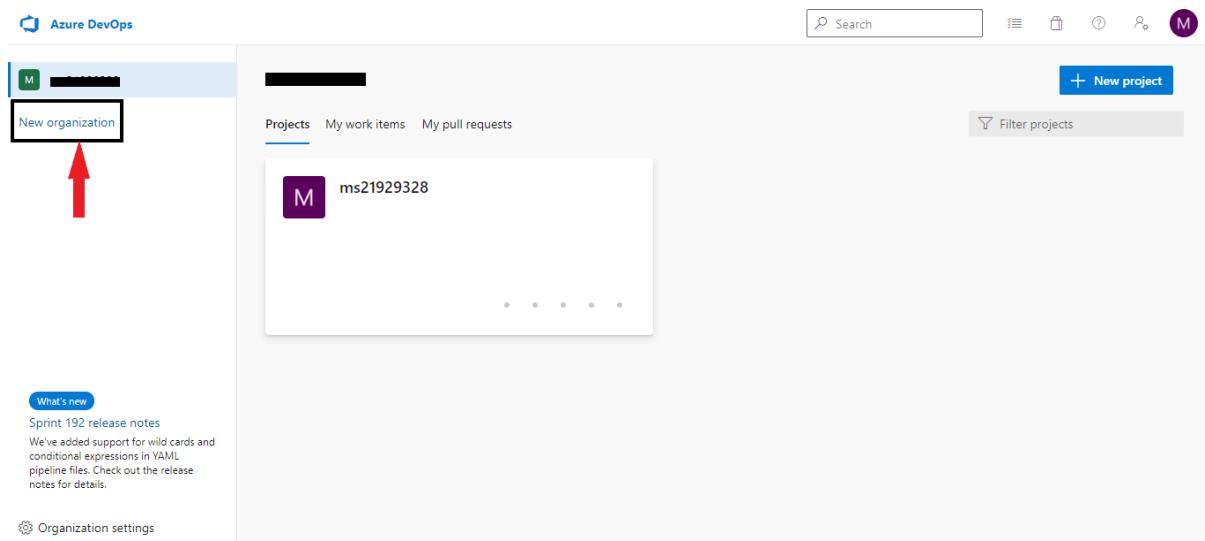
Your Azure DevOps account is now available to use.

A screenshot of the Azure DevOps interface. The top navigation bar shows "Azure DevOps" and the project path "ms21929328 / ms21929328 / Overview / Summary". The left sidebar includes links for Overview, Summary (which is selected), Dashboards, Wiki, Boards, Repos, Pipelines, Test Plans, and Artifacts, along with a "Project settings" link. The main content area features a central illustration of a person working at a desk with a dog, and the text "Welcome to the project! What service would you like to start with?". To the right is a "Project stats" section with a large circular icon and the message "No stats are available at this moment. Setup a service to see project activity.".

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.

The screenshot shows a web-based project creation interface. At the top right is a search bar with a magnifying glass icon and the word 'Search'. To its right are icons for a list, a shopping cart, a question mark, a gear, and a user profile with a 'M' monogram. Below the header is a section titled 'Create a project to get started'. It contains three input fields: 'Project name *' with a placeholder 'Project name', 'Description' with a large empty text area, and 'Visibility' with two options: 'Public' and 'Private'. The 'Private' option is selected, indicated by a blue border around its box and a blue dot next to the radio button.

Project name *

Description

Visibility

Public

Anyone on the internet can view the project. Certain features like TFVC are not supported.

Private

Only people you give access to will be able to view this project.

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.

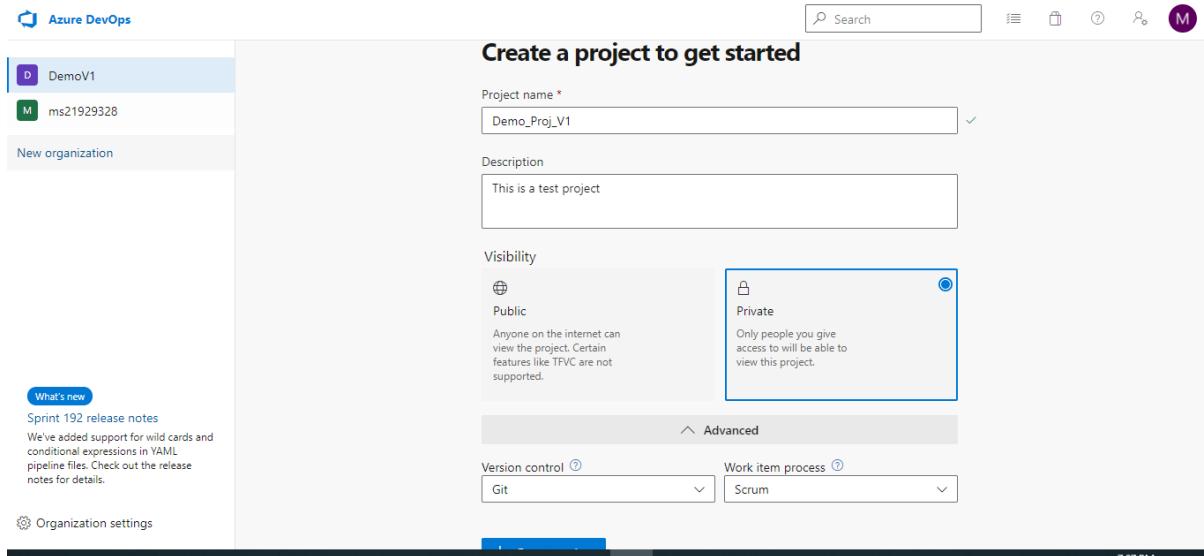
From the Advanced menu, choose

Version control: Git

Work item process: Scrum

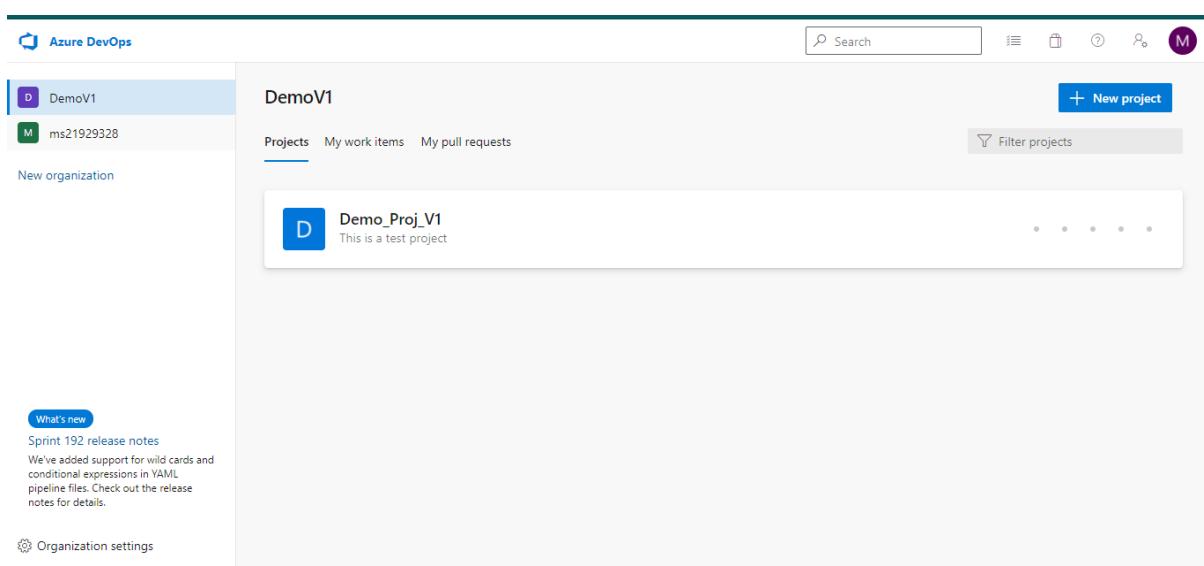
In this form, the project name is a required field, while the other elements are optional to complete.

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."

The screenshot shows the Azure DevOps interface for the 'Demo_Proj_V1' project. The left sidebar contains a navigation menu with options: Overview, Summary (which is selected and highlighted in blue), Dashboards, Wiki, Boards, Repos, Pipelines (highlighted with a pink box), Test Plans, and Artifacts. The main content area features a 'Welcome to the project!' message with a sub-instruction 'What service would you like to start with?'. To the right, there is a 'Project stats' section with a note: 'No stats are available at this moment. Setup a service to see project activity.' The top navigation bar includes a search bar, a 'Private' button, an 'Invite' button, and a user profile icon.

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

This screenshot is identical to the one above, showing the 'Demo_Proj_V1' project summary page. However, the 'Pipelines' option in the left sidebar navigation menu is now highlighted with a red rectangular box and a red arrow points to it, indicating the step to take.

