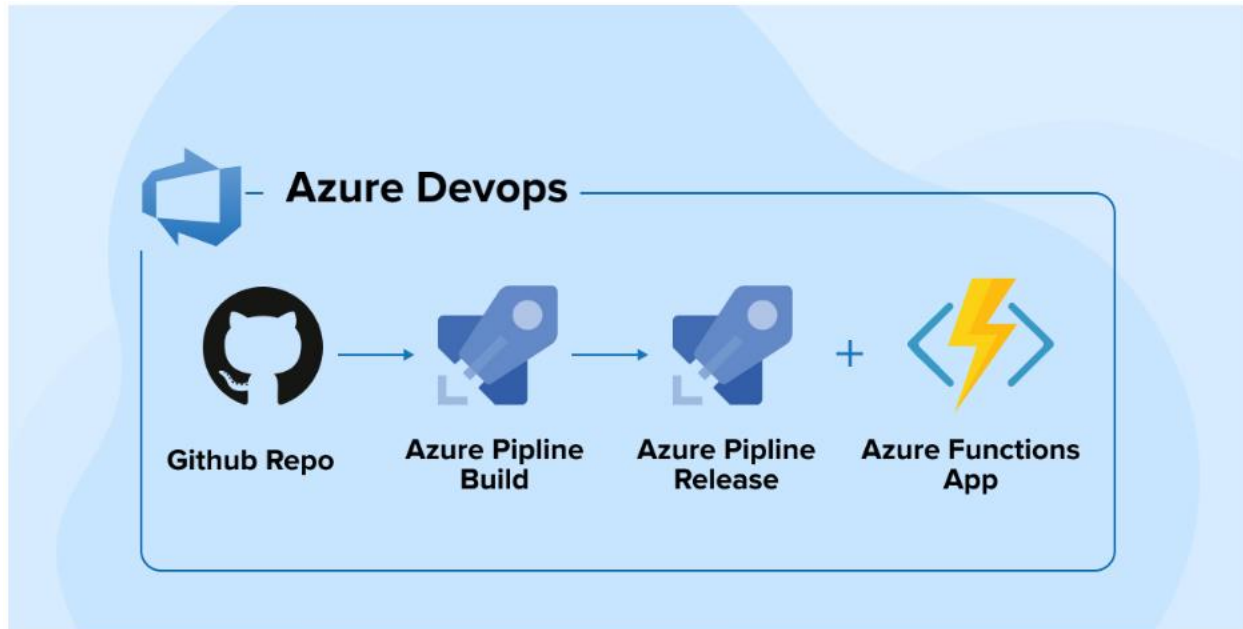


INTRODUCTION OF AZURE DEVOPS PIPELINES



Technology ingestion happens now and then. The main reason for discovering a new way or method to work is to decrease human efforts. As humans, our efforts are always driven towards excellence that would increase productivity and make the process simpler than ever before. Various tools, software, and apps we use in day-to-day life to keep our things managed. But even custom software development and deployment needs some toolchain that keeps your source code in a private repository on the cloud, keeps track of Task-list, generates reports, can do agile planning, and provides an integrated testing solution that would help in the deployment process and that's where Azure DevOps – Software as a Service (SaaS) platform from Microsoft comes to into picture. The launch of Azure DevOps was around 2018 and hence it is not a newbie in the world of technology. Its origin can be traced using the Visual Studio Team system launched in 2006.

1. What is Azure DevOps?

Azure DevOps is an all-inclusive package offering a wide-ranging service that covers the entire SDLC or Application development life cycle. Let's see some substantial features of Azure DevOps:

- **Azure Boards:** Allows Work item tracking, Agile planning, Power BI visualization, and similar other reporting tools.
- **Azure Pipelines:** Defines CI/CD- Continuous Integration and Continuous deployment process with support for containers and Kubernetes.
- **Azure Repos:** Provides full support for cloud-hosted private repositories.
- **Azure Artifacts:** Package management Support for Maven, npm, NuGet, and Python package feeds from private or public sources.
- **Azure Test Plans:** Provides integrated planning and investigation of testing solutions.

Azure DevOps is the feature-rich mature model offered by Microsoft for businesses to simultaneously manage multiple tasks.

2. What is the Azure Pipeline?

Azure Pipeline is an automated set of processes that helps developers to compile, build, and deploy codes on other computation platforms. It is a continuous delivery tool similar to open source Jenkins or CodeShip. The sole goal of this pipeline is that there is no manual intervention, all the changes are automatically

executed in the project. Whenever humans are handling the process, there is a possibility of human errors while doing the same repetitive boring tasks, but with automation, it works seamlessly once it's configured.

A pipeline is normally broken down into the following categories:

- Source Control
- Build Tools
- Package creation
- Configuration management
- Monitoring

This Pipeline can be used with many application types such as Java, Go, Python, JavaScript, Node.js, .Net, C++, and XCode. To use it, you will need a source control to attach to it. Moreover, it supports various source controls such as GitHub, Azure Repos, Bitbucket, Subversion, etc.

This pipeline is based on the strong foundations of CI/CD pipelines which consistently test, build, and deploy. They also perform constant deliveries automatically by continuously testing and deploying the codes to the desired targets in the project.

Continuous Integration (CI)

Continuous Integration helps to catch the bugs and issues in the early stage of the development cycle when fixing errors is simpler and fast. Developers can check their codes in version-controlled repositories, check-in their code to test and locate errors. The main benefits are:

- Minor changes are easier to merge in large codes.
- Easier for big teams to see what each one has been working on.
- Locate bugs and make them easier to fix.
- Continuous code compilation and testing.
- Making the integration process smooth thus improving productivity.

Continuous Delivery (CD)

Continuous Delivery is the process of integrating code with infrastructure, ensuring that all tests are performed, and policies are followed, and finally deploying the code into the desired environment. The CD helps developers to deliver new features, fix bugs and configure changes reliably and quickly. CD pipeline provides routine deliveries as per configuration or can also be called on-demand. The key benefits are:

- Decreases risks in releases.
- Faster delivery of bug fixes and issues.
- Delivery can be feasible at any scale.

Azure Pipelines provides YAML syntax and a classic interface to set up CI-CD pipelines.

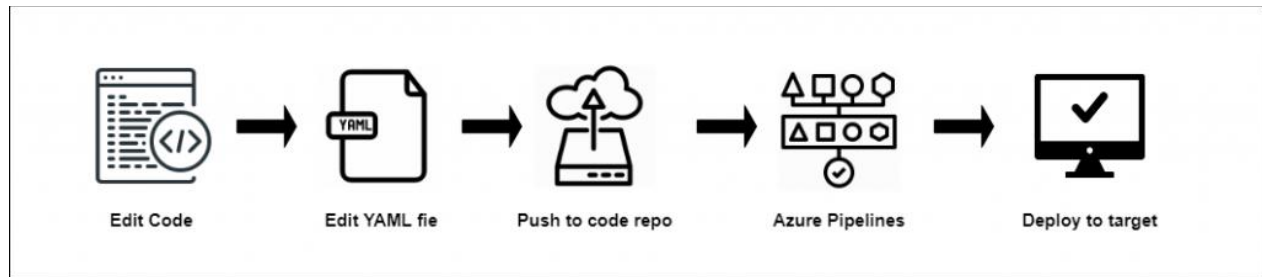
3. Advantages of Using Azure DevOps Pipeline:

Azure pipeline includes a variety of elements and when implied with Azure DevOps CI/CD practice, Azure DevOps Pipeline offers many benefits to the user. Some of them are as mentioned below:

1. Version Control Systems – The first thing you need to do to create an Azure CI/CD pipeline is to put your code in a version control system. Any kind of Git repository including GitHub, Github Enterprise, Subversion, Bitbucket Cloud, and more can help you manage your source code. Developers are also known to leverage it to support Team Foundation Version Control (TFVC).
2. Programming Languages and types of Application – With the Azure DevOps pipeline, you can use it with most application types and various kinds of programming languages including Javascript, Python, Java, Ruby, PHP, C, C++, and so on. It can be used with any platform like Linux, Windows, and macOS.
3. Deployment Targets – You can deploy Azure CI/CD pipelines with apps on environments with multiple targets like any On-prem or cloud platform, virtual machines, container registries, Azure services, containers, etc.

