Documentation of Azure Functions

Version Number:1.1

Table of Contents

[Introduction: 2](#_Toc1314551833)

[Scope: 2](#_Toc1969515250)

[CI/CD Architecture: 2](#_Toc1787246206)

[Build Pipeline for UploadAzureToFtp\_AzureFunction 3](#_Toc128591569)

[Pipeline: 5](#_Toc385876262)

[Tasks: 5](#_Toc1489042803)

[SonarQubePrepare@4 -This task use prepare a SonarQube analysis Configuration. 5](#_Toc1312423341)

[Release Pipeline for UploadAzureToFtp\_AzureFunction 8](#_Toc1199100421)

[Continuous deployment trigger: 10](#_Toc889557009)

[Pull request trigger 11](#_Toc1169042292)

[Pre-deployment conditions : 11](#_Toc594185773)

[Agent: Provisioning 13](#_Toc1501687498)

[Deployment Process: 14](#_Toc1127411292)

[Azure function: 14](#_Toc1194900763)

[Validation: 15](#_Toc972239286)

[Revision History 16](#_Toc1459450986)

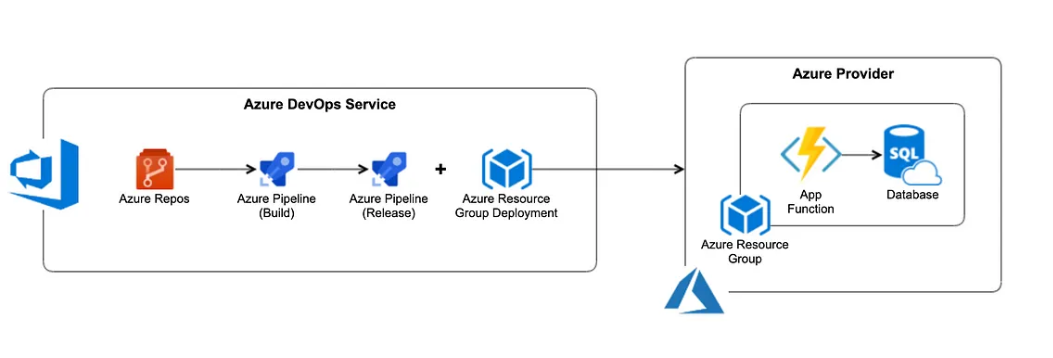
# Introduction:

This is the documentation for the release pipelines for Azure Function. 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:



Build Pipeline for UploadAzureToFtp\_AzureFunction

This pipeline - UploadAzureToFtp\_AzureFunction will demonstrate the build process for UploadAzureToFtp\_AzureFunction repository.

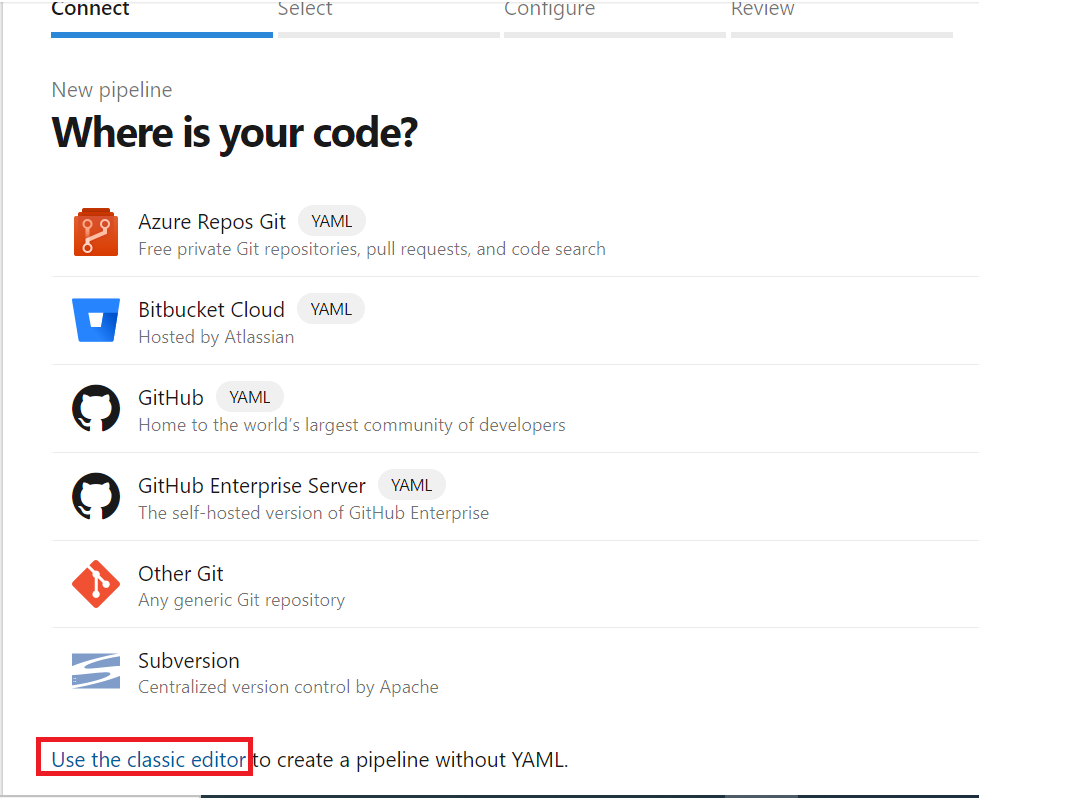
Steps:

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”.

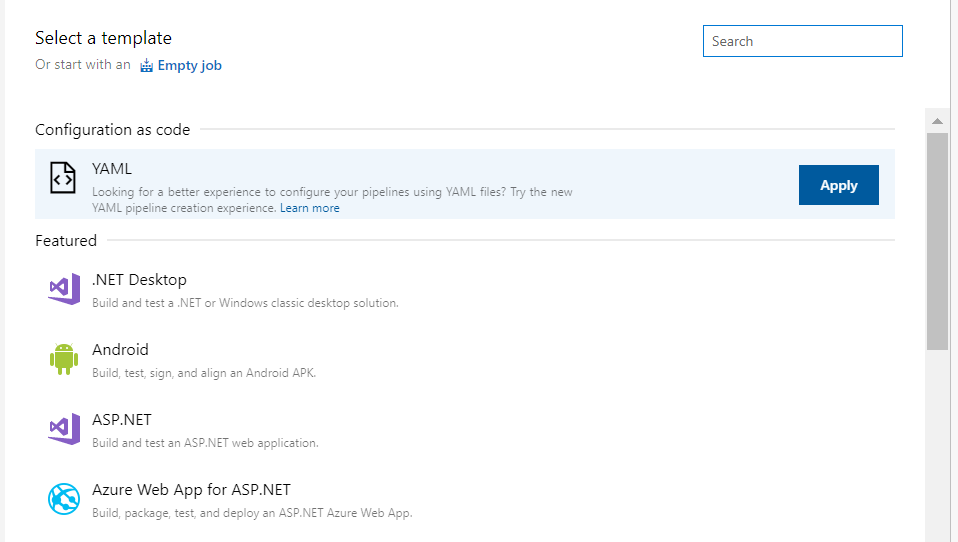


3) Select the Source as “Azure Repos Git” and followed by Project, Repository and branch.

Graphical user interface, text, application, email

Description automatically generated

4)Select the template as “YAML”



## Pipeline:

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

Background pattern

Description automatically generated with low confidence

## Tasks:

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

Graphical user interface, application

Description automatically generated with medium confidence

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

A picture containing graphical user interface

Description automatically generated

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

Graphical user interface, application

Description automatically generated

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

A picture containing text

Description automatically generated

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.

Graphical user interface

Description automatically generated

ArchiveFiles@2 -This task use for create an archive file from a source folder.

Graphical user interface, text, application

Description automatically generated

PublishBuildArtifacts@1

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, text

Description automatically generated with medium confidence

Trigger :