Click on the “Create Pipeline”

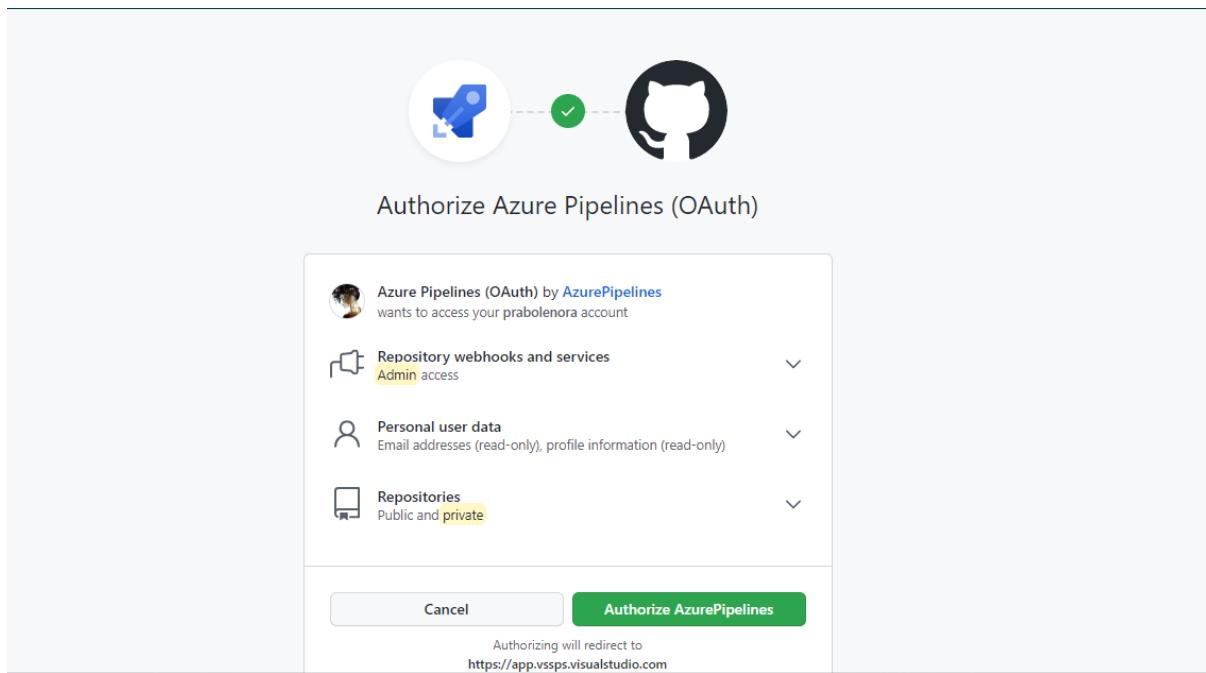
The screenshot shows the Azure DevOps interface for a project named "Demo_Proj_V1". The left sidebar has "Pipelines" selected. The main area features a cartoon illustration of a person launching a rocket. Below it, the text "Create your first Pipeline" is displayed, followed by the sub-instruction "Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes." A prominent blue "Create Pipeline" button is at the bottom right of this section.

Choose GitHub as the repository for your source code.

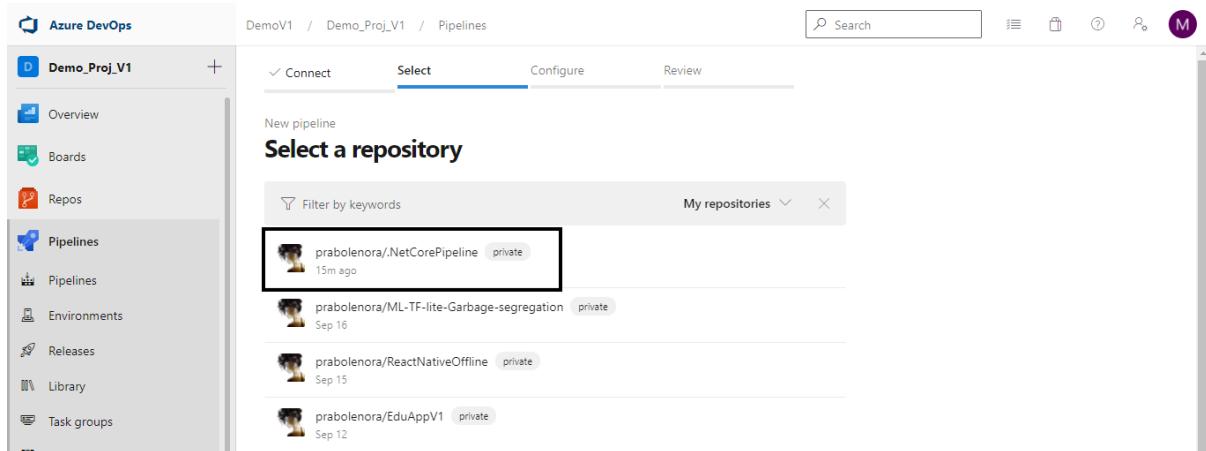
This screenshot shows the "Where is your code?" step in the pipeline creation wizard. The "Connect" tab is selected. A red arrow points to the "GitHub" option, which is highlighted with a black box. Other options listed include "Azure Repos Git", "Bitbucket Cloud", "GitHub Enterprise Server", "Other Git", and "Subversion".

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 screenshot shows the 'Approve & Install Azure Pipelines' page. At the top, there's a circular icon with a blue and white design. Below it, the title 'Approve & Install Azure Pipelines' is centered. A message box says 'suggested installation of this GitHub App 1 minute ago'. There are two radio button options: 'All repositories' (unchecked) and 'Only select repositories' (checked). A 'Select repositories' button with a dropdown arrow is shown. It indicates 'Selected 1 repository' and lists 'prabolenora/.NetCorePipeline' with a 'suggested' badge. Below this, it says 'with these permissions:' followed by two checked items: 'Read access to metadata' and 'Read and write access to checks, code, commit statuses, deployments, issues, and pull requests'. At the bottom, there are 'Approve & Install' and 'Reject' buttons. A note at the bottom states: 'Next: you'll be directed to the GitHub App's site to complete setup.'

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

This screenshot shows the 'Configure your pipeline' window in Azure DevOps. The left sidebar is titled 'Demo_Proj_V1' and includes options like Overview, Boards, Repos, Pipelines (which is selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Project settings. The main area has tabs at the top: 'Connect', 'Select', 'Configure' (which is underlined in blue), and 'Review'. Below the tabs, it says 'New pipeline' and 'Configure your pipeline'. There are four items listed: 1. 'ASP.NET Core' (selected, highlighted with a purple circle) - Build and test ASP.NET Core projects targeting .NET Core. 2. 'ASP.NET Core (.NET Framework)' - Build and test ASP.NET Core projects targeting the full .NET Framework. 3. 'Starter pipeline' - Start with a minimal pipeline that you can customize to build and deploy your code. 4. 'Existing Azure Pipelines YAML file' - Select an Azure Pipelines YAML file in any branch of the repository. A 'Show more' button is at the bottom.

From the configure window choose the ASP .Net Core

This screenshot is identical to the one above, showing the 'Configure your pipeline' window. However, the first item, 'ASP.NET Core', is now highlighted with a black rectangular box and a red arrow points to it from the left, indicating it should be selected.

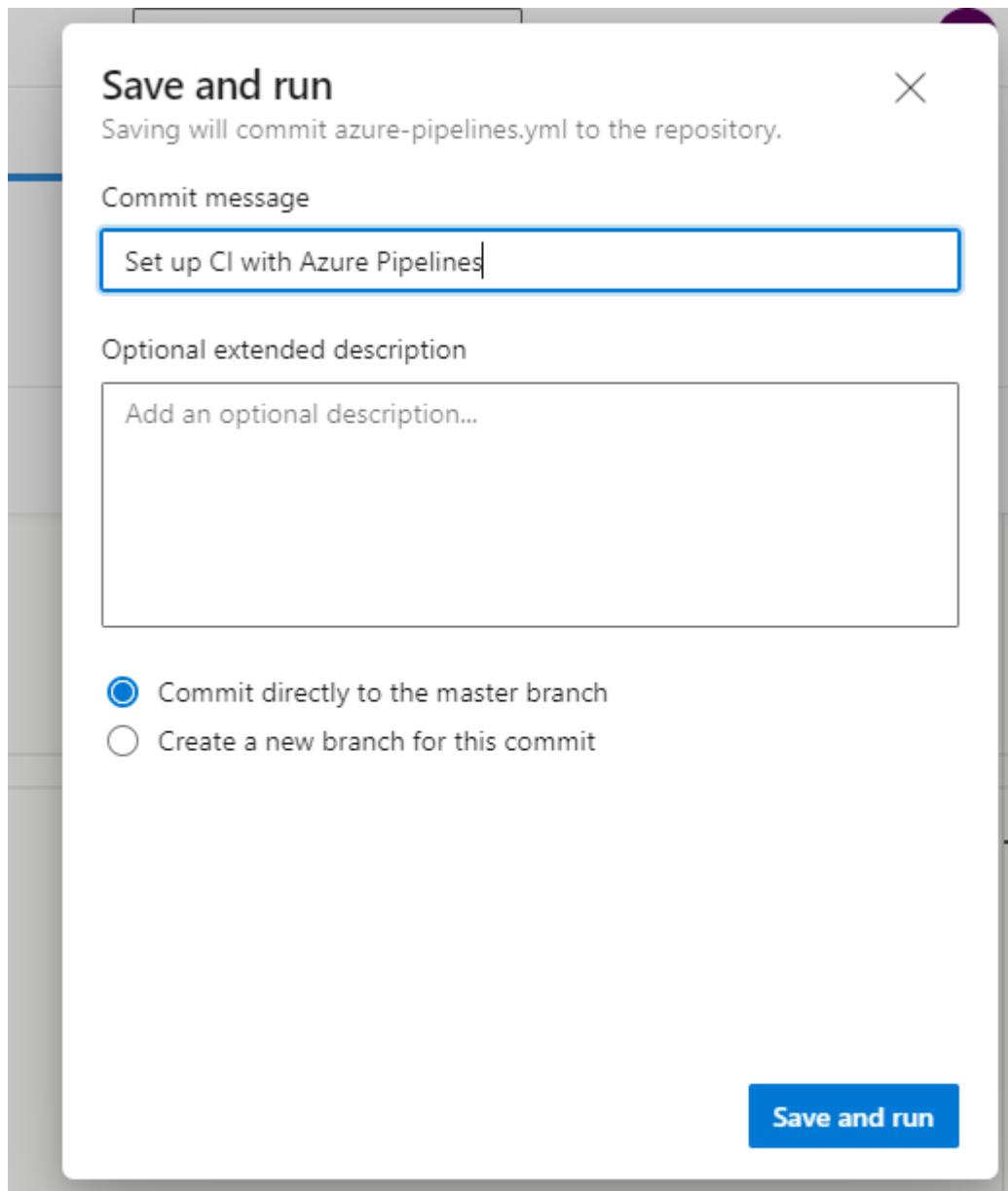
Your pipeline file will now display as a YAML file on the review tab of the application. To make the YAML file executable, choose "Save & Run."