4. Pricing – If you are working on a public project then the Azure DevOps pipeline comes free of charge but in the case of private projects you have to subscribe to its services not before you run up to 1800 minutes of pipeline jobs for free every month.
5. Progressive Deployment – During the development and testing phase, the Azure DevOps pipeline allows you to set as many stages as you want to control the quality of the project before you have to move on to the next stage. This helps you in accurately pinpointing the bugs and understanding the problems, where they are occurring, and how to solve them. The completely customized deployment controls of the Azure DevOps pipeline not only keep you in the controlling position for your project but also enable you to make deliberate delivery of CI/CD solutions to the client.

4. Define Pipelines Using YAML Syntax



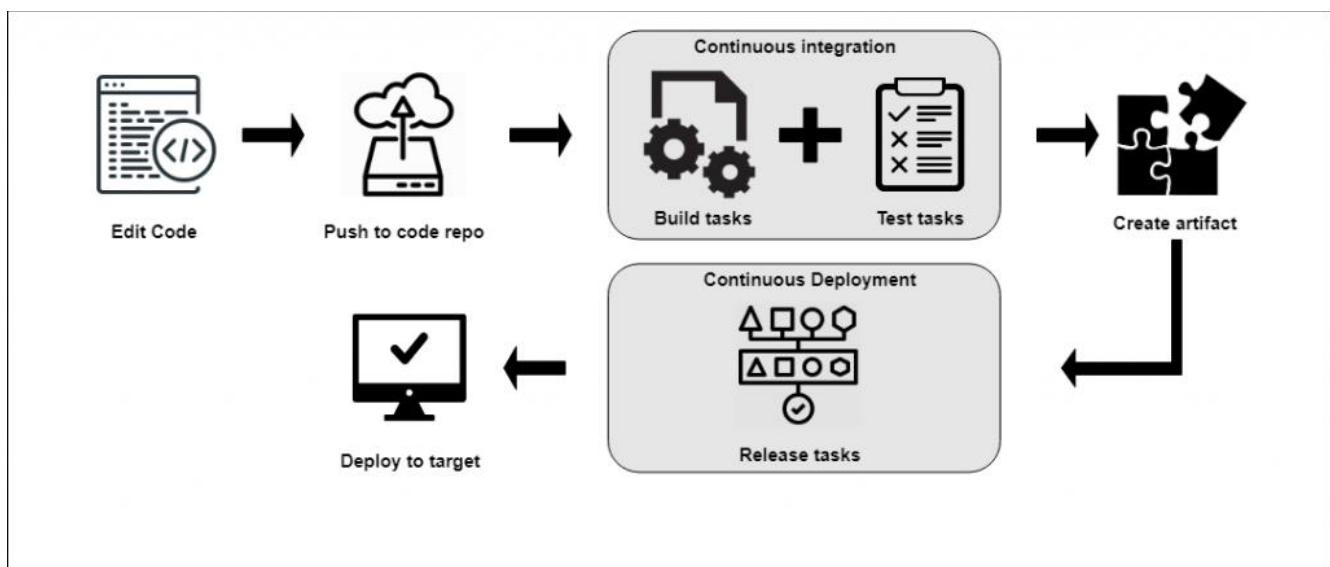
The pipeline depends on the versioning of the code, similar codes will have a similar branching structure. You can set triggers to start the process when any pull request is accepted. You can trigger the build process and identify issues in integration with your existing codebase.

azure-pipelines.yml can be used to add your pipeline in a YAML file with your application.

These are the high-level steps:

- Configure Azure Pipeline to use your repository.
- Edit the YAML file to specify your build.
- Push your code in a specific branch to trigger the process.

5. Define Pipelines Using Classic Interface



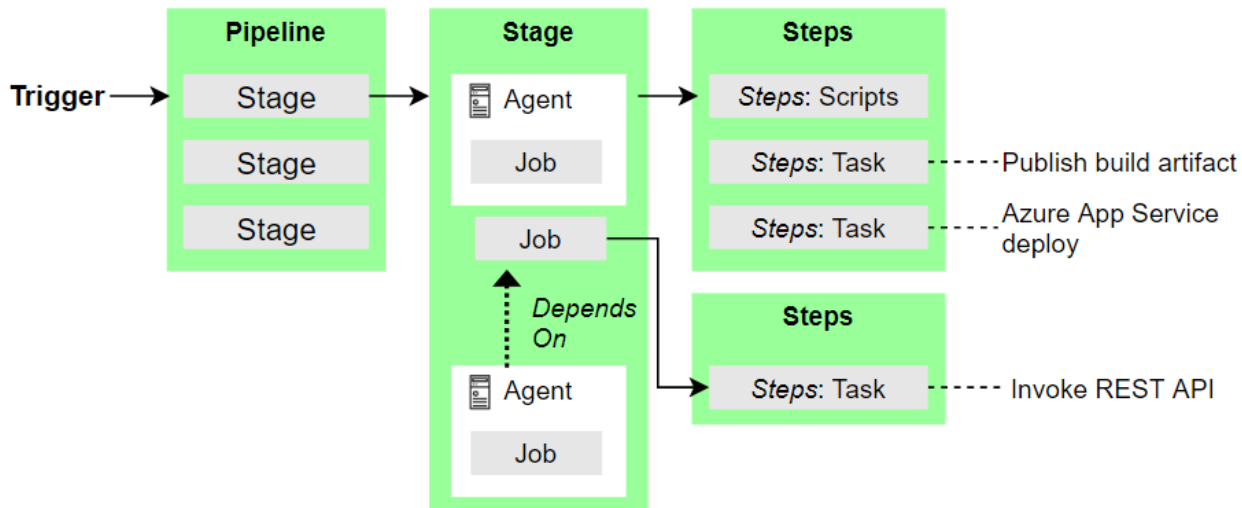
Users can define the pipeline in the Azure DevOps portal with the classic editor. They can also define a “Build Pipeline” for such activities as to build and test your code, and then publish artifacts. Additionally, they can also define the “Release Pipeline” to consume these artifacts and deploy them in targets.

These are the high-level steps:

- Configure Azure DevOps Pipeline to use your repository.
- Define build and release pipelines using the Azure DevOps web portal.
- Push your code to a specific branch to trigger the process.

Here, we saw high levels of steps to define pipelines in both ways. Let see a few terms to be used in Azure DevOps Pipelines before we jump to create a new pipeline.

6. Key Components of Azure DevOps Pipeline



- A trigger initiates an Azure DevOps Pipeline to run.
- A pipeline can have many stages. A pipeline can deploy to single or multiple environments.
- A stage can be specified to manage jobs in a pipeline and each stage has various jobs.
- Each job runs on one agent. It's also possible that a job doesn't have an agent.
- Each agent runs a job that may have various steps.
- A step can be anything like a script or task and it is the compact part of a pipeline.
- A task is a pre-bundled script that acts like to publish a build artifact or to call a REST API.
- A run publishes a bunch of files or bundles called an artifact.

Agents and Agent pools

An Agent is a computing architecture with installed agent software that runs a set of steps when a job attached to the agent runs. Rather than handling each agent separately, you group them into agent pools. For detailed information on this component check- Build and release agents.

Approvals

It specifies a set of validations needed before a deployment can be executed. It is used to manage deployments to production environments. Once checks are configured, the pipelines will pause and stop before starting the deployment process. They will only restart the process once all the uncheck options are checked and completed successfully. For any information, visit Define Approvals and checks.

Artifacts

Artifacts are a bundle of packages or files published by build pipelines. They let the team quickly collect the dependencies and other artifacts required for the application development. Also, you can utilize diverse artifacts like Build Artifacts, Maven, Nuget, PyPI, Universal Packages, etc. Learn more about Artifacts in Azure Pipelines.

Trigger

A trigger is considered as a setup for letting the pipeline know when to operate. Experts can configure a pipeline to work against a force to the repository or at scheduled times. Therefore, build triggers and release operations are perceived as triggers. For further information, explore build triggers and release triggers.

Deployment group

It is a series of target machines that have agents installed. Setting a deployment target for pipelines in machines is easy with the deployment group. See more about deployment groups.

Environment

When a bundle of different resources comes together and deploys the application then it is called an environment. It can be a collection of various resources such as virtual machines, containers, web apps or any services.

Job

The stage consists of many jobs, each job is run by an agent. Job defines an execution boundary for a set of steps to be performed.

