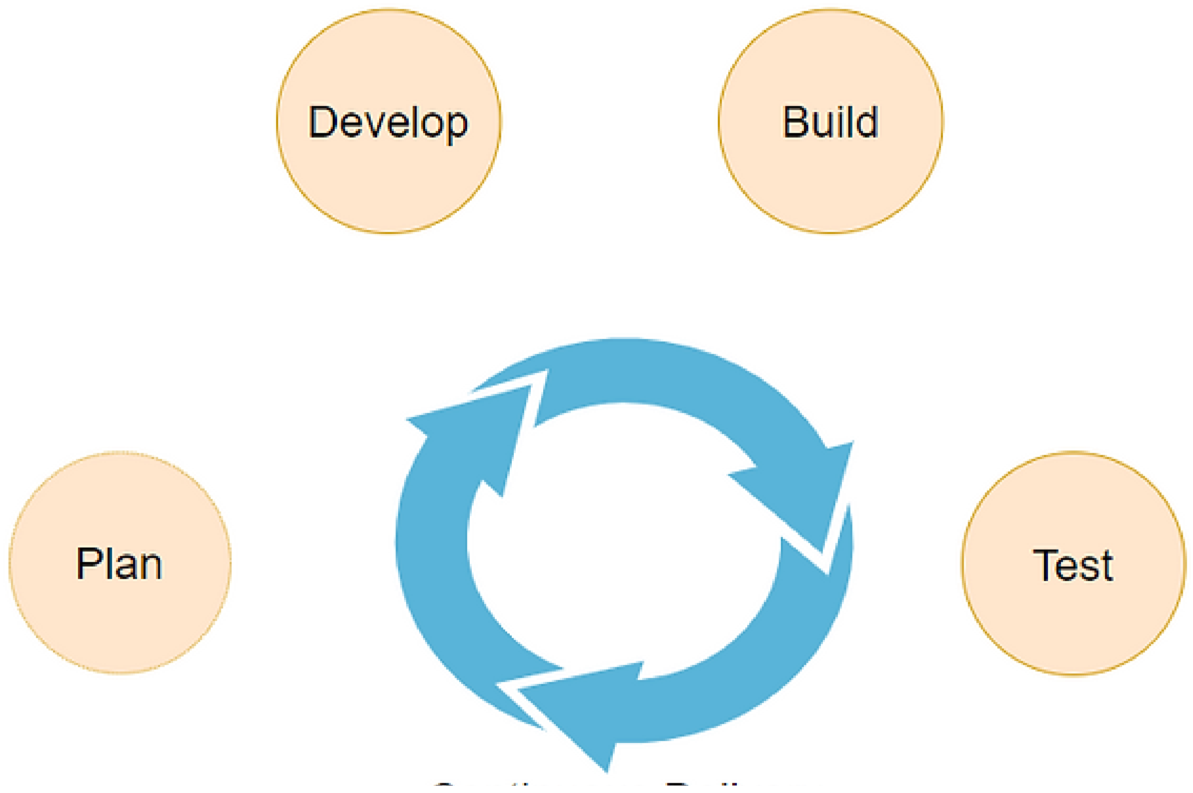
LetsDevOps: Introduction to Azure DevOps for Beginners - Create CI/CD Pipelines, Setup Repository

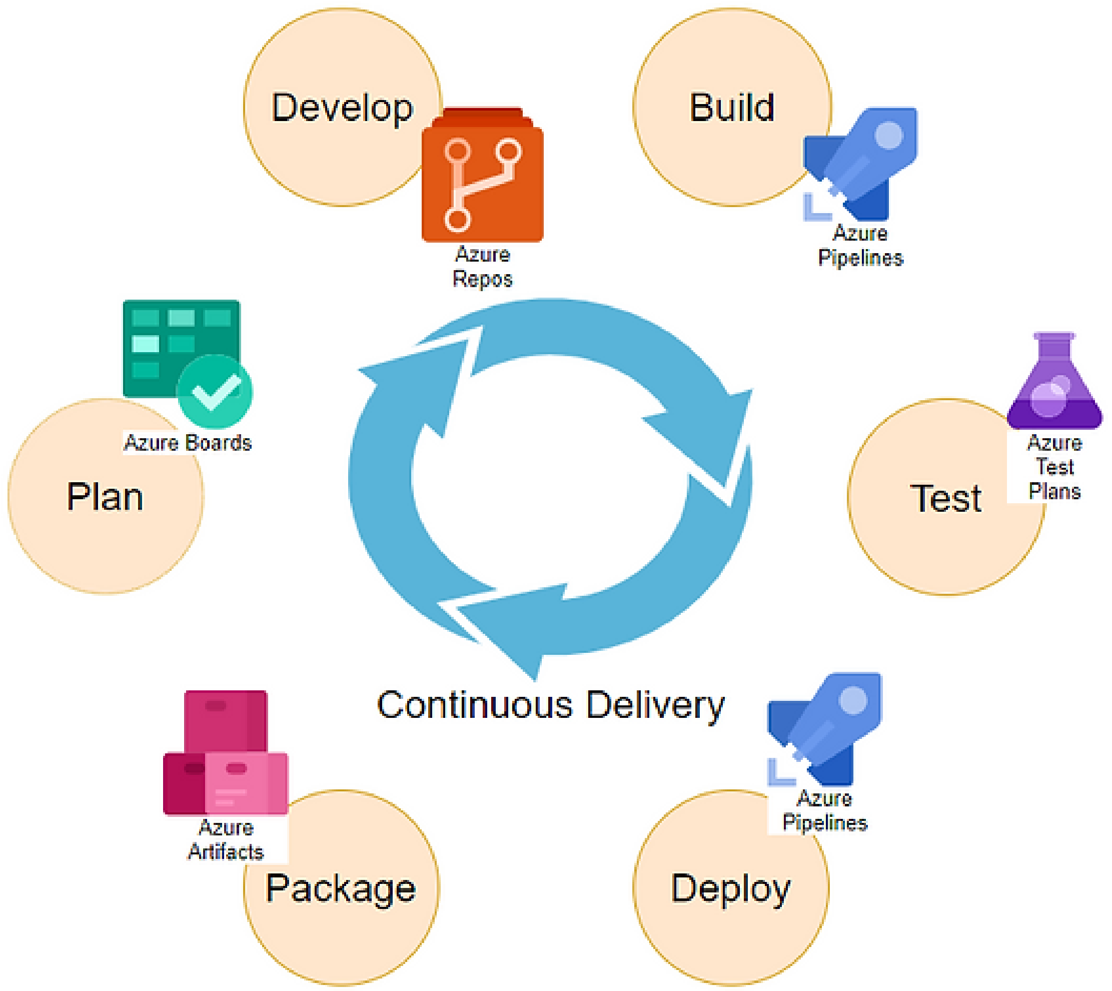
Introduction

DevOps is a process which helps to enable continuous delivery of value to our end users. Delivery of Software is a process of Planning, Developing, Build, Test, Deploy and Retrospective. Azure DevOps gives all these capabilities to achieve the Continuous Delivery under a single platform.



Azure DevOps Components

1. Azure Boards
2. Azure Repos
3. Azure Pipelines
4. Azure Test Plans
5. Azure Artifact



1. Azure Boards

This component helps to plan your Backlog, Sprint and track it across the team.

Azure Boards

2. Azure Repos

This helps to manage the Code Repository with unlimited private Git Repo. It can be integrated with Centralized [TFS] as well as decentralized [Git] repository with Source Code Management

Azure
Repos

3. Azure Pipelines

This is very important feature for the DevOps practice. Azure Pipelines helps to setup the **[CI/CD] Continuous Integration** and **Continuous Deployment**. The beauty of this component is that it can be used for any platform like Windows, Linux, Mac OS as well as with any Cloud provider.

Azure
Pipelines

4. Azure Test Plans

This component helps to integrate the TestPlans for your application to validate the changes before shipping to the Customer.

Azure
Test
Plans

5. Azure Artifact

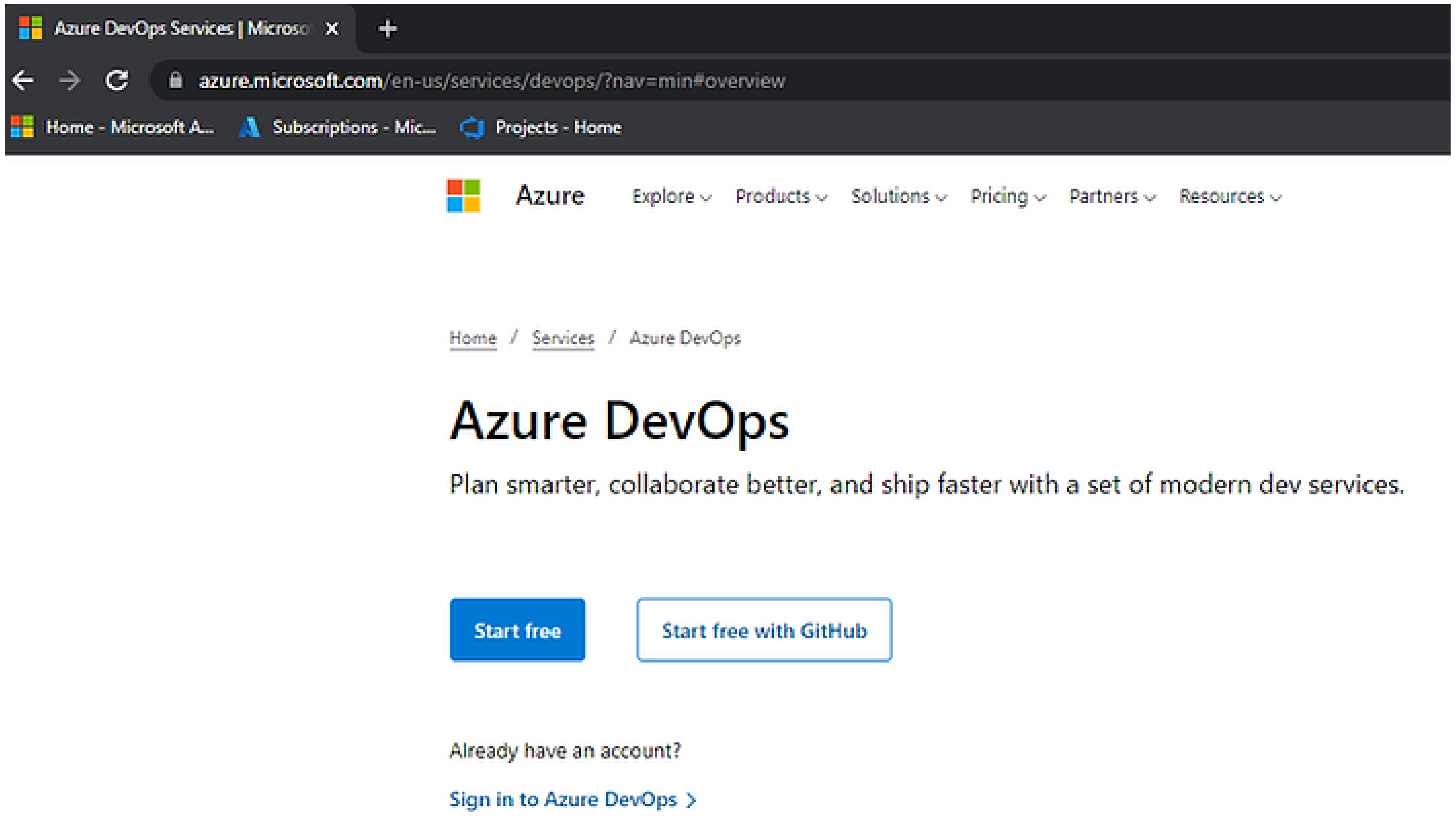
Azure Artifact helps to create package which can be shared by different Teams also this can be integrated with CI/CD pipeline.

Azure
Artifacts

Account Setup

If you do not have Azure DevOps setup you can start for free but you need to sign up using any valid account. Follow below link and start for Free.

https://azure.microsoft.com/en-us/services/devops/?nav=min


Azure DevOps Services | Microso
azure.microsoft.com/en-us/services/devops/?nav=min#overview
Home - Microsoft A...
Subscriptions - Mic...
Projects - Home
Azure
Explore
Products
Solutions
Pricing
Partners
Resources
Home / Services / Azure DevOps

Azure DevOps

Plan smarter, collaborate better, and ship faster with a set of modern dev services.

Start free
Start free with GitHub

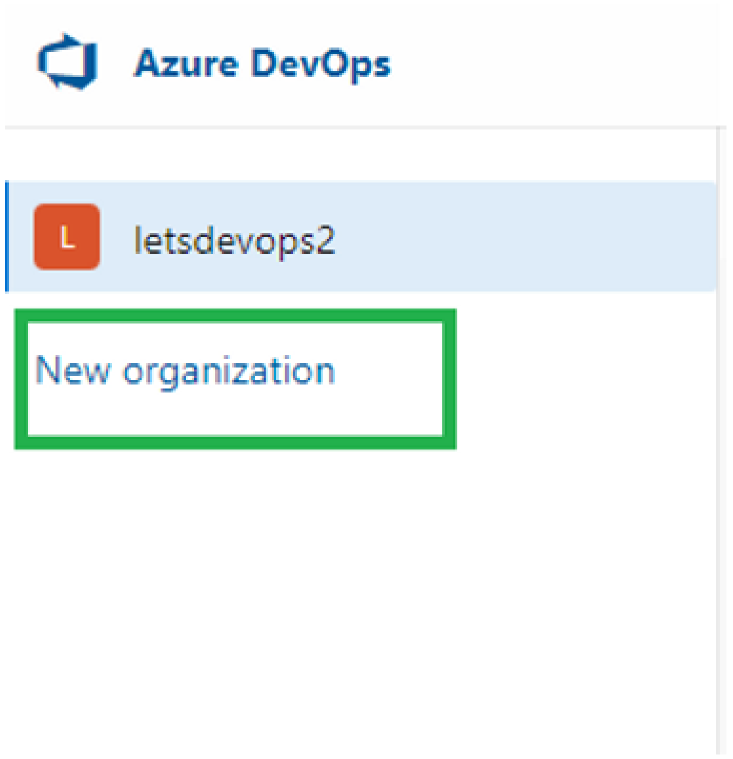
Already have an account?

Sign in to Azure DevOps >

Organization.

Organization is collection of related projects. Also it helps to organize the different Projects.

Once you are signed up there will be Organization created. You can create Multiple Organization with **Create New Organization.**



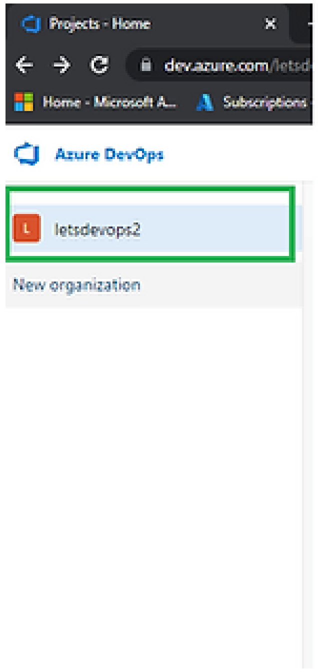
Project

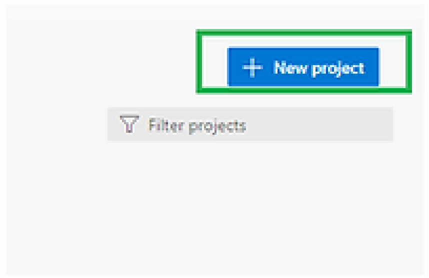
Project helps to build the Software Solution which includes Planning, Tracking, Source Code Management, Setting Up CI/CD, Releasing Artifact. Once you create Project team will be created Automatically.