The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with options like Overview, Boards, Repos, Pipelines (which is selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Project settings. The main area has a breadcrumb navigation path: DemoV1 / Demo_Proj_V1 / Pipelines. Below the path, there are four tabs: Connect, Select, Configure, and Review (the last one is underlined). A red arrow points down to the 'Save and run' button, which is located in a box on the right side of the screen. The 'Variables' button is also visible in the same box. The code editor displays a YAML pipeline configuration:

```
1  # ASP.NET Core
2  # Build and test ASP.NET Core projects targeting .NET Core.
3  # Add steps that run tests, create a NuGet package, deploy, and more:
4  # https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core
5
6  trigger:
7  - master
8
9  pool:
10 | - vmImage: ubuntu-latest
11
12 variables:
13 | - buildConfiguration: 'Release'
14
15 steps:
16 | - script: dotnet build --configuration $(buildConfiguration)
17 | - displayName: 'dotnet build $(buildConfiguration)'
```

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.

The screenshot shows the Azure DevOps interface for a project named 'Demo_Proj_V1'. The 'Pipelines' tab is selected in the sidebar. A single pipeline run is displayed under the heading '#20210928.1 Set up CI with Azure Pipelines'. The run was triggered by 'prabolenora'. The summary card shows the repository as 'prabolenora/.NetCorePipeline' and the commit as 'master b0ab20d0'. The run started at 8:52 PM today and completed successfully in 32 seconds. There are 0 work items and 0 artifacts. A 'View change' button is available. Below the summary is a 'Jobs' section showing one job named 'Job' which completed successfully in 24 seconds.

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.

Azure DevOps Parallelism Request

This form is for users to request increased parallelism in Azure DevOps.

* Required

1. What is your name? *

Enter your answer

2. What is your email address? *

Enter your answer

3. What is the name of your Azure DevOps Organization? *
(E.g. for <https://myorganization.visualstudio.com> or <https://dev.azure.com/myorganization> link formats - organization name would be 'myorganization')

Enter your answer

4. Are you requesting a parallelism increase for Public or Private projects? *

Private
 Public

Submit

Never give out your password. Report abuse

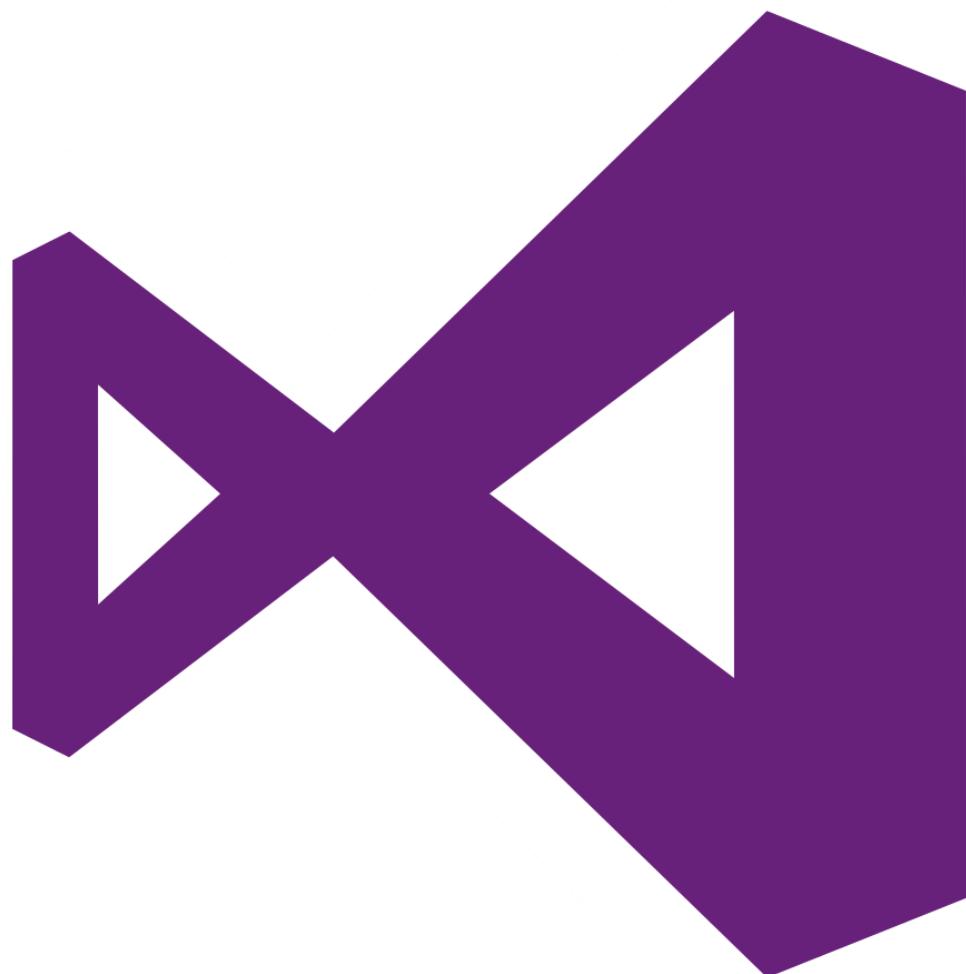
After Microsoft has approved your request, you will get an email confirming your permission.

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.

The screenshot shows the Azure DevOps Pipelines interface for a project named 'Demo_Proj_V1'. The pipeline run '#20210928.1 Set up CI with Azure Pipelines' is highlighted with a red arrow pointing to its title bar. Below the title bar, a message indicates that the run has been retained forever by 1 Pipeline. The 'Jobs' section shows a single job named 'Job' with a status of 'Success' and a duration of 24s. A red arrow points down to the 'Job' entry in the table.

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.

The screenshot shows the Azure DevOps interface for the 'CoreWeb' repository. On the left, there's a sidebar with options like Overview, Boards, Repos (selected), Files, Commits, Pushes, Branches, Tags, Pull requests, and Pipelines. The main area displays the message 'CoreWeb is empty. Add some code!'. Below this, there are two sections: 'Clone to your computer' and 'Push an existing repository from command line'. In the 'Clone to your computer' section, there are tabs for HTTPS and SSH, and a dropdown menu showing various cloning options. A red arrow points to the 'Clone in Visual Studio' option, which is highlighted with a black border. Other options listed in the dropdown include Android Studio, CLion, DataGrip, Eclipse, IntelliJ IDEA, PhpStorm, PyCharm, RubyMine, Tower, and Visual Studio.

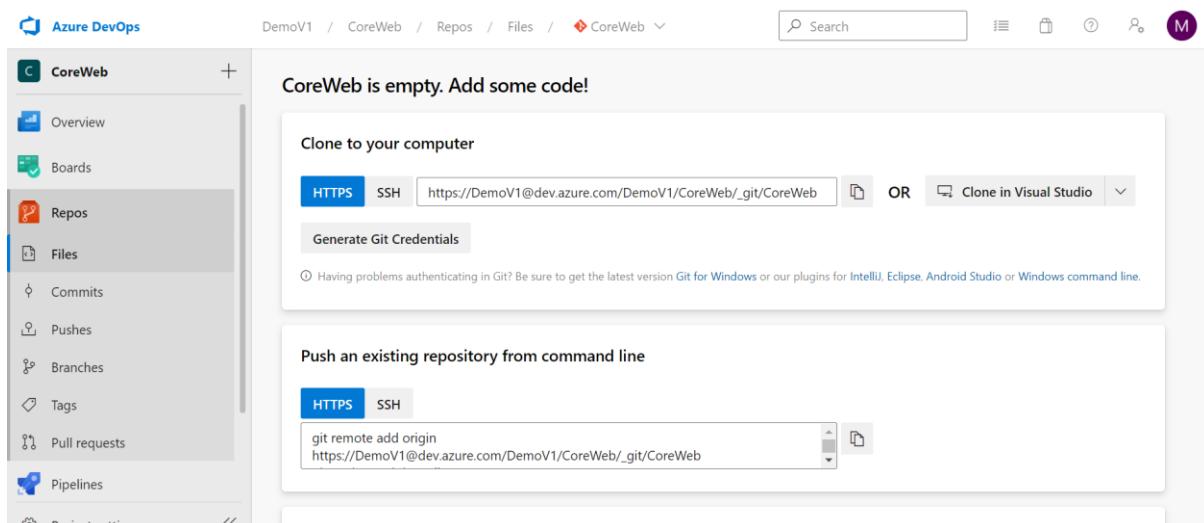
Following your selection of a concept, you will be routed to that idea. Your suggestion will be immediately accepted. You may either choose a local path on your computer to clone your project or it will clone your project to a default directory on your computer from there. The Azure DevOps clone URL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

