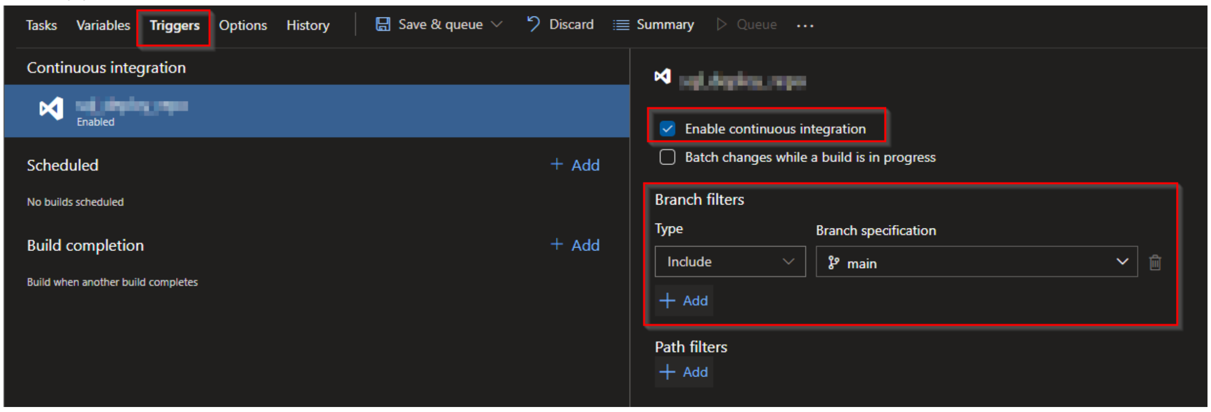
Q1

1) – In Classic Model
Go to Triggers, Check enable continuous integration and select the branches where trigger needs to happen



Tasks Variables Triggers Options History | Save & queue | Discard | Summary | Queue ...

Continuous integration
[redacted]
Enabled

Scheduled + Add
No builds scheduled

Build completion + Add
Build when another build completes

[redacted]

☑ Enable continuous integration
☐ Batch changes while a build is in progress

Branch filters

Type	Branch specification
Include	main

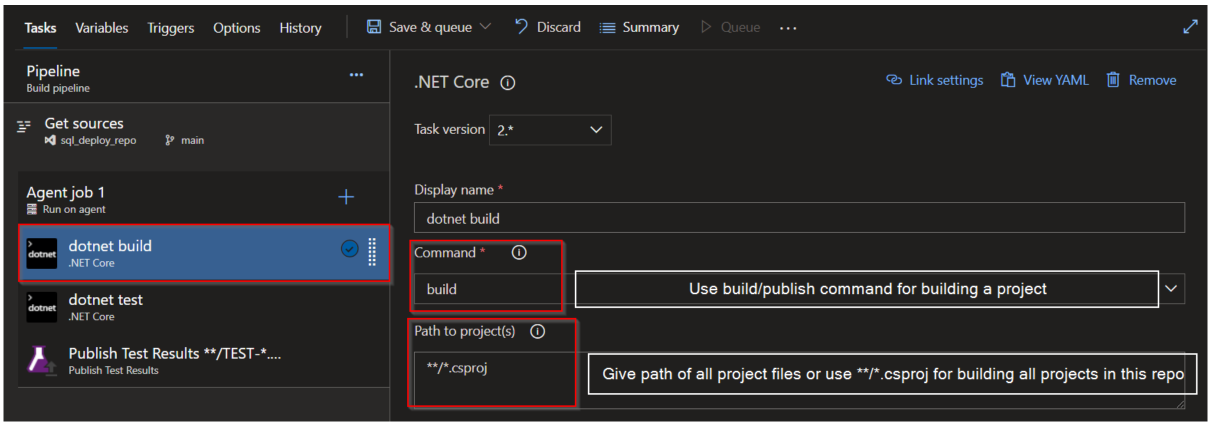
+ Add

Path filters
+ Add

In YAML based model, add following code in YAML code

```
trigger:

    batch: true
    branches:
        include:
            - master
```

2) For Classic Model
Use task -> .NET Core
Add three tasks in pipelines, Build, Test and Publish Test Results
Build



Tasks Variables Triggers Options History | Save & queue | Discard | Summary | Queue ...