Graphical user interface, application

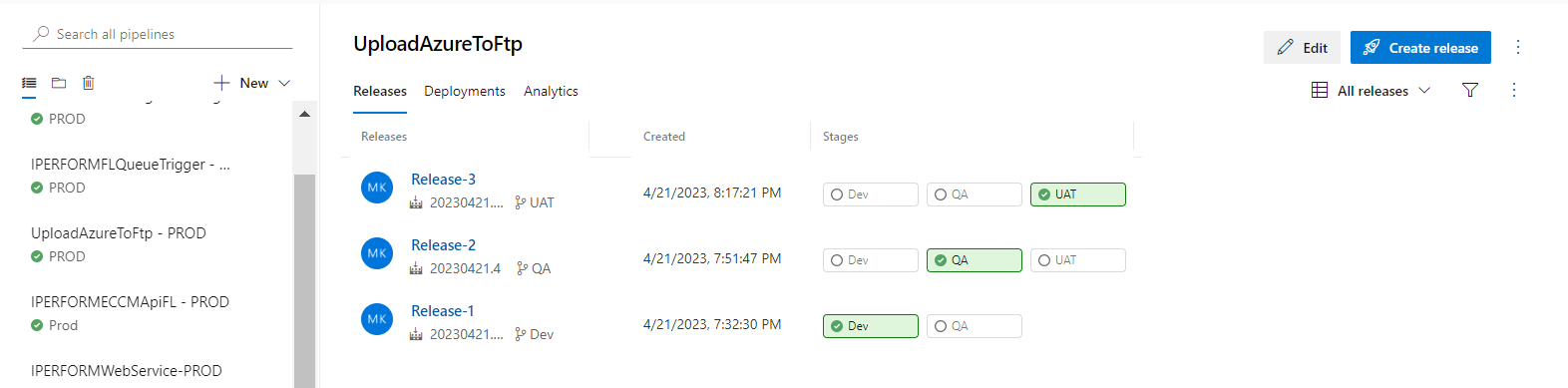
Description automatically generated

# Release Pipeline for UploadAzureToFtp\_AzureFunction

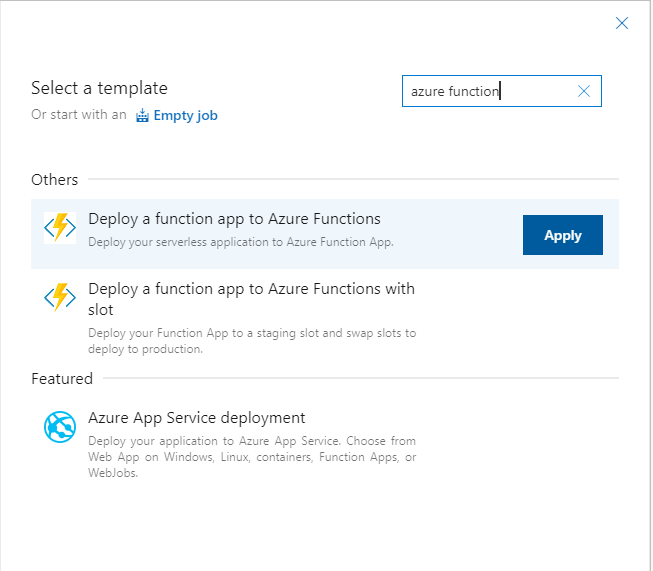
This pipeline will demonstrate the release pipeline of IPERFORMInputValidationTrigger

Steps:

1. Navigate to release pipelines and click on "New" to create a new release pipeline, which will be IPERFORMInputValidationTrigger in this case.



1. Select an deploy a function app to azure function , 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 that produced the artifact that is going to be used in this release pipeline.

Graphical user interface, application, email

Description automatically generated

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

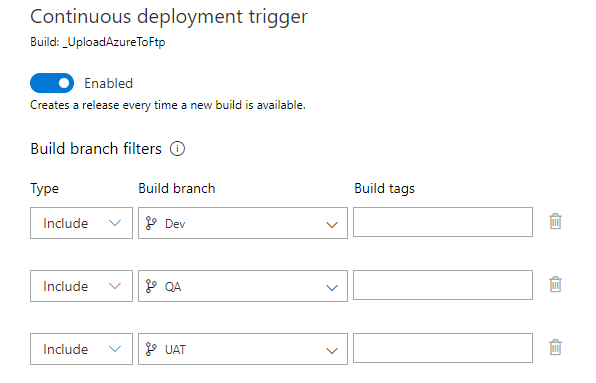
Graphical user interface, text

Description automatically generated

5) The next step is to select the jobs that are needed.

## Continuous deployment trigger:

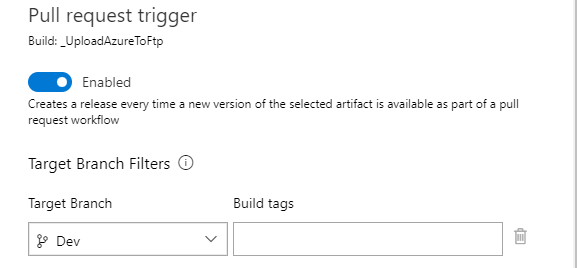
Continuous deployment triggers allow you to create a release every time a new build artifact is available.



