Documentation on IPERFORMUploadFTPtoAzure

Version Number:1.1

Table of Contents

[Introduction: 3](#_Toc133946039)

[Scope: 3](#_Toc133946040)

[CI/CD Architecture: 3](#_Toc133946041)

[Deployment Group: 3](#_Toc133946042)

[Build Pipeline for IPERFORMUploadFTPtoAzure 6](#_Toc133946043)

[Pipelines: 9](#_Toc133946044)

[Variables: 9](#_Toc133946045)

[Tasks: 9](#_Toc133946046)

[Publish Artifact 11](#_Toc133946047)

[Validation 12](#_Toc133946048)

[Release pipeline for IPERFORMUploadFTPtoAzure 12](#_Toc133946049)

[Steps: 12](#_Toc133946050)

[Continuous deployment trigger: 14](#_Toc133946051)

[Pull request trigger 15](#_Toc133946052)

[Pre-deployment conditions : 15](#_Toc133946053)

[Tasks: 17](#_Toc133946054)

[Agent Job: 17](#_Toc133946055)

[Azure powershell script:Inline Script 18](#_Toc133946056)

[Copy files – 20](#_Toc133946057)

[File Transform – 21](#_Toc133946058)

[Variables: 21](#_Toc133946059)

Revision History: 22

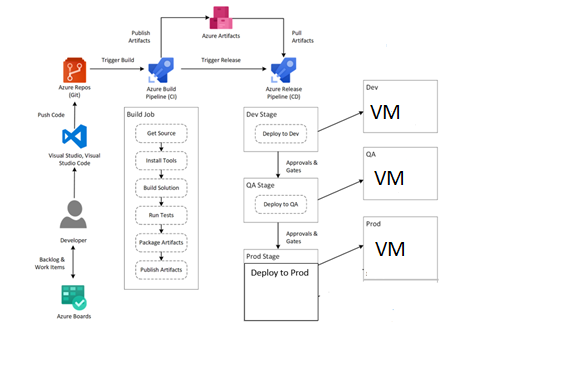
# Introduction:

# This is the documentation for the release pipelines for IPERFORMEUploadFTPtoAzure. It includes the build and release pipeline processes. Build pipelines include SonarQube-related tasks, artifact generation, etc. Once the build is successful, the release pipeline starts triggering it

# Scope:

* CI/CD is used because code can be easily deployed into a specific environment in a short period of time.
* Through the build pipeline, we can generate the artifacts, and through the release pipeline, we are able to deploy them in a specific location with the required changes.

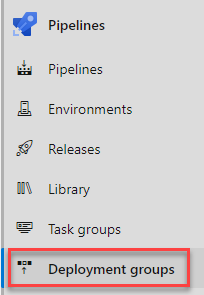
# CI/CD Architecture:



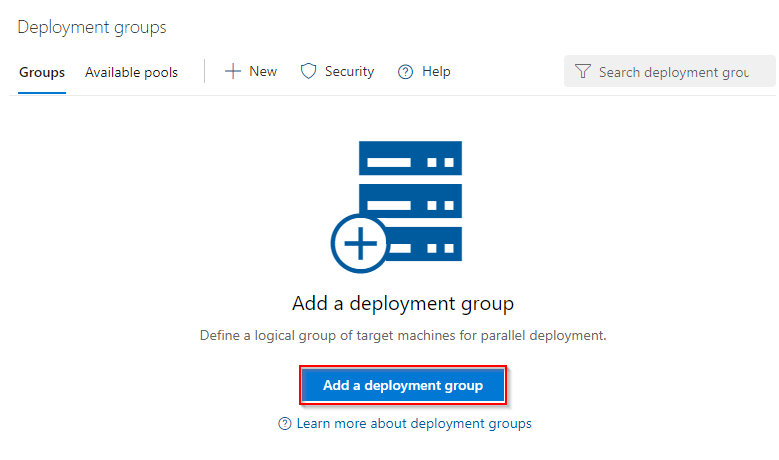
## Deployment Group:

A deployment group is a logical set of deployment target machines that have agents installed on each one.

