|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

How-To-Do Setup DevOps Process

|  |  |
| --- | --- |
| **Purpose** | This document describes in detail how to setup DevOps pipelines for Continuous Build and Deployment process on Azure DevOps service for publishing PowerBI reports |
| **Document Version** | v0.1 |

Document Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Reviewed by** | **Comments** |
| 0.1 | Ronnie Kapoor | 22-02-2019 |  | Initial Draft |

Contents

[1 Introduction 4](#_Toc1743885)

[1.1 Management Summary 4](#_Toc1743886)

[2 Implementation Process 5](#_Toc1743887)

[2.1 Pre-Requisites 5](#_Toc1743888)

[2.1.1 Softwares/Services 5](#_Toc1743889)

[2.1.2 Access rights 5](#_Toc1743890)

[2.2 Setting up Azure DevOps Organization 5](#_Toc1743891)

[2.3 Setting up a new project in an organization 6](#_Toc1743892)

[2.4 Setting up the repository in the project 7](#_Toc1743893)

[2.5 Set up new branches on the Repository 8](#_Toc1743894)

[2.6 Set up repository for domain wise use. 9](#_Toc1743895)

[2.7 Setting up Azure Build Pipeline 12](#_Toc1743896)

[2.8 Setting up Azure Deployment Pipeline 25](#_Toc1743897)

# Introduction

## Management Summary

This document describes in detail how to setup DevOps pipelines for Continuous Build and Deployment process on Azure DevOps service for publishing PowerBI reports.

# Implementation Process

## Pre-Requisites

### Softwares/Services

Below are the softwares and services which are required to set up complete DevOp Continuous Build and Deployment pipelines

Azure DevOps Service (Basic Level Access). For more information on licensing of Azure DevOps Service please visit [this](https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/) link.

### Access rights

Admin access rights on PowerBI API on Azure Active Directory application

AD user with Admin access rights for setting up the PowerBI service connection in Azure DevOps

## Setting up Azure DevOps Organization

The first step to get started up with Azure DevOps is to create an Organization on Azure Dev Portal - <https://dev.azure.com/>

Click on the link above and login to the portal using Radiometer account ID. Create the new organization by clicking on new organization as shown below.

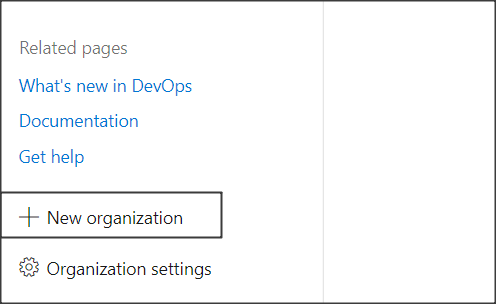


Figure 1: Setting up the new organization

Click continue and enter the name of the organization you want to create.

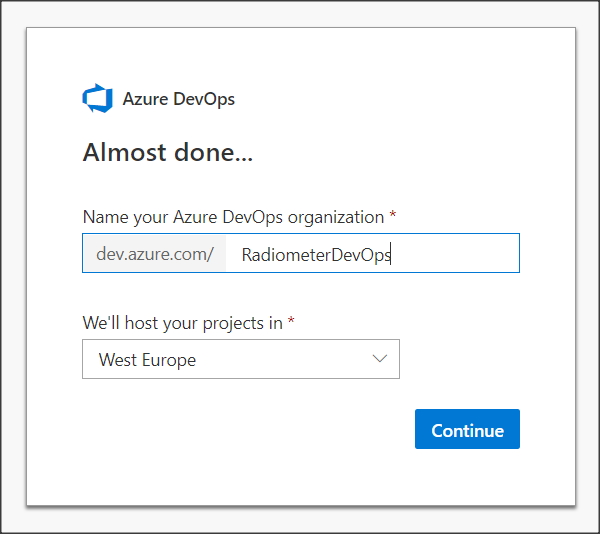


Figure 2: Setting up the new organization

Now that the organization has been created, it can be used to set up projects where DevOps pipelines and processes can be set up. More information on how to create an organization can be found [here](https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/create-organization?view=azure-devops).

## Setting up a new project in an organization

Organization serves as an umbrella to host innumerable number of projects on which teams can work. Continuous Build and Deployment pipelines set up can be done in a project.

After creating an organization, the page to create a project will appear automatically as the home page of the organization as shown in the snap shot below.

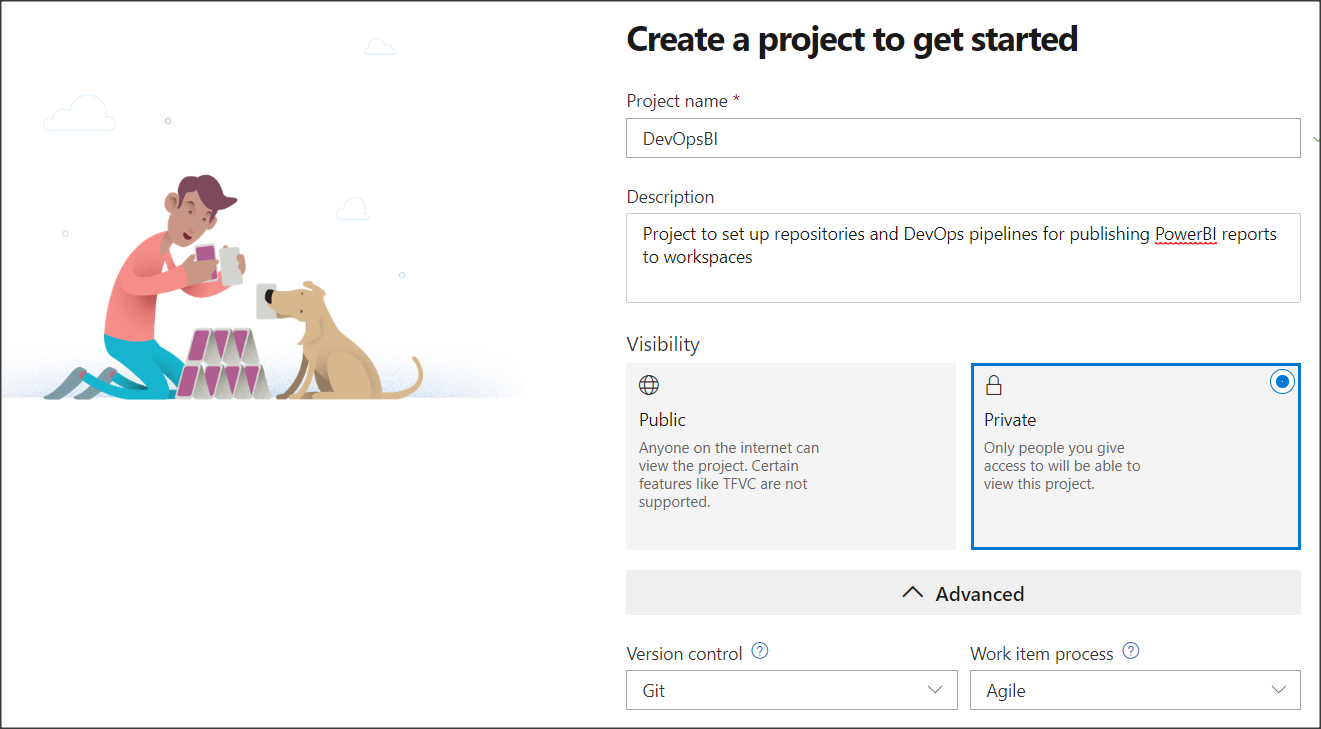


Figure 3: Setting up a new project

Type the name of the project (DevOpsBI in snap shot) as desired and the description accordingly. Choose ***private*** as Visibility and ***Git*** in Version control and ***Agile*** in Work Item process in the options available in Advanced Tab.