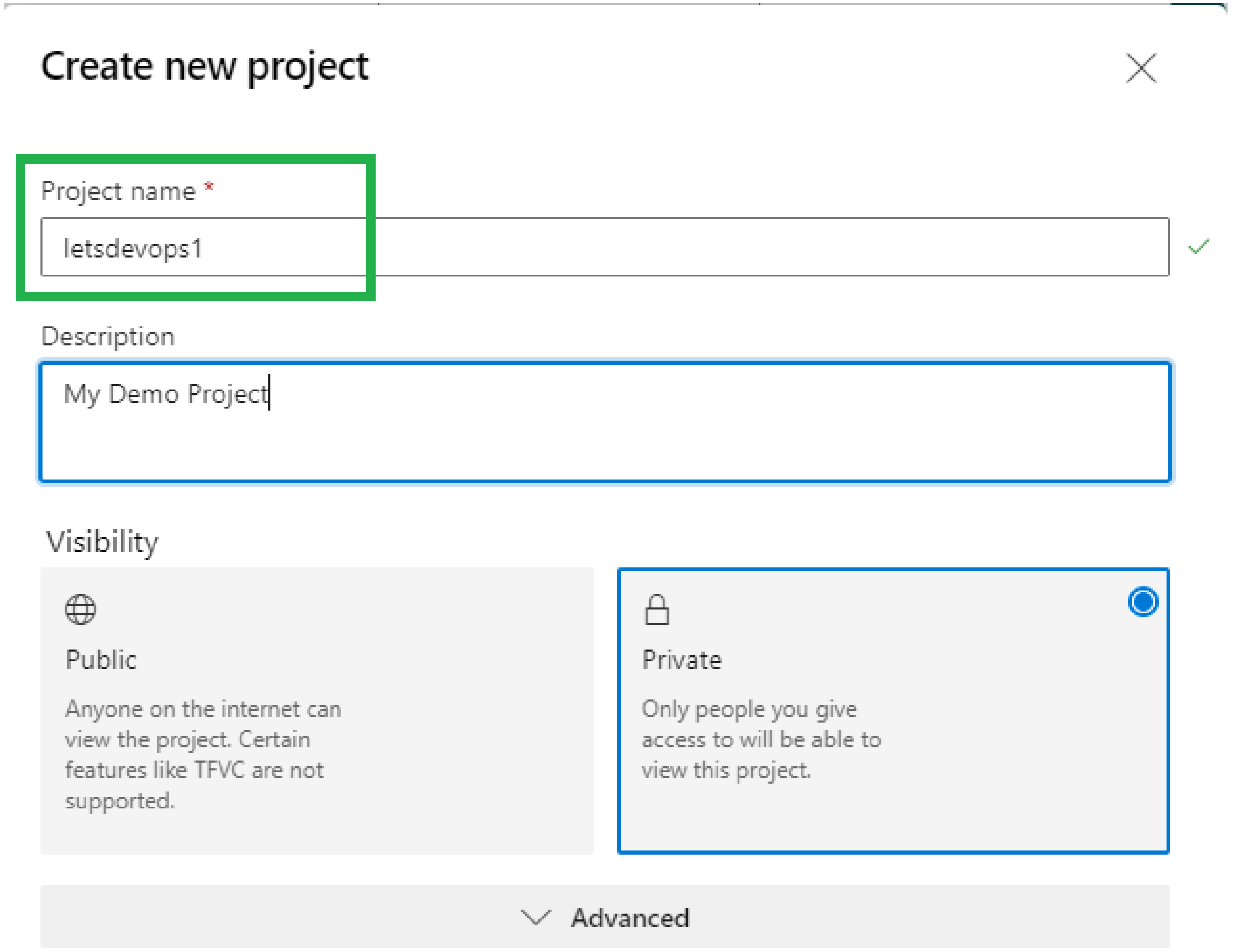
Under the Organization you can create Project.

Create Project

Organization --> New Project --> Create







Create new project

Project name *

letsdevops1

Description

My Demo Project

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.

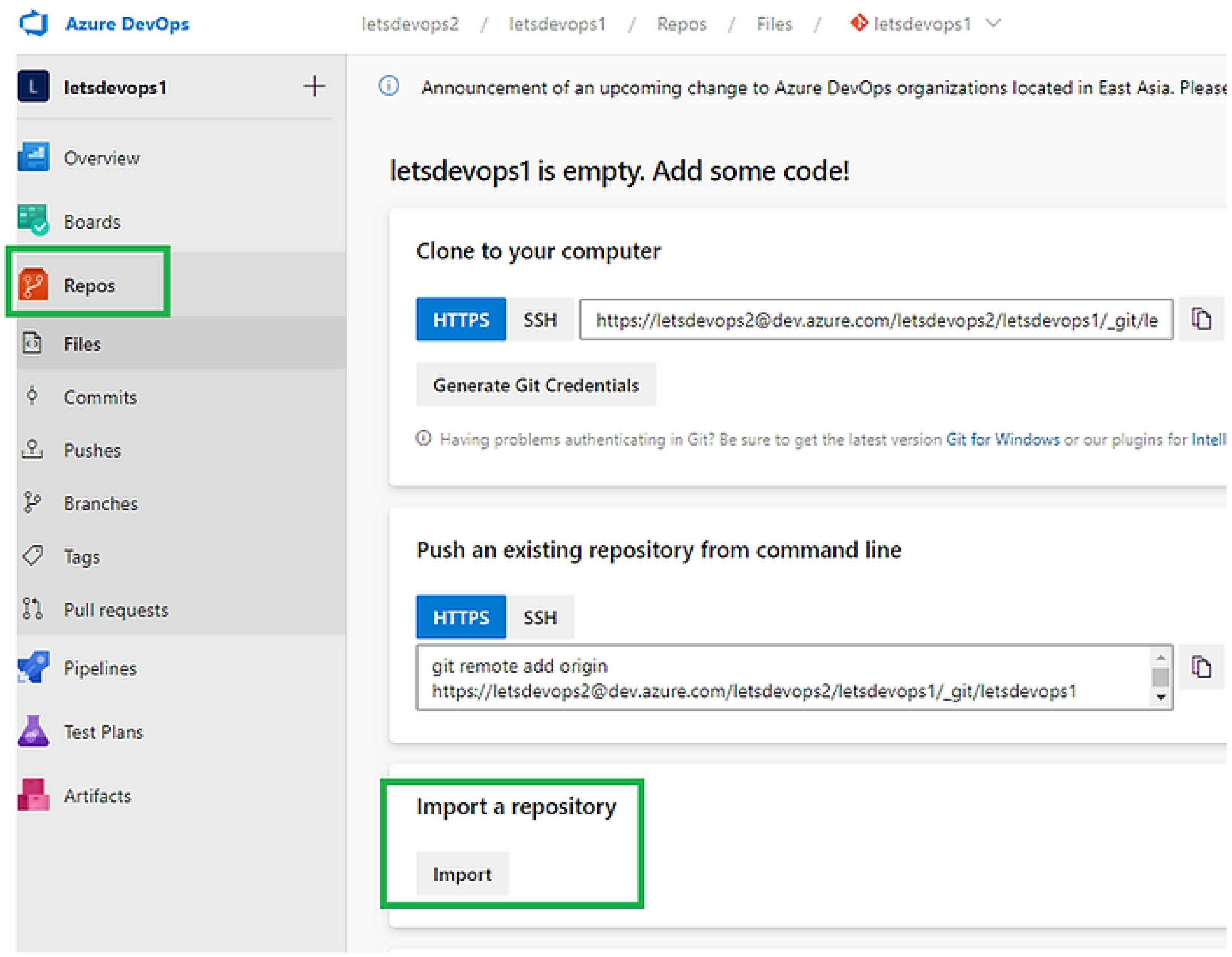
Azure Repos

Azure Repos helps to create and Manage Source Code which further will be used to Build and Deploy.

For example we can import any sample Project to setup the CI/CD. Use below git repo to import.

Git Repo: https://github.com/sumitraj0103/Letsdevops.git

Step 1: I will select import Repository to import the Source Code.



Announcement of an upcoming change to Azure DevOps organizations located in East Asia. Please

letsdevops1 is empty. Add some code!

Clone to your computer

HTTPS SSH https://letsdevops2@dev.azure.com/letsdevops2/letsdevops1/_git/le

Generate Git Credentials

Having problems authenticating in Git? Be sure to get the latest version Git for Windows or our plugins for Intell

Push an existing repository from command line

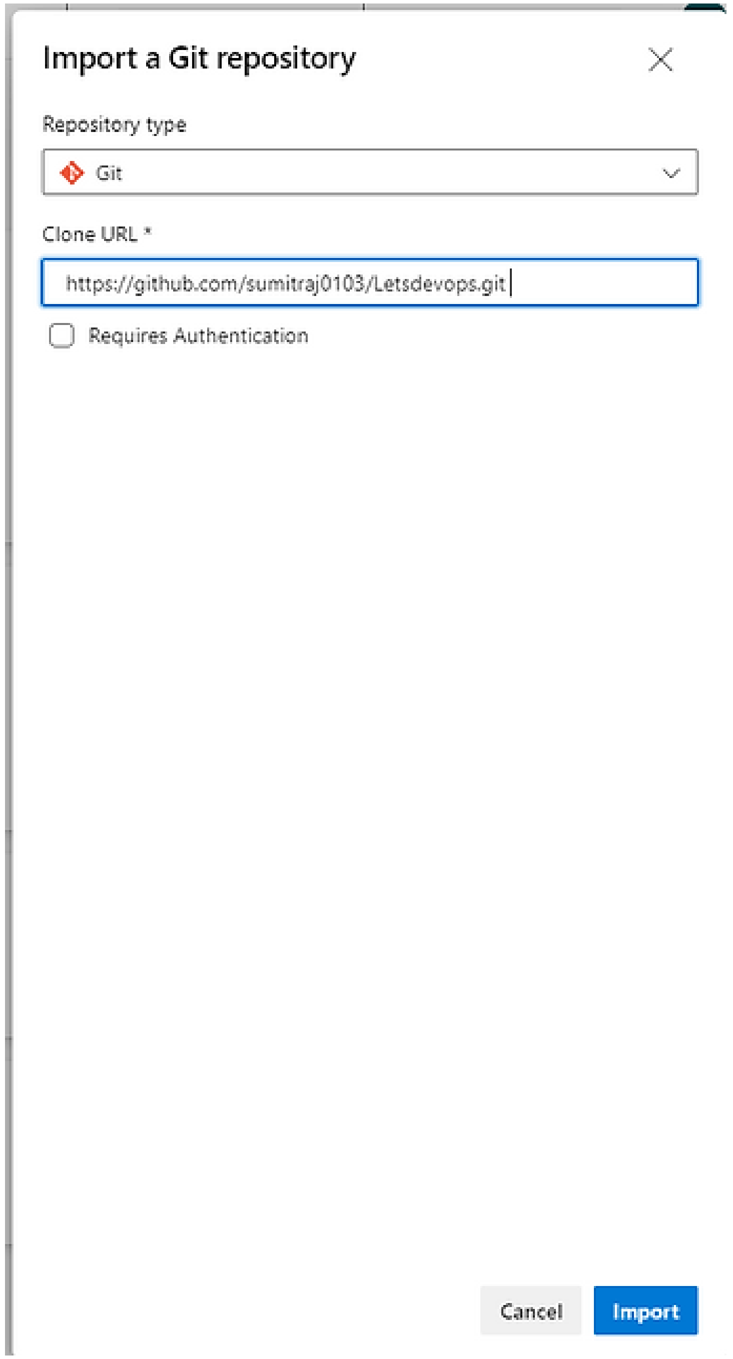
HTTPS SSH

```
git remote add origin
https://letsdevops2@dev.azure.com/letsdevops2/letsdevops1/_git/letsdevops1
```

Import a repository

Import

Step 2: Copy the URL https://github.com/sumitraj0103/Letsdevops.git and Import.

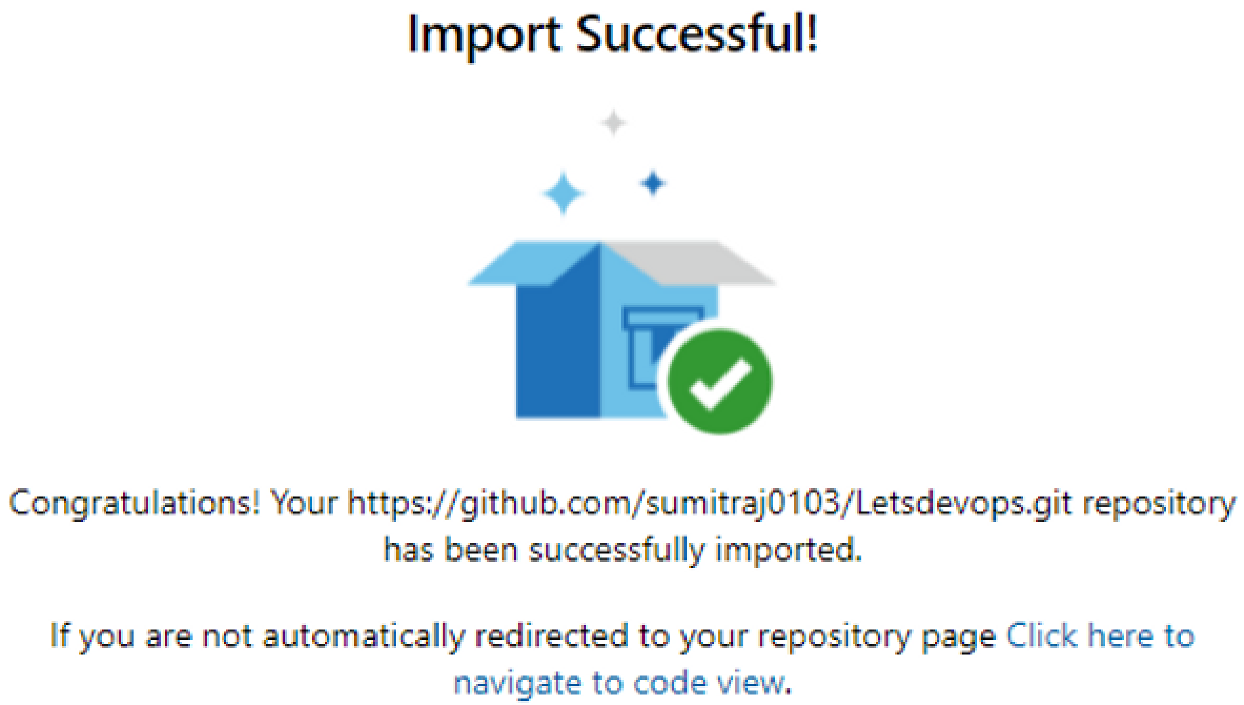

Import a Git repository

Repository type: Git

Clone URL *: https://github.com/sumitraj0103/Letsdevops.git

Requires Authentication

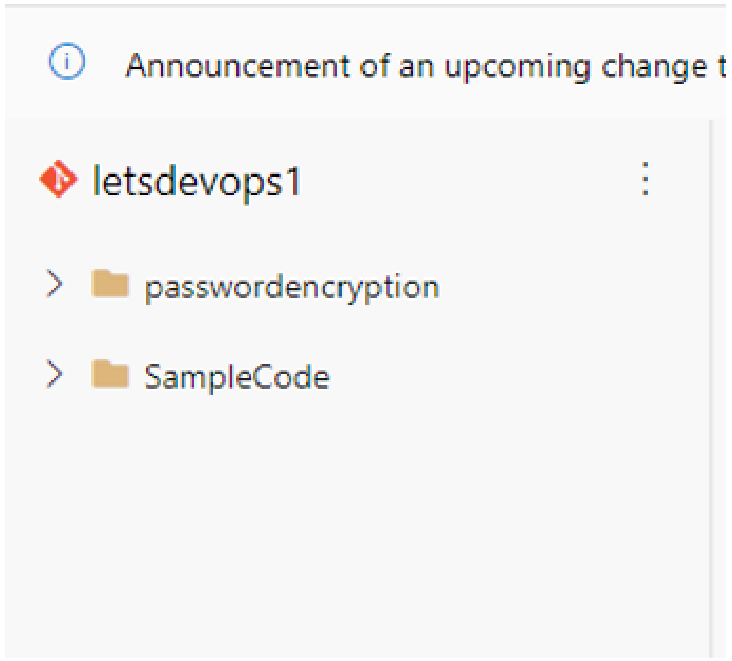
Step 3: On Successful import you can see the Repository Created.