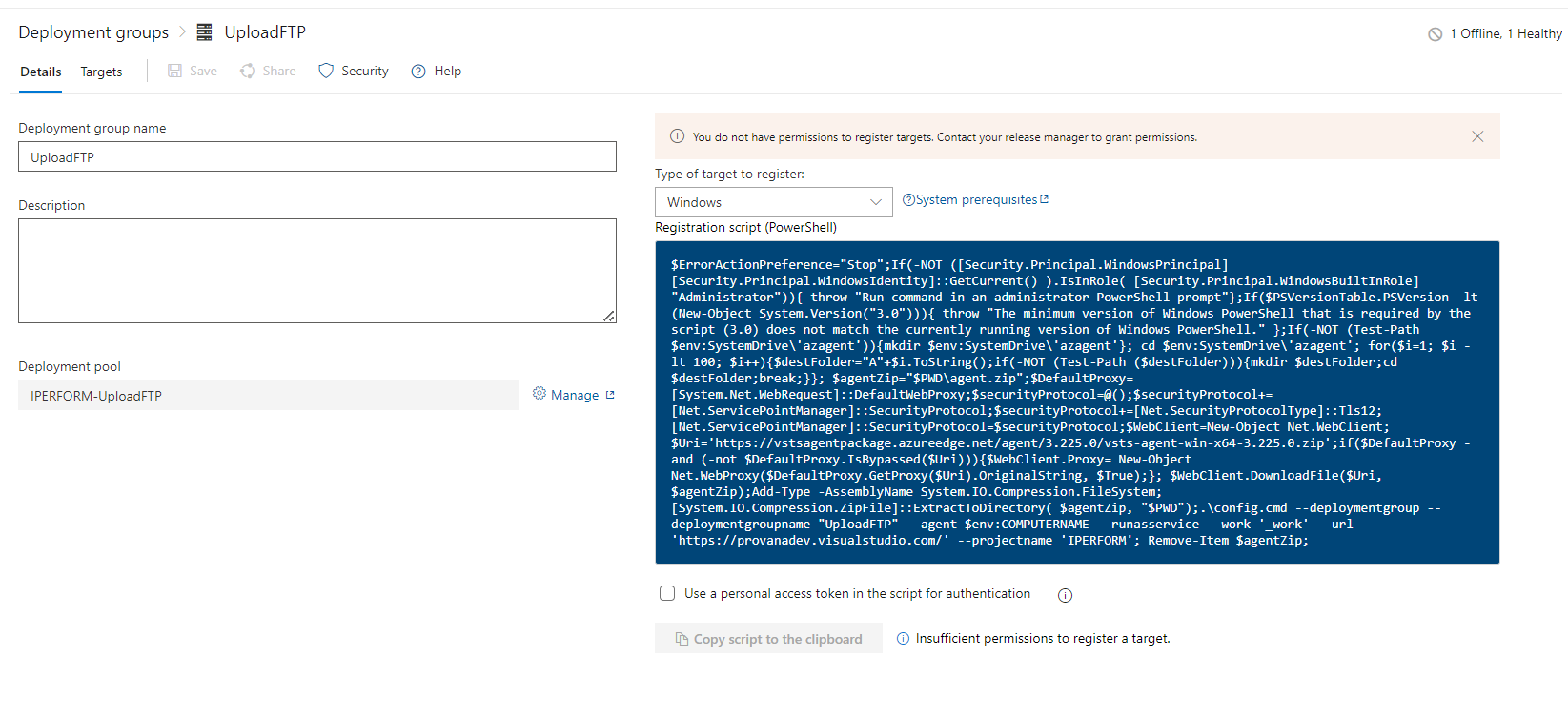
1. Create a deployment group.



1. Select Add a deployment group

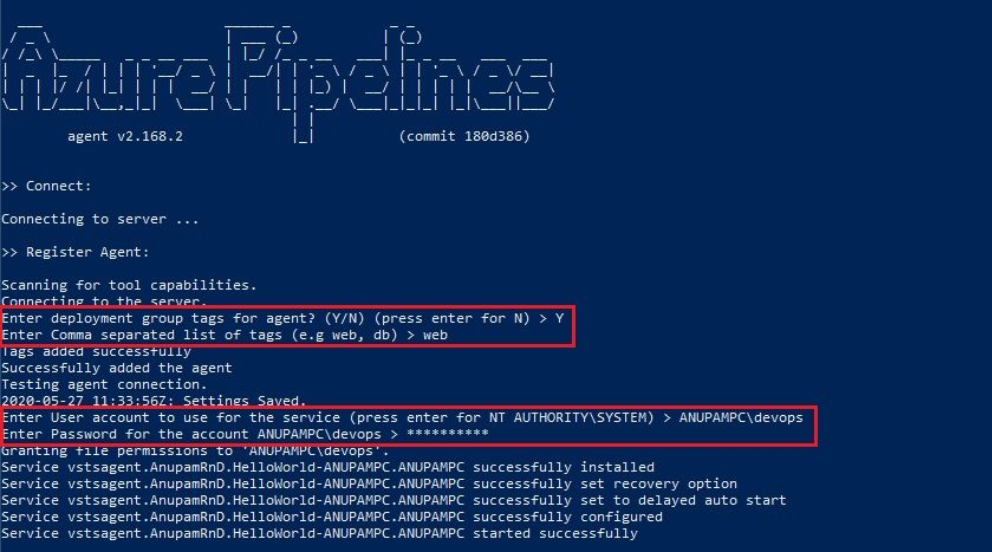


1. Enter a Deployment group name and then select Create. A registration script will be generated. Select the Type of target to register and then select Use a personal access token in the script for authentication. Finally, select Copy script to the clipboard.