Click on create and the project will be created. You can find more information on how to create a project on this [link](https://docs.microsoft.com/en-us/azure/devops/organizations/projects/create-project?view=azure-devops).

New members can be added to this project by clicking on invite button as shown below.

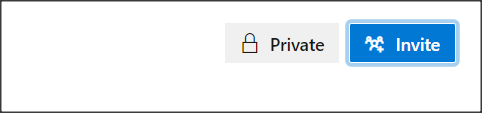


Figure 4: Adding a new member to the team

Search for the user using their information and add them to the team

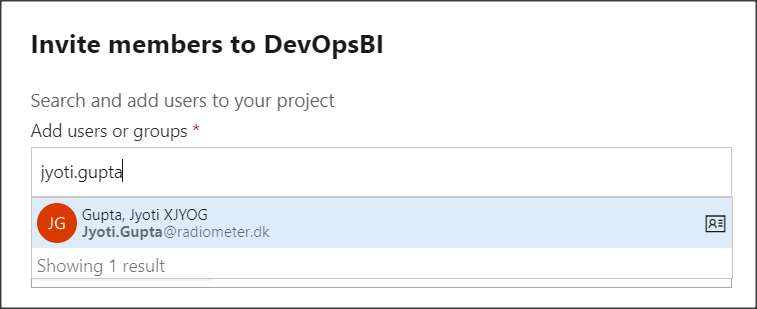


Figure 5: Example of adding a member

More information on how to add users can be found on [this](https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/add-organization-users?view=azure-devops) link.

## Setting up the repository in the project

Click on the Repos from left menu and initialize the Repo as shown below.

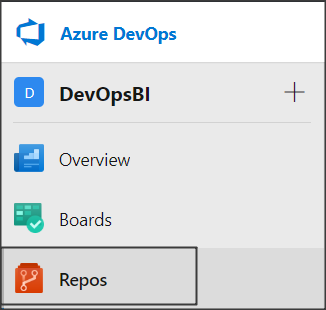


Figure 6: Setting up the azure repository

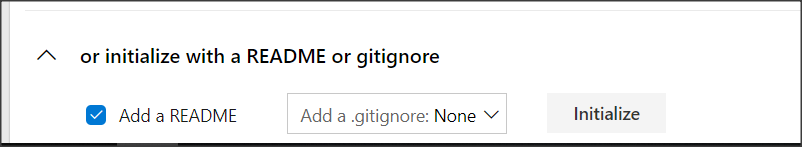


Figure 7: Initialize a new repository

## Set up new branches on the Repository

The proposed solution demands creating branches on the repository based on Environments available which are INT, UAT and PROD (Production)

To create the branches please follow below steps

Go to create branch from the branches drop down

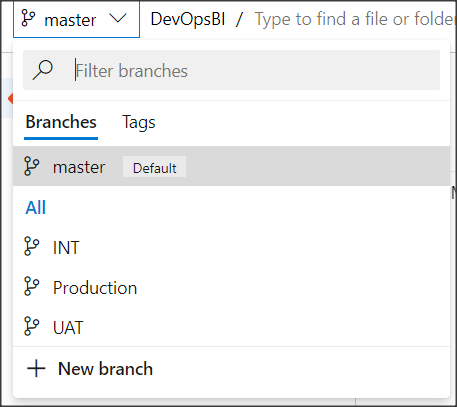


Figure 8: Creating a new branch

Name the new branch as desired

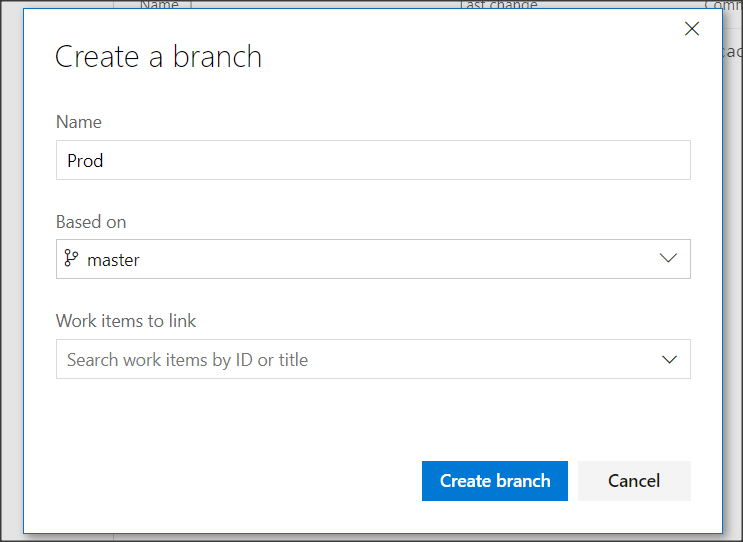


Figure 9: Naming a branch

By the end you should have three branches as required

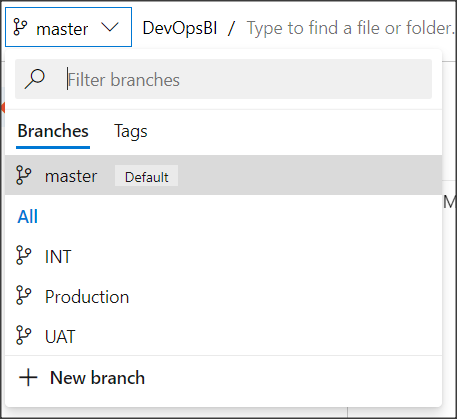


Figure 10: Showing all branches

## Set up repository for domain wise use.

Once all the branches have been created, it needs to be cloned to a developer’s machine and for that Git needs to be installed on the machine (assuming it is windows OS)

Please download and install Git from [this](https://git-scm.com/download/win) link if the Git is not installed already on the system the repository is being set up.

Copy the clone URL from the repository from Azure DevOps portal and clone the repository branch wise on the system.

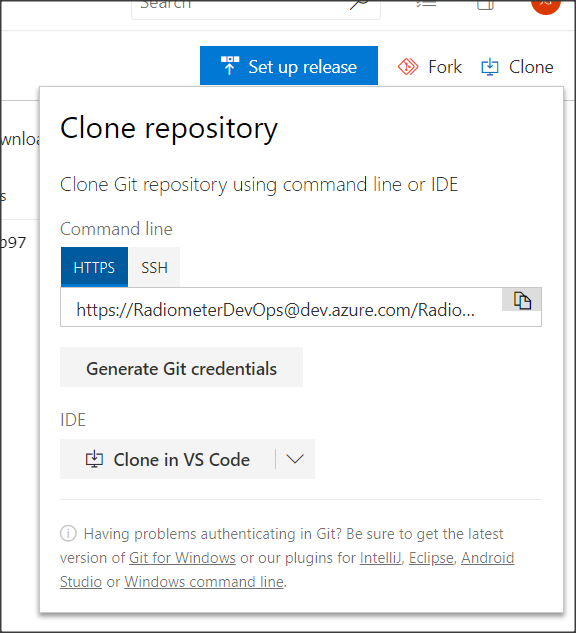


Figure 11: Copying the clone URL

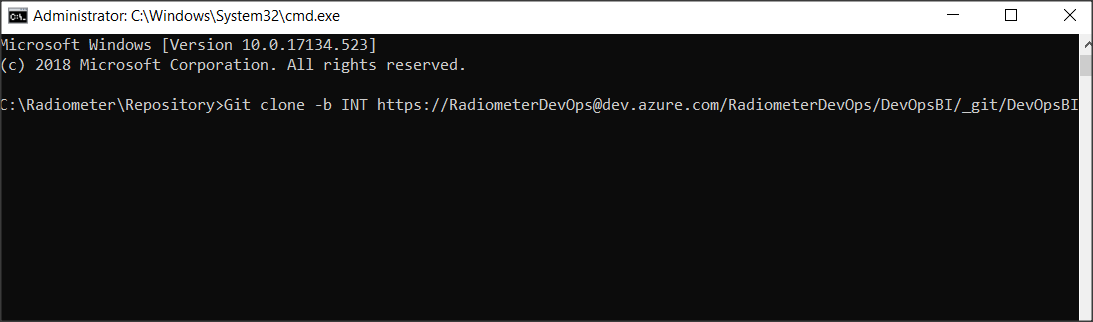


Figure 12: Cloning the Repository to a system

***Note****- The path on which the repository is cloned on a system can be any path.*

Example directory structure after cloning of all the branches can look like this

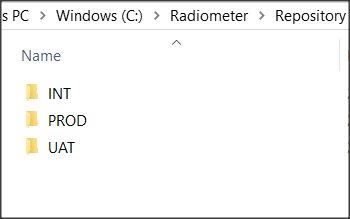


Figure 13: Example directory structure

Please refer to [this](https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell) link to know more of Git branches and how it works.