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

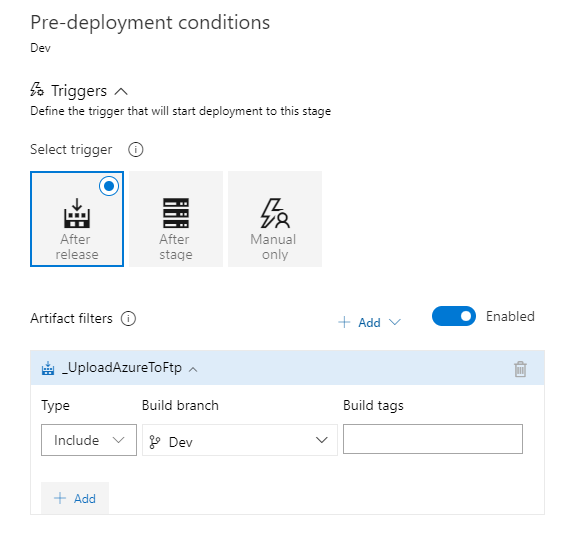
## Pull request trigger

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.

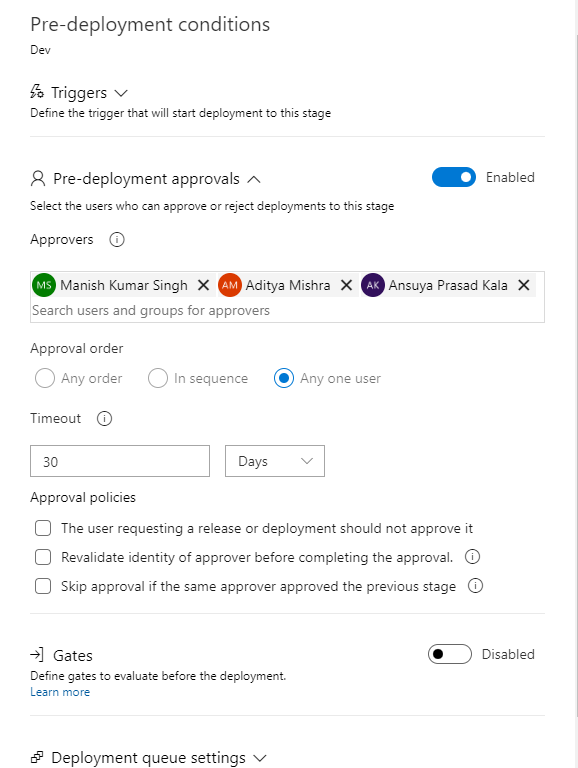


Note –

* 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.