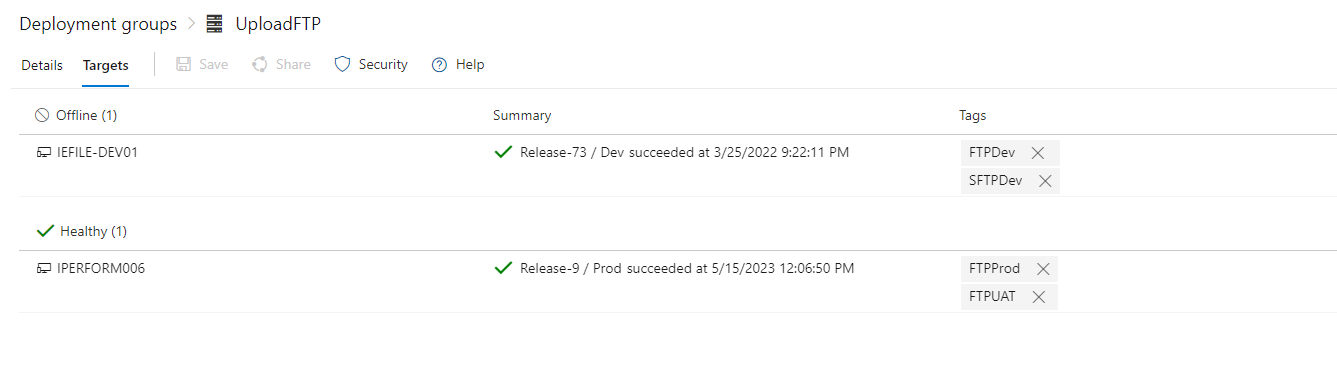


At this stage our deployment group is created but not online yet.

1. Log onto each of your target machines and run the script from an elevated PowerShell command prompt to register it as a target server. When prompted to enter tags for your agent, press Y and enter the tag(s) you will use to filter subsets of the servers.



At this stage, our deployment group creation gets completed and now if you go and check the status of the deployment group in the Azure DevOps, it should show as Online.



# Build Pipeline for IPERFORMUploadFTPtoAzure

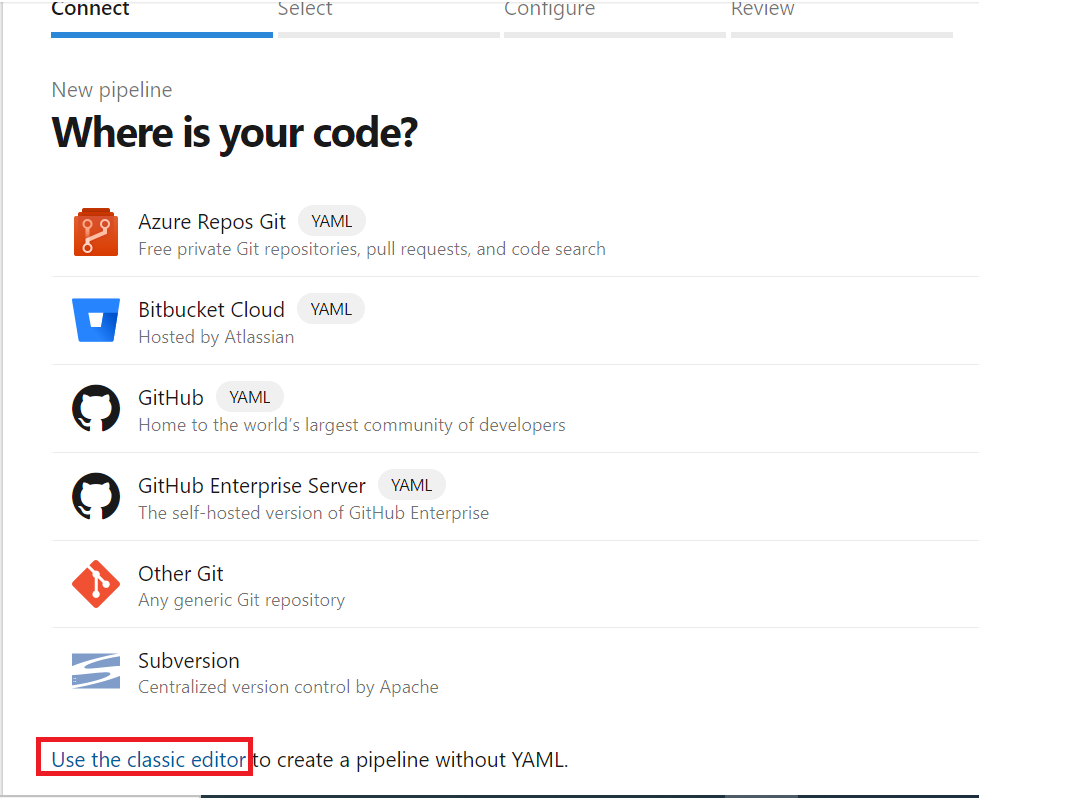
Steps:

1) Navigate the page to Pipelines and click on the “New Pipeline”.

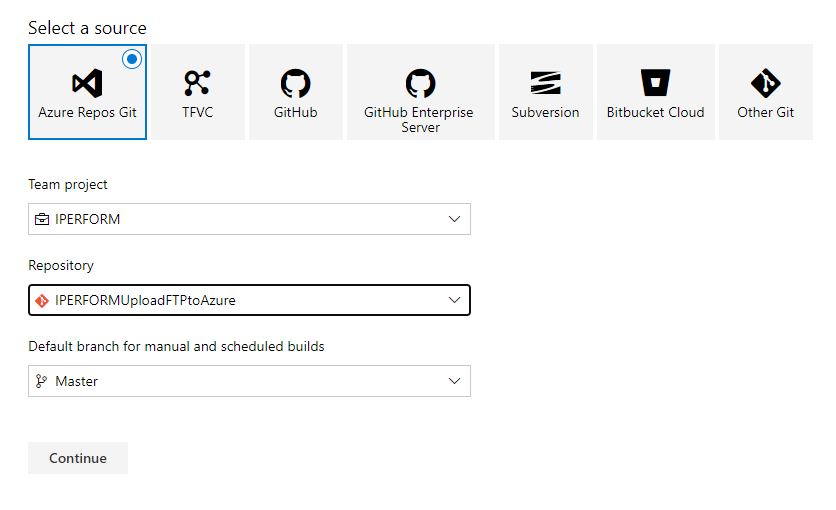
Graphical user interface, text, application, email

Description automatically generated

2) Click on the “Use the classic editor”.