Commit the needed reports in the folder structure as desired for different domains In different folder structure. (***Note*** *- This folder structure is based on the requirement and can be changed if needed)*

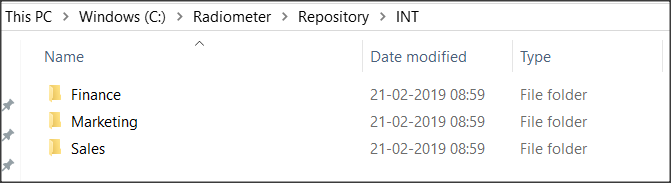


Figure 14: Example folder structure inside the repository

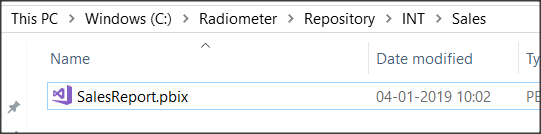


Figure 15: Example report inside sales folder

Once all the files have been added to the respective branch of the repository, the files are committed to Azure Repo with the help if Git commands.

Go to Command prompt and navigate to the INT branch folder of the repository. For e.g. cd C:\Radiometer\Repository\INT

Run command “*git Add .*”

Run command “*git commit -m ‘<your message for the commit>’* ”. For e.g. *git commit -m ‘adding INT reports’*

Run command “*git push*”

Once the files have been committed and pushed to the repository, it will show up in the Azure Repos as shown below

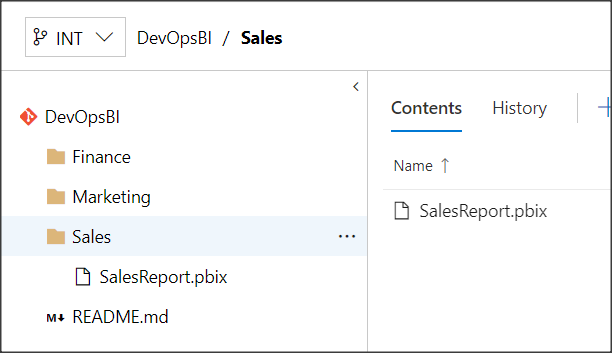


Figure 16: Files shown on Azure Repos

Follow similar steps in future for other domain folders as well.

## Setting up Azure Build Pipeline

To set up Continuous build process for Radiometer PowerBI application, a build definition needs to be created first targeting the reports to be published to PowerBI and generating the Artifacts of the reports in the build. The tasks needed to set up are shown below.

Create a Task Group for copying and publishing the domain wise report Artifacts to Azure pipeline Artifacts.

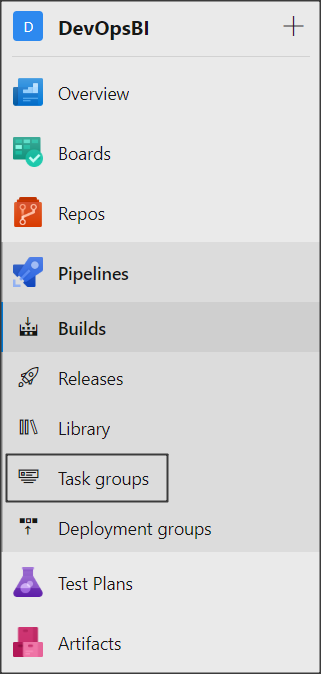


Figure 17: Setting up a task group

Import a task group which from a json file which is attached below and click on save.



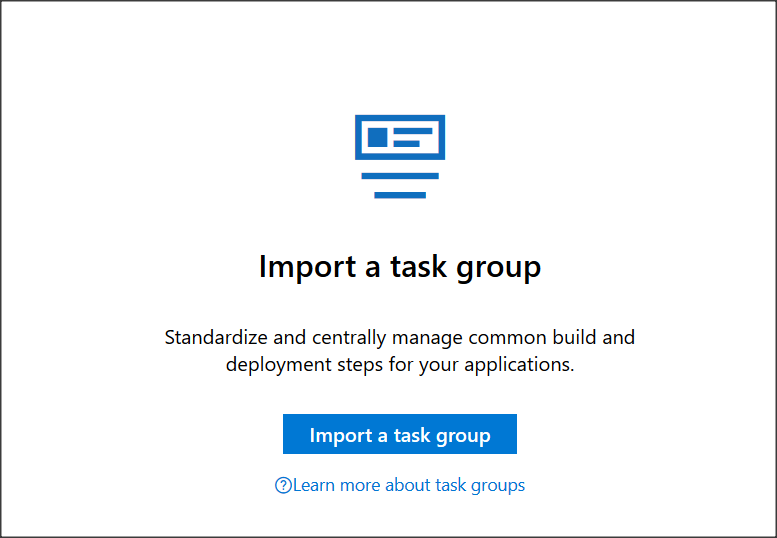


Figure 18: Importing a task group

This task group contains two tasks

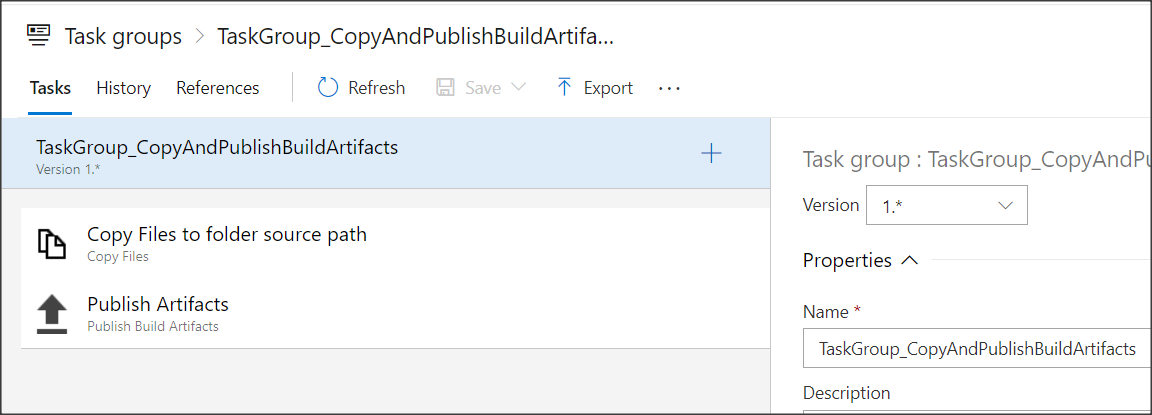


Figure 19: Task group

Copying domain wise reports from source to target folder for generating artifacts