Import Successful!

Congratulations! Your https://github.com/sumitraj0103/Letsdevops.git repository has been successfully imported.

If you are not automatically redirected to your repository page Click here to navigate to code view.


Announcement of an upcoming change t

letsdevops1

passwordencryption

SampleCode

Azure Pipelines

Once the Source Control setup is completed now we can setup the Continuous Integration and Continuous Deployment

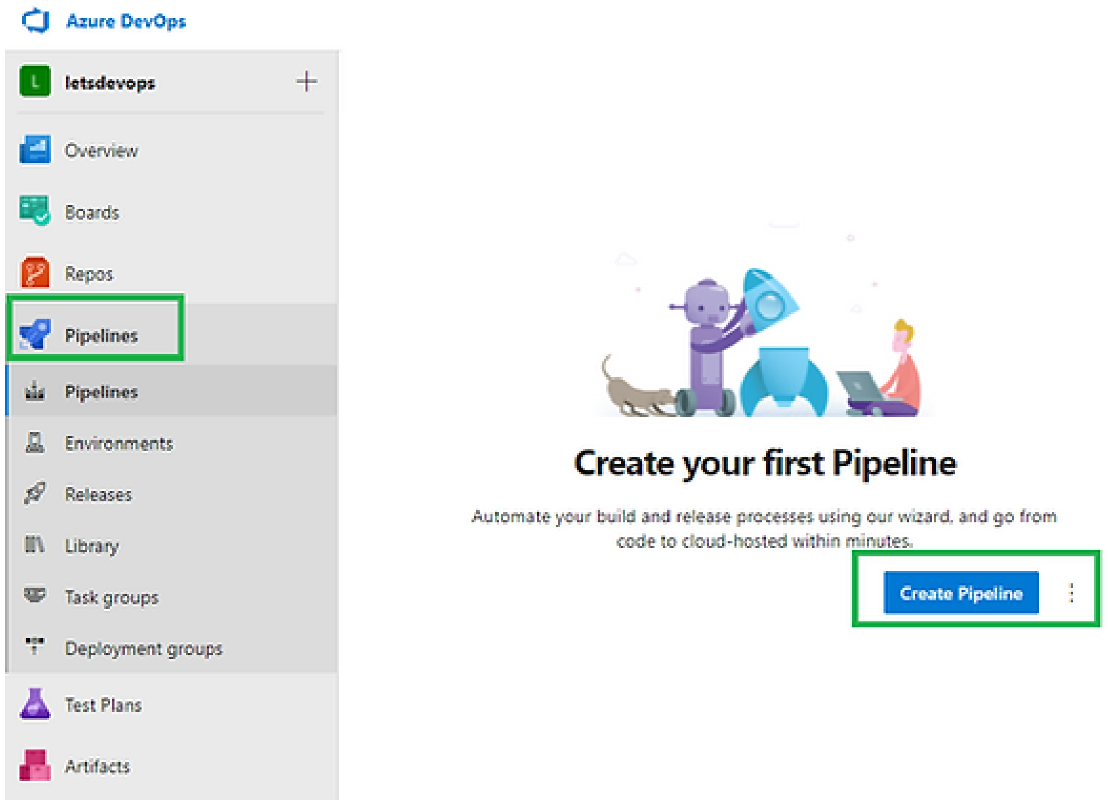
Continuous Integration

Continuous Integration is a process to setup the Build for your application Project, which enables to trigger build pipeline automatically on each source code Check-In. This helps to track the Build status and and identify build failure caused by specific change.

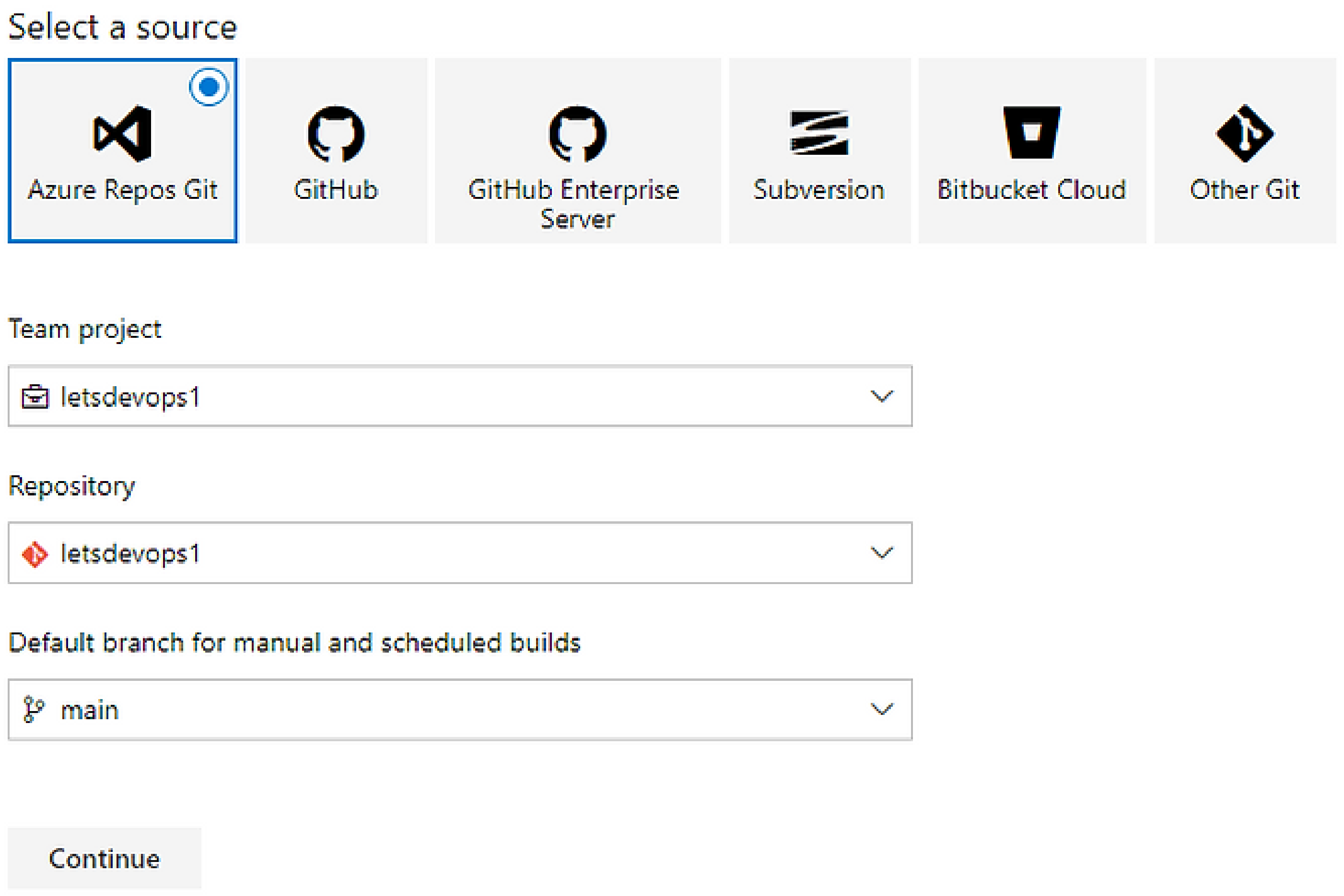
Lets setup the CI step by Step.

Step :1 Create Build Pipeline

Pipelines --> Create Pipeline



Step 2: Select the respective repository and the branch.


Select a source

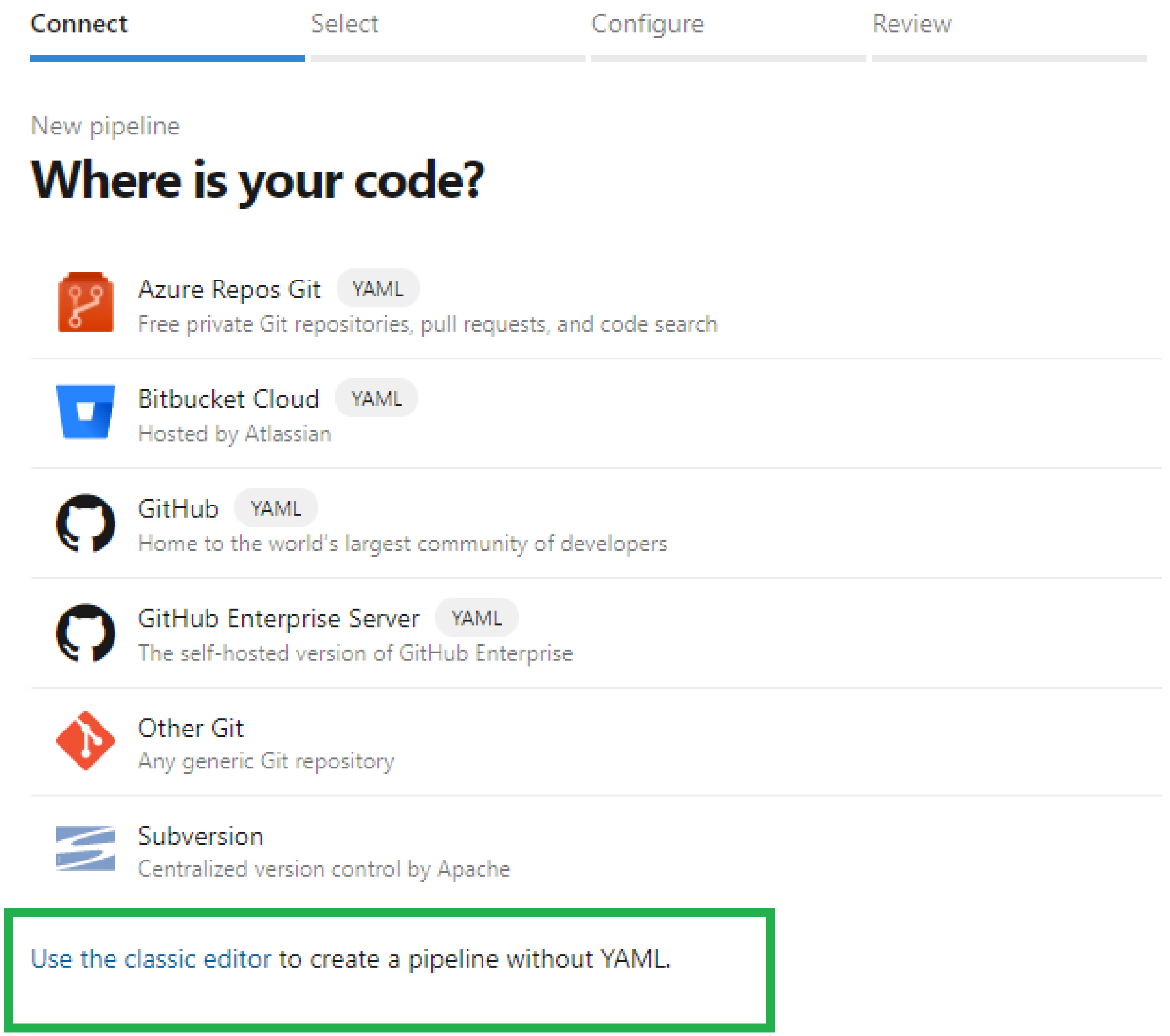
Azure Repos Git [✓] | GitHub | GitHub Enterprise Server | Subversion | Bitbucket Cloud | Other Git

Team project: letsdevops1

Repository: letsdevops1

Default branch for manual and scheduled builds: main

Step 3: Create the build using the Classic Editor



Connect Select Configure Review

New pipeline

Where is your code?