3) Select the Source as “Azure Repos Git” and followed by Project, Repository and branch.



4)Select the template as “YAML”

Graphical user interface, text, application, email, Teams

Description automatically generated

## Pipelines:

We are using Azure pipelines (Microsoft hosted agent) as the agent pool for this pipeline. Microsoft-2019 OS version will be used in the agent.

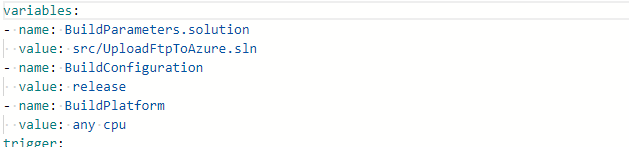
Background pattern

Description automatically generated with low confidence

## Variables:

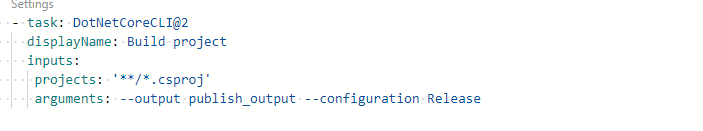
you can create variables that can be used in every step of the pipeline. For builds, you could set the username and password of an external service that you want to call. Azure DevOps already comes with a couple of predefined variables like the build configuration or platform.

|  |  |  |
| --- | --- | --- |
| Variable | Description | Used in Task |
| BuildPlatform | CPU configs | Build Solution |
| BuildConfiguration | Configuration | Build Solution |



### Tasks:

DotNetCoreCLI@2 - This task for Build, test, package, or publish a dotnet application, or run a custom dotnet command

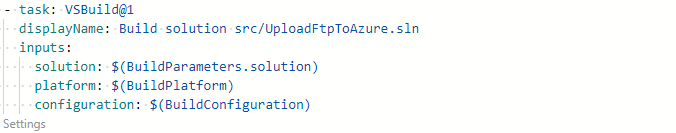


SonarQubePrepare@4 - This task use prepare a SonarQube analysis Configuration.

Graphical user interface, text, application

Description automatically generated

VSBuild@1 - The Visual Studio Build step builds your application using the solution file

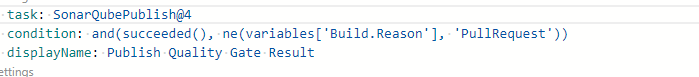


SonarQubeAnalyze@4 - This Use this task to run the scanner and upload the results to the SonarQube server.

A picture containing text

Description automatically generated

SonarQubePublish@4 - This Use this task to publish SonarQube's Quality Gate result on the Azure DevOps build result.



sonar-buildbreaker@8 - This task for has better error handling and you probably want to browse to SonarQube if this task makes your build fail.

Text

Description automatically generated

VSTest@2 - Use this task to run unit and functional tests (Selenium, Appium, Coded UI test, etc.

Graphical user interface, text, application

Description automatically generated

CopyFiles@2 - Use this task to copy files from a source folder to a target folder using match patterns.

Graphical user interface, text, application

Description automatically generated

# Publish Artifact

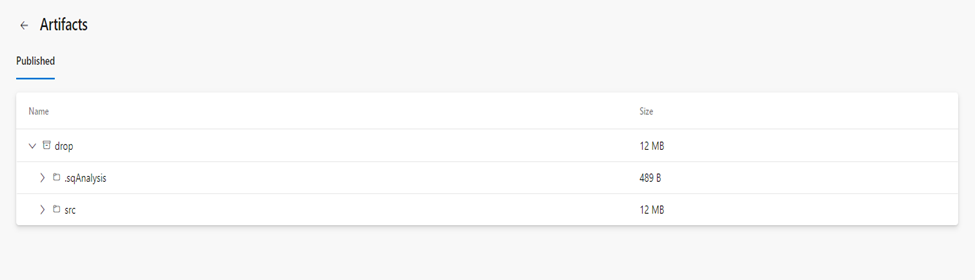
Publish Artifact is necessary for an automated deployment. This step publishes all the files which you want to deploy later in the release pipeline.