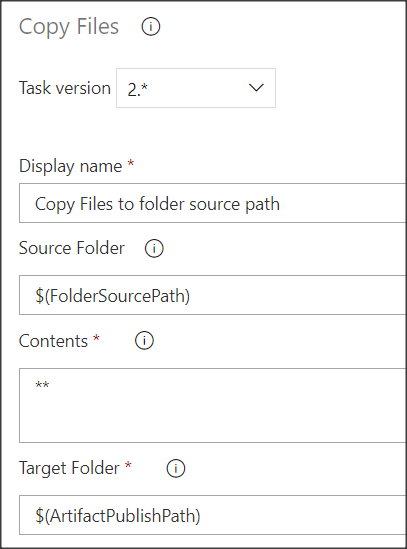


Figure 20: Copy source files

Generating domain wise artifacts from target folder to Azure pipelines

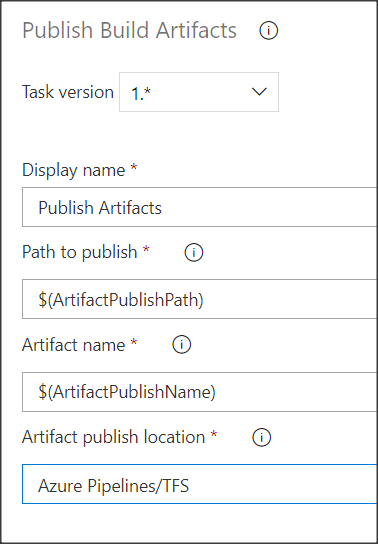


Figure 21: Publishing build artifacts

Create a build pipeline

Go to builds from the left menu and create a new build pipeline

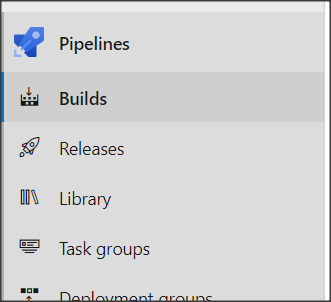


Figure 22: Build pipeline option in menu

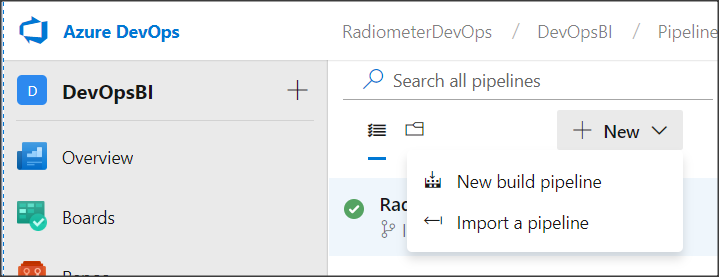


Figure 23: Create a new build pipeline

Select the visual designer in the below option as shown

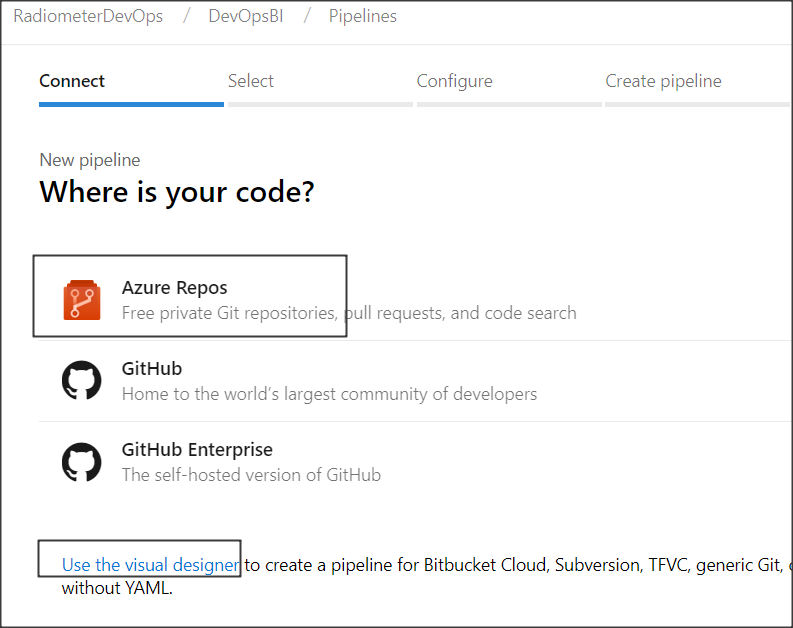


Figure 24: Selecting Azure repos and using visual designer

Select DevOpsBI Repository and default branch as INT and click on Continue

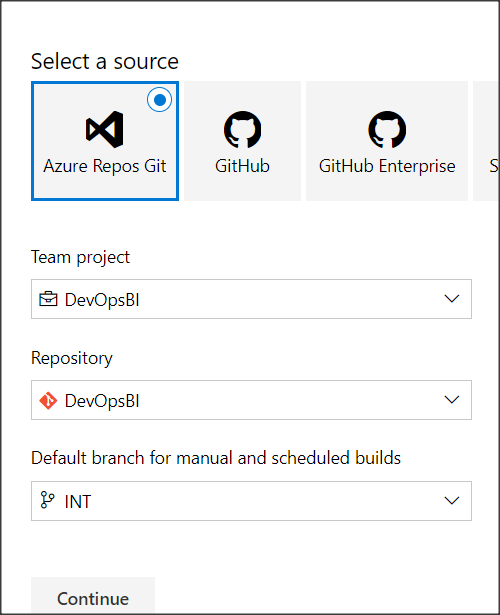


Figure 25: Selecting Azure repo for pipeline

Select an Empty Job in the template in next option

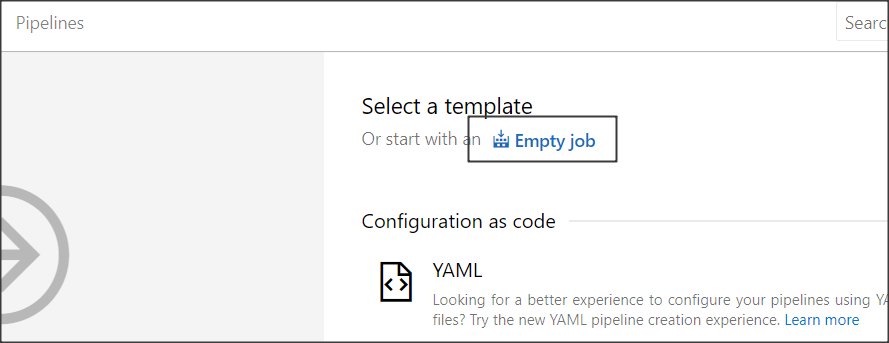


Figure 26: Select Empty job in template

Name the pipeline accordingly and choose Hosted VS2017 as agent pool

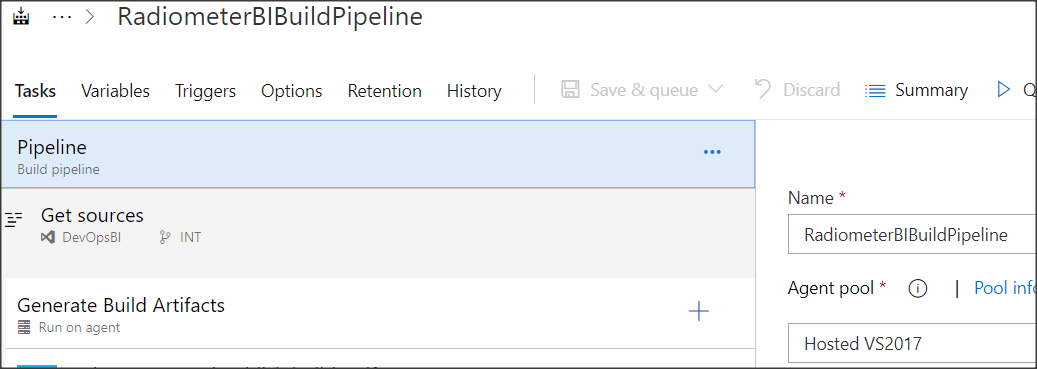


Figure 27: Naming the pipeline

Name first agent job in the pipeline as “*Generate Build Artifacts*” and add a task for task group which is created in above steps