Azure Repos Git YAML
Free private Git repositories, pull requests, and code search

Bitbucket Cloud YAML
Hosted by Atlassian

GitHub YAML
Home to the world's largest community of developers

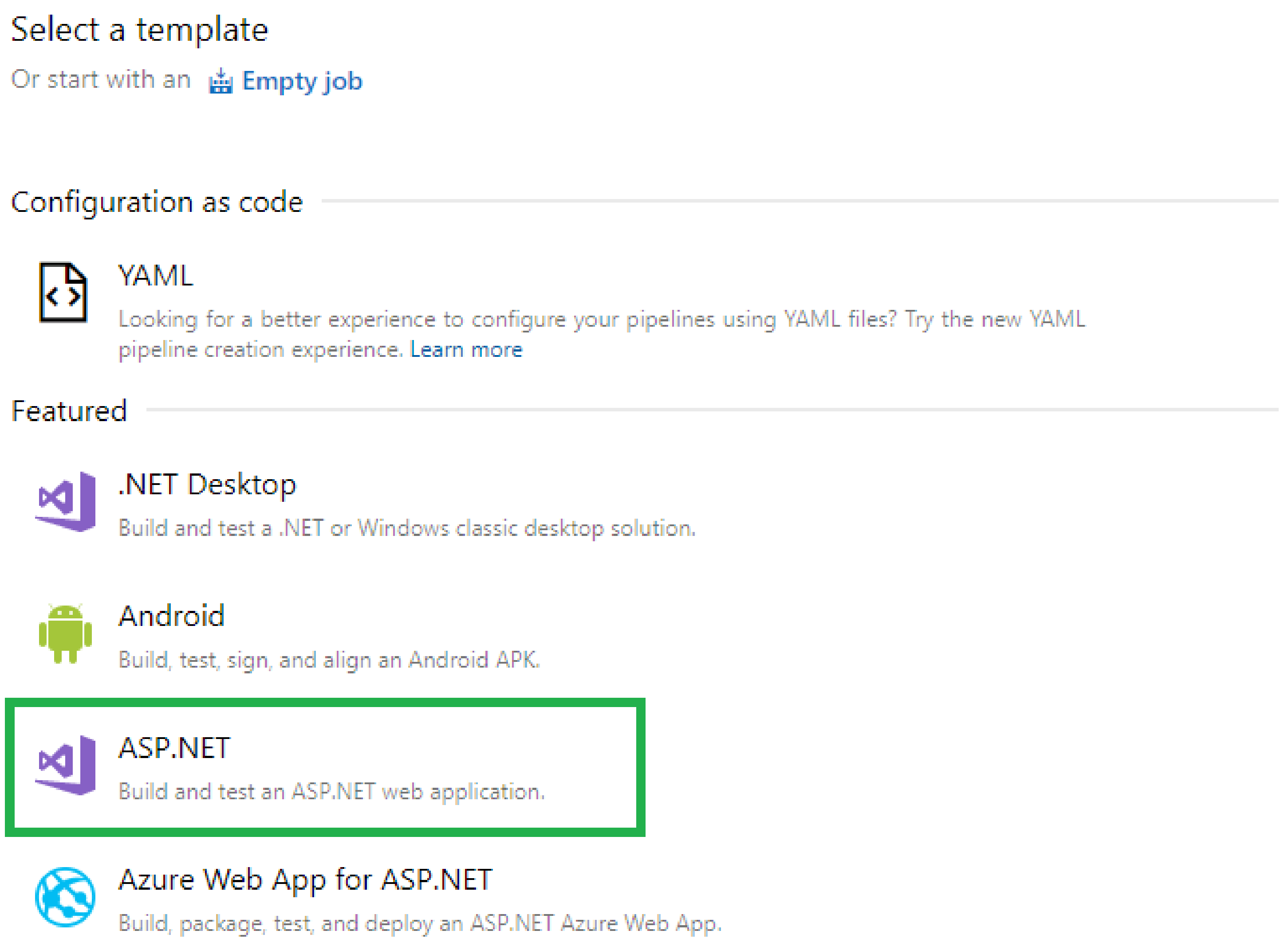
GitHub Enterprise Server YAML
The self-hosted version of GitHub Enterprise

Other Git
Any generic Git repository

Subversion
Centralized version control by Apache

Use the classic editor to create a pipeline without YAML.

Step 4: Use the ASP.NET Build template since the Sample Repo is ASP.NET web Application.


Select a template

Or start with an Empty job

Configuration as code

YAML

Looking for a better experience to configure your pipelines using YAML files? Try the new YAML pipeline creation experience. Learn more

Featured

.NET Desktop

Build and test a .NET or Windows classic desktop solution.

Android

Build, test, sign, and align an Android APK.

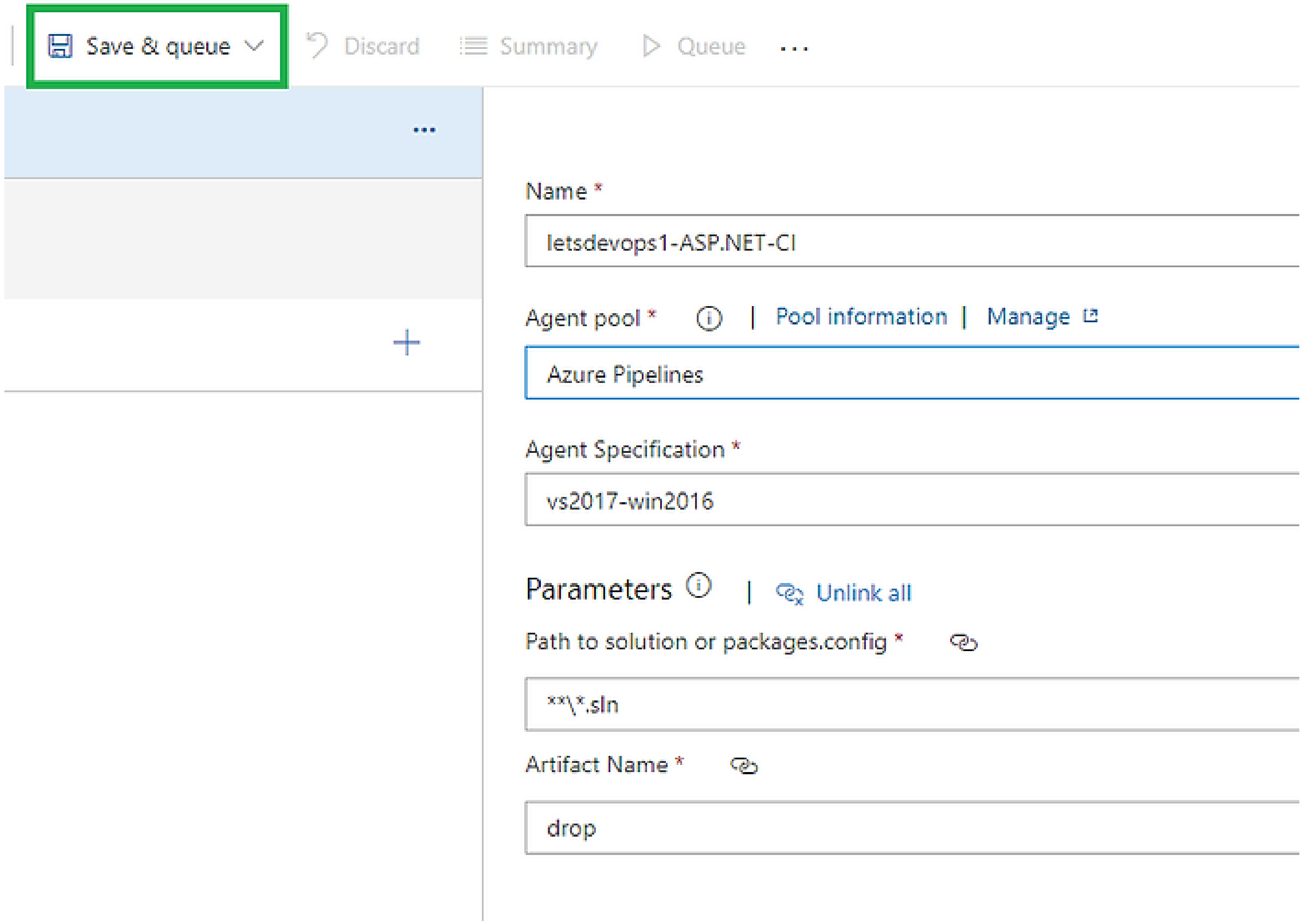
ASP.NET

Build and test an ASP.NET web application.

Azure Web App for ASP.NET

Build, package, test, and deploy an ASP.NET Azure Web App.

Step 5: Save the Pipeline and Queue the Build.



Name *

letsdevops1-ASP.NET-CI

Agent pool *

Azure Pipelines

Agent Specification *

vs2017-win2016

Parameters

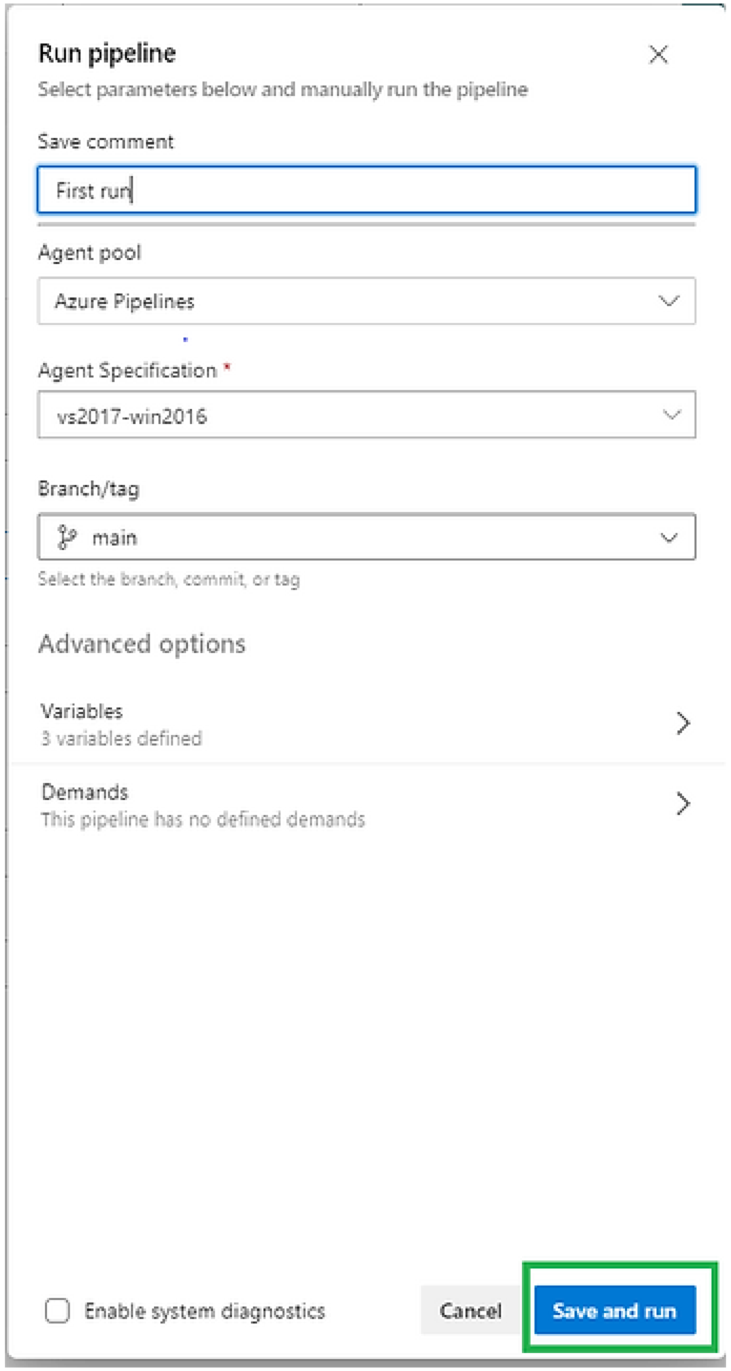
Path to solution or packages.config *

```
**\*.sln
```

Artifact Name *

drop

Save


Run pipeline

Select parameters below and manually run the pipeline

Save comment

First run

Agent pool

Azure Pipelines

Agent Specification *

vs2017-win2016

Branch/tag

main

Select the branch, commit, or tag

Advanced options

Variables

3 variables defined

Demands

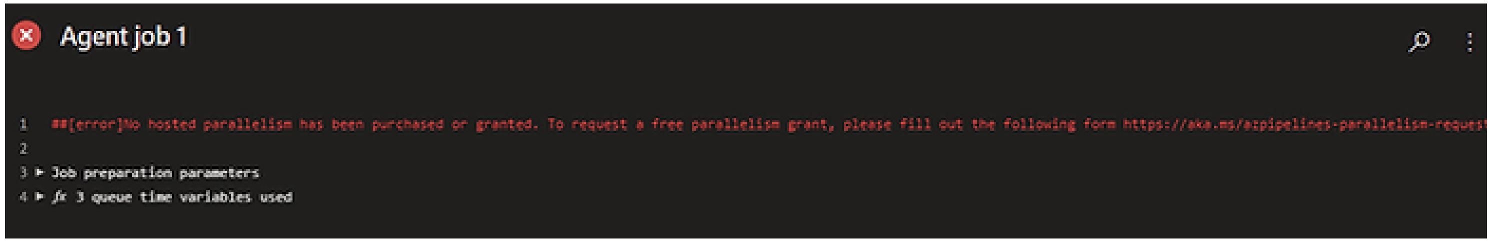
This pipeline has no defined demands

Enable system diagnostics

Cancel

Save and run

Step 6: If you get below error it mean the Free tier is not approved by Microsoft. If you are using MSDN Account you will not receive this error.


Agent job 1

```
##[error]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

▶ Job preparation parameters
▶ fx 3 queue time variables used
```

Solution 1: To solve this error you need to send email to Microsoft. Follow below instruction.

Private Project:
You could send email to azpipelines-freetier@microsoft.com in order to get your free tier.

- Your name
- Name of the Azure DevOps organization

Public Project:
You could send email to azpipelines-ossgrant@microsoft.com in order to get your free tier.

- Your name
- Azure DevOps organization for which you are requesting the free grant
- Links to the repositories that you plan to build
- Brief description of your project

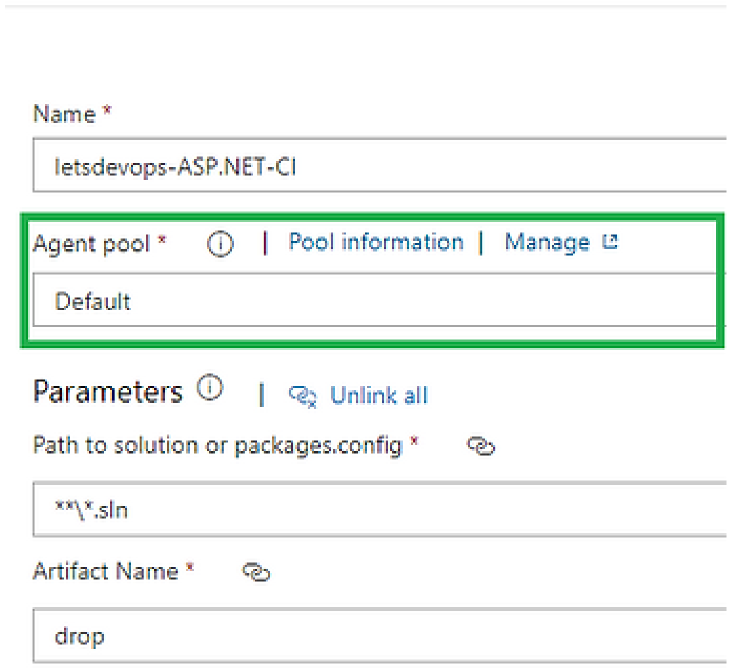
Solution 2: If you have self hosted agent you can configure to run the Build. In my case I will use configured self hosted Agent.

If you want to learn on How to Configure Self Hosted Agent please follow below article.