Graphical user interface, application

Description automatically generated

# Validation

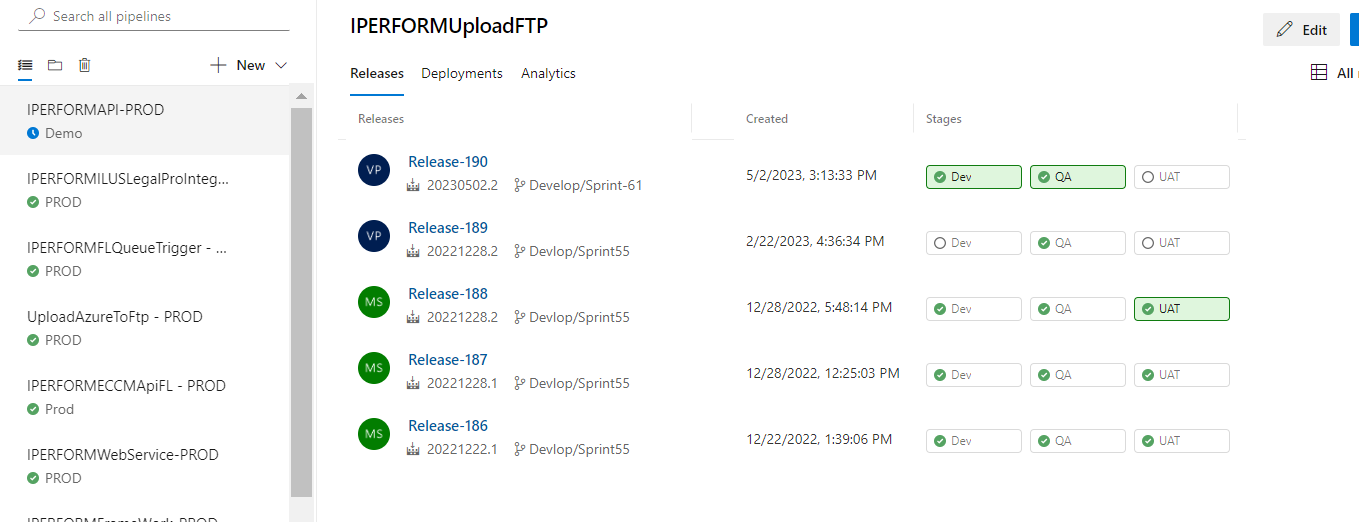
So, here “drop’ folder should include the following files:



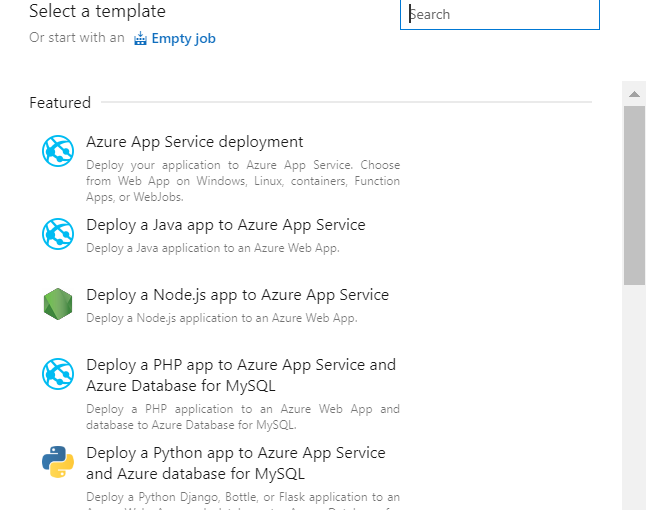
Release pipeline for IPERFORMUploadFTPtoAzure

## Steps:

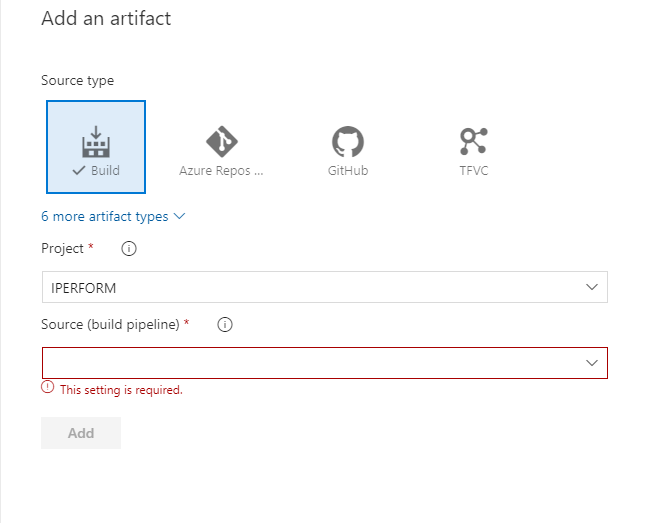
1. Navigate to release pipelines and click on “New“so as to create a new release pipeline which will be IPERFORMUploadFTP in this case.



2) Select an empty job because we need to add customized jobs later which will support our project’s needs and functionality.



3)Before customizing the jobs, we will need to add the artifact that was built through our build pipeline. So, click on “Add an artifact” and then select the build pipeline which produced the artifact that is going to get used in this release pipeline.



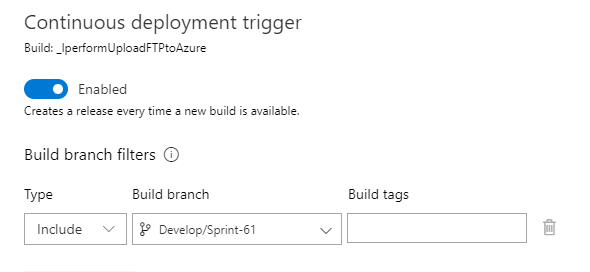
4) As soon as the artifact is added, we must customize jobs according to the functionality of the project.