Figure 28: Add a task in pipeline

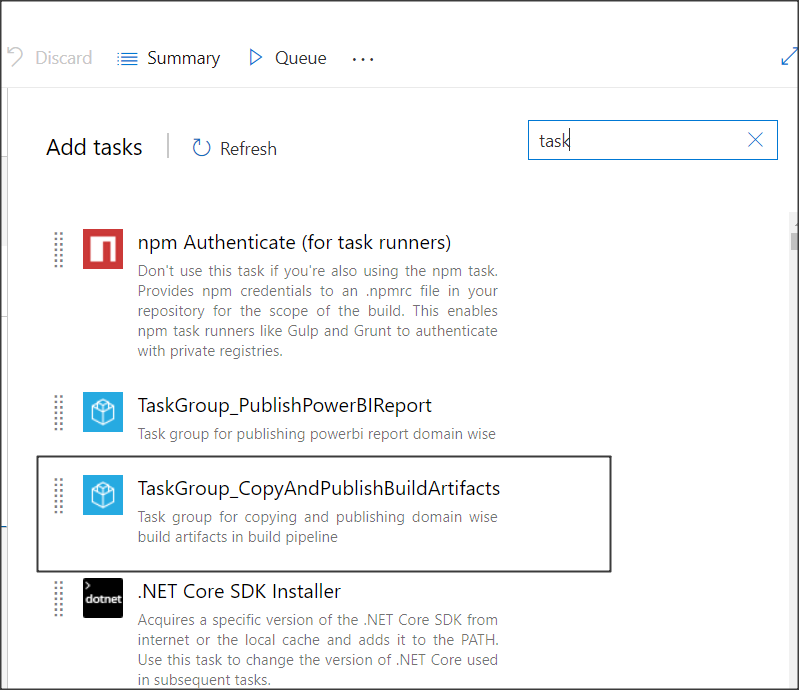


Figure 29: Add a task group task in pipeline

The task group requires inputs on three parameters

1. ArtifactPublishName - The name of the artifact to create in the publish location.
2. ArtifactPublishPath - Target folder or UNC path files will copy to.
3. FolderSourcePath - The source folder that the copy pattern(s) will be run from. Empty is the root of the repo

These values will be provided with the help of custom build variables to be created in Variables tab as shown below

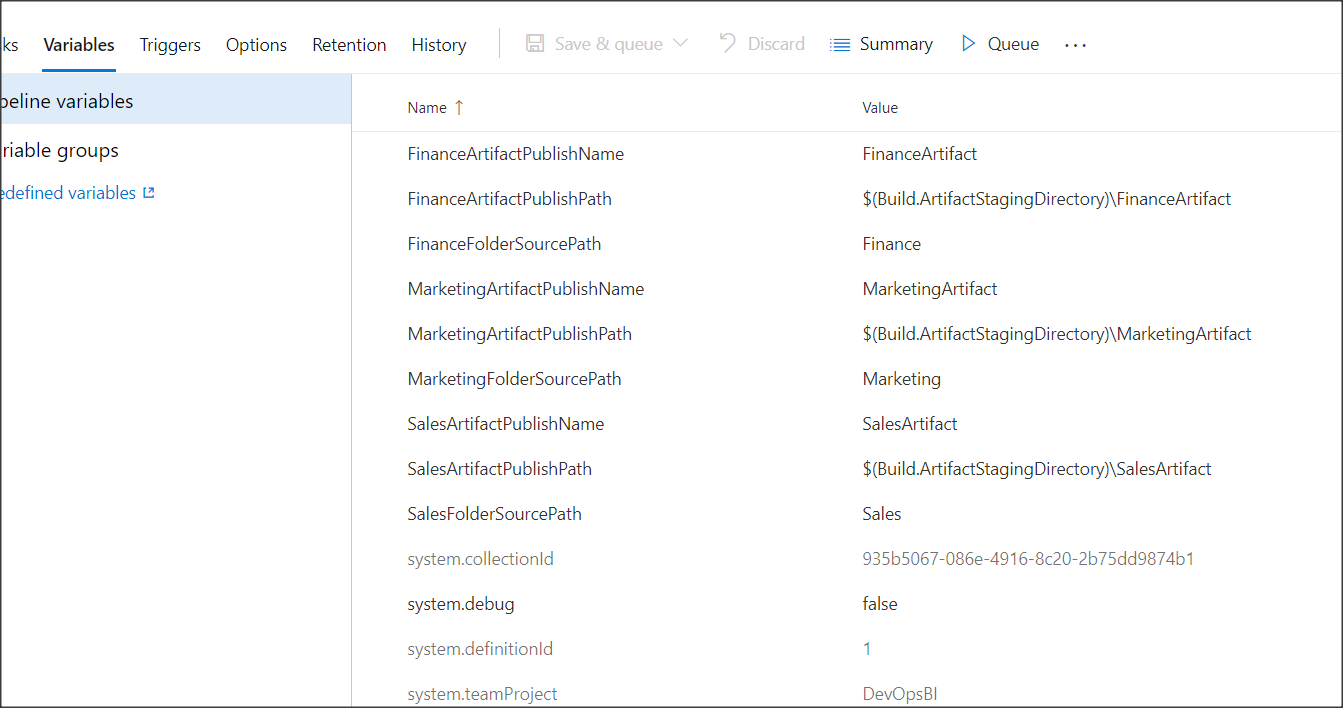


Figure 30: Adding variables to build definition

The *FolderSourcePath* is going to be the path of the folder domain wise checked in in the repository. For e.g. the sales domain source path is “*Sales*” etc.

The *ArtifactPublishPath* is going to be the target path on the artifact staging directory as shown above.

To know more on build variables please visit [this](https://docs.microsoft.com/en-us/azure/devops/pipelines/build/variables?view=azure-devops&tabs=yaml) link.

Use these variables in the task added in the build pipeline as shown below.