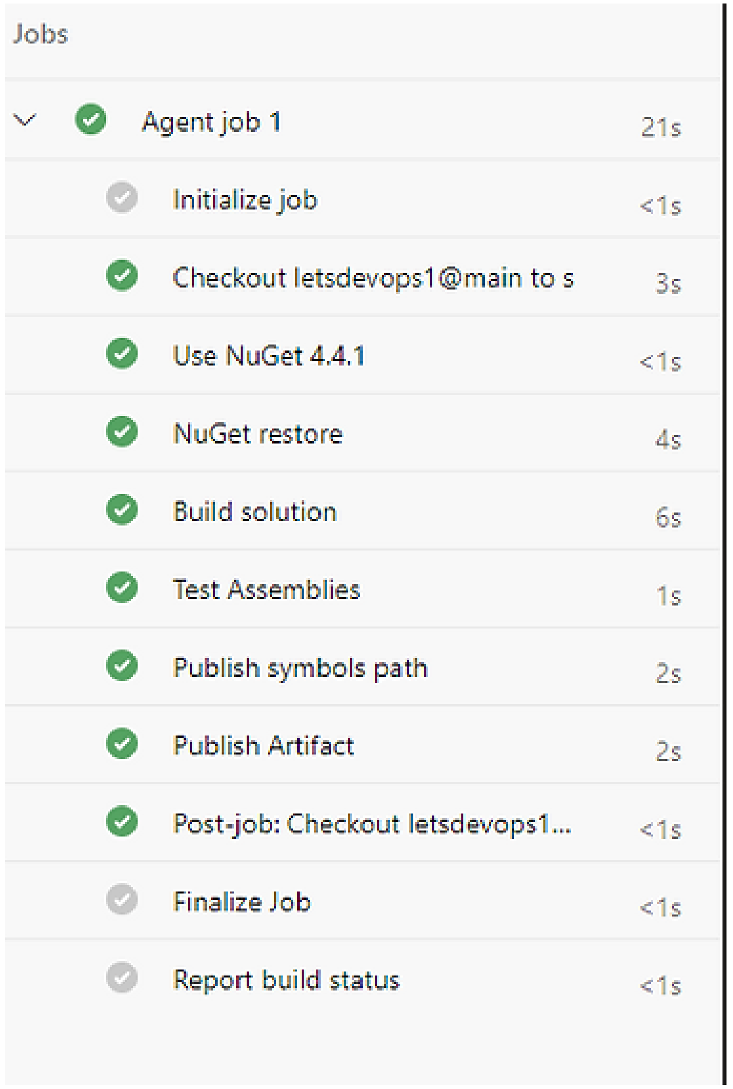
https://www.letsdevops.net/post/azure-devops-configure-self-hosted-agent-on-the-azure-devops

Demo: https://www.youtube.com/watch?v=zWwnXR5CqYM&t=170s

Step 7: Once you configured the Agent you can update the pipeline to run with self Hosted Agent Pool.


Name *
letsdevops-ASP.NET-CI
Agent pool *
Default
Path to solution or packages.config *
***.sln
Artifact Name *
drop

Step 8: Save and queue the Build and on successful run you will see the output like below.

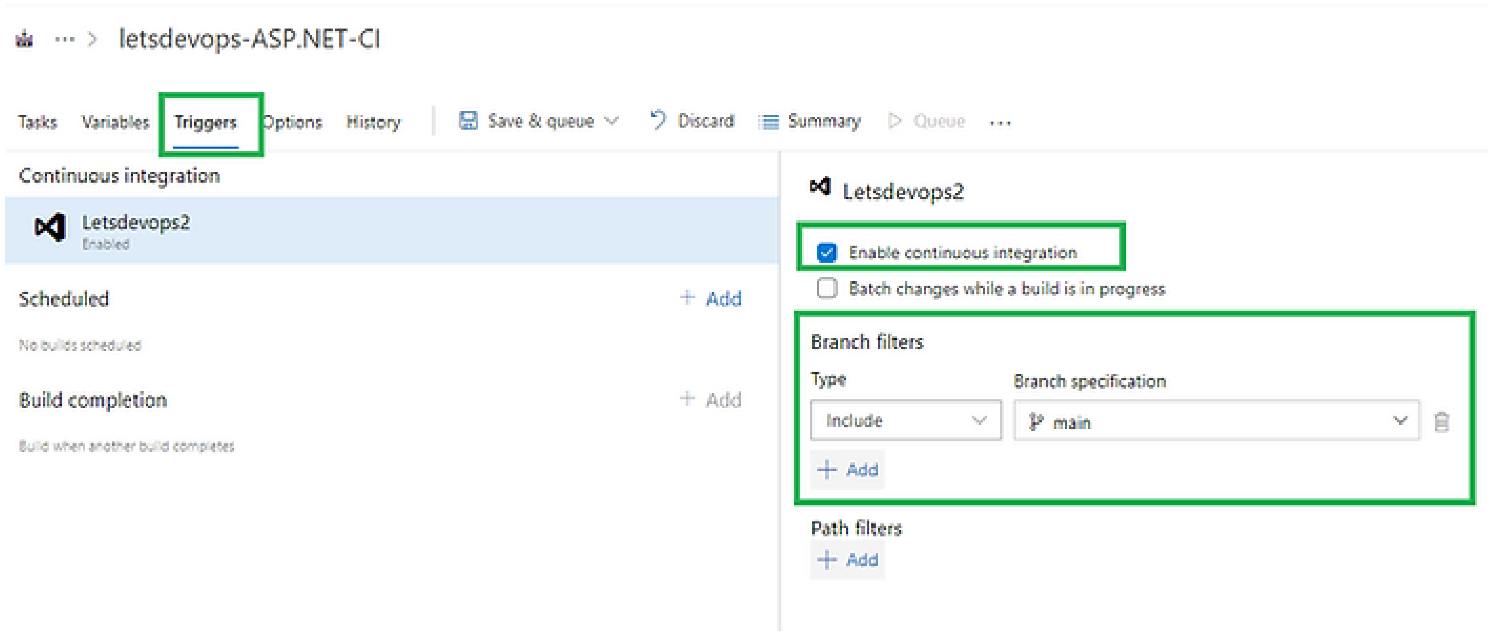


Jobs

✓	Agent job 1	21s
✓	Initialize job	<1s
✓	Checkout letsdevops1@main to s	3s
✓	Use NuGet 4.4.1	<1s
✓	NuGet restore	4s
✓	Build solution	6s
✓	Test Assemblies	1s
✓	Publish symbols path	2s
✓	Publish Artifact	2s
✓	Post-job: Checkout letsdevops1...	<1s
✓	Finalize Job	<1s
✓	Report build status	<1s

Step 9: Now you can enable the Continuous Integration and save the changes. With this changes on each check-in the Build will be triggered.

Edit Pipeline --> Triggers --> Enable continuous integration

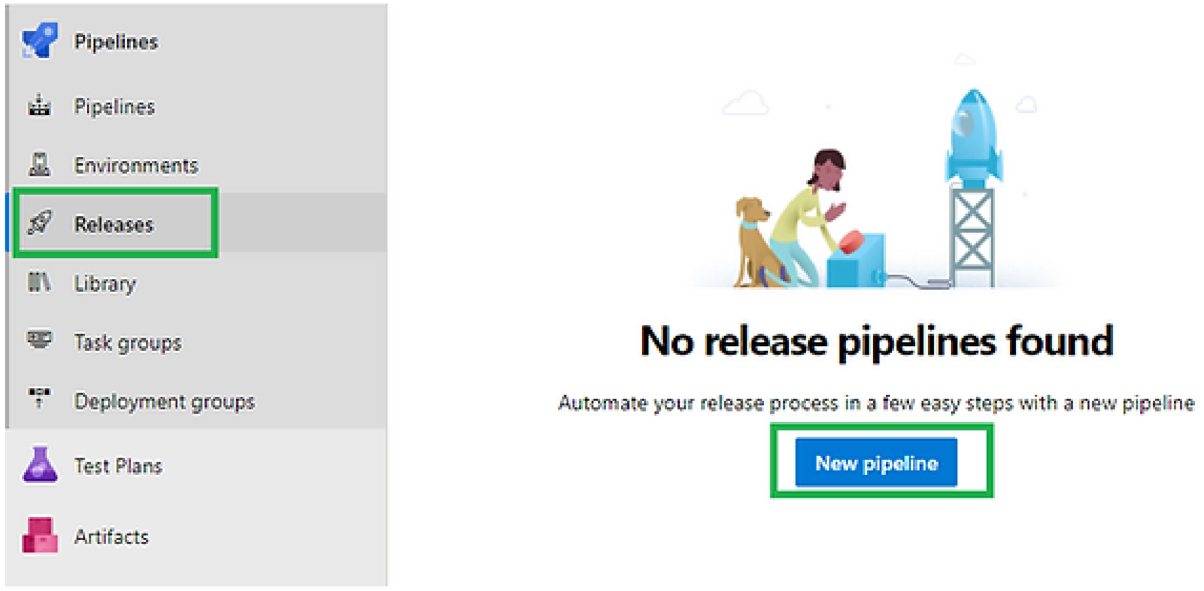


Continuous Deployment

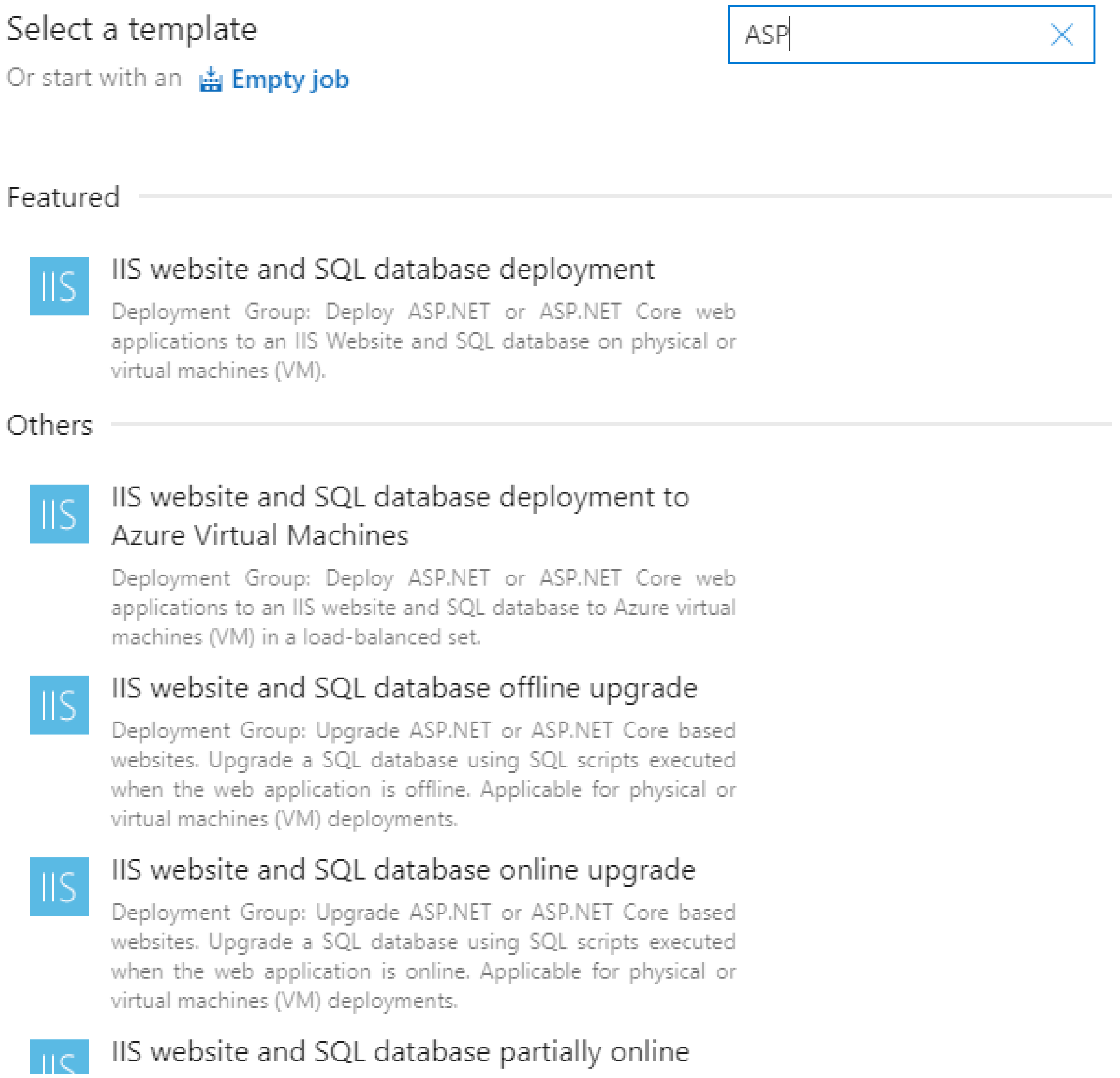
Continuous deployment is a process to deploy the changes on the different environment.

Once the Build (CI) is completed now its time to deploy the package. This will help to make your application available for use.

Step 1: Create Release Pipeline



Step 2: Since we have build the ASP.NET Solution. I will be selecting the below IIS Website Deployment template.


Select a template

Or start with an Empty job

ASP

Featured

IIS website and SQL database deployment

Deployment Group: Deploy ASP.NET or ASP.NET Core web applications to an IIS Website and SQL database on physical or virtual machines (VM).

Others

IIS website and SQL database deployment to Azure Virtual Machines

Deployment Group: Deploy ASP.NET or ASP.NET Core web applications to an IIS website and SQL database to Azure virtual machines (VM) in a load-balanced set.

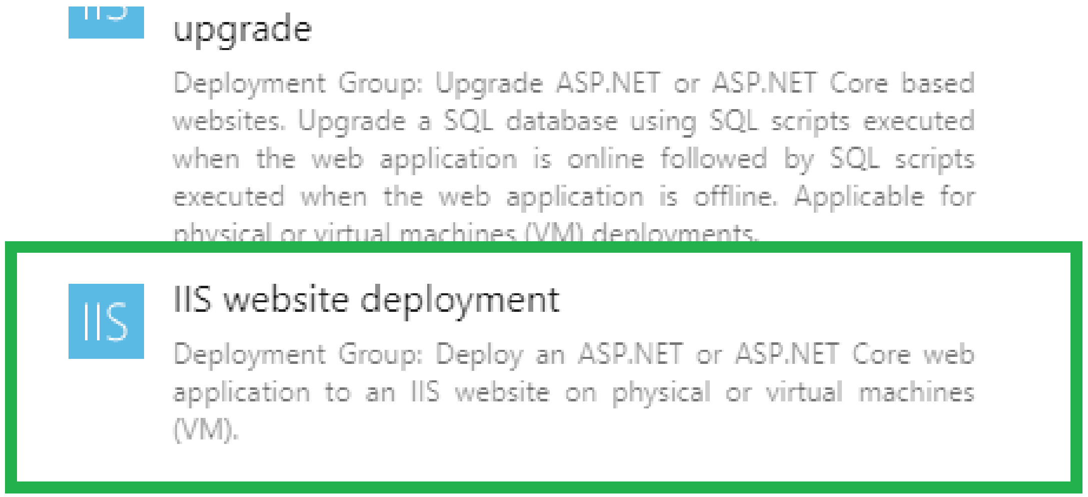
IIS website and SQL database offline upgrade

Deployment Group: Upgrade ASP.NET or ASP.NET Core based websites. Upgrade a SQL database using SQL scripts executed when the web application is offline. Applicable for physical or virtual machines (VM) deployments.

IIS website and SQL database online upgrade

Deployment Group: Upgrade ASP.NET or ASP.NET Core based websites. Upgrade a SQL database using SQL scripts executed when the web application is online. Applicable for physical or virtual machines (VM) deployments.