Graphical user interface, application, PowerPoint

Description automatically generated

## Continuous deployment trigger:

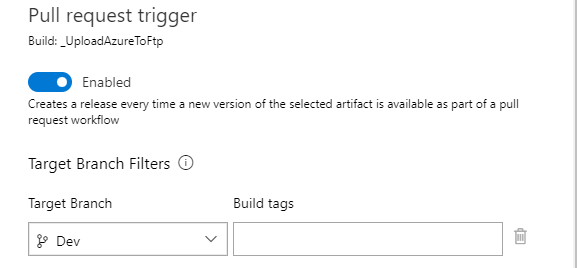
Continuous deployment triggers allow you to create a release every time a new build artifact is available.



Note -A release will be triggered only for a build that is from one of the branches selected here.  For example, selecting "Develop/Sprint-61" will trigger a release for every build from the Develop/Sprint-61 branch.

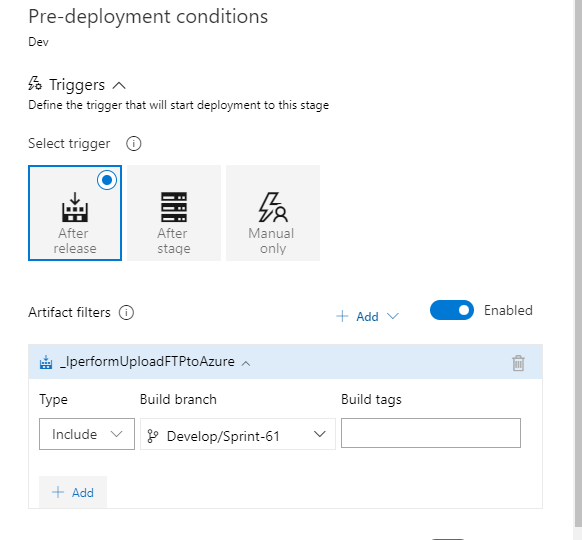
## Pull request trigger

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



## Pre-deployment conditions :

Pre-deployment gates ensures there are no active issues in the work item or problem management system before deploying a build to an environment.

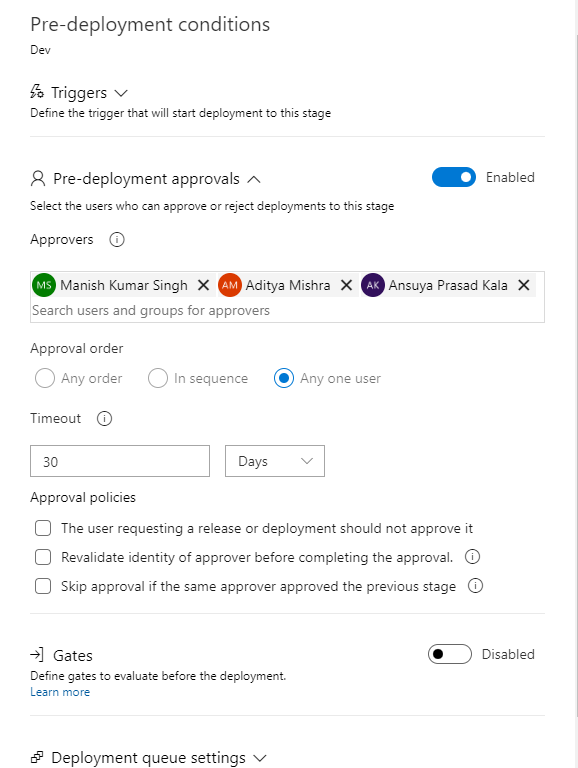


Note –

* Triggers - Set the trigger that will start the deployment to this stage automatically. Select "Release" to deploy to the stage every time a new release is created. Use the "Stage" option to deploy after deployments to selected stages are successful. To allow only manual deployments, select "Manual".
* Artifact filters - Select artifact condition(s) to trigger a new deployment. A release will be deployed to this stage only if all artifact conditions match.

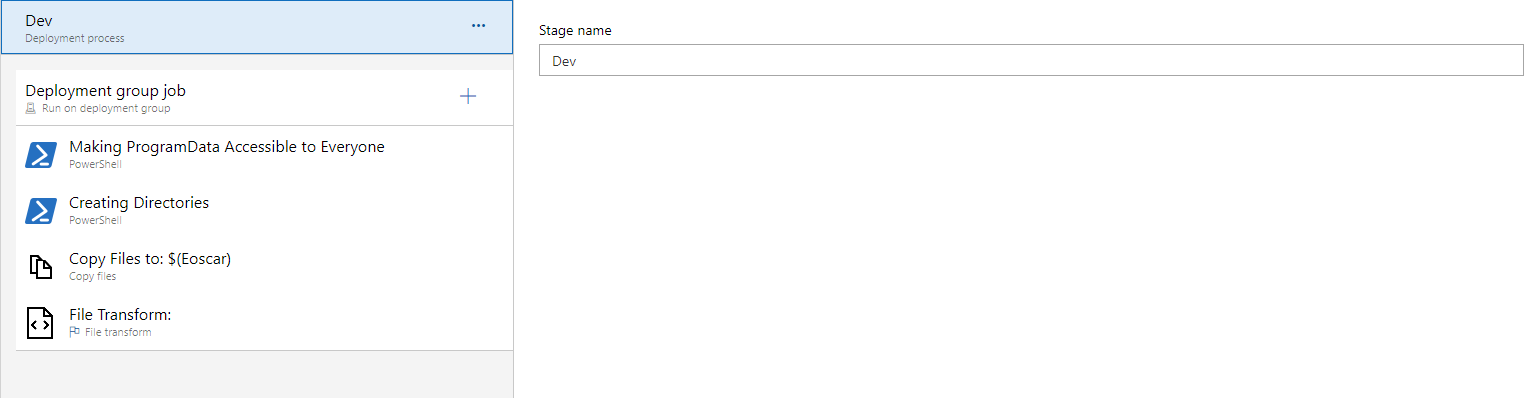
Pre-deployment approvals –

Select the users who can approve or reject deployments to this stage.