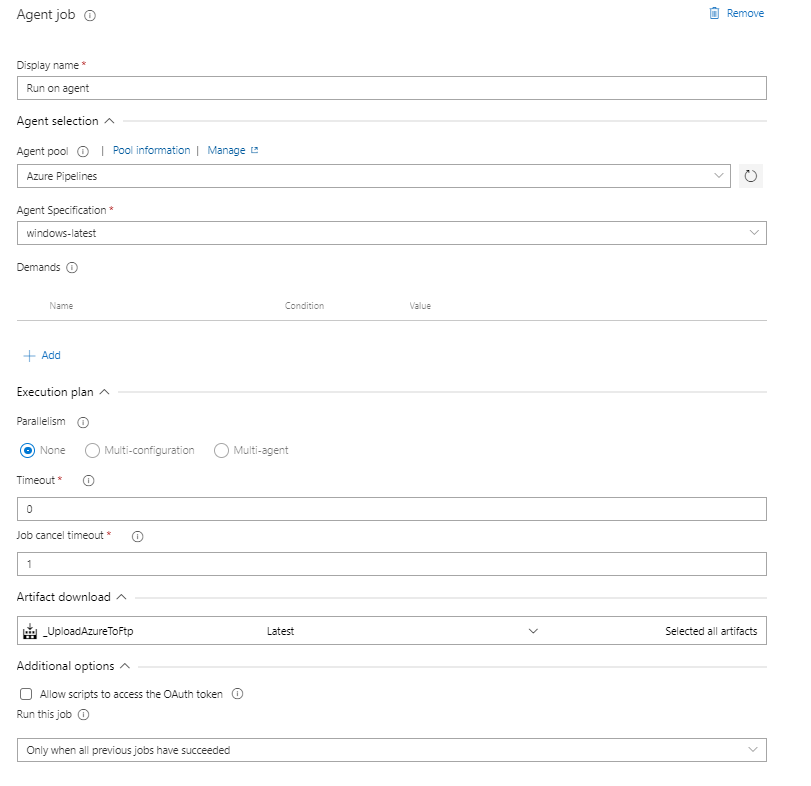
Pre-deployment approvals –

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



## Agent: Provisioning

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



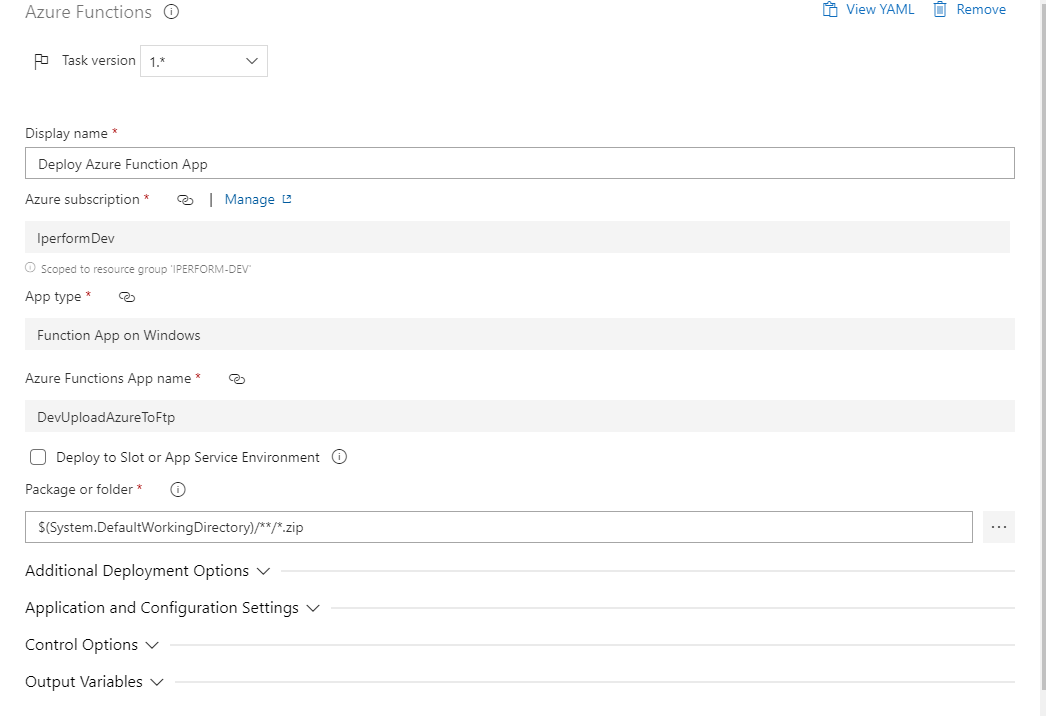
## Deployment Process:

A deployment source is the location of your application code. For production apps, the deployment source is usually a repository hosted by version control software



## Azure function:

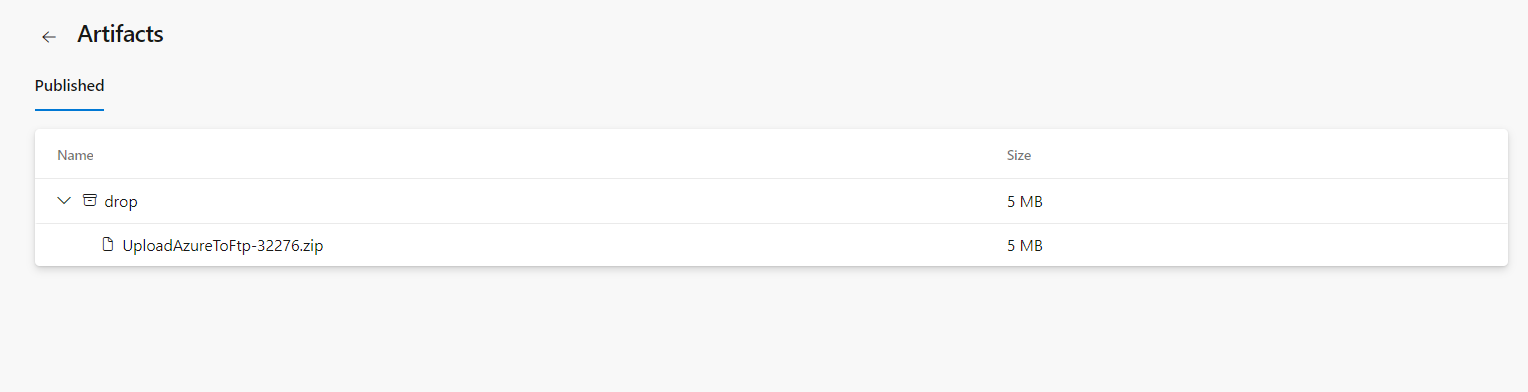
Azure Functions is a cloud service available on-demand that provides all the continually updated infrastructure and resources needed to run your applications. You focus on the code that matters most to you, in the most productive language for you, and Functions handles the rest.