IIS website and SQL database partially online

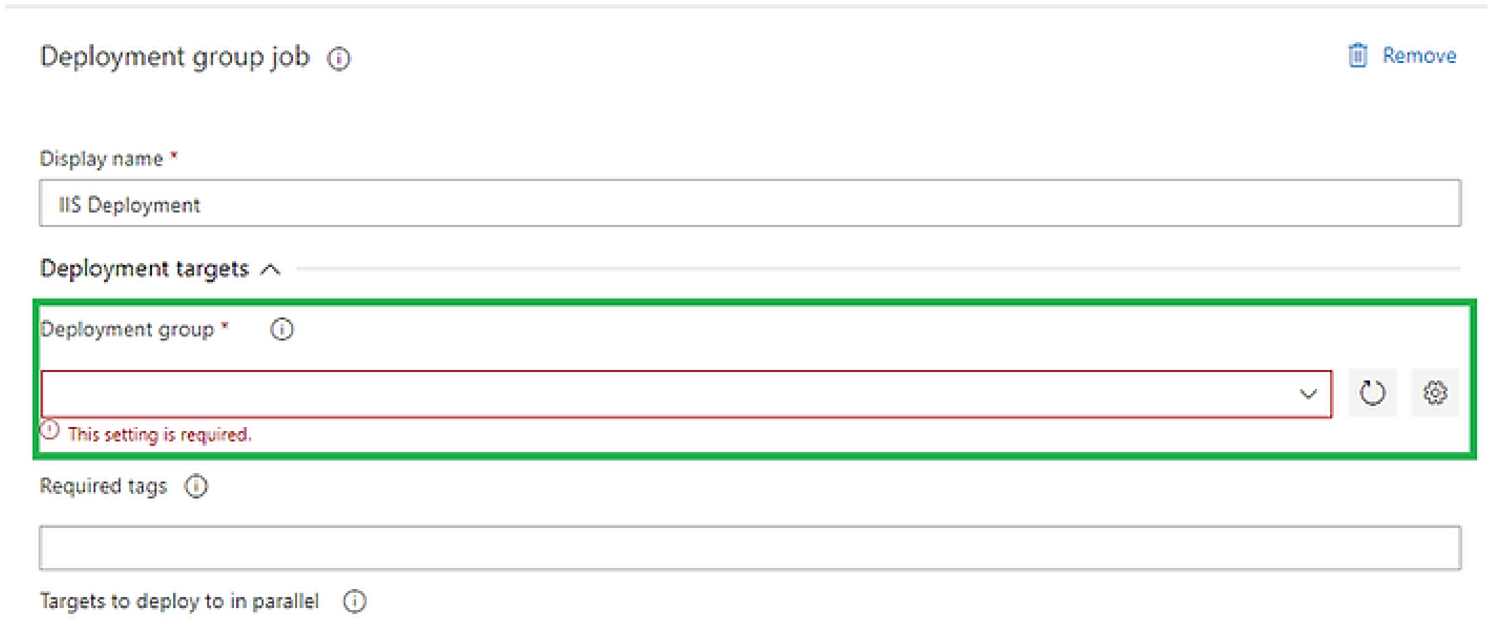

upgrade

Deployment Group: Upgrade ASP.NET or ASP.NET Core based websites. Upgrade a SQL database using SQL scripts executed when the web application is online followed by SQL scripts executed when the web application is offline. Applicable for physical or virtual machines (VM) deployments.

IIS website deployment

Deployment Group: Deploy an ASP.NET or ASP.NET Core web application to an IIS website on physical or virtual machines (VM).

Step 3: Now we need to configure the deployment group. Deployment group is the collection of Machine on which deployment will run. With this required package will be deployed to the respective target machine.



Deployment group job

Remove

Display name *

IIS Deployment

Deployment targets

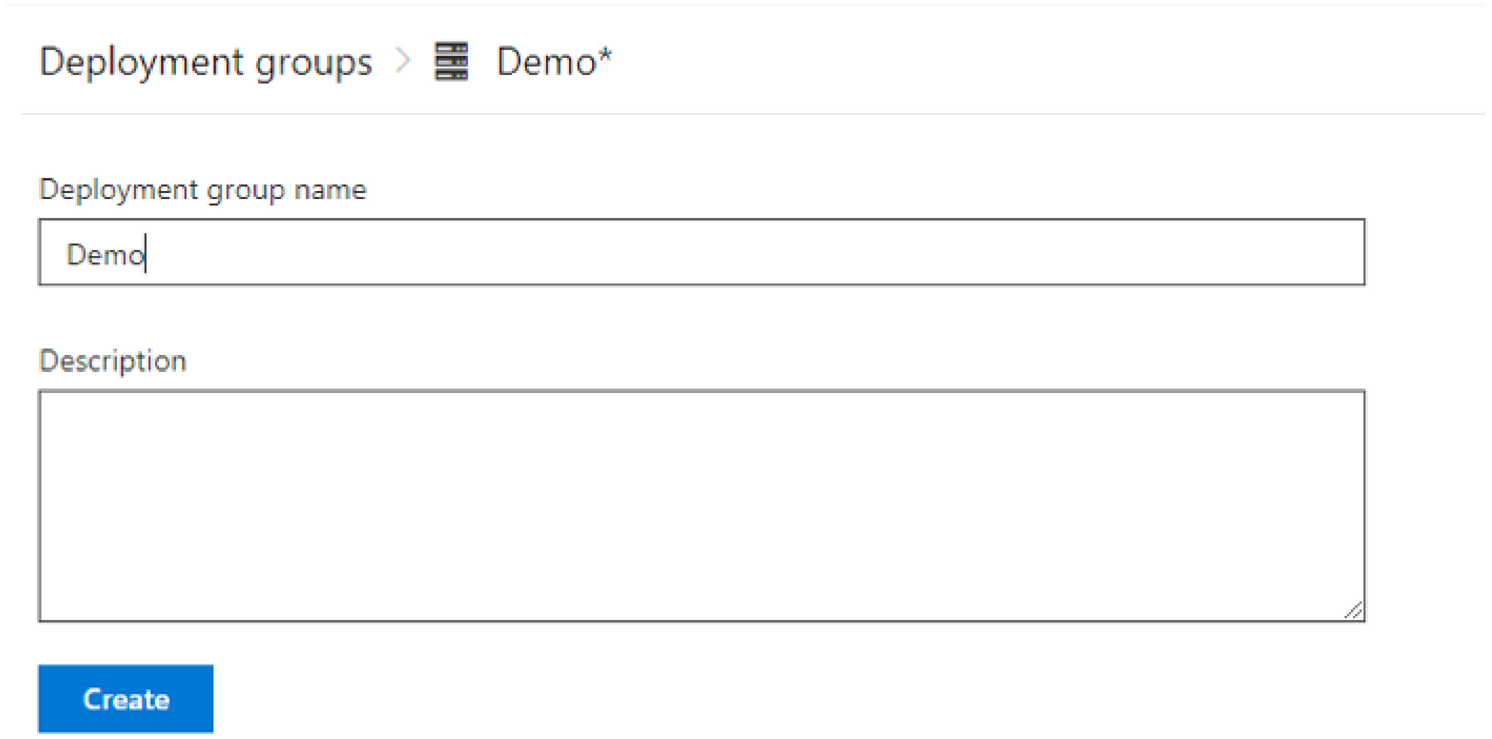
Deployment group *

This setting is required.

Required tags

Targets to deploy to in parallel

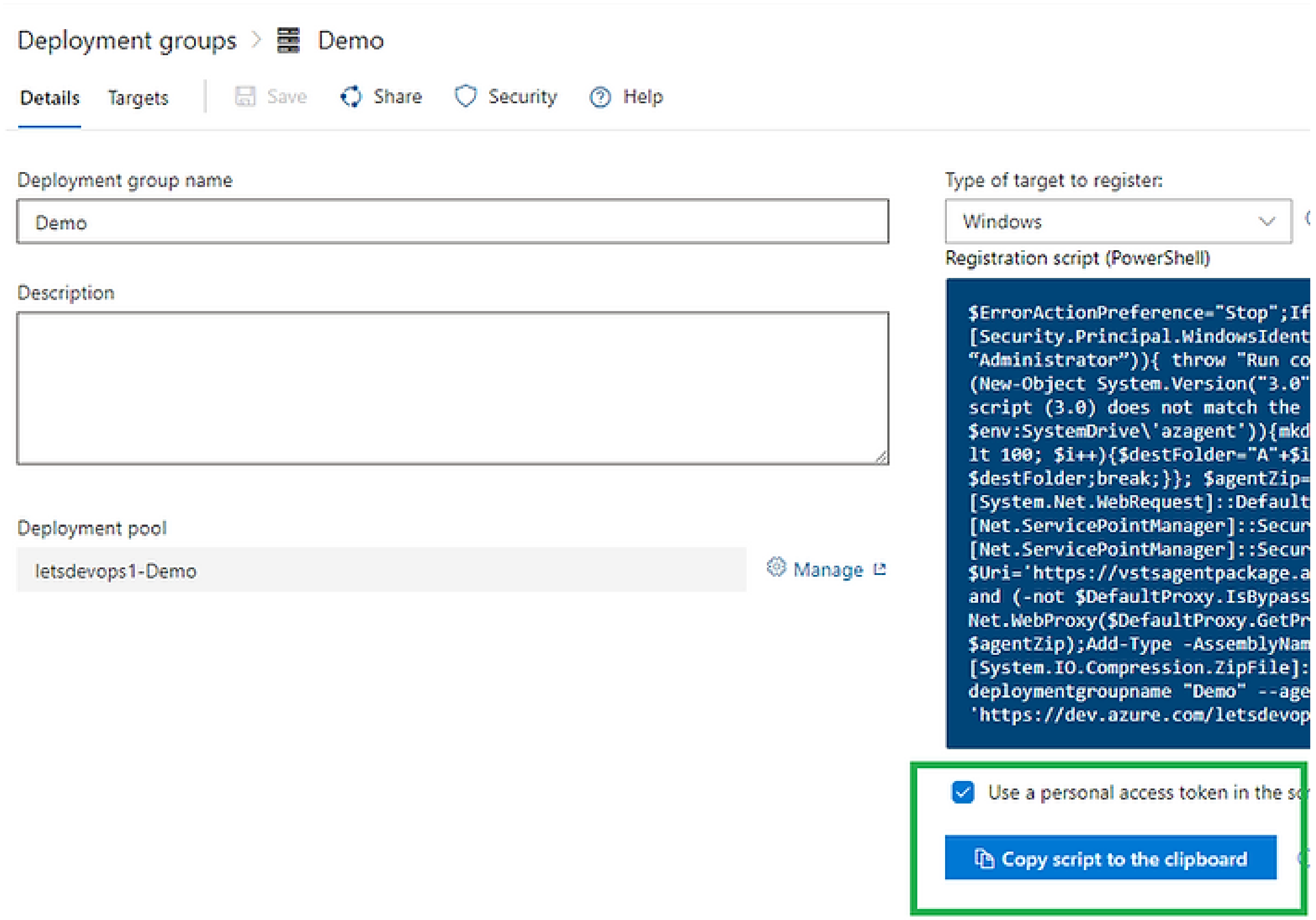
Step 4: Click on the Setting Button and create the Group.


Deployment groups > Demo*

Deployment group name: Demo

Description:

Step 5: Once group created now you will see PowerShell command that needs to be run on the machine where you want to deploy the Application.



Deployment group name

Demo

Description

Deployment pool

letsdevops1-Demo

Type of target to register:

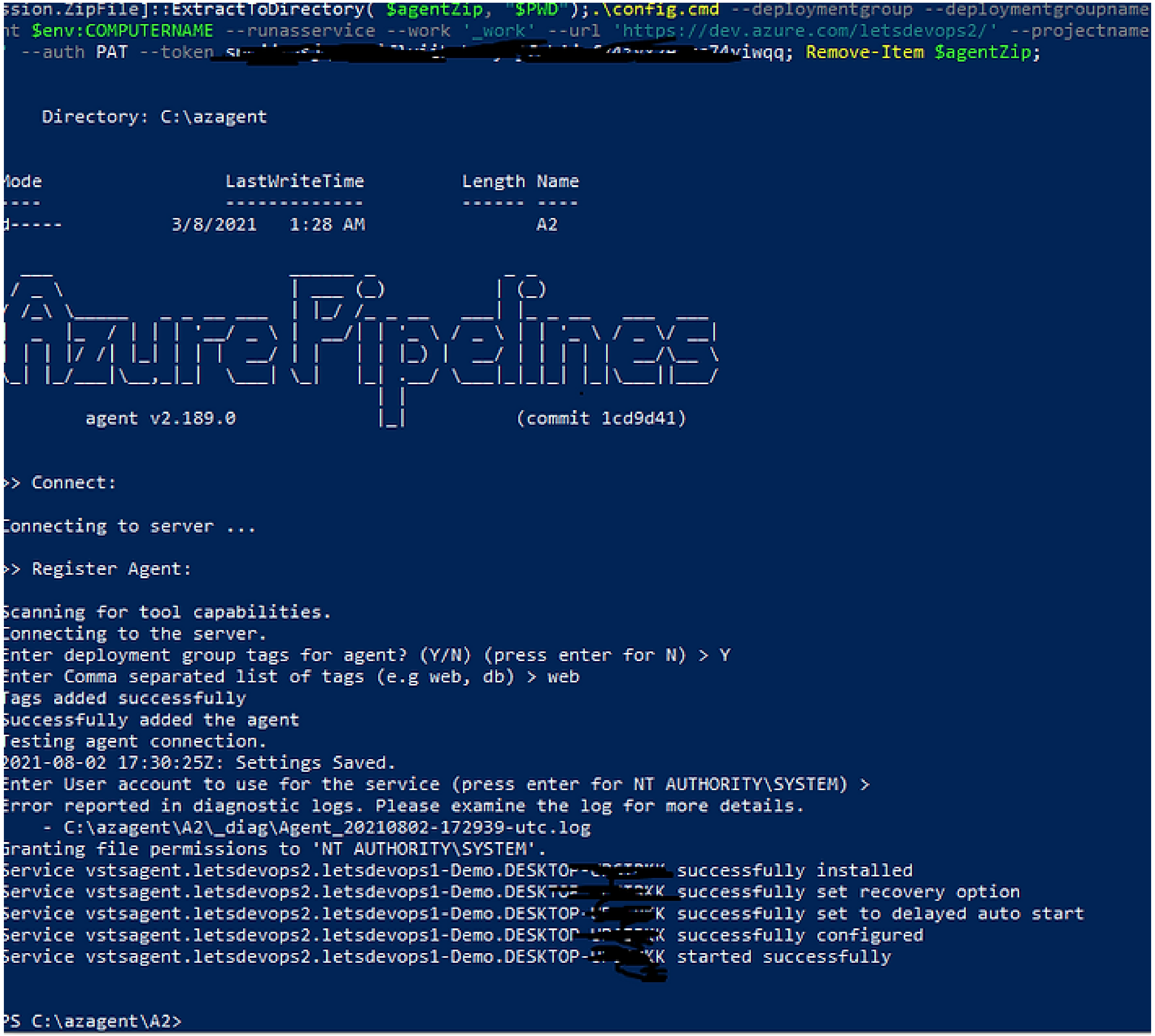
Windows

Registration script (PowerShell)

```
$ErrorActionPreference="Stop";If
[Security.Principal.WindowsIdent
"Administrator")){ throw "Run co
(New-Object System.Version("3.0"
script (3.0) does not match the
$env:SystemDrive\'azagent')){mkd
lt 100; $i++){$destFolder="A"+$i
$destFolder;break;}}; $agentZip=
[System.Net.WebRequest]::Default
[Net.ServicePointManager]::Secur
[Net.ServicePointManager]::Secur
$Uri='https://vstsagentpackage.a
and (-not $DefaultProxy.IsBypass
Net.WebProxy($DefaultProxy.GetPr
$agentZip);Add-Type -AssemblyNam
[System.IO.Compression.ZipFile]:
deploymentgroupname "Demo" --age
'https://dev.azure.com/letsdevop
```