Pipeline
Build pipeline

Get sources
sql_deploy_repo main

Agent job 1
Run on agent

dotnet build
.NET Core

dotnet test
.NET Core

Publish Test Results **/TEST-*....
Publish Test Results

.NET Core ⓘ | Link settings | View YAML | Remove

Task version: 2.*

Display name *: dotnet build

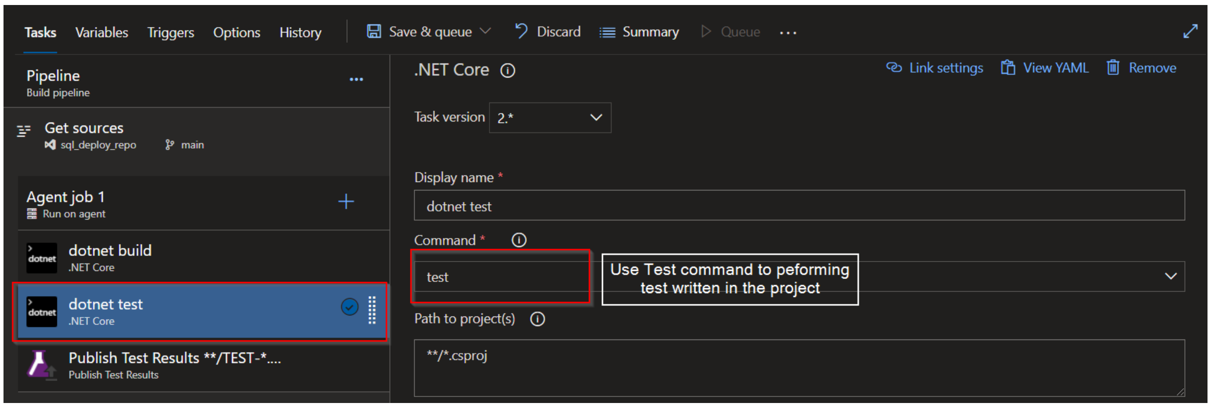
Command *: build

Use build/publish command for building a project

Path to project(s): **/*.csproj

Give path of all project files or use **/*.csproj for building all projects in this repo