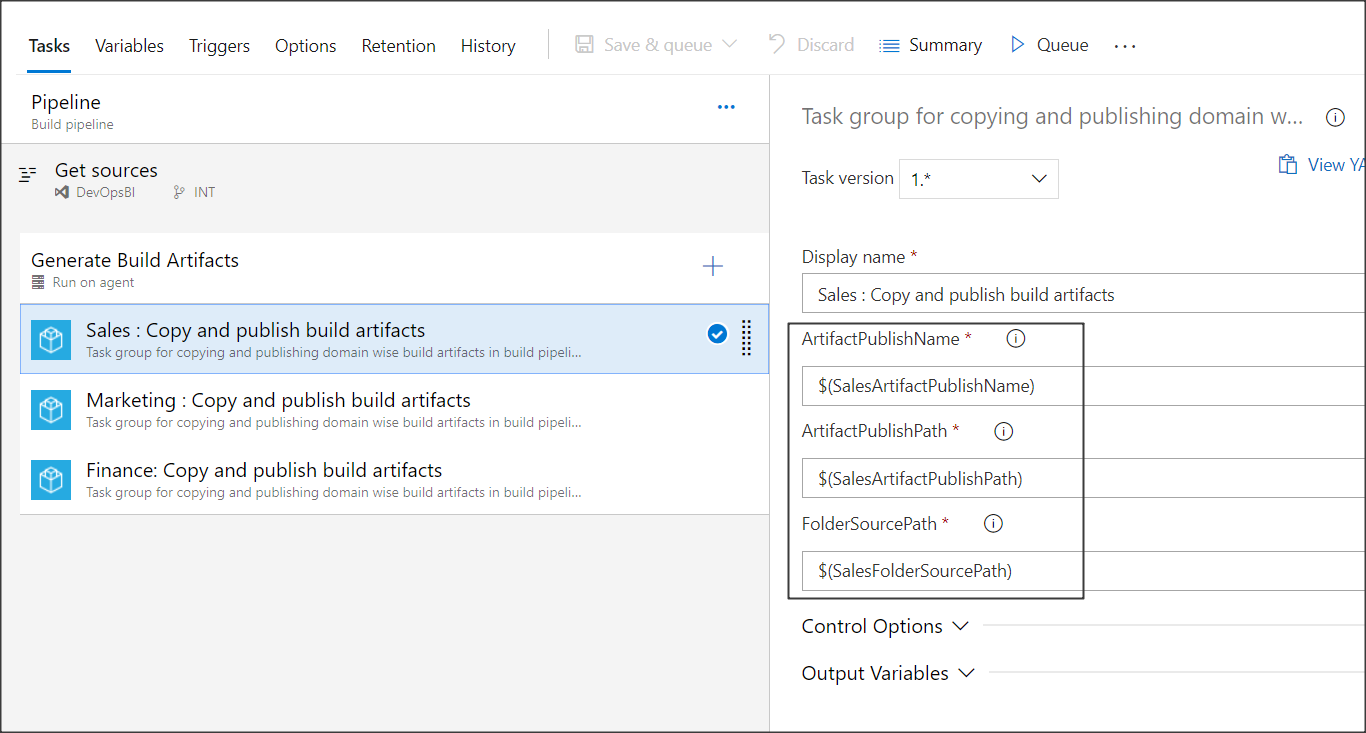


Figure 31: Usage of variables in the build task

Repeat these above steps for all the domains/modules possible in the requirement.

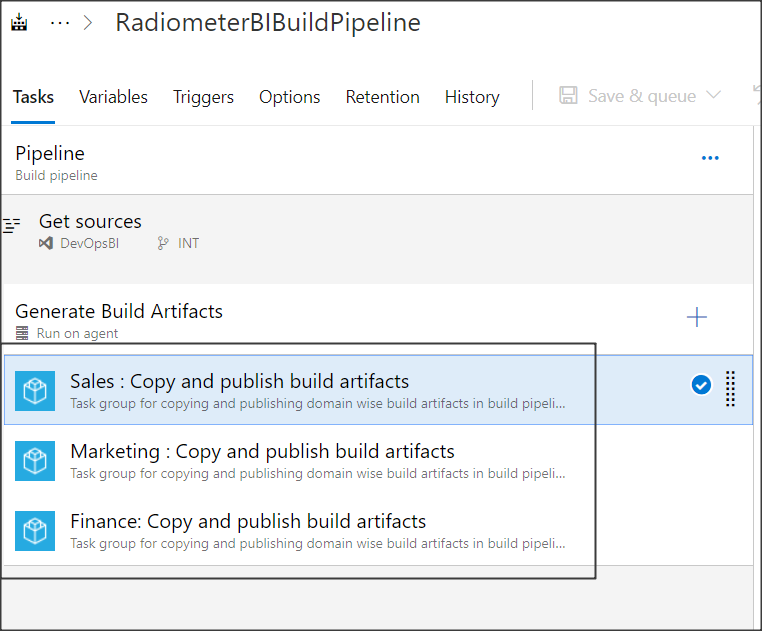


Figure 32: Repeating the task for rest of the domains

Enable continuous integration for this pipeline from the trigger tab and enable it for all the branches in the repository. Type **\* (asterisk)** in the Branch specification field and save the pipeline. This will automatically trigger the build once anything is committed on any of the branch of the repository by any one.

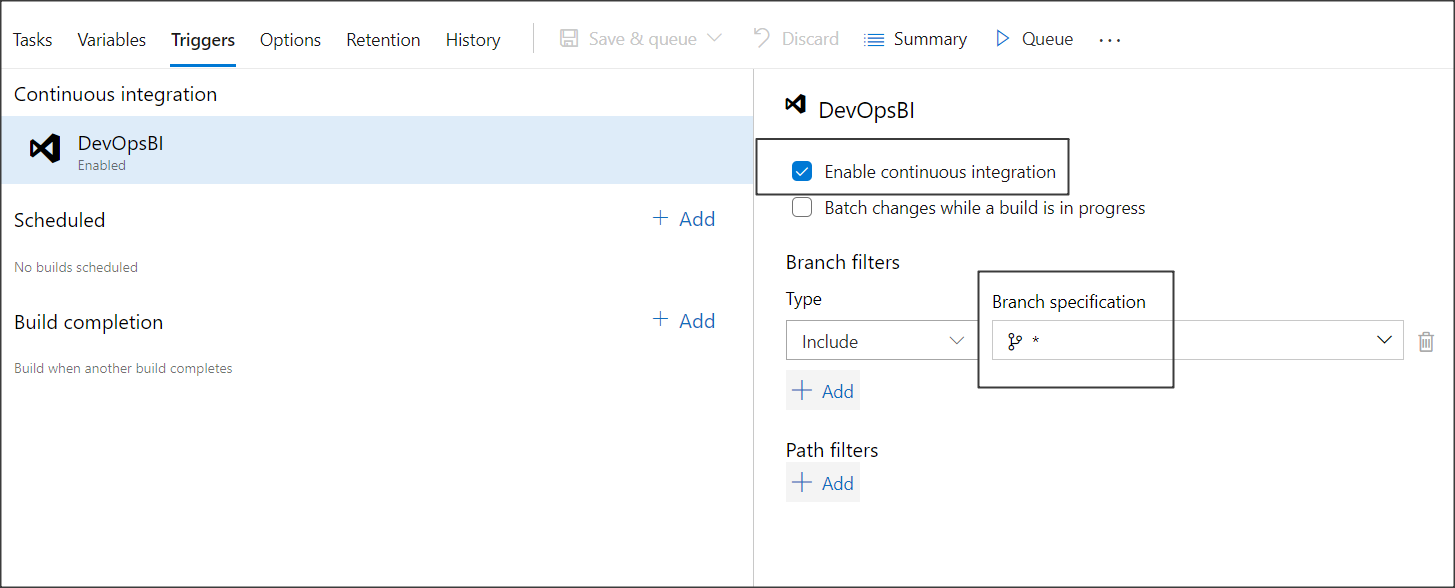


Figure 33: Enabling CI

In the options tab, set the description of the pipeline as desired and keep the build number format as desired. You can use in built variables for setting up build number format. For e.g. “*Build-$(Build.SourceBranchName)-$(date:yyyyMMdd)-$(rev:r)”*  will generate a build with name “***Build-INT-20190221-4***” where INT is the name of the branch triggering it and 4 is the revision number of the build.