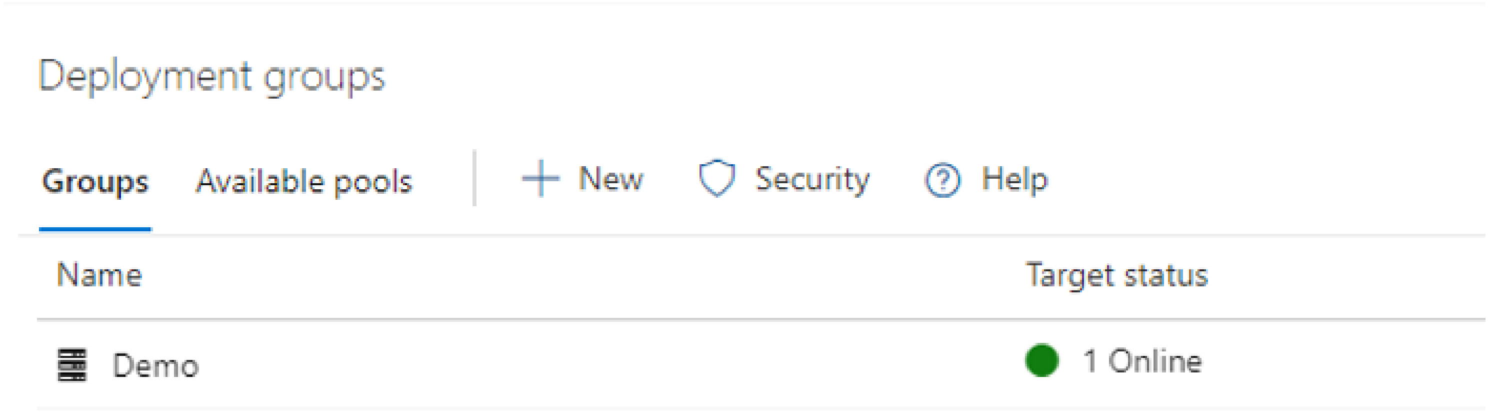
[✓] Use a personal access token in the s

Step 6: Open PowerShell as admin on the deployment machine and run the script.



```
ssion.ZipFile]::ExtractToDirectory( $agentZip, "$PWD");.\config.cmd --deploymentgroup --deploymentgroupname
nt $env:COMPUTERNAME --runasservice --work '_work' --url 'https://dev.azure.com/letsdevops2/' --projectname
' --auth PAT --token [redacted]74viwqq; Remove-Item $agentZip;


    Directory: C:\azagent


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          3/8/2021   1:28 AM                A2

 ___                      ______ _            _ _
/ _ \                     | ___ (_)          | (_)
/ /_\ \_____   _ _ __ ___  | |_/ /_ _ __   ___| |_ _ __   ___  ___
  _  |_  / | | | '__/ _ \ |  __/| | '_ \ / _ \ | | '_ \ / _ \/ __|
 | | |/ /| |_| | | |  __/ | |   | | |_) |  __/ | | | | |  __/\__ \
_| |_/___|\__,_|_|  \___| \_|   |_| .__/ \___|_|_|_| |_|\___||___/
                                  | |
        agent v2.189.0           |_|          (commit 1cd9d41)


>> 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) > Y
Enter Comma separated list of tags (e.g web, db) > web
Tags added successfully
Successfully added the agent
Testing agent connection.
2021-08-02 17:30:25Z: 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\A2\_diag\Agent_20210802-172939-utc.log
Granting file permissions to 'NT AUTHORITY\SYSTEM'.
Service vstsagent.letsdevops2.letsdevops1-Demo.DESKTOP-[redacted] successfully installed
Service vstsagent.letsdevops2.letsdevops1-Demo.DESKTOP-[redacted] successfully set recovery option
Service vstsagent.letsdevops2.letsdevops1-Demo.DESKTOP-[redacted] successfully set to delayed auto start
Service vstsagent.letsdevops2.letsdevops1-Demo.DESKTOP-[redacted] successfully configured
Service vstsagent.letsdevops2.letsdevops1-Demo.DESKTOP-[redacted] started successfully


PS C:\azagent\A2>
```

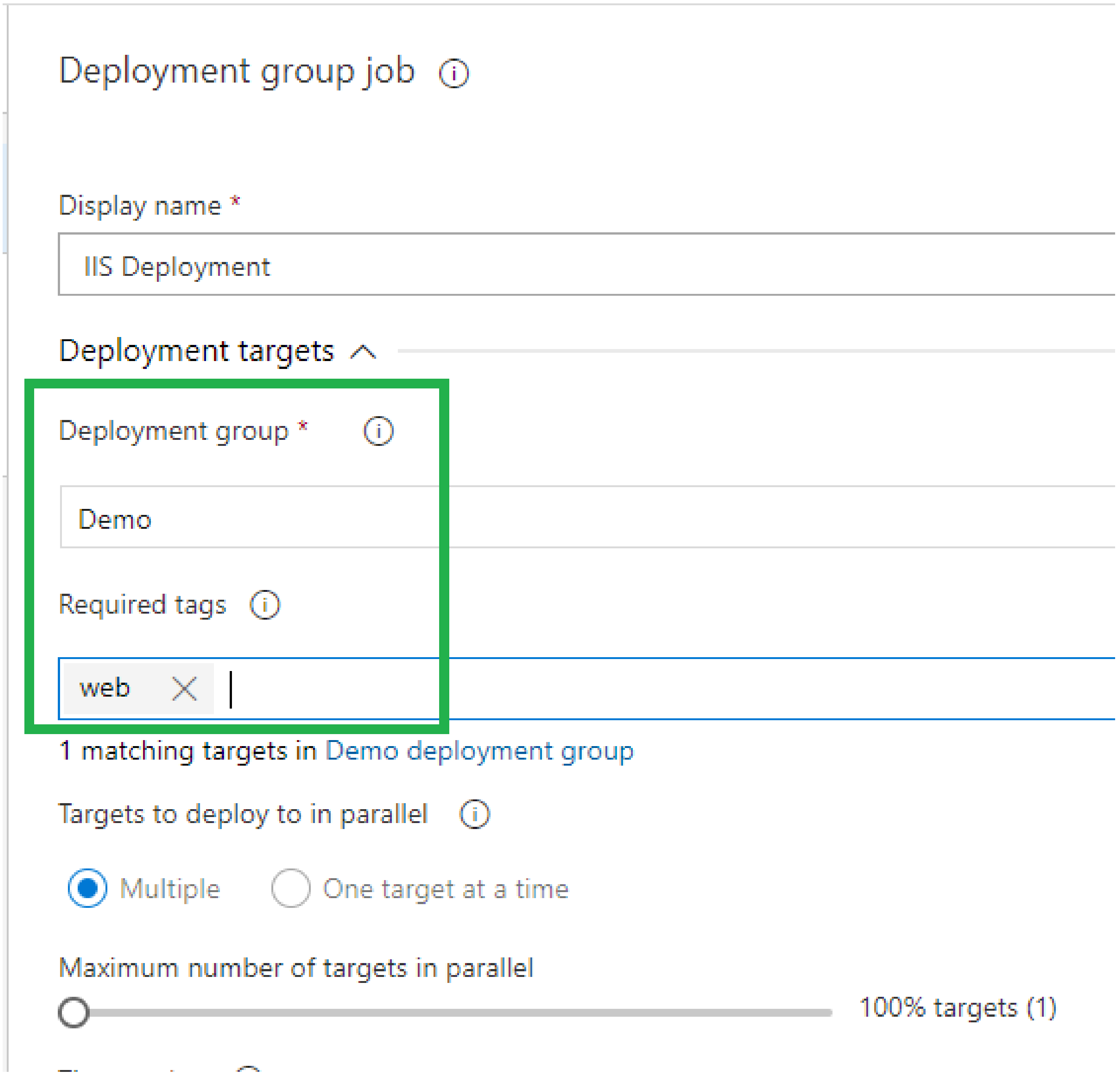
Step 7: Once the Deployment Machine is configured you can see it Online.


Deployment groups

Groups Available pools | + New Security Help

Name	Target status
Demo	● 1 Online

Step 7: Now Complete the Release pipeline setup and save the changes after adding the deployment group details. Required Tag you can setup during t



Deployment group job

Display name *

IIS Deployment

Deployment targets

Deployment group *

Demo

Required tags

web

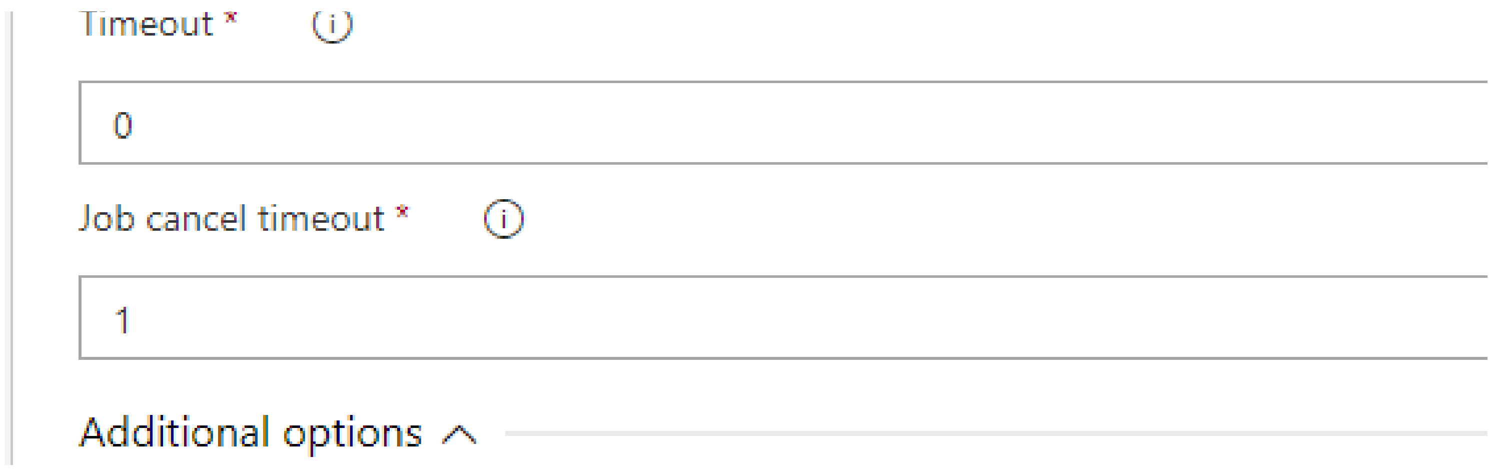
1 matching targets in Demo deployment group

Targets to deploy to in parallel

[✓ Multiple] One target at a time

Maximum number of targets in parallel

100% targets (1)