Test



Pipeline
Build pipeline

Get sources
sql_deploy_repo main

Agent job 1
Run on agent

dotnet build
.NET Core

dotnet test
.NET Core

Publish Test Results **/TEST-*....
Publish Test Results

.NET Core

Task version 2.*

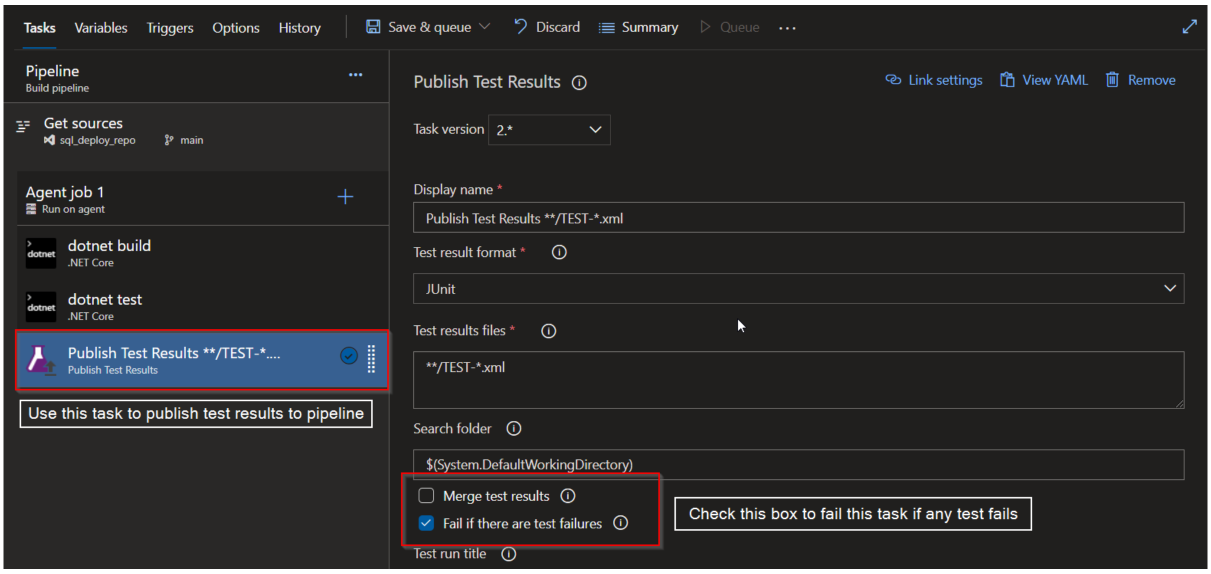
Display name * dotnet test

Command * test

Use Test command to peforming test written in the project

Path to project(s) **/*.csproj

Publish



Pipeline
Build pipeline

Get sources
sql_deploy_repo main

Agent job 1
Run on agent

dotnet build
.NET Core

dotnet test
.NET Core

Publish Test Results **/TEST-*....
Publish Test Results

Use this task to publish test results to pipeline

Publish Test Results

Task version 2.*

Display name * Publish Test Results **/TEST-*.xml

Test result format * JUnit

Test results files * **/TEST-*.xml

Search folder $(System.DefaultWorkingDirectory)

☐ Merge test results

☑ Fail if there are test failures

Check this box to fail this task if any test fails

Test run title

For YAML based Model

Build – add this code in YAML pipeline

```
steps:
- task: DotNetCoreCLI@2
  displayName: 'dotnet build'
  inputs:
    projects: '**/*.csproj'
```

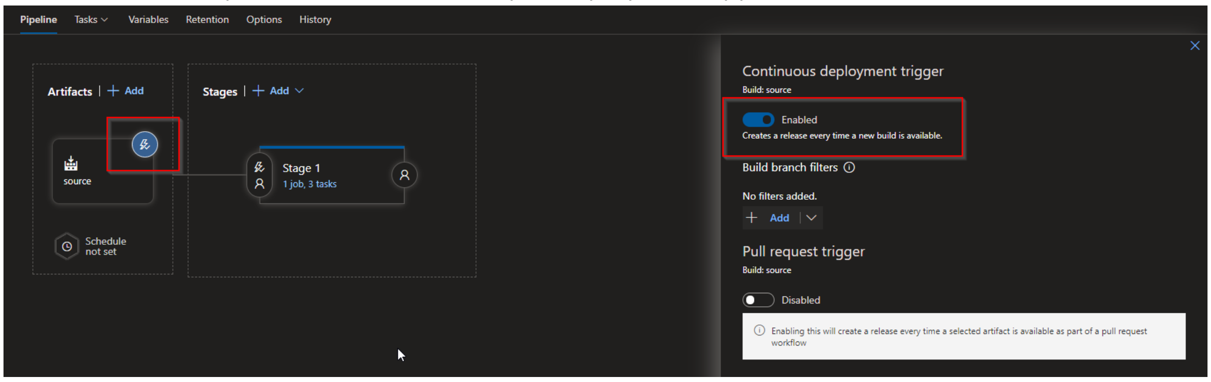
Test – add this code in YAML pipeline

```
steps:

- task: DotNetCoreCLI@2
  displayName: 'dotnet test'
  inputs:
    command: test
    projects: '**/*.csproj'
```

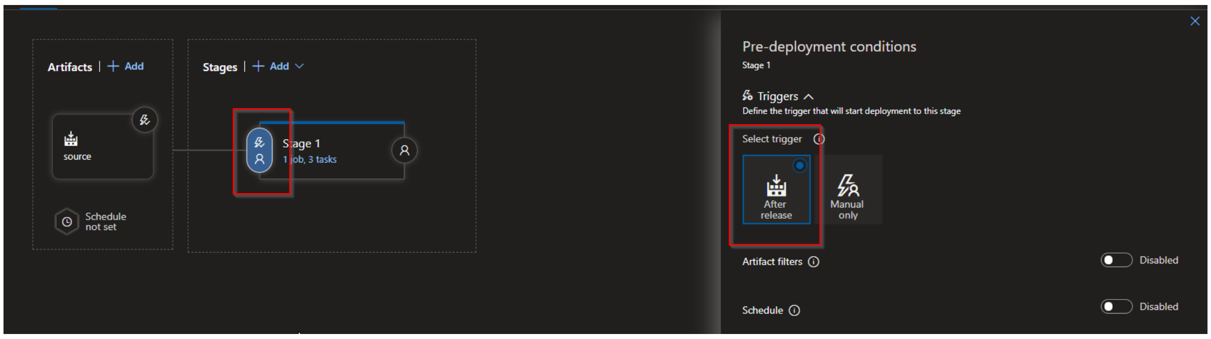
Publish

```
steps:

- task: PublishTestResults@2
  displayName: 'Publish Test Results **/TEST-*.xml'
  inputs:
    failTaskOnFailedTests: true
```