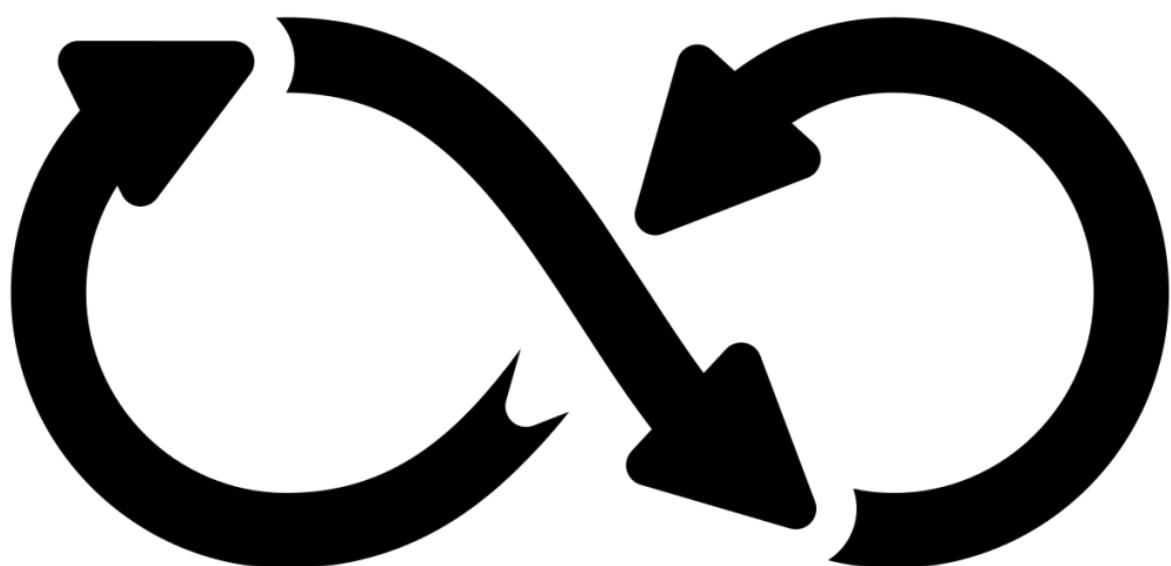
The screenshot shows the Azure DevOps interface for the CoreWeb repository. The left sidebar is visible with options like Overview, Boards, Repos, Files, Commits, Pushes, Branches, Tags, Pull requests, Pipelines, and Project settings. The main area displays the 'CoreWeb' repository under the 'master' branch. A search bar at the top right says 'Search'. Below it is a 'Build' button and a 'Clone' button. The 'Branches' tab is selected, showing the 'master' branch and its sub-branches: 'DevOpsCoreV1', 'TestProject1', and 'WebApplication1'. The 'Default' tag is also listed. A table below shows commit details:

Commit	Date	Author	Message
a6904db4	2h ago	prabodha	First Commit pr...
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...

This screenshot shows the same Azure DevOps interface, but the 'History' tab is selected under the 'Files' section for the 'CoreWeb' repository. The left sidebar remains the same. The main area now displays the commit history for the 'DevOpsCoreV1.sln' file. A 'Set up build' button and a 'Clone' button are present at the top right. The 'History' tab is active, showing a single commit:

Graph	Commit	Pull Request	Status
●	First Commit a6904db4 prabodha Today at 5:02 PM		

Chapter 07



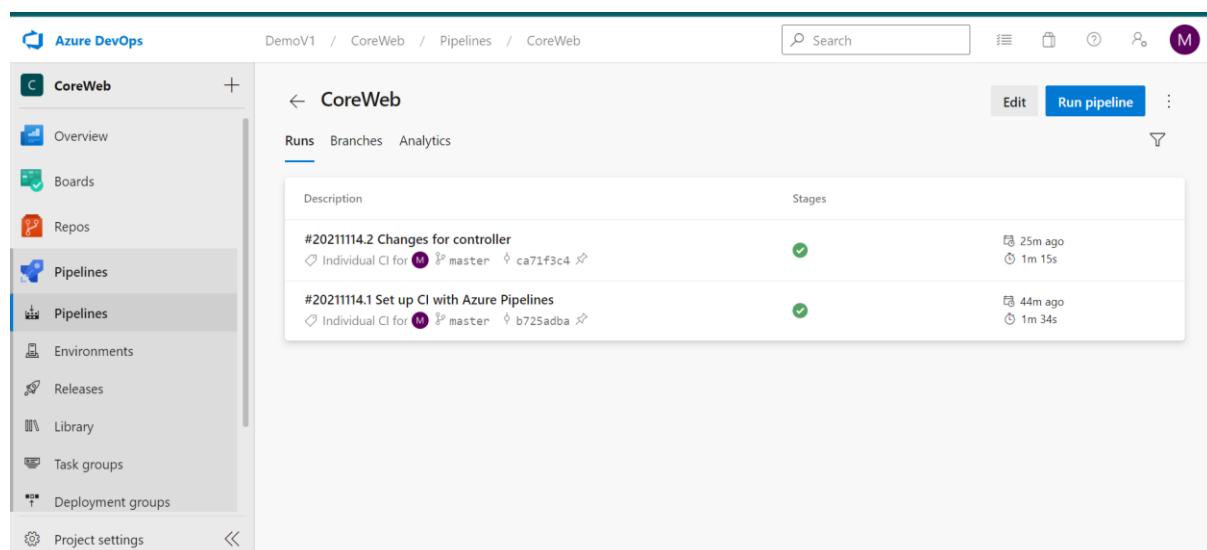
Continuous Integration

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.



The screenshot shows the Azure DevOps interface for the 'CoreWeb' project. The left sidebar has 'Pipelines' selected. The main area displays two recent pipeline runs under the 'Runs' tab:

Description	Stages	Created	Last Run
#20211114.2 Changes for controller Individual CI for master ↗ ca71f3c4 ↘	✓	25m ago	1m 15s
#20211114.1 Set up CI with Azure Pipelines Individual CI for master ↗ b725adba ↘	✓	44m ago	1m 34s

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.

```

1 # ASP.NET Core (.NET Framework)
2 # Build and test ASP.NET Core projects targeting the full .NET Framework
3 # Add steps that publish symbols, save build artifacts, and more:
4 # https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core
5
6 trigger:
7 - master
8
9 pool:
10 - vmImage: 'windows-latest'
11
12 variables:
13 - solution: '**/*.sln'
14 - buildPlatform: 'Any CPU'
15 - buildConfiguration: 'Release'
16
17 steps:
18 - task: NuGetToolInstaller@1
19

```

The screenshot shows the Azure DevOps interface for managing pipelines. The left sidebar has 'Pipelines' selected. The main area shows the YAML configuration for the 'CoreWeb' pipeline. On the right, there's a 'Tasks' pane with various options like 'NET Core', 'Android signing', 'Ant', etc.

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.

Description	Stages
#20211114.2 Changes for controller ⌚ Individual CI for M 🌐 master ↴ ca71f3c4	⌚ Just now ⌚ 5s
#20211114.1 Set up CI with Azure Pipelines ⌚ Individual CI for M 🌐 master ↴ b725adba ↴	⌚ 19m ago ⌚ 1m 34s

The screenshot shows the 'Runs' tab in the Azure DevOps pipeline interface. It lists two recent runs: one for changes to the controller and another for setting up CI with Azure Pipelines. The first run is currently 'Running', indicated by a green checkmark and a progress bar.

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

The screenshots illustrate the Azure DevOps Pipelines interface. The top screenshot shows the 'Runs' page for the 'CoreWeb' pipeline, listing two recent runs. The bottom screenshot provides a detailed view of the first run, showing its trigger, repository, and execution details.

Top Screenshot: CoreWeb Pipeline Runs

Description	Stages	Time
#20211114.2 Changes for controller Individual CI for master ↗ ca71f3c4 ✅	1	44m ago 1m 15s
#20211114.1 Set up CI with Azure Pipelines Individual CI for master ↗ b725adba ✅	1	1h ago 1m 34s

Bottom Screenshot: Run Details for #20211114.2

Triggered by	Repository and version	Time started and elapsed	Related	Tests and coverage
M [REDACTED]	CoreWeb master ↗ ca71f3c4	Today at 8:10 PM 1m 15s	0 work items 0 artifacts	Get started

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.

The screenshot shows the Azure DevOps interface for a project named 'CoreWeb'. The left sidebar is titled 'CoreWeb' and includes links for Overview, Boards, Repos, Pipelines (which is selected), Environments, Releases, Library, Task groups, Deployment groups, and Project settings. The main content area displays the 'Jobs in run #20211114.2' for CoreWeb. A summary table for the 'Job' step shows the following details:

Step	Description	Duration
1	Pool: Azure Pipelines	
2	Image: windows-latest	
3	Agent: Hosted Agent	
4	Started: today at 8:11 PM	
5	Duration: 1m 6s	
6	▶ Job preparation parameters	

Below this, a list of tasks is shown:

- Initialize job (6s)
- Checkout Cor... (3s)
- NuGetToolInst... (3s)
- NuGetComm... (26s)
- VSBuild (17s)
- VSTest (9s)
- Post-job: Ch... (<1s)
- Finalize Job (<1s)
- Report build