Pipeline

The Pipeline is a part of the CI/CD process where there are numerous stages. You can imagine it as a workflow in a sequential form where you run tests, build and deploy your app.

Run

A run command is used to implement any function in a pipeline. It gathers the logs linked with the steps currently running and the results of the tests are currently implemented. In the course of a run, pipelines will first evaluate the pipeline and then transfer the run to various agents. Jobs will be executed by each agent. See more about the pipeline run sequence.

Script

A script can be a Bash, command line or PowerShell that runs code in your pipeline.

Stage

In a pipeline, there is a logical perimeter for monitoring performance which is called Stage.

Task

A task is a bundled script or method that has been engrossed with a pair of inputs.

7. Simple Steps For Implementing The CI/CD Process With One .NET Core Application

Let's deploy this application in 2 different environments:

- Azure App services
- On-premise /Self-Hosted server

Before we start with building a new project, please check below prerequisite points and make sure you have all things available:

- Microsoft Azure account with DevOps. You can start it free from [here](#)
- Visual Studio 2019
- Azure Repos
- On-premise server
- Internet connection

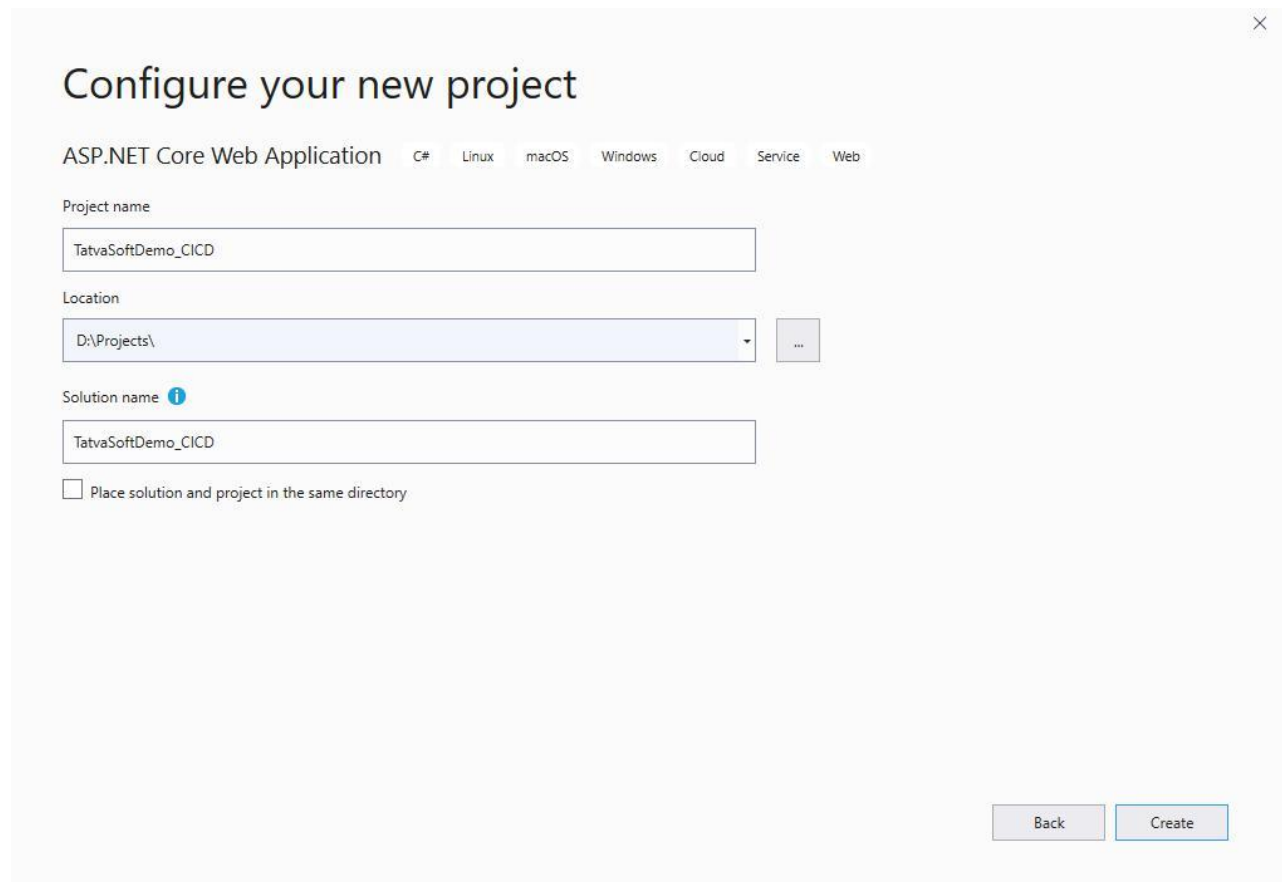
the whole process is divided into 4 parts:

1. Setup project in source control
2. Configure Azure CI/CD pipelines to deploy the application in Azure Services
3. Configure Release pipeline to deploy same application parallelly in on-premises/Self-Hosted server.
4. Run Azure DevOps Pipeline to deploy the application in 2 environments.

7.1 Setup Project in Source Control

Sign into the Microsoft Azure portal. I have used Azure Repos as source control for demo projects. Azure DevOps supports a vast range of source controls available in the market. You can use any of them.

1. Let's create a new .NET Core Project in Visual Studio 2019. You can also use your existing project. Here, I'm using a template project with basic coding for example and demo purposes.




Configure your new project

ASP.NET Core Web Application C# Linux macOS Windows Cloud Service Web

Project name
TatvaSoftDemo_CICD

Location
D:\Projects\

Solution name 
TatvaSoftDemo_CICD

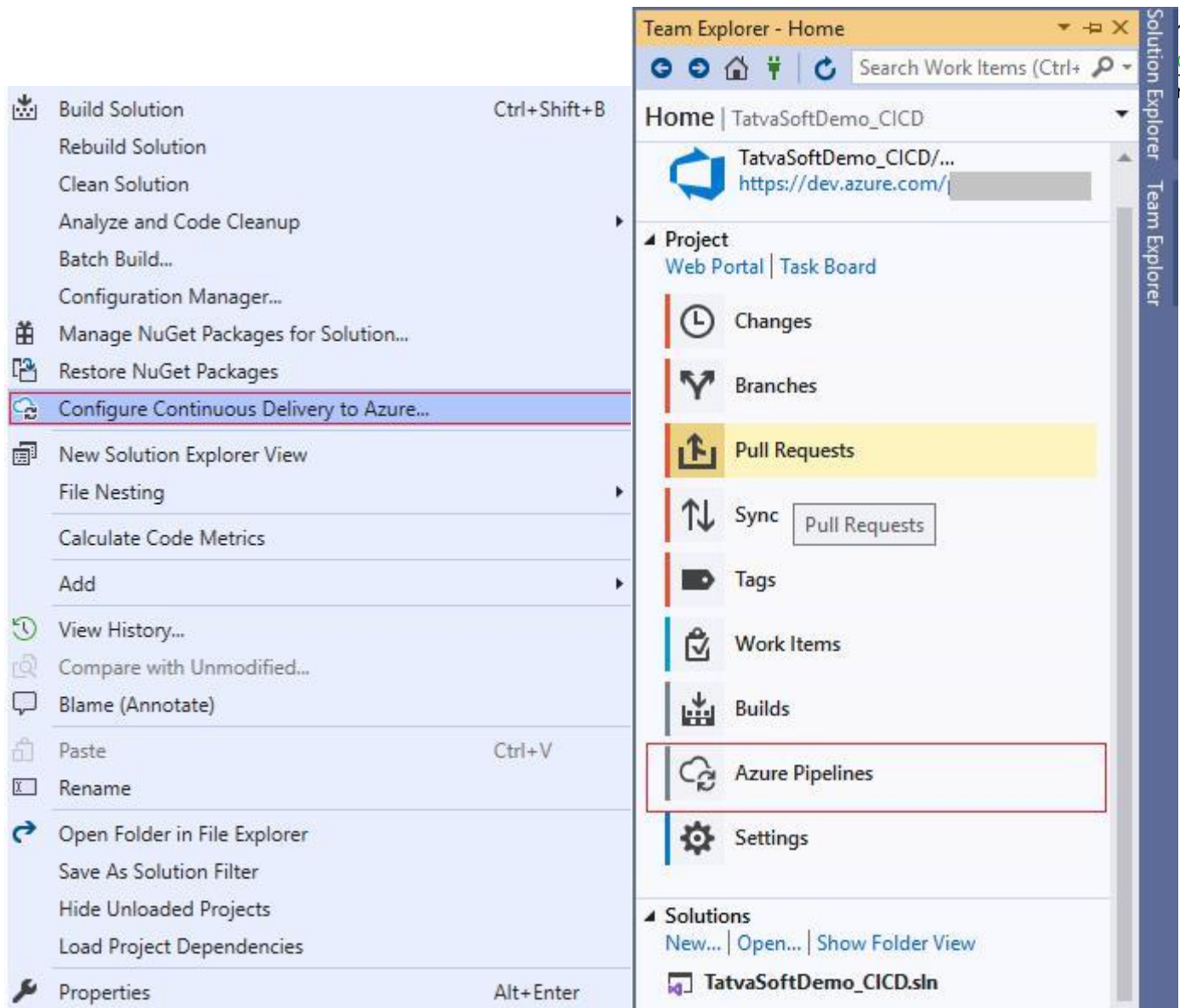
☐ Place solution and project in the same directory