3) In Release pipeline
   Make sure these options are selected and pre-deployment approval is not enabled

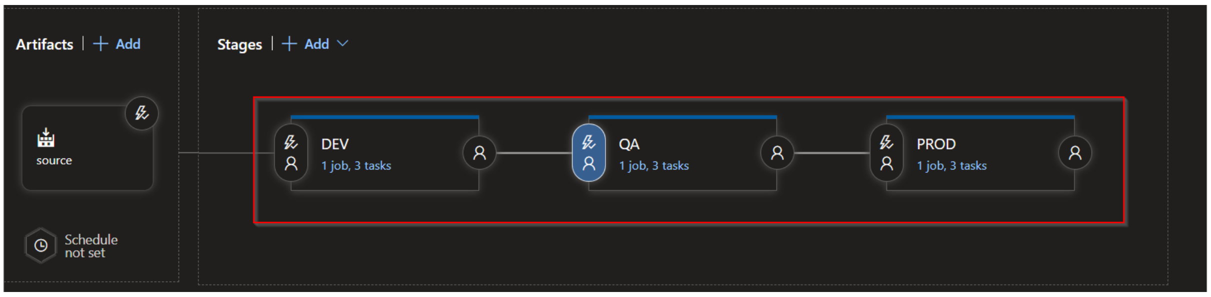


Continuous deployment trigger
Build: source
Enabled
Creates a release every time a new build is available.
Build branch filters
No filters added.
Add
Pull request trigger
Build: source
Disabled
Enabling this will create a release every time a selected artifact is available as part of a pull request workflow