## Tasks:

1. We must provide Stage Name, Azure subscription and App type and Name.



## Agent Job:

A job is a logical grouping of tasks that defines the runtime target on which the tasks will execute. An agent job executes tasks on an agent in an agent pool.

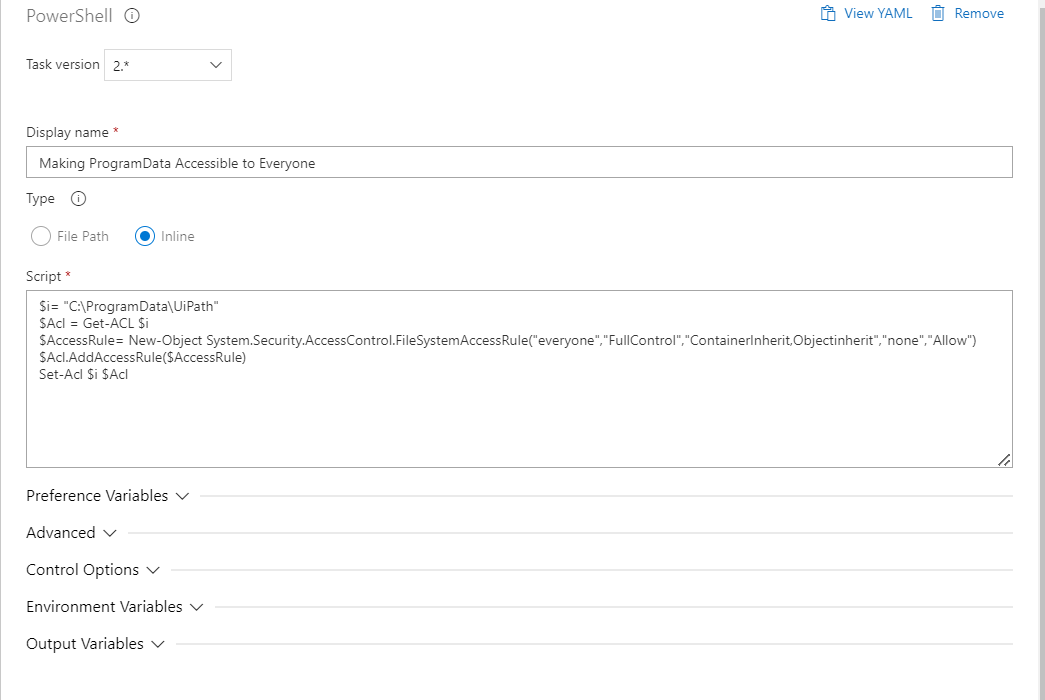
Graphical user interface, text, application, email

Description automatically generated

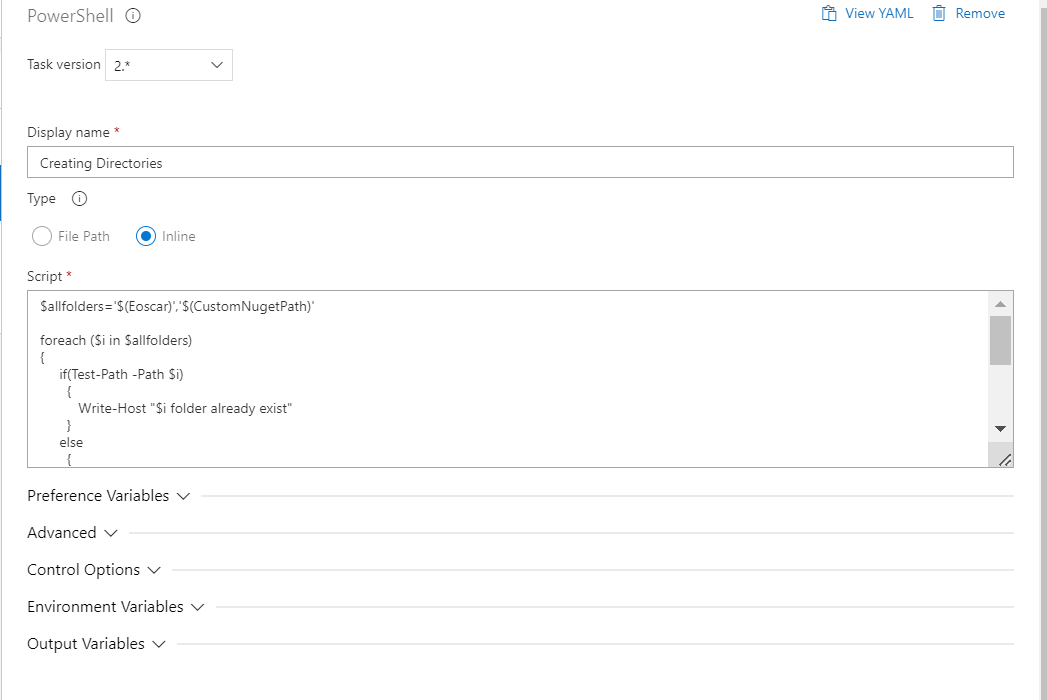
## Azure powershell script:Inline Script

The PowerShell task allows you to add PowerShell code directly within the YAML pipeline or execute an existing script in the source repo .