Back Create

2. Commit the source code in Azure Repository that is already configured with Visual Studio. [Some steps are skipped to make this article smaller.]

7.2 Configure Azure CI/CD Pipelines to Deploy The Application in Azure Services

1. Visual studio 2019 provides a default feature for configuring CI-CD in any project. You can find this feature by right-clicking on the project solution file and selecting **Configure continues delivery to azure**. Other than this you can also find this feature under **Team Explorer** as images are given below.



2. By following previous steps, one popup for **Setup Azure Pipeline** will open in which you can select the appropriate branch and App Service.

Setup Azure Pipelines

Start a build and release now or whenever you check in code.

Select a Repository Branch

Repository: TatvaSoftDemo_CICD

Branch: master

Select Target Azure Resources

Subscription: Visual Studio Professional Subscription

App Service: TatvaSoftDemoCICD-dev-as (new) [Edit...](#)

Setting up Azure Pipelines will:

- Create an Azure App Service with the S1 Service Plan
- Create a build pipeline on Azure DevOps that targets the selected repository.
- Create a release pipeline targeting your selected Azure App Service

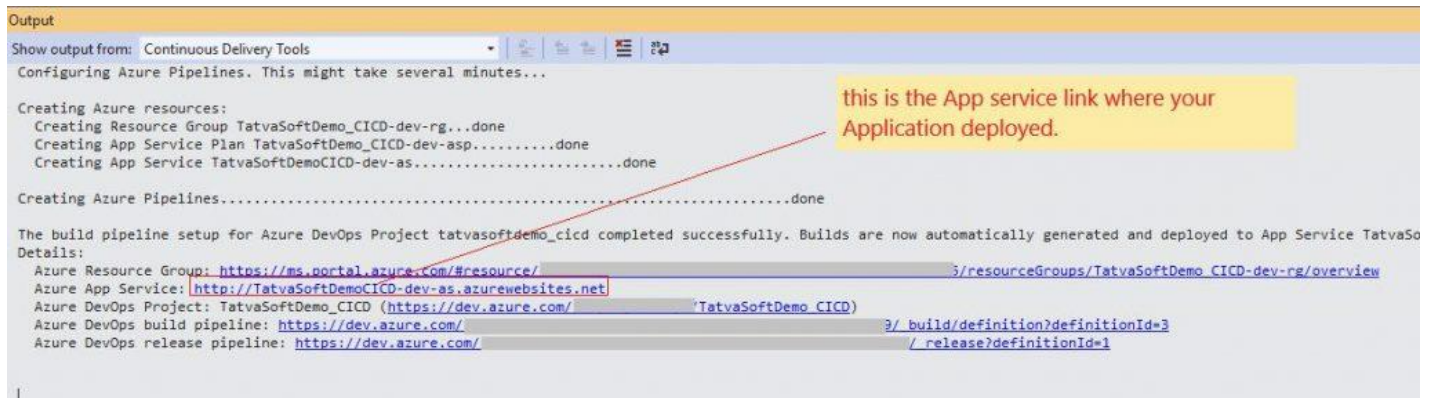
Azure Pipelines is a free service

[What is Azure Pipelines?](#)

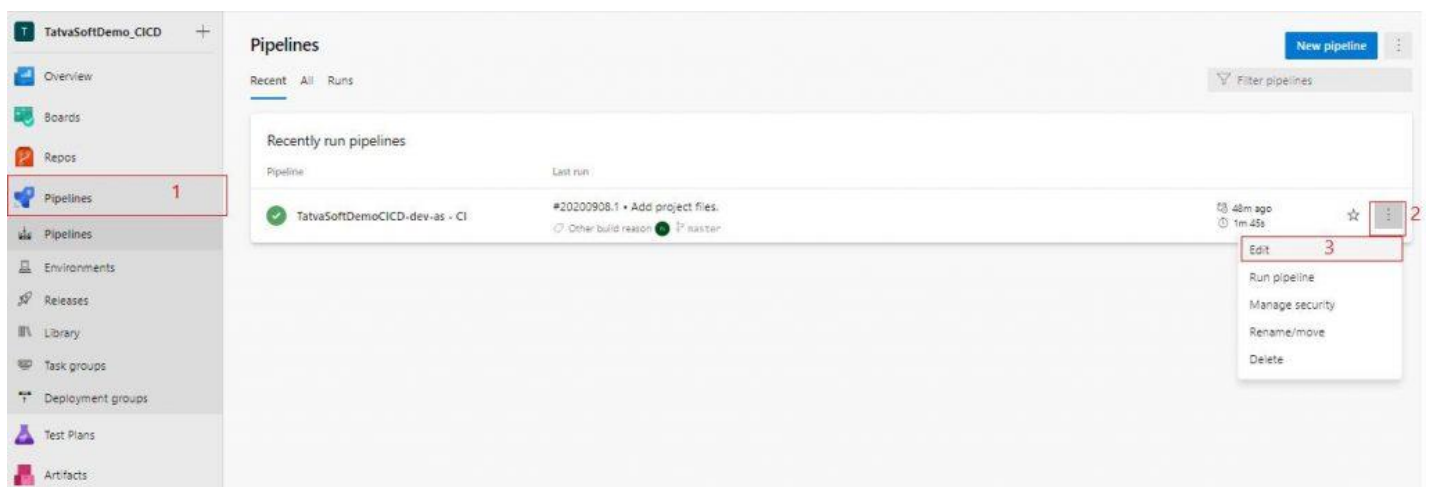
[Review Azure App Service pricing](#)

OK
Cancel

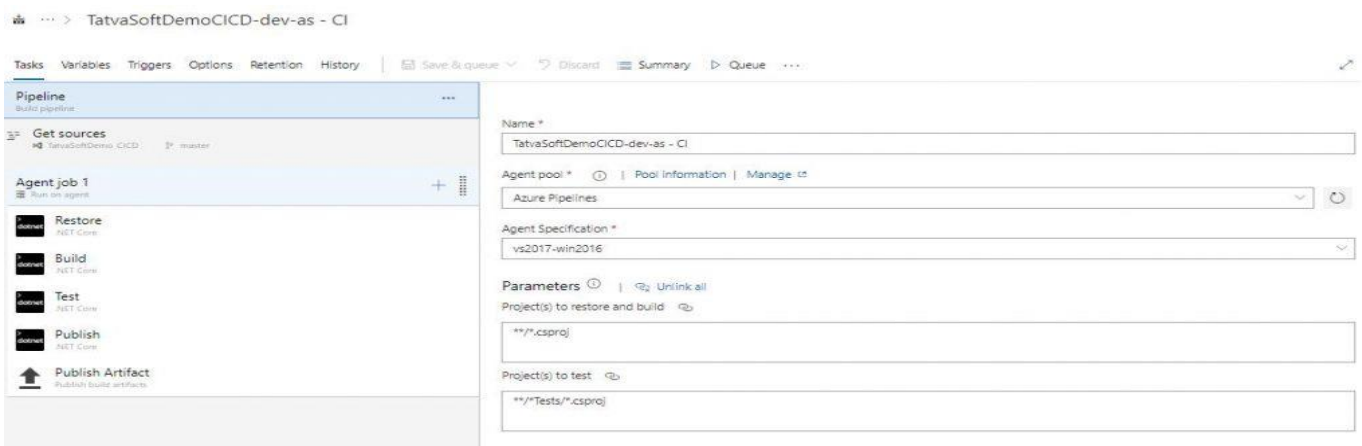
- You can also change the advanced setting for App service by clicking on the **Edit** button given besides an App service drop-down.
- After all necessary steps click the OK button, it will run some background processes to create Azure Pipeline.
- You can track this ongoing background process in the **Output** window as shown in the below image.