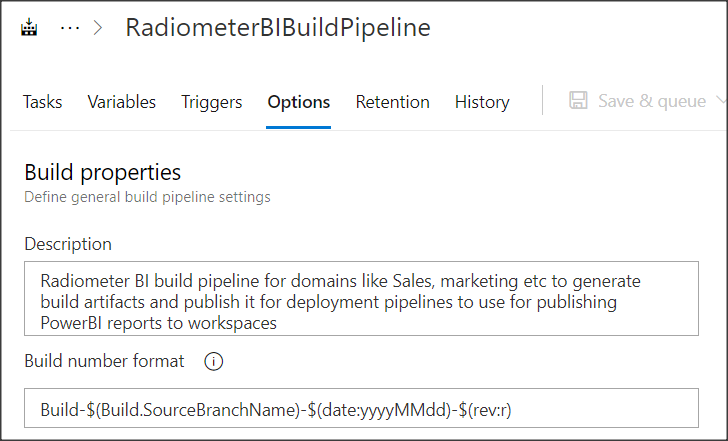


Figure 34: Build options tab

Queuing a build manually and browsing for Artifacts

The build can be manually queued by clicking on queue



Figure 35: Queuing a build manually

Click on any of the build for viewing its outputs and artifacts

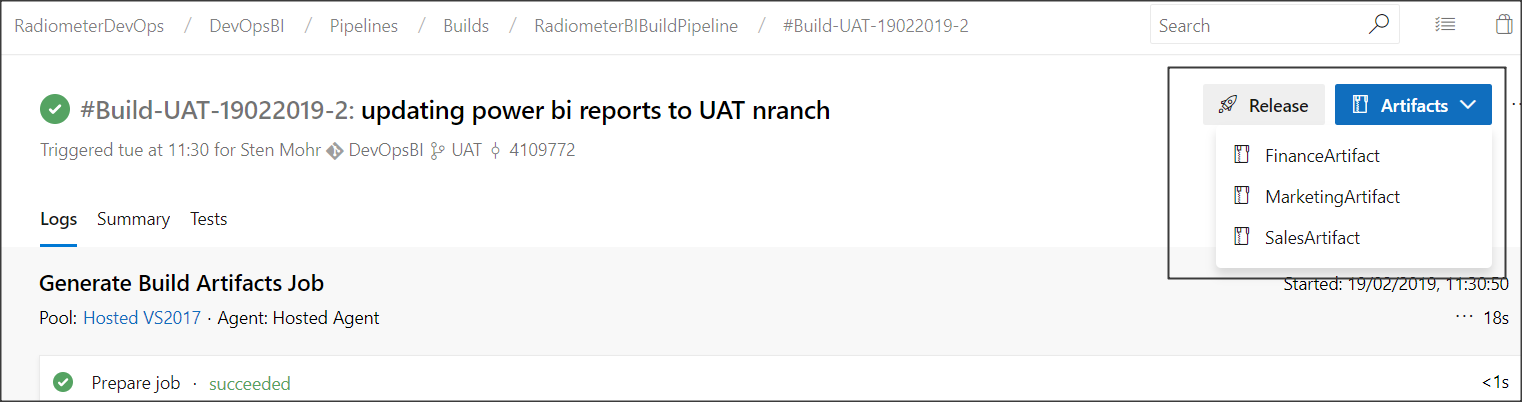


Figure 36: Browsing for Artifacts

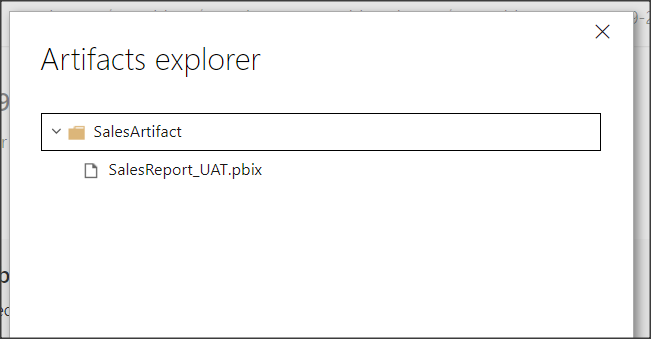


Figure 37: Exploring domain artifacts

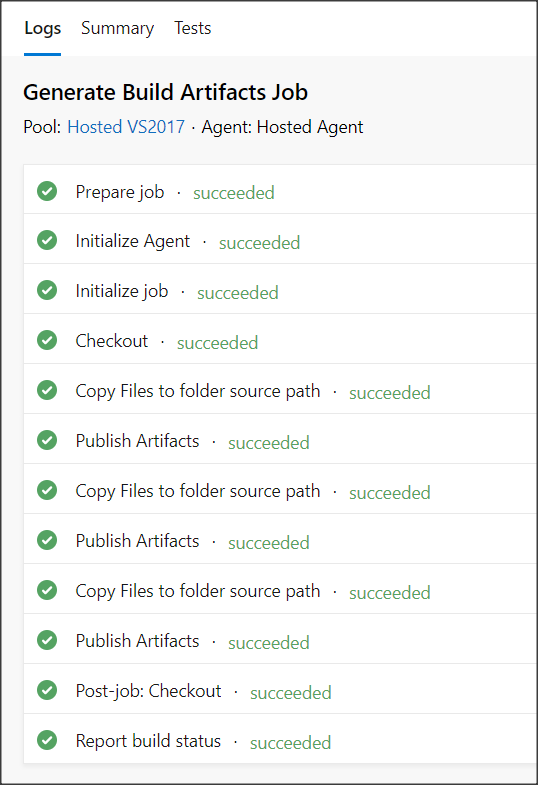


Figure 38: Exploring build logs

Set up retention policies

Set the retention policies to 30 Days with minimum number of days to 10 to retain.

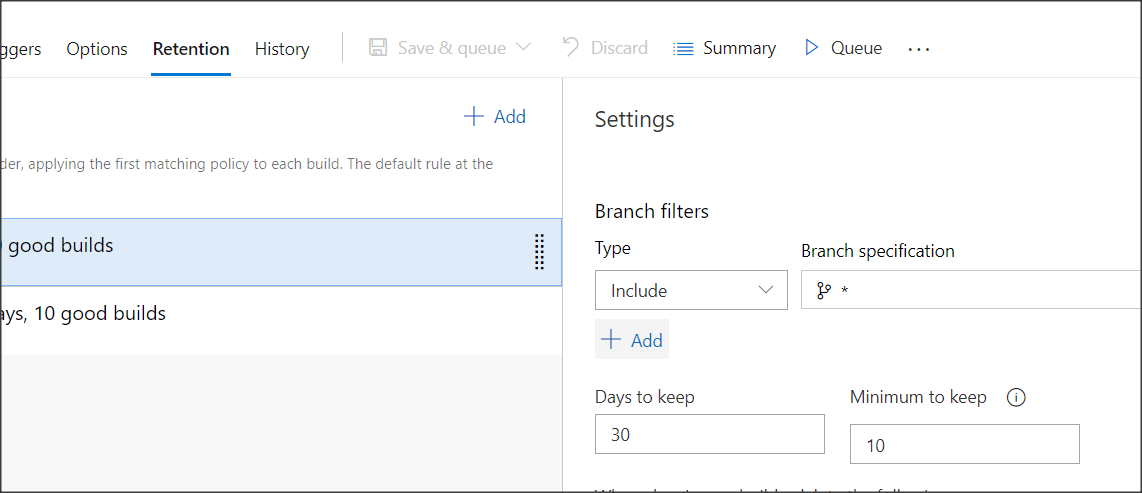


Figure 39: Setting up retention policies

Please refer to [this](https://docs.microsoft.com/en-us/azure/devops/pipelines/policies/retention?view=azure-devops) link to know more on retention policies.

A build definition can also be imported from a file which can be created by exporting an already created build definition. The JSON file for this build definition Is attached below.



The pipeline can be imported from below mentioned option

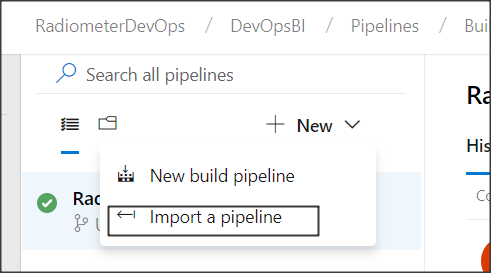


Figure 40: Import a build pipeline

Definition and use of variables defined in the build definition

For inbuilt variables, see [this](https://docs.microsoft.com/en-us/vsts/build-release/concepts/definitions/build/variables?tabs=batch#control-variables) link.