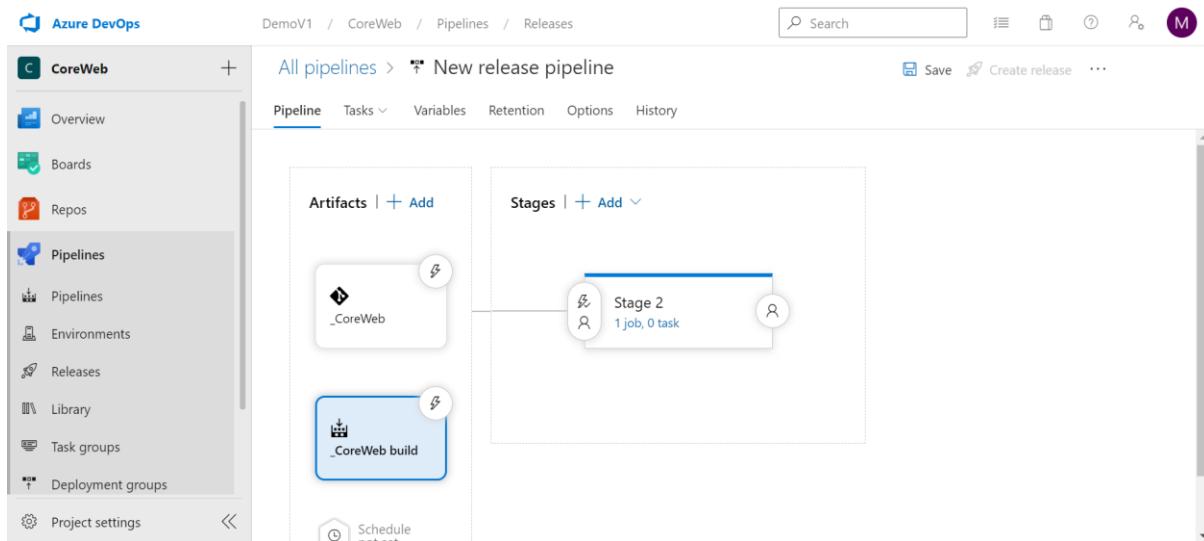
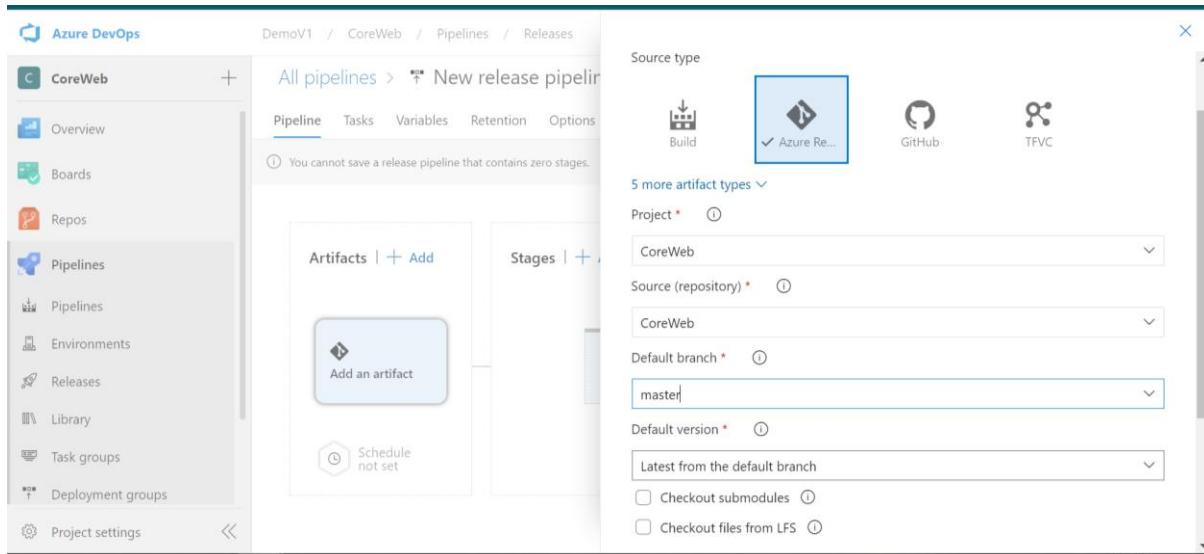
Chapter 08



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.

The screenshot shows the Azure DevOps interface for the CoreWeb project under the Pipelines / Releases section. The left sidebar has 'Releases' selected. The main area displays a 'New release pipeline' card with the message 'No deployments found'. A central illustration shows a person launching a rocket from a box. Below the illustration, a button says 'Create a release'.

Create a release for the pipeline at this point.

The screenshot shows the same Azure DevOps interface after creating a release. A green notification bar at the top says 'Release Release-1 has been queued'. The main area now lists 'Release-1' in the 'Releases' table, which includes columns for Created (11/14/2021, 9:42:03 PM), Stage (Stage 2), and a preview link (ca71f3c4). The preview link shows a commit from the master branch.

Let's edit the release pipeline. Click on the edit button.

The screenshot shows the Azure DevOps Pipelines interface for a project named "CoreWeb". The left sidebar has "Releases" selected. A green banner at the top says "Release Release-1 has been queued". The main area is titled "New release pipeline" with a sub-section "Release-1". A red arrow points to the "Edit" button in the top right corner of the pipeline card.

"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.

The screenshot shows the "Tasks" tab for the "New release pipeline". It lists a single "Agent job" under "Stage 2". On the right, a search bar finds the "Visual Studio Test" task, which is described as running unit and functional tests using the Visual Studio Test runner. A large "Add" button is visible at the bottom right of the task card.

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\**
```

The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with options like Overview, Boards, Repos, Pipelines, Environments, Releases, Library, Task groups, Deployment groups, and Project settings. The main area shows a pipeline named 'New release pipeline' with a single stage named 'Stage 2'. This stage contains an 'Agent job' with a task named 'VsTest - testAssemblies' (Visual Studio Test). To the right of the task, there's a detailed configuration pane. In the 'Test files' field, the text is set to:

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

The configuration pane also includes fields for 'Search folder', 'Test results folder', and other parameters.

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.

The screenshot shows the Azure DevOps interface for creating a new release pipeline. On the left, there's a sidebar with project navigation options like Overview, Boards, Repos, Pipelines, Releases, Library, Task groups, and Project settings. The main area shows a pipeline diagram with an 'Artifacts' section containing a 'CoreWeb build' artifact and a 'Stages' section with one stage labeled 'Stage 1 job'. To the right of the diagram is the 'Triggers' configuration panel. It includes a title 'Triggers ^ Define the trigger that will start deployment to this stage', a 'Select trigger' dropdown with two options: 'After release' (selected) and 'Manual only', and three toggle switches for 'Artifact filters', 'Schedule', and 'Pull request deployment', all of which are currently disabled.

Chapter 09



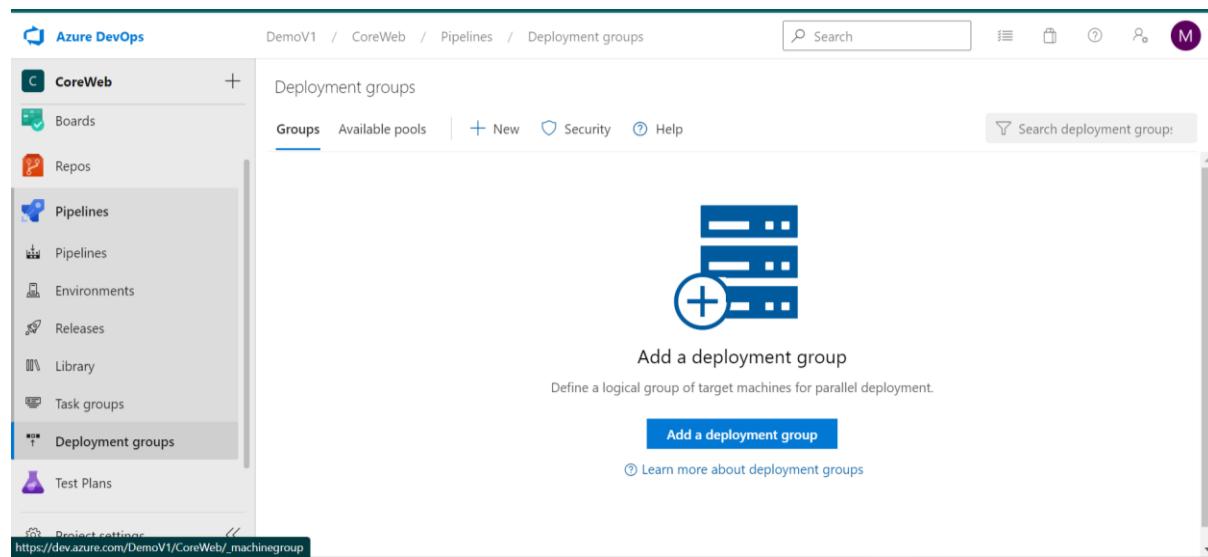
Deployment

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.

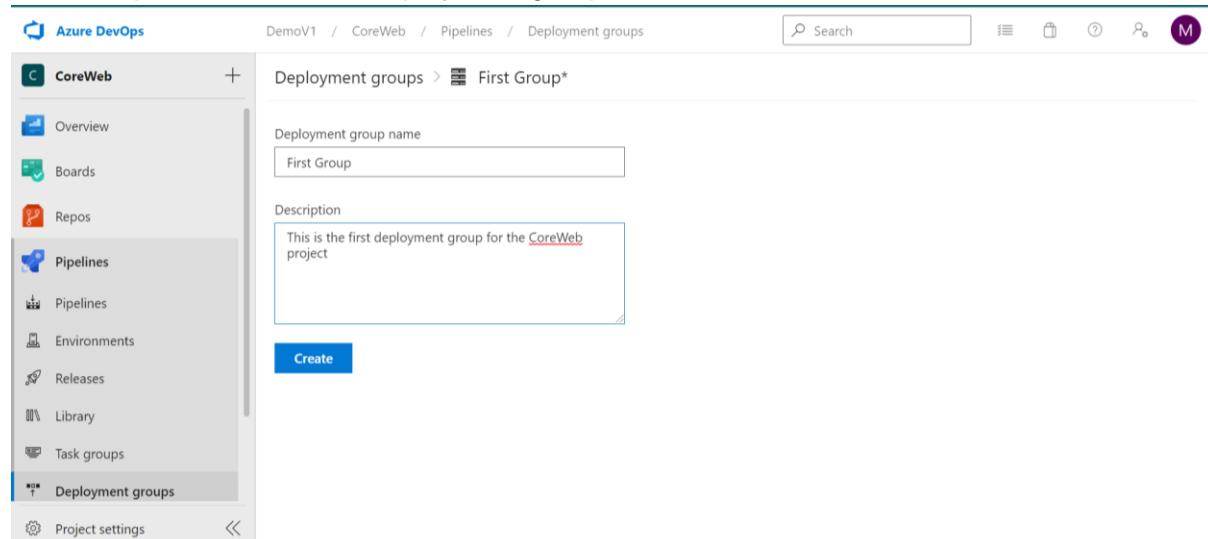
To begin, you must first establish a group.

"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."



The screenshot shows the Azure DevOps interface for creating a deployment group. The left sidebar has 'CoreWeb' selected. The main navigation bar shows 'DemoV1 / CoreWeb / Pipelines / Deployment groups'. Below the navigation is a search bar and some icons. The main content area is titled 'Deployment groups' and shows a large blue icon of two servers with a plus sign. Below the icon is the text 'Add a deployment group' and 'Define a logical group of target machines for parallel deployment.' A prominent blue button labeled 'Add a deployment group' is centered. At the bottom right is a link 'Learn more about deployment groups'.

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.



The screenshot shows the 'Add a deployment group' page for 'First Group'. The left sidebar has 'CoreWeb' selected. The main navigation bar shows 'DemoV1 / CoreWeb / Pipelines / Deployment groups / Deployment groups > First Group*'. The main content area has 'Deployment group name' set to 'First Group' and 'Description' set to 'This is the first deployment group for the CoreWeb project'. A blue 'Create' button is at the bottom.

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

The screenshot shows the Azure DevOps interface for the 'CoreWeb' project. The left sidebar has 'Pipelines' selected. The main area is titled 'Deployment groups > First Group'. It shows a form with fields for 'Deployment group name' (set to 'First Group'), 'Description' (containing a note about being the first deployment group for the CoreWeb project), and 'Deployment pool' (set to 'CoreWeb-First Group'). A large text box contains PowerShell registration script code. The top navigation bar shows the path 'DemoV1 / CoreWeb / Pipelines / Deployment groups' and includes a search bar and various icons.

Deployment group name

First Group

Description

This is the first deployment group for the [CoreWeb](#) project

Deployment pool

CoreWeb-First Group

Deployment groups > First Group

Details Targets Save Share Security Help

Type of target to register:

Windows

Registration script (PowerShell)