3. Open Azure DevOps portal and navigate to the Azure Pipelines section to see the newly created Azure pipeline by Visual Studio. Follow the steps shown as per below image to edit the pipeline.



- You can see the edit pipeline screen where you can edit the configuration of the build pipeline in the future as you wish.
- In the task tab, you will be able to add/edit tasks and agent jobs as per your requirement for your new project as per below image. These task tabs are created automatically as we have setup the pipeline from Visual Studio. You can create such tasks manually also.



- You can add/edit variables or variable groups to be used in various tasks in the Variables tab.

The screenshot shows the 'Variables' tab in Azure DevOps. The left sidebar has 'Pipeline variables' selected. The main area displays a table of variables:

Name	Value	Settable at queue time
BuildConfiguration	release	<input checked="" type="checkbox"/>
BuildPlatform	any cpu	<input checked="" type="checkbox"/>
system.collectionId		
system.debug	false	<input checked="" type="checkbox"/>
system.definitionId	3	
system.teamProject	TatvaSoftDemo_CICD	

Below the table is a '+ Add' button.

- In the Triggers tab, you can set the trigger to indicate when to run the build pipeline. You can also set specific branches from the CI process, path filters and schedules for the pipeline to run, etc.

The screenshot shows the 'Triggers' tab in Azure DevOps. The left sidebar has 'Continuous integration' selected. The main area shows the 'TatvaSoftDemo_CICD' trigger configuration:

- Enable continuous integration:** ☒
- Batch changes while a build is in progress:** ☒
- Branch filters:**
 - Type: Include
 - Branch specification: master
- Path filters:**
 - Type: Include
 - Path specification: /

4. Follow the below steps to see a newly created Release Pipeline (CD) by Azure DevOps Project and also edit as per your need.

The screenshot shows the 'Releases' view in Azure DevOps. The left sidebar has 'Releases' selected. The main area shows the 'TatvaSoftDemoCICD-dev-as - CD' release pipeline. A table lists the releases:

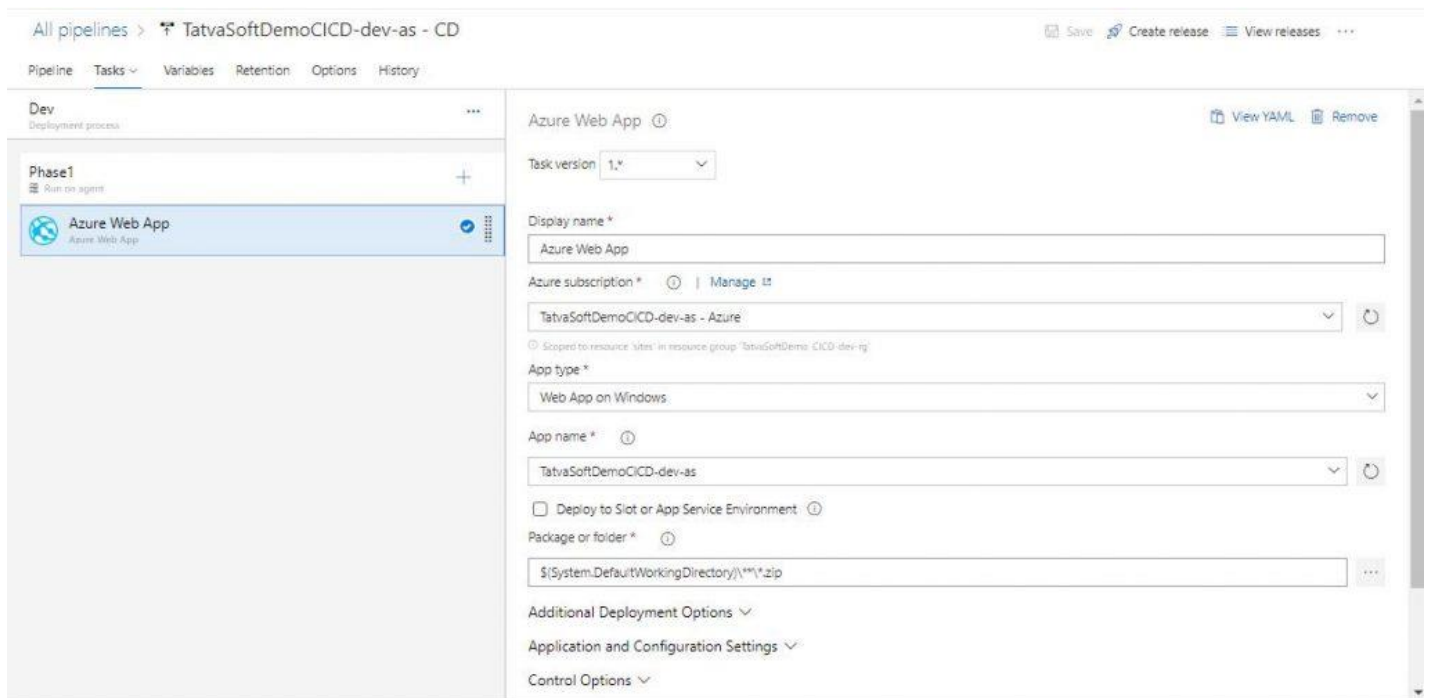
Release	Created	Stages
Release-1 20200908... [master]	9/8/2020, 1:59:54 PM	Dev

At the top right, there are buttons for 'Edit' and 'Create release'.

- After this in the next screen, follow the steps as per below screen to go to the edit task.



- Here you can add/edit tasks or phases in the release pipeline.



Now, we can deploy a .NET Core application in Azure App Services using the Azure CI/CD process. Let's see the steps to deploy the same application on a local server.

7.3 Configure Release Pipeline to Deploy Same Application Parallely in On-Premise/Self-Hosted Server

Before we start configuring the pipeline, you will have to perform a few tasks in your on-premise server to add it to the deployment group of Azure DevOps. And then you can add the same stage/phase in the release pipeline to deploy a build-in IIS server.

- Go to the deployment group in Azure DevOps. And click on **add a deployment group** to add a new one.



- Add an appropriate name and description then click on the **Create** button to create a new one.

Deployment groups > CICD_Demo*

Deployment group name

CICD_Demo

Description

Create

- It will open the screen to register your on-premises server, so select the type of machine which you are targeting, check the checkbox and click on the **Copy to Clipboard** button as suggested in the image. It will copy the power shell command in the clipboard.

Deployment groups > CICD_Demo

Details Targets Save Share Security Help

Deployment group name

CICD_Demo

Description

Deployment pool

TatvaSoftDemo-CICD-CICD_Demo Manage

Type of target to register:

Windows System prerequisites!!

Registration script (PowerShell)

```
$ErrorActionPreference="Stop";if(-NOT ([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole( [Security.Principal.WindowsBuiltInRole]
"Administrator")){ throw "Run command in an administrator PowerShell
prompt";if($PSVersionTable.PSVersion -lt (New-Object System.Version("3.0"))){ throw "The minimum
version of Windows 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:SystemDrive\azagent}; cd $env:SystemDrive\azagent; for($i=1; $i -lt 100; $i++){
($destFolder="AP+$i.ToString()).if(-NOT (Test-Path ($destFolder))){mkdir $destFolder;cd
$destFolder;break}}; $agentZip="$PWD\agent.zip";$defaultProxy=
[System.Net.WebRequest]::DefaultWebProxy;$securityProtocol=@();$securityProtocol+=
[Net.ServicePointManager]::SecurityProtocol;$securityProtocol+=[Net.SecurityProtocolType]::Tls12;
[Net.ServicePointManager]::SecurityProtocol=$securityProtocol;$webClient=New-Object Net.WebClient;
$uri="https://vstsagentpackage.azureedge.net/agent/2.174.1/vsts-agent-win-x64-
2.174.1.zip";if($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 "CICD_Demo" --agent $env:COMPUTERNAME --runasservice --work 'work' --url
'https://dev.azure.com/[organization]/[projectname]' --projectname 'TatvaSoftDemo-CICD'; Remove-Item $agentZip;
```

You can change the type of target to Windows or Linux

Make sure that this option is checked.