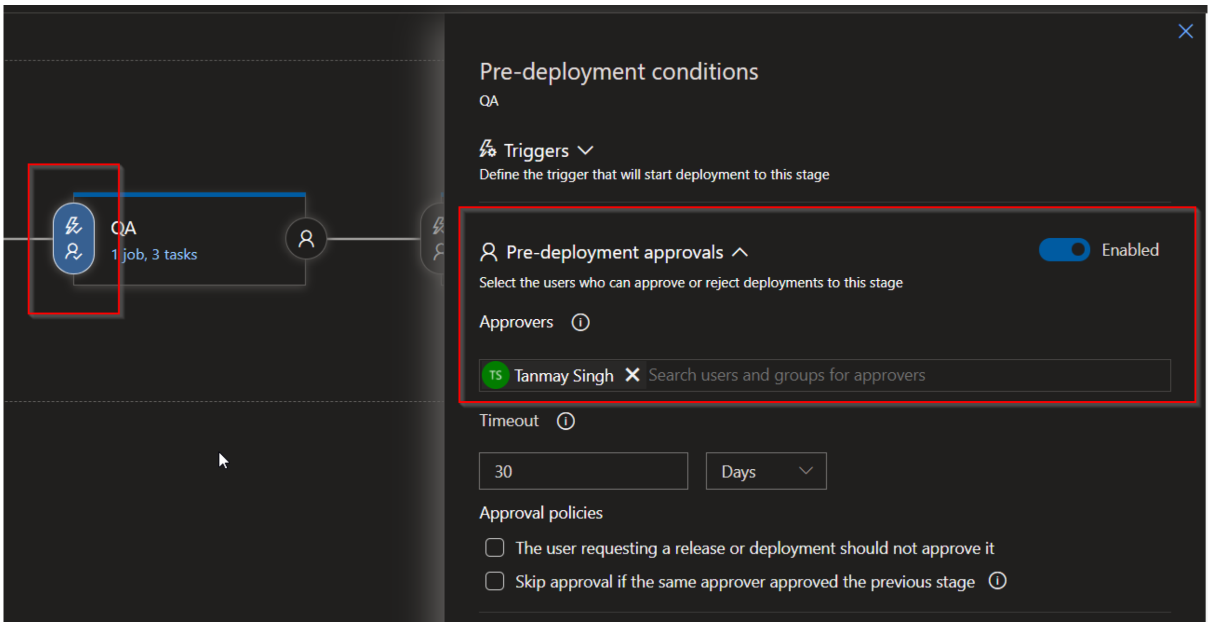


Pre-deployment conditions
Stage 1
Triggers
Define the trigger that will start deployment to this stage
Select trigger
After release
Manual only
Artifact filters Disabled
Schedule Disabled

4) This is how Release pipeline should be configured. For easy promotion of build to next stage there should be no post deployment approvals



5) In Pre-deployment section of QA and PROD stage enable Pre-Deployment Approvals and select approvers/stakeholders



Pre-deployment conditions
QA

Triggers
Define the trigger that will start deployment to this stage

Pre-deployment approvals — Enabled
Select the users who can approve or reject deployments to this stage

Approvers: Tanmay Singh — Search users and groups for approvers

Timeout: 30 Days

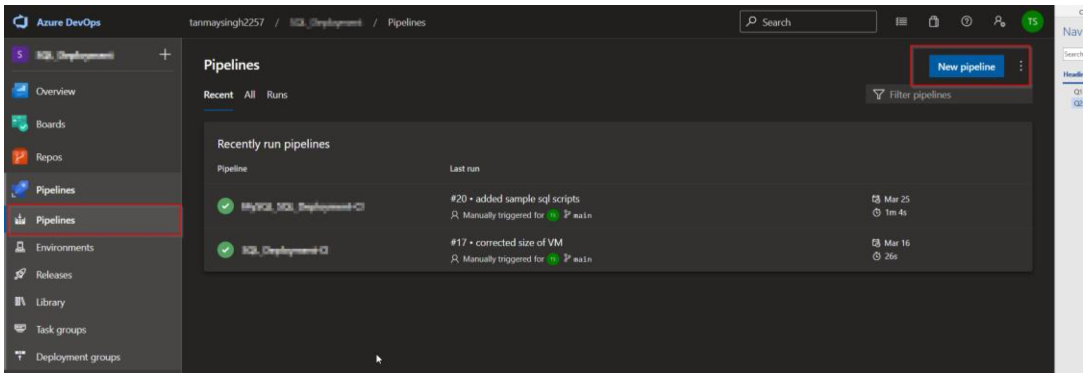
Approval policies
- [ ] The user requesting a release or deployment should not approve it
- [ ] Skip approval if the same approver approved the previous stage

Q2-

1) Artifacts to be created –
   (When assuming artifacts mean “Azure Artifacts” which would contain terraform modules)
   Terraform modules are written to reuse a terraform code.
   Terraform modules also help in maintaining a process/security/protocols when creating any resource
   a. Module for Virtual Network
   b. Module for Subnet
   c. Module for NIC
   d. Module for Public and Private IP
   e. Module for NSG
   f. Module for any Azure resource which needs to be created multiple times or created in multiple projects.

   (When assuming artifacts mean “Pipeline Artifacts” which would help in deployment and tracking deployment history)
   Build Artifacts - Containing Terraform code which has been validated and has a plan file.

2) Tools to be used to create and store terraform templates
   a. VSCode
   b. GIT/S3
   c. Terraform SDK
3) Creating a deployment pipeline can be done with Classic and YAML Model both and requires a Build Pipeline and Release Pipeline
   a. Creating a Build Pipeline
      i. Go to dev.azure.com -> Pipelines -> New Pipeline