```
$ErrorActionPreference="Stop";If(-NOT ([Security.Principal.WindowsPrincipal]::GetCurrent().IsInRole([Security.Principal.WindowsBuiltInRole] "Administrator")){ throw "Run an administrator PowerShell prompt";}If($PSVersionTable.PSVersion -lt System.Version("3.0")){ throw "The minimum version of Windows PowerShell required by the script (3.0) does not match the currently running version. PowerShell." };If(-NOT ($testPath = Test-Path $env:SystemDrive\azagent)){ $env:SystemDrive\azagent}; cd $env:SystemDrive\azagent; For($i=1; $i++);{$destFolder = "$i.ToString()"; If(-NOT (Test-Path $destFolder)){ $destFolder=cd $destFolder;break;}}; $agentZip="$PwD\Agent-$i.zip";$defau [System.Net.WebRequest]::DefaultWebProxy;$securityProtocol=$securityProtocol+=[Net.ServicePointManager]::SecurityProtocol;[SecurityProtocol]$securityProtocol+= [Net.SecurityProtocolType]::Tls12; [Net.ServicePointManager]::SecurityProtocol=$securityProtocol;[$webClient Object] $webClient; $uri="https://vstsagentpackage.azureedge.net/agent/2.194.0/vsts-agent-2.194.0.zip";if($DefaultProxy -and (-not $DefaultProxy.IsBypassed($uri [WebClient] Proxy New-Object
```

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.

The screenshot shows the Azure DevOps interface for a 'CoreWeb' project under 'Deployment groups'. The 'First Group' is selected. On the right, a large block of PowerShell script is displayed, and a red arrow points to a checkbox labeled 'Use a personal access token in the script for authentication'. Below the script, there is a blue button labeled 'Copy script to the clipboard'.

```

[System.Net.WebRequest]::DefaultWebProxy;$securityProtocol=@();$securi
[Net.ServicePointManager]::SecurityProtocol;$securityProtocol+=
[Net.SecurityProtocolType]::Tls12;
[Net.ServicePointManager]::SecurityProtocol=$securityProtocol;$WebClient
Object,Net.WebClient;
$Uri="https://vstsagentpackage.azureedge.net/agent/2.195.0/vsts-agent-
2.195.0.zip";if($DefaultProxy -and (-not $DefaultProxy.IsBypassed($Uri
{$WebClient.Proxy} New-Object
Net.WebProxy($DefaultProxy.GetProxy($Uri).OriginalString, $True)););
$WebClient.DownloadFile($Uri, $agentZip);Add-Type -AssemblyName
System.IO.Compression.FileSystem;
[System.IO.Compression.ZipFile]::ExtractToDirectory( $agentZip,
"$PWD");<config.cmd --deploymentgroup --deploymentgroupname "First Gr
agent $env:COMPUTERNAME --runnasservice --work '_work' --url
'https://dev.azure.com/DemoV1/' --projectname 'CoreWeb'; Remove-Item $
pvxhmmmdhjwseloe2dcrngocukrvu7erjpnbk4yta; Remove-Item $agentZip;

```

Use a personal access token in the script for authentication

Copy script to the clipboard

Run from an administrator PowerShell command prompt

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/powershell

```

PS C:\Users\praboo> $ErrorActionPreference="Stop";If(-NOT ((Security.Principal.WindowsPrincipal)[Security.Principal.WindowsIdentity]::GetCurrent() ).IsInRole( [Security.Principal.WindowsBuild
tInRole] "Administrator")){ throw "Run command in an administrator PowerShell prompt"};If($PSVersionTable.PSVersion -lt (New-Object System.Version("3.0"))){ throw "The minimum version of Wi
ndows PowerShell that is required by the script (3.0) does not match the currently running version of Windows PowerShell." };If(-NOT (Test-Path $env:SystemDrive"\azagent)){mkdir $env:Sy
stemDrive"\azagent"};cd $env:systemdrive\azagent";for($i=1; $i -lt 100; $i++){${destFolder}"\"$i.ToString();If(-NOT (Test-Path ${destFolder})){$mdir ${destFolder};break}}; $ag
entZip=$PWD\agent.zip;$DefaultProxy=[System.Net.WebRequest]::DefaultWebProxy;$securityProtocol=$securityProtocol@();$securityProtocol+= [Net.ServicePointManager]::SecurityProtocol;$securityProtocol+= [Net.S
ecurityProtocolType]::Tls12;[Net.ServicePointManager]::SecurityProtocol=$securityProtocol;$WebClient=New-Object Net.WebClient; $Uri="https://vstsagentpackage.azureedge.net/agent/2.195.0/vsts-
agent-win-x64-2.195.0.zip";if($DefaultProxy -and (-not $DefaultProxy.IsBypassed($Uri))){$WebClient.Proxy= New-Object Net.WebProxy($DefaultProxy.GetProxy($Uri).OriginalString, $True)}; $Web
client.DownloadFile($Uri, $agentZip);Add-Type -AssemblyName System.IO.Compression.FileSystem;[System.IO.Compression.ZipFile]::ExtractToDirectory( $agentZip, '$PWD');<config.cmd --depl
oymentgroup --deploymentgroupname "First Group" --agent $env:COMPUTERNAME --runnasservice --work '_work' --url 'https://dev.azure.com/DemoV1/' --projectname 'CoreWeb' --auth PAT --token $ncnogd
pvxhmmmdhjwseloe2dcrngocukrvu7erjpnbk4yta; Remove-Item $agentZip;

```

Directory: C:\azagent

Mode	LastWriteTime	Length	Name
d----	11/18/2021 5:03 AM		A5

/A ZURE|DEVICES

agent v2.195.0 (commit 805596a)

>> Connect:
Connecting to server ...
>> Register Agent:
Scanning for tool capabilities.
Connecting to the server.
Enter deployment group tags for agent? (Y/N) (press enter for N) >

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

```

Administrator: Windows PowerShell

Directory: C:\azagent

Mode LastWriteTime Length Name
---- -- -- A5
d--- 11/18/2021 5:03 AM
agent v2.195.0
                               (commit 805596a)

>> Connect:
Connecting to server ...
>> Register Agent:
Scanning for tool capabilities.
Connecting to the server.
Enter deployment group tags for agent? (Y/N) (press enter for N) > n
Successfully added the agent
Testing agent connection.
2021-11-18 05:05:17Z: Settings Saved.
Enter User account to use for the service (press enter for NT AUTHORITY\SYSTEM) >
Error reported in diagnostic logs. Please examine the log for more details.
- C:\azagent\A5\diagAgent_20211118-050346-utc.log
Granting file permissions to 'NT AUTHORITY\SYSTEM'.
Service vstsagent.DemoV1.CoreWeb-First Group.WindowsServer successfully installed
service vstsagent.DemoV1.CoreWeb-First Group.WindowsServer successfully set recovery option
Service vstsagent.DemoV1.CoreWeb-First Group.WindowsServer successfully set to delayed auto start
service vstsagent.DemoV1.CoreWeb-First Group.WindowsServer successfully configured
Enter whether to prevent service starting immediately after configuration is finished? (Y/N) (press enter for N) >
Service vstsagent.DemoV1.CoreWeb-First Group.WindowsServer started successfully