Use a personal access token in the script for authentication

Copy script to the clipboard

Run from an administrator PowerShell command prompt

- Open PowerShell console as an Administrator and paste the script copied in the previous step.
- Run the script and wait for some time to execute it completely.

```

PS C:\> $ErrorActionPreference="Stop";if(-NOT ([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole]"Administrator")){ throw "Run command in an administrator PowerShell prompt";};if($PSVersionTable.PSVersion -lt (New-Object System.Version(3.0))){ throw "The minimum version of Windows 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:SystemDrive\azagent}; cd $env:SystemDrive\azagent; for($i=1; $i -lt 100; $i++){($destFolder=$env:systemdrive\$i.ToString());if(-NOT (Test-Path ($destFolder))){mkdir $destFolder;cd $destFolder;break;}}; $agentZip=$PWD;agent.zip;$DefaultProxy=[System.Net.WebRequest]::DefaultWebProxy;$SecurityProtocol=[Net.ServicePointManager]::SecurityProtocol;$SecurityProtocol--[Net.SecurityProtocolType]::Tls12;[Net.ServicePointManager]::SecurityProtocol=$SecurityProtocol;$WebClient-New-Object Net.WebClient; $Uri=https://azagentopen.azureedge.net/agent/2.174.1/vsts-agent-win-x64-2.174.1.zip;if($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 "CICD_Demo" --agent $env:COMPUTERNAME --runasservice --work ".\work" --url "https://dev.azure.com/[redacted]"; Remove-Item $agentZip;

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
d-----          08/09/2020         17:44         azagent

Directory: C:\azagent

Mode                LastWriteTime         Length Name
----                -
d-----          08/09/2020         17:44             A1

Azure Pipelines
agent v2.174.1 (commit eaf1360)
  
```

```

PS C:\> $ErrorActionPreference="Stop";if(-NOT ([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole]"Administrator")){ throw "Run command in an administrator PowerShell prompt";};if($PSVersionTable.PSVersion -lt (New-Object System.Version(3.0))){ throw "The minimum version of Windows 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:SystemDrive\azagent}; cd $env:SystemDrive\azagent; for($i=1; $i -lt 100; $i++){($destFolder=$env:systemdrive\$i.ToString());if(-NOT (Test-Path ($destFolder))){mkdir $destFolder;cd $destFolder;break;}}; $agentZip=$PWD;agent.zip;$DefaultProxy=[System.Net.WebRequest]::DefaultWebProxy;$SecurityProtocol=[Net.ServicePointManager]::SecurityProtocol;$SecurityProtocol--[Net.SecurityProtocolType]::Tls12;[Net.ServicePointManager]::SecurityProtocol=$SecurityProtocol;$WebClient-New-Object Net.WebClient; $Uri=https://azagentopen.azureedge.net/agent/2.174.1/vsts-agent-win-x64-2.174.1.zip;if($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 "CICD_Demo" --agent $env:COMPUTERNAME --runasservice --work ".\work" --url "https://dev.azure.com/[redacted]"; Remove-Item $agentZip;

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
d-----          08/09/2020         17:44         azagent

Directory: C:\azagent

Mode                LastWriteTime         Length Name
----                -
d-----          08/09/2020         17:44             A1

Azure Pipelines
agent v2.174.1 (commit eaf1360)
  
```

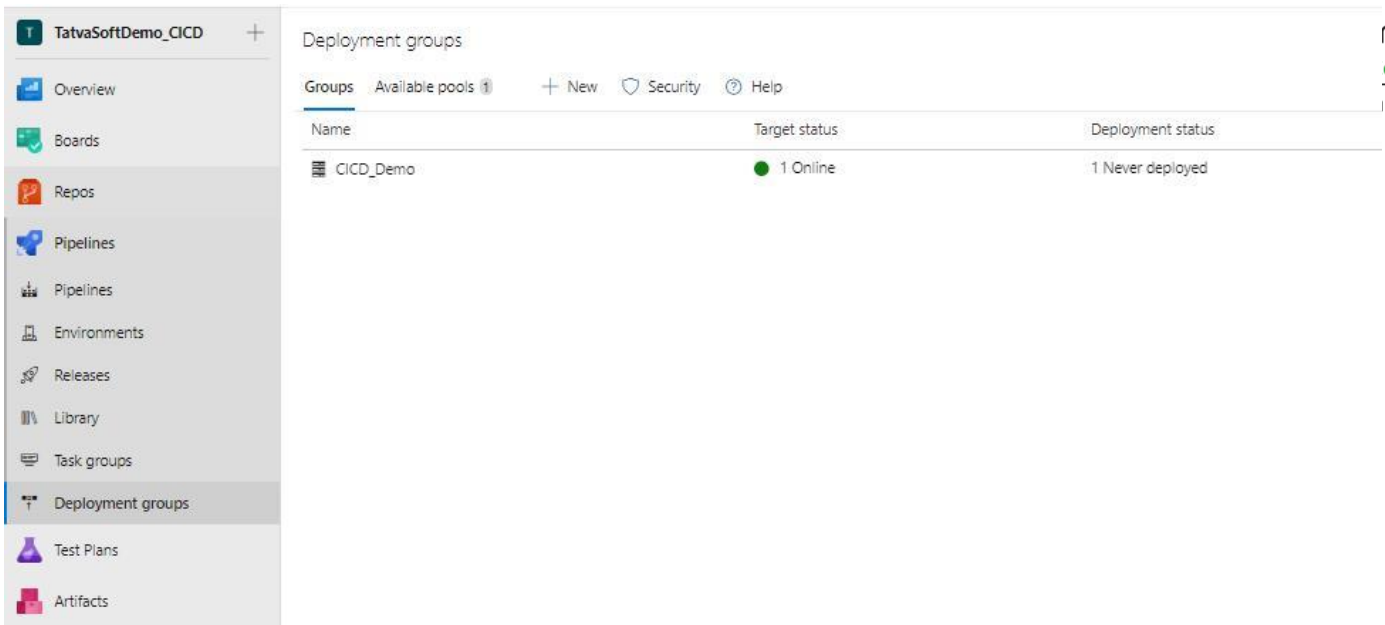
```

>> Connect:
Connecting to server ...

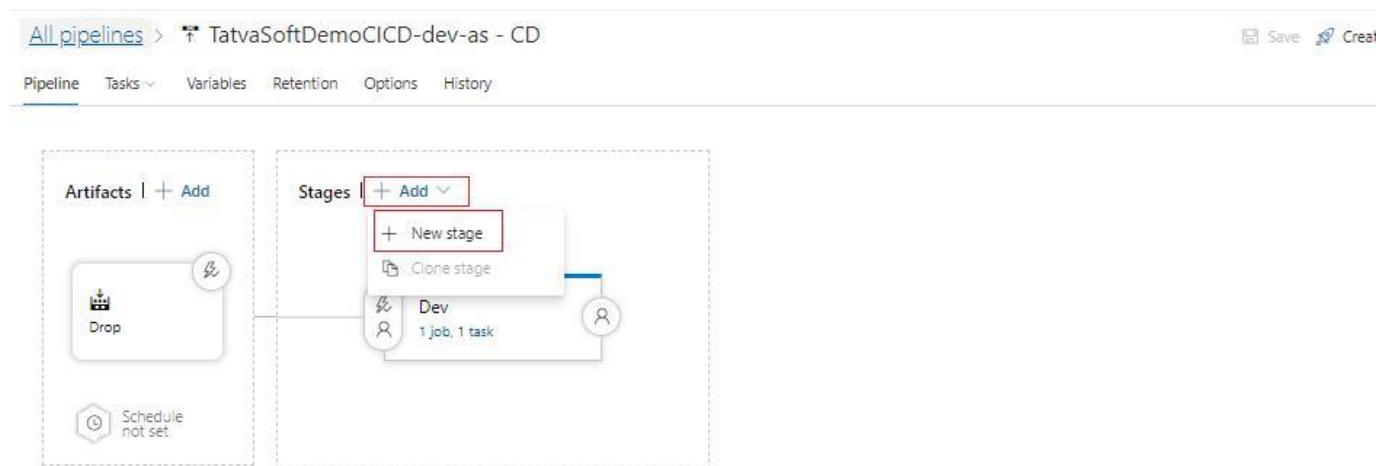
>> Register Agent:
Scanning for tool capabilities.
Connecting to the server.
Enter deployment group tags for agent? (Y/N) (press enter for N) >
Successfully added the agent
Testing agent connection.
2020-09-08 12:18:36Z: Settings Saved.
Enter User account to use for the service (press enter for NT AUTHORITY\SYSTEM) >
Granting file permissions to 'NT AUTHORITY\SYSTEM'.
Service vstsagent. [redacted] .TatvaSoftDemo_CICD-CICD_Demo.PCA186 successfully installed
Service vstsagent. [redacted] .TatvaSoftDemo_CICD-CICD_Demo.PCA186 successfully set recovery option
Service vstsagent. [redacted] .TatvaSoftDemo_CICD-CICD_Demo.PCA186 successfully set to delayed auto start
Service vstsagent. [redacted] .TatvaSoftDemo_CICD-CICD_Demo.PCA186 successfully configured
Service vstsagent. [redacted] .TatvaSoftDemo_CICD-CICD_Demo.PCA186 started successfully

PS C:\azagent\A1>
  