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Variable Name** | **Description** | **Value** |
|  | {Domain}ArtifactPublishName | Name of the publish Artifact (domain wise). For E.g. FinanceArtifactPublishName | {Domain}Artifact for e.g. FinanceArtifact |
|  | {Domain}ArtifactPublishPath | Path of the publish Artifact (domain wise). For E.g. FinanceArtifactPublishPath | For E.g*. $(Build.ArtifactStagingDirectory)\FinanceArtifact* |
|  | {Domain}FolderSourcePath | Source path of the files from where Artifact are to be generated | For e.g. Finance |

***Note****- Replace {Domain} with the actual names such as Finance, Marketing or Sales etc.*

## Setting up Azure Deployment Pipeline

To set up Continuous deployment process for Radiometer PowerBI application, a deployment pipeline needs to be created first targeting the reports to be published to PowerBI from the Artifacts generated by a build pipeline. The tasks needed to set up are shown below.

Setting up a task group having common tasks of managing workspaces and reports to PowerBI

The task group required can be imported from below attached file



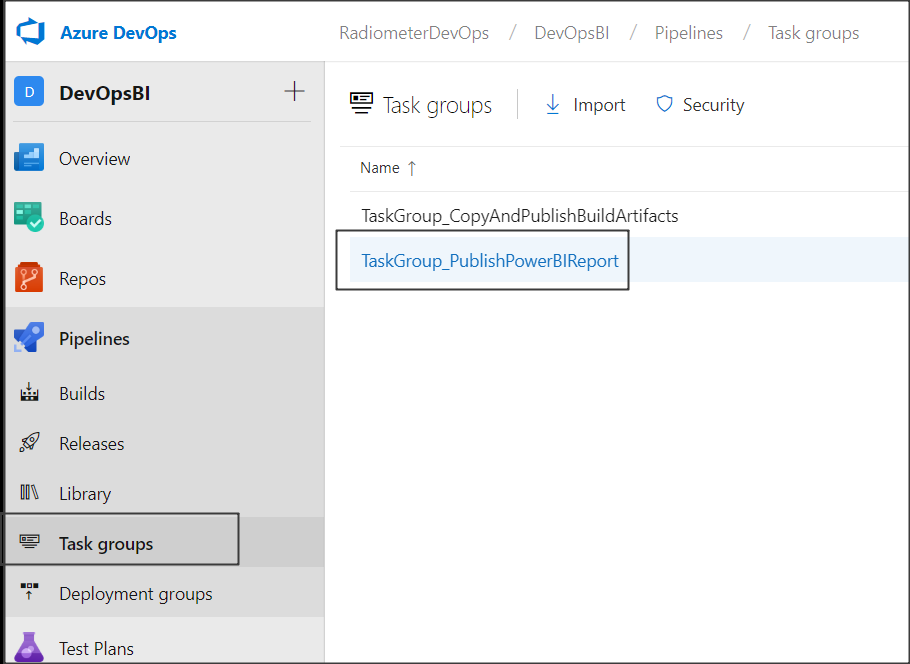


Figure 41: Import a task group for deployment

Please refer [this](https://docs.microsoft.com/en-us/azure/devops/pipelines/library/task-groups?view=azure-devops) link to know more on creating and managing task group

Tasks in the imported task group

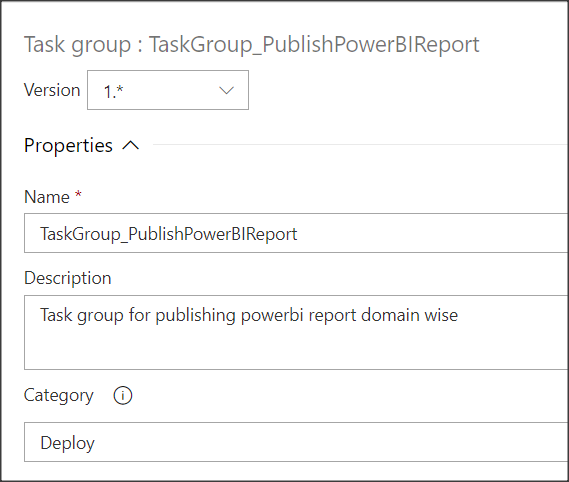


Figure 42:Name and the description of the task group

Below are the tasks which are included in the task group

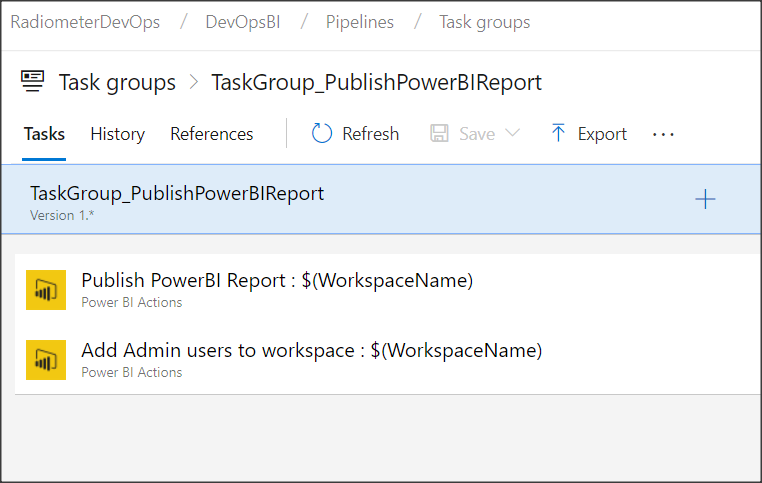


Figure 43: Task group tasks for deployment process

Publish PowerBI Report

Task to publish power BI report to PowerBI workspace and create a workspace if it doesn’t exist.

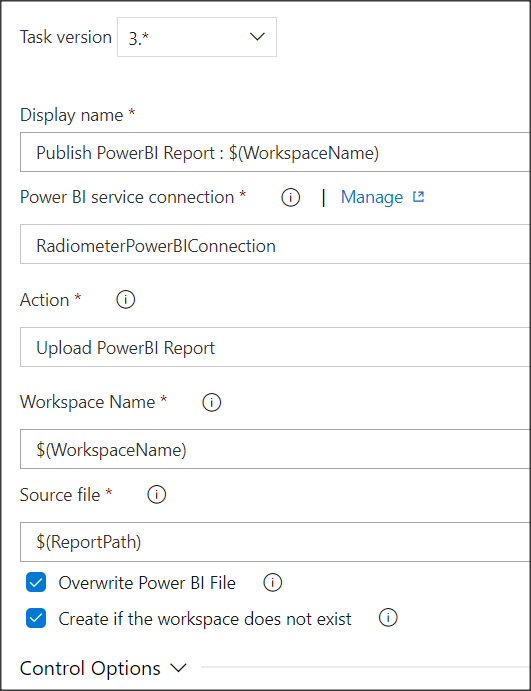


Figure 44: Publish PowerBI Report Task

Add admin users to workspace

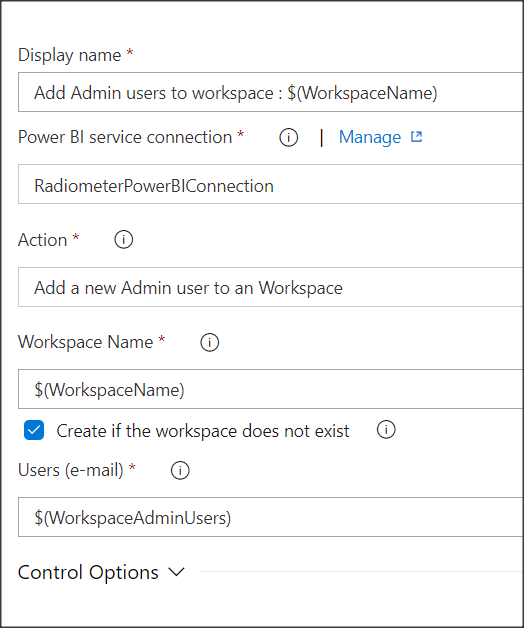


Figure 45: Adding admin users to a workspace