PS C:\azagent\A5> 

```

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.

The screenshot shows the Azure DevOps interface. The left sidebar has a navigation menu with items like Overview, Boards, Repos, Pipelines, Environments, Releases, Library, Task groups, and Deployment groups. The Deployment groups item is currently selected. The main content area shows a breadcrumb path: DemoV1 / CoreWeb / Pipelines / Deployment groups. Below this, it says 'Deployment groups > First Group'. Underneath, there are tabs for Details, Targets (which is selected), Save, Share, Security, and Help. The 'Targets' section shows a summary: 'Healthy (1)'. It lists a single target named 'WindowsServer' with the status 'No deployments yet'. There are buttons for 'Register' and 'Add tags'.

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

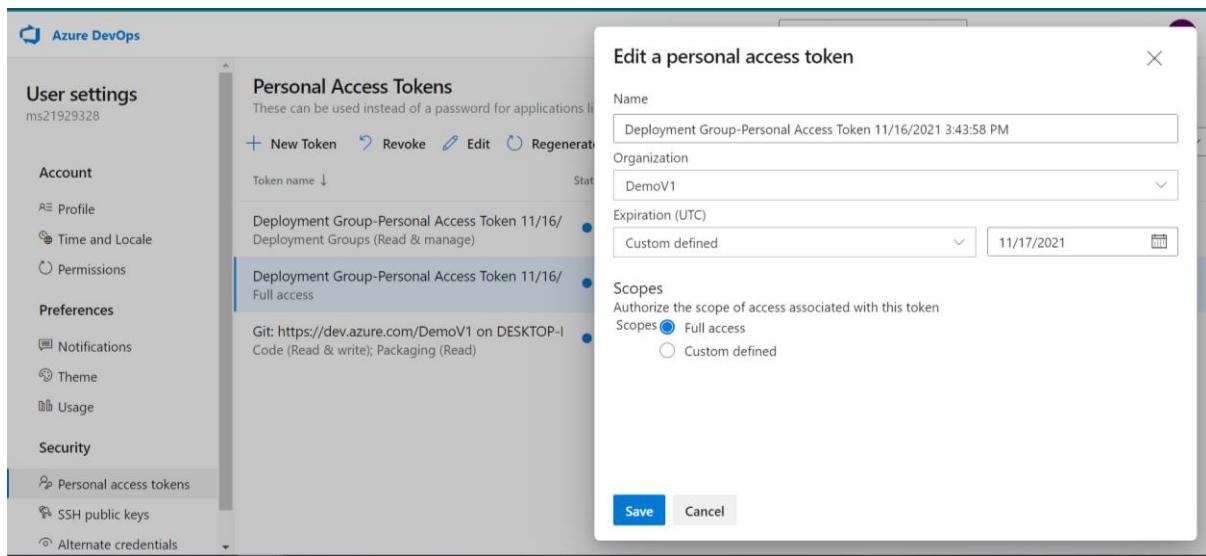
The screenshot shows the Azure DevOps interface for managing deployment groups. On the left, there's a sidebar with project navigation. The main area displays a deployment group named 'First Group' under 'Deployment groups'. It includes fields for 'Deployment group name' (set to 'First Group'), 'Type of target to register' (set to 'Windows'), and a 'Registration script (PowerShell)' pane containing PowerShell code. A vertical sidebar on the right lists various user profile and access settings.

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.

The screenshot shows the 'User settings' page in Azure DevOps. The 'Security' section is selected, specifically the 'Personal access tokens' subsection. It lists three active tokens:

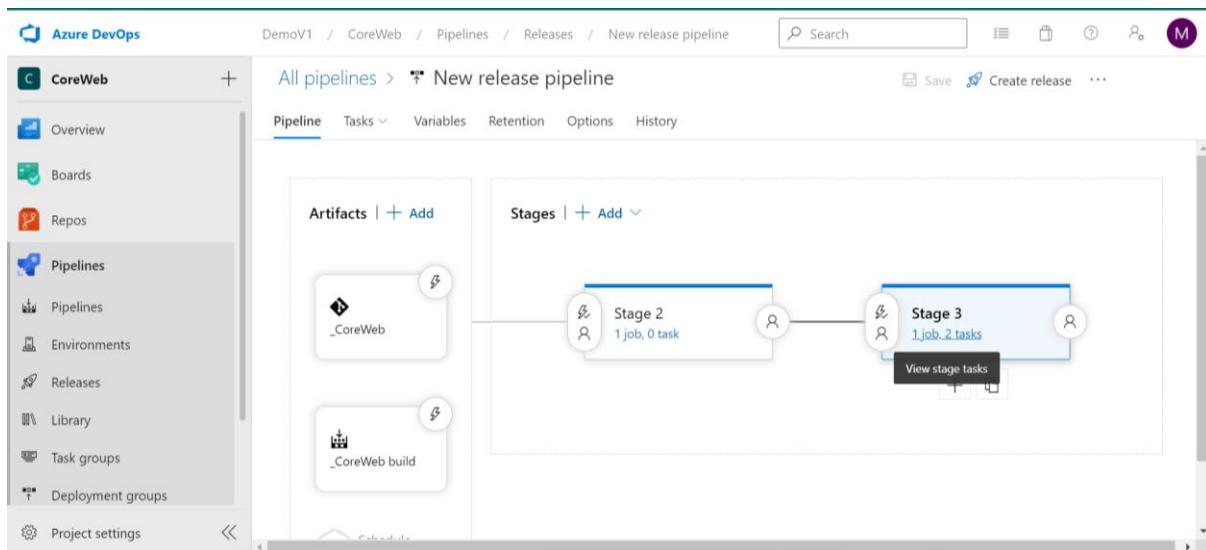
Token name	Status	Organization	Expires on
Deployment Group-Personal Access Token 11/16/2021	Active	DemoV1	11/17/2021
Deployment Group-Personal Access Token 11/16/2021	Active	DemoV1	11/17/2021
Git: https://dev.azure.com/DemoV1 on DESKTOP-I	Active	DemoV1	11/14/2022

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.

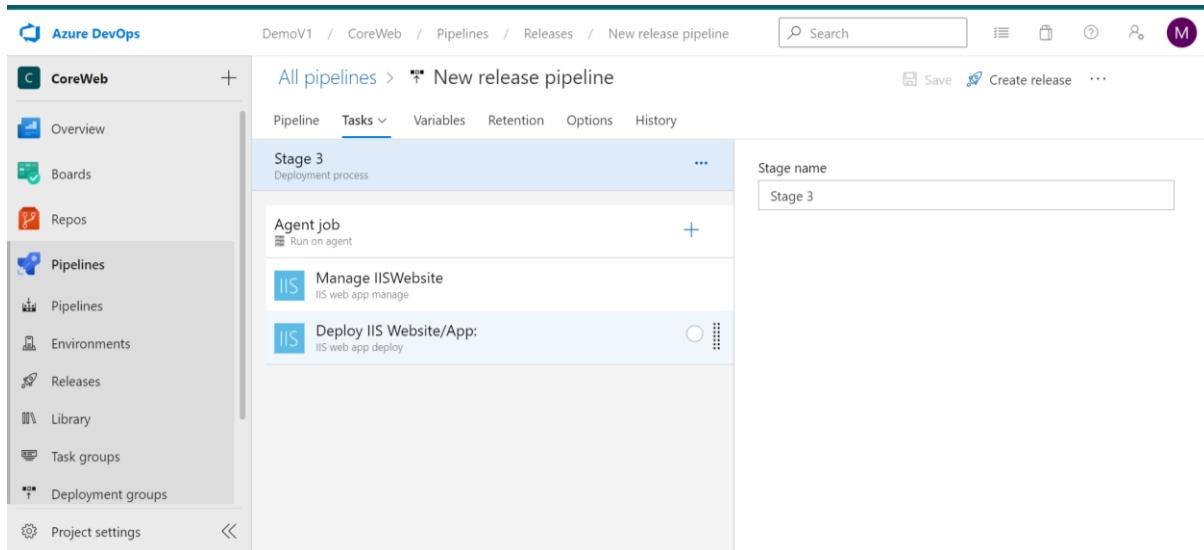


9.2 IIS Deployment Manage

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.

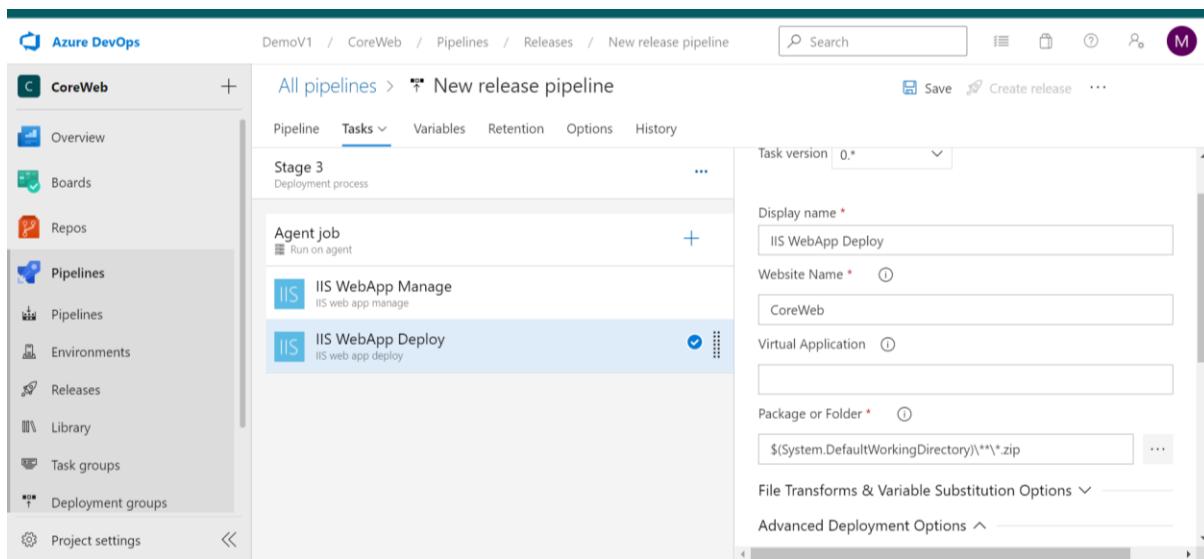


The screenshot shows the Azure DevOps Pipeline Editor. On the left, there's a sidebar with project navigation: CoreWeb, Overview, Boards, Repos, Pipelines (selected), Environments, Releases, Library, Task groups, Deployment groups, and Project settings. The main area shows a pipeline named 'New release pipeline' with three stages: Stage 1 (Deployment process), Stage 2 (Deployment process), and Stage 3 (Deployment process). Stage 3 contains an 'Agent job' task labeled 'Run on agent'. Underneath it are two 'IIS' tasks: 'Manage IISWebsite' and 'Deploy IIS Website/App'. The 'Deploy IIS Website/App' task has a small ellipsis icon next to it.

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.



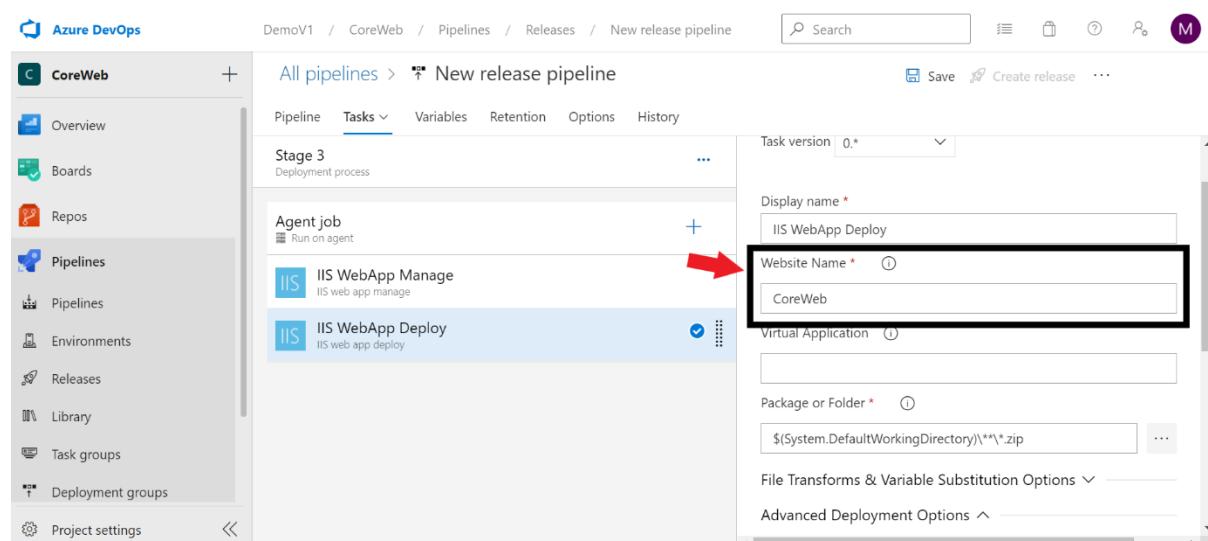