```

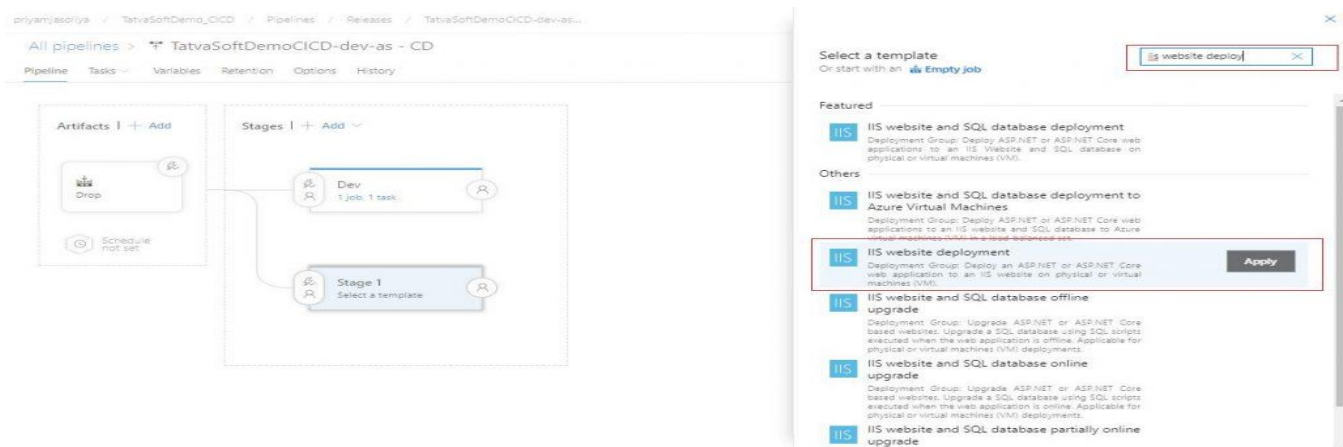
- So now, it's registered successfully. You can verify it in the Azure DevOps deployment group section where you can find the target status is **Online**. See the below image for reference.



2. Now, add a new stage in the release pipeline to deploy the build in this server. For that go back to the release pipeline section and click on the **Edit** button to edit a pipeline. Then follow the steps shown as per image to add a new stage in the pipeline.



- Select the template for **IIS Website Deployment** as we are going to publish build on IIS server of self-hosted server.



- After this, it will open a new popup, give the appropriate name to the newly created stage.

All pipelines > TatvaSoftDemoCICD-dev-as - CD
Save Create release View releases

Pipeline
Tasks Variables Retention Options History

Artifacts | + Add

Drop

Schedule not set

Stages | + Add

Dev
1 job, 1 task

SelfHosted-pca186
1 job, 2 tasks

Stage

SelfHosted-pca186

Properties

Name and owners of the stage

Stage name

SelfHosted-pca186

Stage owner

[User Avatar]

- Go to the **Tasks** tab to edit the template as shown in the image. Change the bindings and website name as you require.

All pipelines > TatvaSoftDemoCICD-dev-as - CD
Save Create release View releases

Pipeline
Tasks Variables Retention Options History

SelfHosted-pca186

Deployment process

IIS Deployment

Some settings need attention

IIS Web App Manage
IIS web app manage

IIS Web App Deploy
IIS web app deploy

Stage name

SelfHosted-pca186

Parameters | Unlink all

Configuration type *

IIS Website

Action *

Create Or Update

Website name *

Default Web Site

☒ Add binding

Add bindings *

http://All Unassigned:80:

This field is linked to 1 setting in 'IIS Web App Manage'

Click here to add new bindings, where you can add IP and Domain name.

- Edit the **IIS Deployment** agent, here you have to set the deployment group to what we have created in previous steps.

All pipelines > TatvaSoftDemoCICD-dev-as - CD

Pipeline Tasks Variables Retention Options History

Selfhosted-pca186

IIS Deployment Run on deployment group

IIS Web App Manage IIS web app manage

IIS Web App Deploy IIS web app deploy

Deployment group job

Display name * IIS Deployment

Deployment targets

Deployment group * CICD_Demo

Required tags

1 matching targets in CICD_Demo deployment group.

Targets to deploy to in parallel

☒ Multiple ☐ One target at a time

Maximum number of targets in parallel 100% targets (1)

Timeout * 0

Job cancel timeout * 1

Artifact download

Drop Latest Selected all artifacts

Additional options

☐ Allow scripts to access the OAuth token

Run this job

Only when all previous jobs have succeeded

- Go to task **IIS Web App Manage**, change the physical path where you want to deploy your build. Fields that are not editable are already set in previous steps.

All pipelines > TatvaSoftDemoCICD-dev-as - CD

Pipeline Tasks Variables Retention Options History

Selfhosted-pca186

IIS Deployment Run on deployment group

IIS Web App Manage IIS web app manage

IIS Web App Deploy IIS web app deploy

Deployment group job

Display name * IIS Deployment

Deployment targets

Deployment group * CICD_Demo

Required tags

1 matching targets in CICD_Demo deployment group.

Targets to deploy to in parallel

☒ Multiple ☐ One target at a time

Maximum number of targets in parallel 100% targets (1)