# Validation:

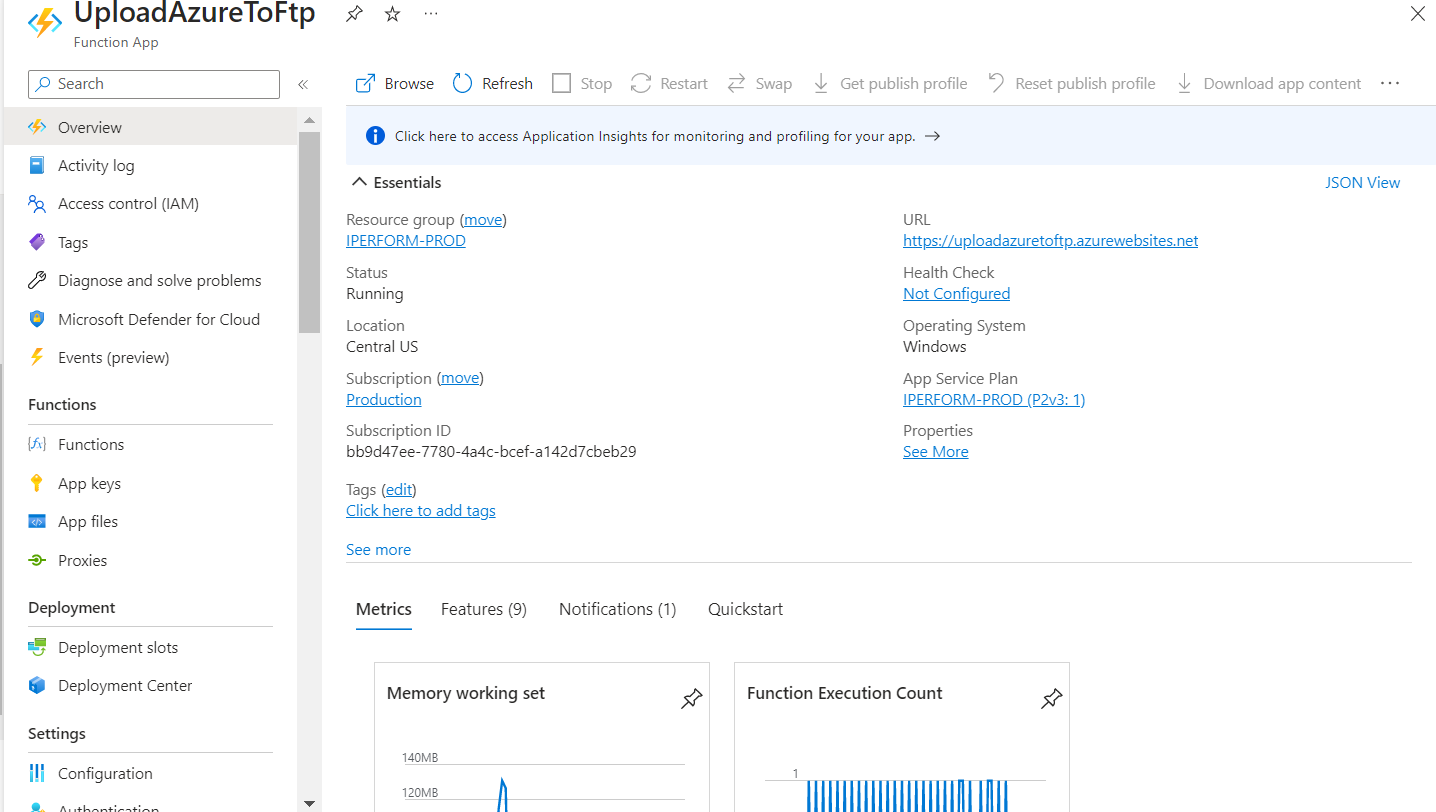
Build Pipeline:

* Once the build is successful, check the generated artifacts. Artifacts should contain the following:



Release Pipeline:

Once the release is successful, it will start running automatically and iperform main website can consume this azure function.



# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| **04/28/2023** | 1.0 | **Draft and Initial** | **Atul Sikarwal** |
| 05/2/2023 | 1.1 | **Review** | Sourabha M |