This screenshot is similar to the previous one, showing the Pipeline Editor for the 'New release pipeline'. Stage 3 now contains two tasks: 'IIS WebApp Manage' and 'IIS WebApp Deploy'. A detailed configuration pane is open for the 'IIS WebApp Deploy' task. It shows the following fields:

- Task version: 0.*
- Display name: IIS WebApp Deploy
- Website Name: CoreWeb
- Virtual Application: (empty)
- Package or Folder: \$(System.DefaultWorkingDirectory)/**/*.zip
- File Transforms & Variable Substitution Options
- Advanced Deployment Options

9.3 IIS Deployment

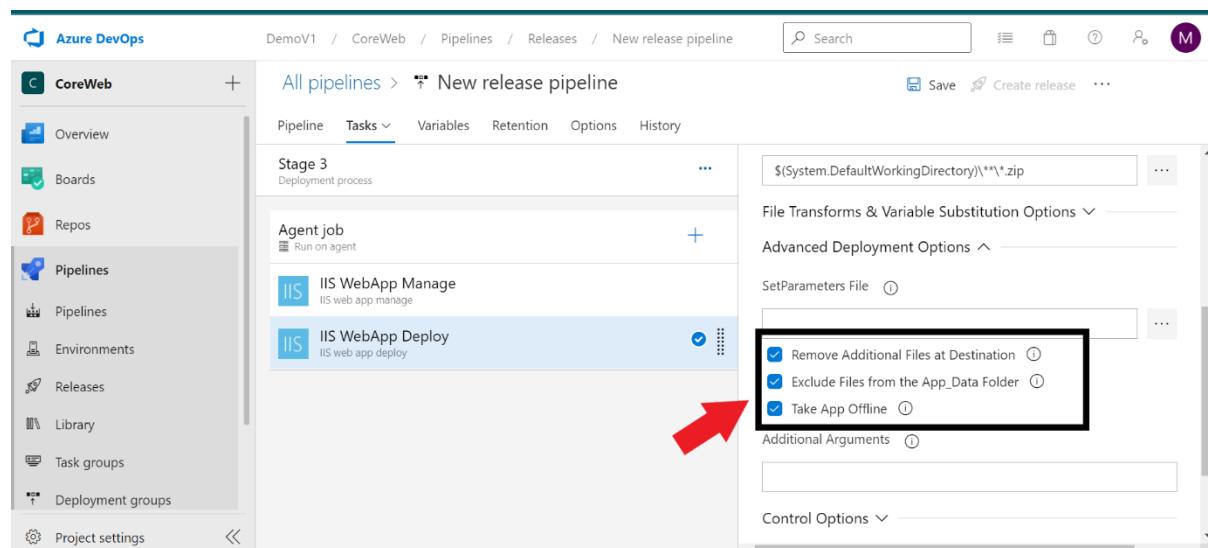
Let's have a look at how we can put the idea into production.

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.



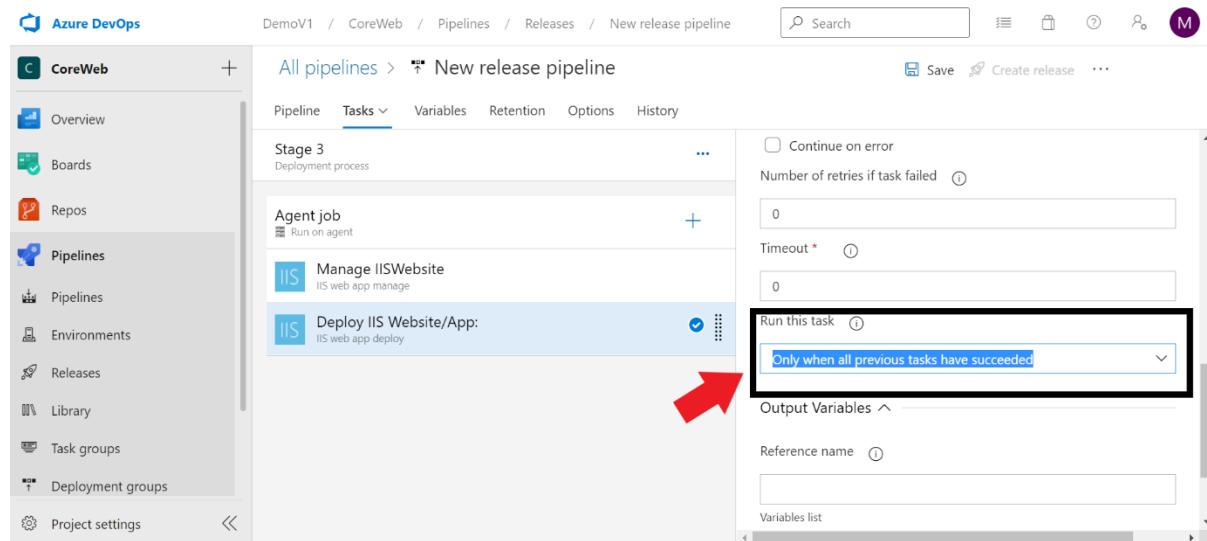
The screenshot shows the Azure DevOps Pipeline Editor. On the left, there is a sidebar with options like Overview, Boards, Repos, Pipelines, Environments, Releases, Library, Task groups, Deployment groups, and Project settings. The Pipelines option is selected. In the center, under 'All pipelines > New release pipeline', there is a 'Stage 3' section. An 'Agent job' is listed, followed by two 'IIS WebApp Deploy' tasks. The second 'IIS WebApp Deploy' task is currently selected. A red arrow points to the 'Website Name' field in the configuration pane on the right, which is set to 'CoreWeb'. Other fields visible include 'Virtual Application' (empty), 'Package or Folder' (\$System.DefaultWorkingDirectory)***.zip), and 'File Transforms & Variable Substitution Options'.

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.



This screenshot is similar to the previous one, showing the Azure DevOps Pipeline Editor with the 'New release pipeline' stage. The 'IIS WebApp Deploy' task is selected. A red arrow points to the 'Advanced Deployment Options' section in the configuration pane. Within this section, a box highlights three checked checkboxes: 'Remove Additional Files at Destination', 'Exclude Files from the App_Data Folder', and 'Take App Offline'. Other sections visible include 'File Transforms & Variable Substitution Options' (with a value of '\$(System.DefaultWorkingDirectory)***.zip') and 'Control Options'.

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.



The screenshot shows the Azure DevOps interface for creating a new release pipeline. On the left, there's a sidebar with project navigation: CoreWeb, Overview, Boards, Repos, Pipelines (selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, and Project settings. The main area is titled 'All pipelines > New release pipeline'. It shows a 'Stage 3 Deployment process' with two tasks: 'Manage IISWebsite' (IIS web app manage) and 'Deploy IIS Website/App:' (IIS web app deploy). A red arrow points from the text below to the 'Run this task' dropdown for the second task, which is set to 'Only when all previous tasks have succeeded'. Other settings visible include 'Continue on error' (unchecked), 'Number of retries if task failed' (0), and 'Timeout' (0).

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/>