Timeout * 0

Job cancel timeout * 1

Artifact download

Drop Latest Selected all artifacts

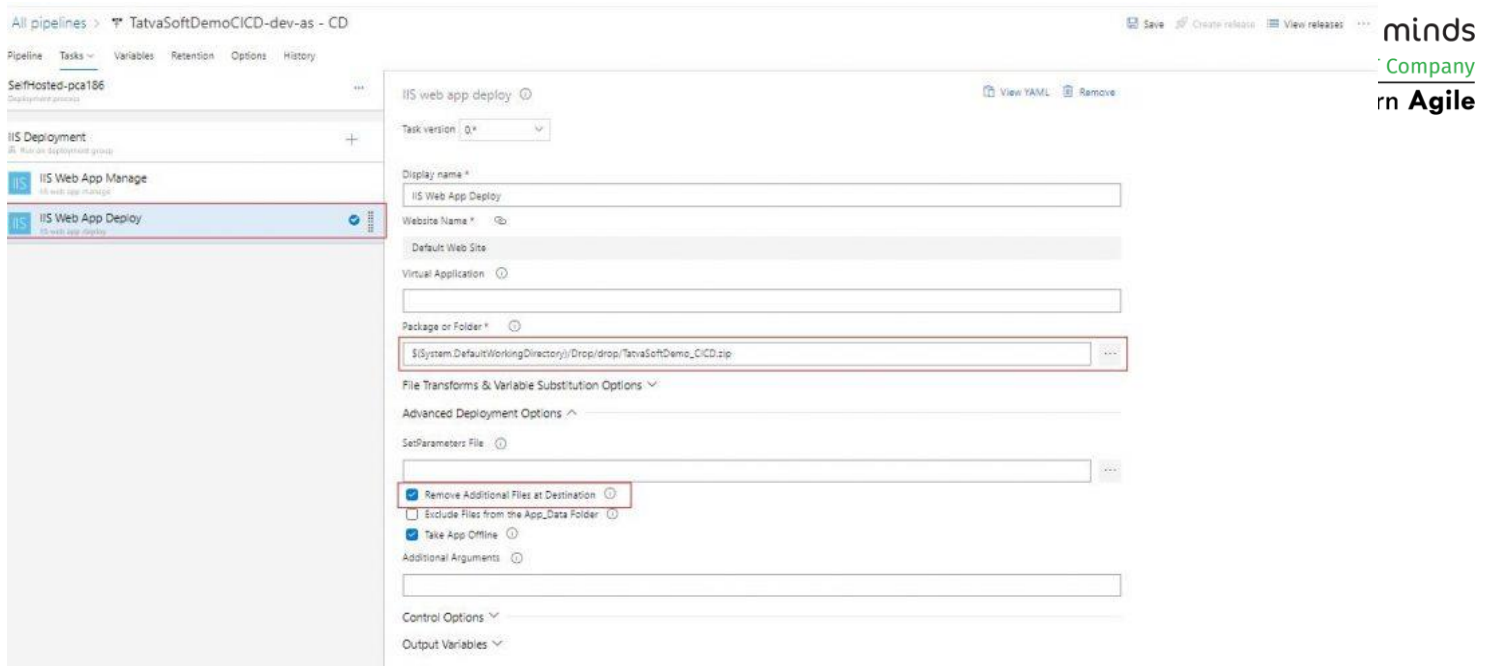
Additional options

☐ Allow scripts to access the OAuth token

Run this job

Only when all previous jobs have succeeded

- Select task **IIS Web App Deploy**, browse the path where the build is placed and set it to **Package or folder** field, check the option of **Remove additional files at destinations**.

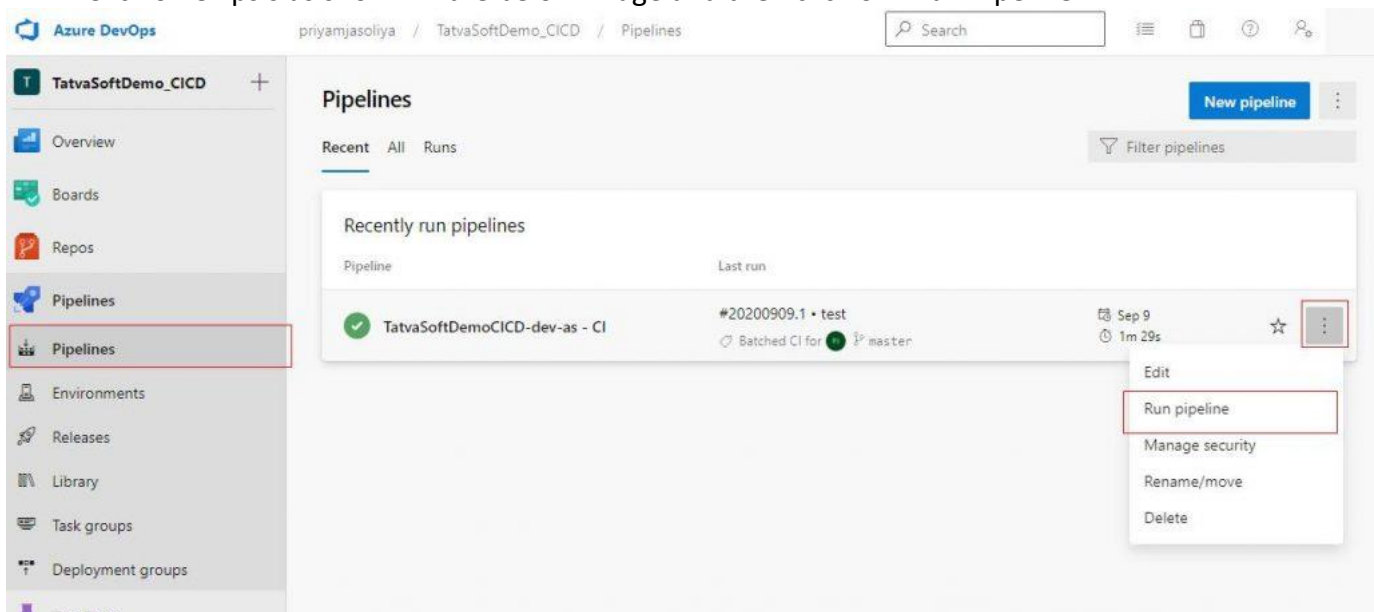


After following all the above steps, don't forget to click on the **Save** button. That's it, you are all set to deploy applications on a self-hosted server using Azure Pipelines.

Let see how you can run your pipeline and where you can see the status of these pipelines in the Azure DevOps portal.

7.4 Run Azure Pipeline to Deploy the Application in 2 Environments

1. You can set up your Pipeline to be run when any changes are committed in a particular branch in the repository as shown earlier in this article.
2. You can also manually run this pipeline from the Azure DevOps portal. Follow the below steps to run it.
 - Open Azure DevOps Portal, navigate to your project.
 - Click on "Pipelines" to open the window.
 - Find your pipeline from Recent or All.
 - Click on ellipsis as shown in the below image and then click on "Run Pipeline".



- Click on the title of your Pipeline to open it where you will see all your runs in descending order latest run on top.

The screenshot shows the Azure DevOps interface. On the left is a sidebar with navigation options: Overview, Boards, Repos, Pipelines (selected), Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area displays the 'TatvaSoftDemoCICD-dev-as - CI' pipeline. Below the pipeline name are tabs for 'Runs', 'Branches', and 'Analytics'. The 'Runs' tab is active, showing a table of pipeline runs. The table has two columns: 'Description' and 'Stages'. Two runs are listed, both with a green checkmark in the 'Stages' column, indicating success.

Description	Stages
#20200909.1 test Batched CI for master 2711daa	✓
#20200908.1 Add project files. Other build reason master 234c7b4	✓

- Select any of the runs to see details of that run.

The screenshot shows the detailed view of a pipeline run. At the top, there's a header with a green checkmark, the run ID '#20200909.1 test', and the pipeline name 'on TatvaSoftDemoCICD-dev-as - CI'. A 'Run new' button is on the right. Below the header are tabs for 'Summary', 'Releases', and 'Code Coverage'. The 'Summary' tab is active, showing a 'Rolling build triggered' message. Below this is a table with build details. The 'Warnings' section shows one warning: 'Project file(s) matching the specified pattern were not found.' The 'Jobs' section shows a table with one job: 'Build', which is successful and took 1m 22s.

Repository and version	Time started and elapsed	Related	Tests and coverage
TatvaSoftDemo_CICD master 2711daa	Sep 9 at 12:40 PM 1m 29s	0 work items 1 published; 1 consumed	Get started

Name	Status	Duration
Build	Success	1m 22s

- In the Releases tab, you will find a release pipeline run for this instance.

#20200909.1 test

on TatvaSoftDemoCICD-dev-as - CI
Retained by release

Run new

inds
mpany
Agile

Summary
Releases
Code Coverage

Name

TatvaSoftDemoCICD-dev-as - CD/Release-2

Dev

- Further going deeper by clicking on the release title, you will finally reach your release summary, where you can see all executed jobs. You can also find errors if they occurred during deployment.

TatvaSoftDemoCICD-dev-as - CD > Release-2 > Dev
 Succeeded

Pipeline
Tasks
Variables
Logs
Tests

Deploy
Cancel
Refresh
Download all logs
Edit

Deployment process
Succeeded

Phase1
Succeeded

Phase1
Started: 9/9/2020, 12:43:52 PM
Pool: Azure Pipelines · Agent: Hosted Agent ... 54s

	Initialize job · succeeded	20s
	Download artifact - Drop - drop · succeeded	18s
	Azure Web App · succeeded	14s
	Finalize Job · succeeded	<1s

- Clicking on each job will get you a console to see all steps executed.

You have seen simple steps for deploying an application in 2 different environments. There are many other factors involved that I haven't covered, and you will find when you deploy an application in a production environment.