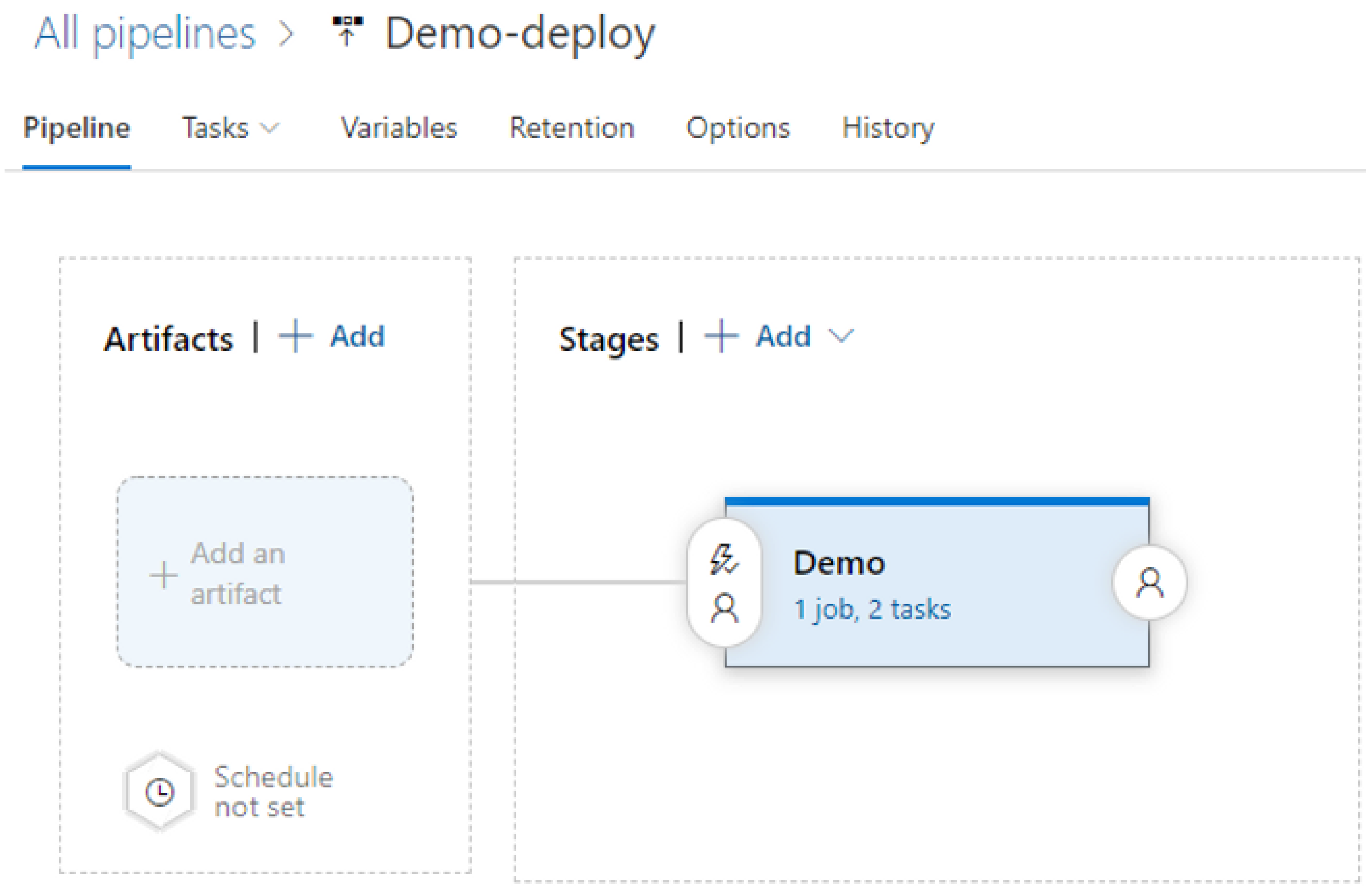
Timeout *

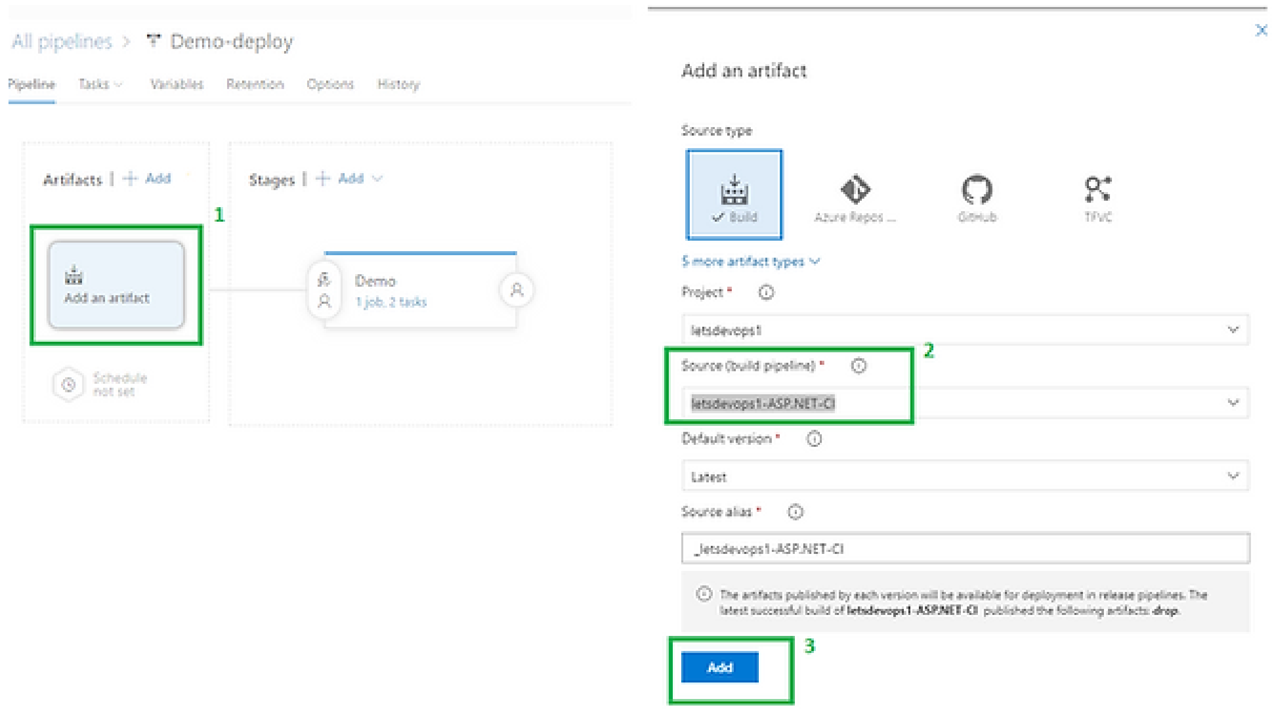
0

Job cancel timeout *

1

Step 8: Add the Artifact so that it will use the package which created during the CI.


All pipelines > Demo-deploy
Pipeline
Tasks
Variables
Retention
Options
History
Artifacts | + Add
Add an artifact
Schedule not set
Stages | + Add
Demo
1 job, 2 tasks



Project *: letsdevops1

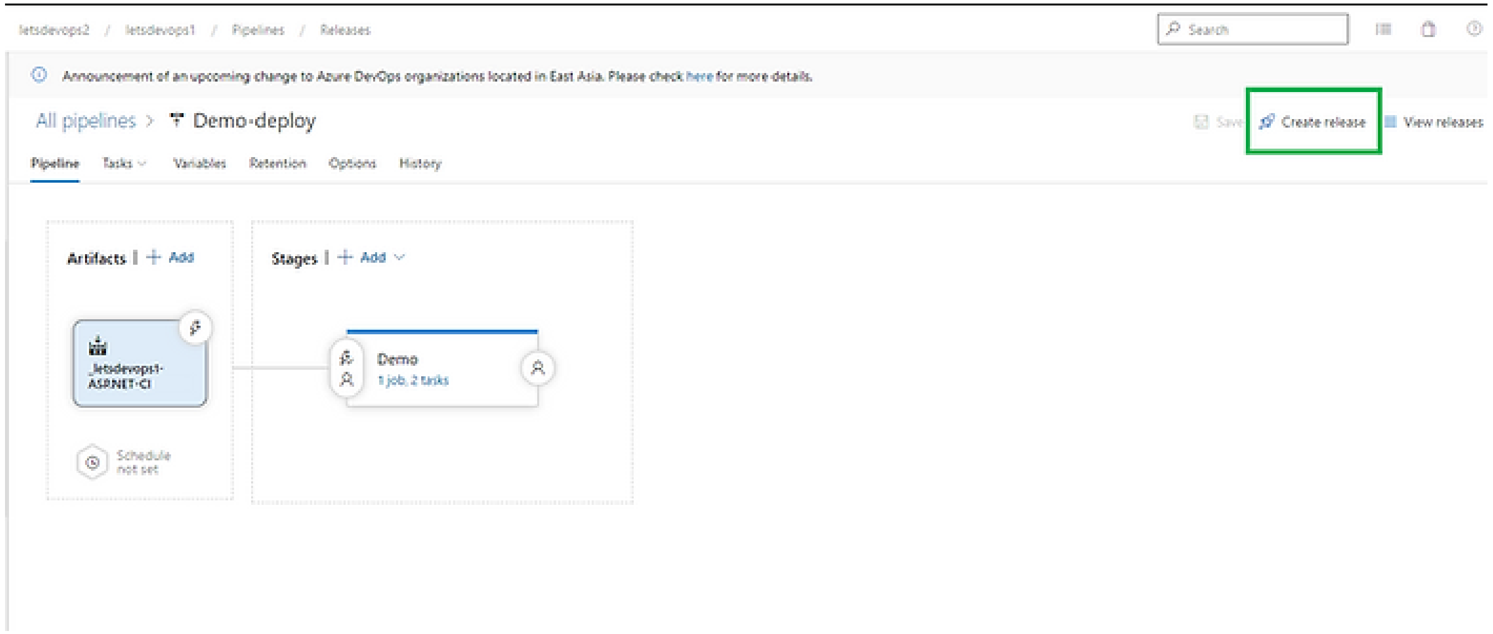
Source (build pipeline) *: letsdevops1-ASP.NET-CI

Default version *: Latest

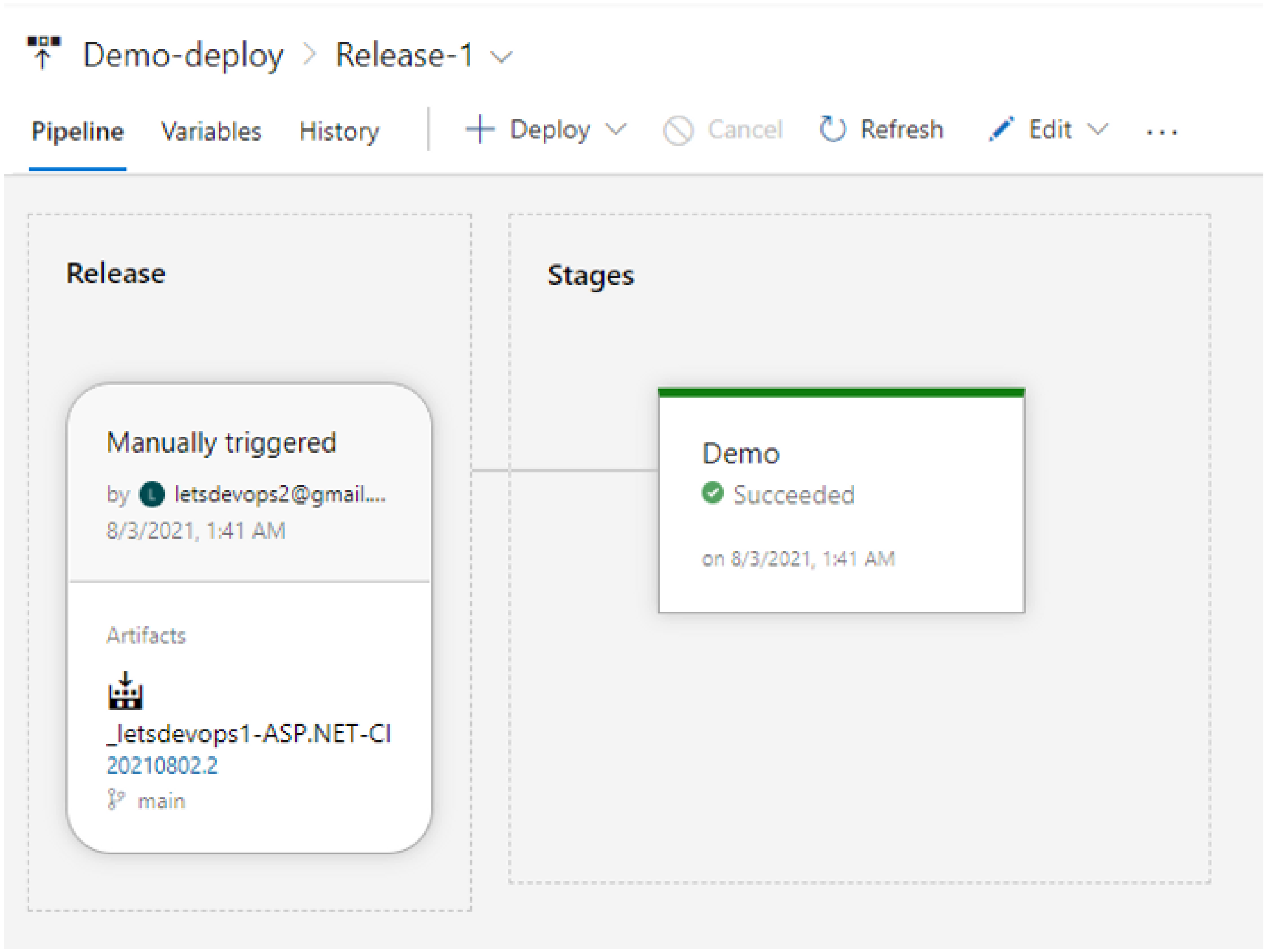
Source alias *: _letsdevops1-ASP.NET-CI

The artifacts published by each version will be available for deployment in release pipelines. The latest successful build of **letsdevops1-ASP.NET-CI** published the following artifacts: **drop**.

Step 9: Create Release and deploy it.


letsdevops2 / letsdevops1 / Pipelines / Releases
Search
Announcement of an upcoming change to Azure DevOps organizations located in East Asia. Please check here for more details.
All pipelines > Demo-deploy
Save
Create release
View releases
Pipeline
Tasks
Variables
Retention
Options
History
Artifacts | + Add
Stages | + Add
_letsdevops1-ASPNET-CI
Schedule not set
Demo
1 job, 2 tasks

Step 10: On Successful deployment you can see the Stage Deployment completed.



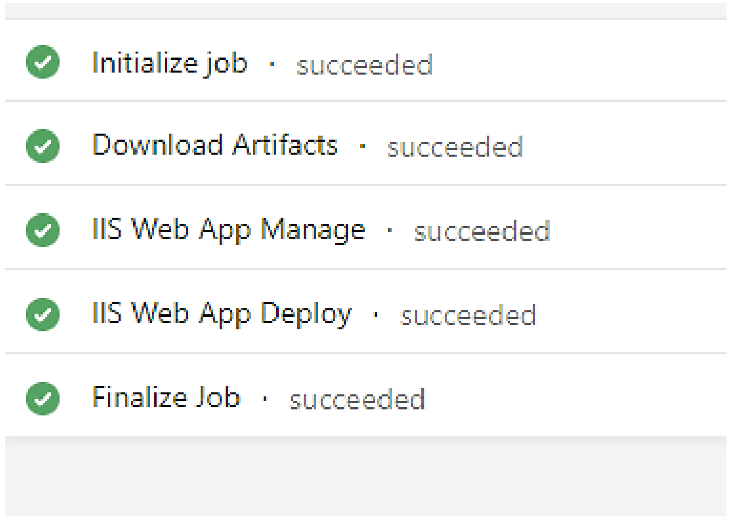
Release

Manually triggered
by letsdevops2@gmail....
8/3/2021, 1:41 AM

Artifacts
_letsdevops1-ASP.NET-CI
20210802.2
main

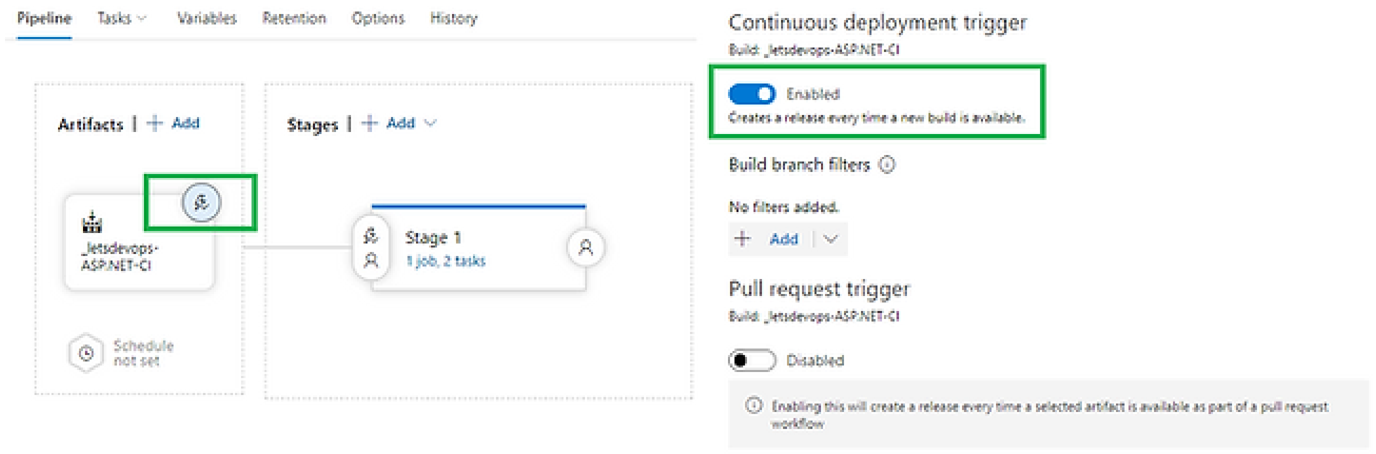
Stages

Demo
Succeeded
on 8/3/2021, 1:41 AM


Initialize job · succeeded
Download Artifacts · succeeded
IIS Web App Manage · succeeded
IIS Web App Deploy · succeeded
Finalize Job · succeeded

Step 11: Enable the Continuous Deployment

Edit Release Pipeline --> Trigger --> Enabled



Continuous deployment trigger

Build: _letsdevops-ASP.NET-CI

Enabled

Creates a release every time a new build is available.

Build branch filters

No filters added.

Add

Pull request trigger

Build: _letsdevops-ASP.NET-CI

Disabled

Enabling this will create a release every time a selected artifact is available as part of a pull request workflow

Continuous Integration and Continuous Deployment Simulation.

Since the CI and CD is enabled and now I will make a simple check -in in the Source Repository and that will trigger the Build as well as deployment without any human intervention.

LetsDevOps: Introduction to Azure DevOps for Beginners - Setup Repository, CI/CD With Demo