1. Making ProgramData Accessible to Everyone.



1. Create Directories - This task for creating Directories.



## Copy files –

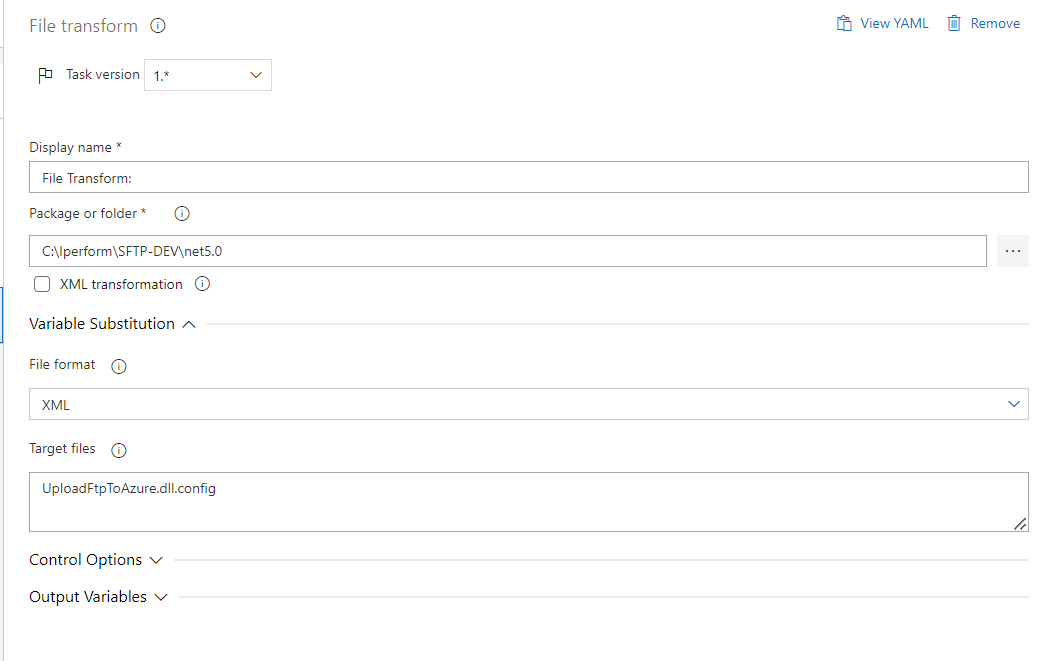
Copying of files from source folder to target folder. $(system.defaultworkingdirectory) to $(UiPath Studio.artifactstagingdirectory).

Graphical user interface, text, application, email

Description automatically generated

## File Transform –

Use this task to replace tokens with variable values in XML or JSON configuration files.



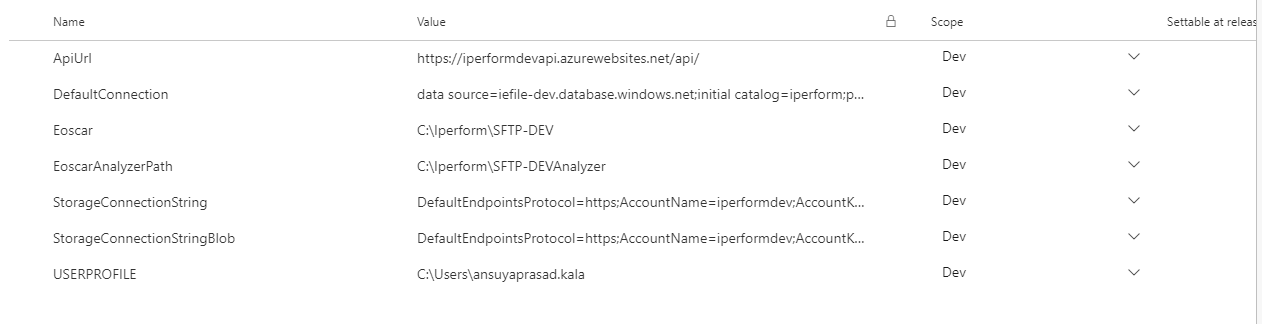
## Variables:

1. StorageConnectionString - we need to add connection string.

Here it is, DefaultEndpointsProtocol=https AccountName=iperformdev; AccountKey=kxKXB7LHbK0iulW8uvqBud7OBsmVDZJGkaRFdNeJi0Oz9jr+Rw8f5ZNeIlHafzowFeV3Z3rqSuiGGTXM7+VJMA==; EndpointSuffix=core.windows.net;

Note – The initial catalog will change according to the environment that we want to deploy.

2)Key-Value pair are:



Validation

Once the release is successful, check on VM FTP functionality is working fine.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| **05/3/2023** | 1.0 | **Draft and Initial** | **Atul Sikarwal** |
| 05/03/2023 | 1.1 | **Review** | Sourabha M |