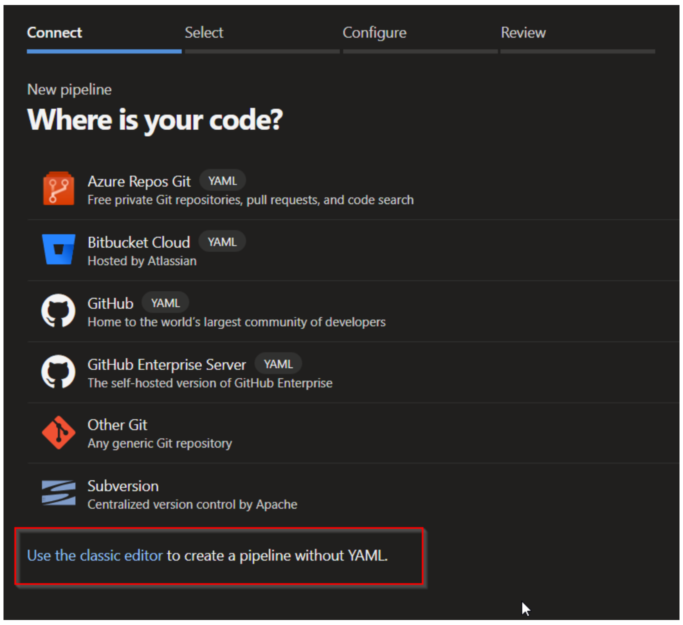


Pipelines

Recently run pipelines

Pipeline	Last run	
[redacted]	#20 • added sample sql scripts Manually triggered for main	Mar 25 1m 4s
[redacted]	#17 • corrected size of VM Manually triggered for main	Mar 16 26s

ii. Select Use 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

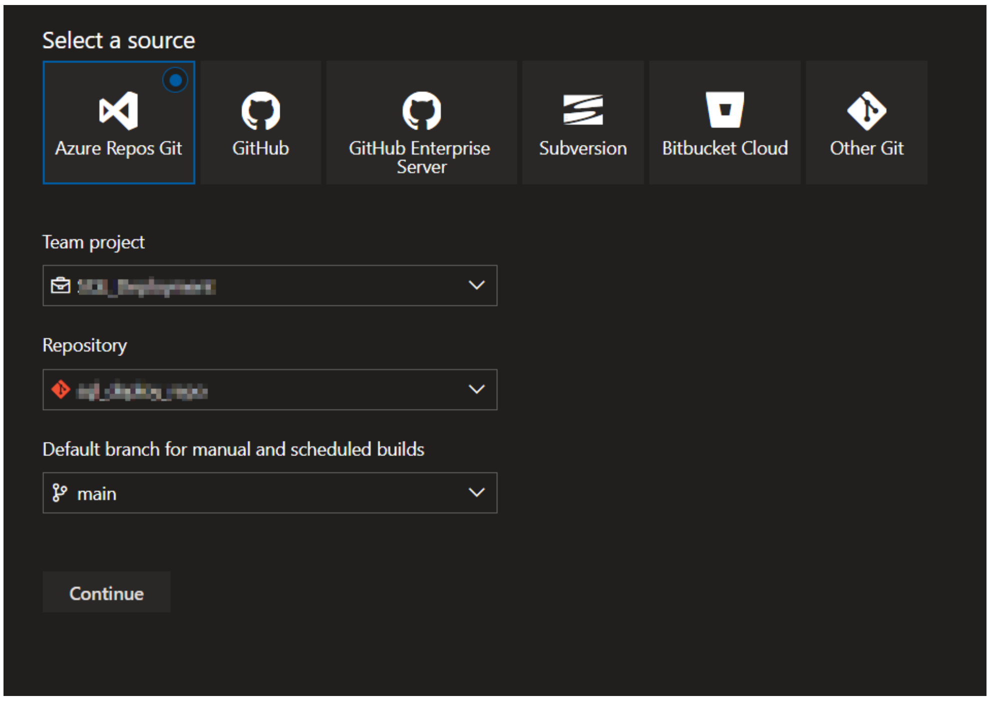
GitHub Enterprise Server YAML
The self-hosted version of GitHub Enterprise

Other Git
Any generic Git repository

Subversion
Centralized version control by Apache

Use the classic editor to create a pipeline without YAML.

iii. Select your source project, repo & default branch, where code for deployment is present
If your code in present in Azure Repo, no further change is needed
If your code is present in Github, Bitbucket or any other SVC tool, you need to create a Service Connection to that service for authentication


Select a source

Azure Repos Git | GitHub | GitHub Enterprise Server | Subversion | Bitbucket Cloud | Other Git

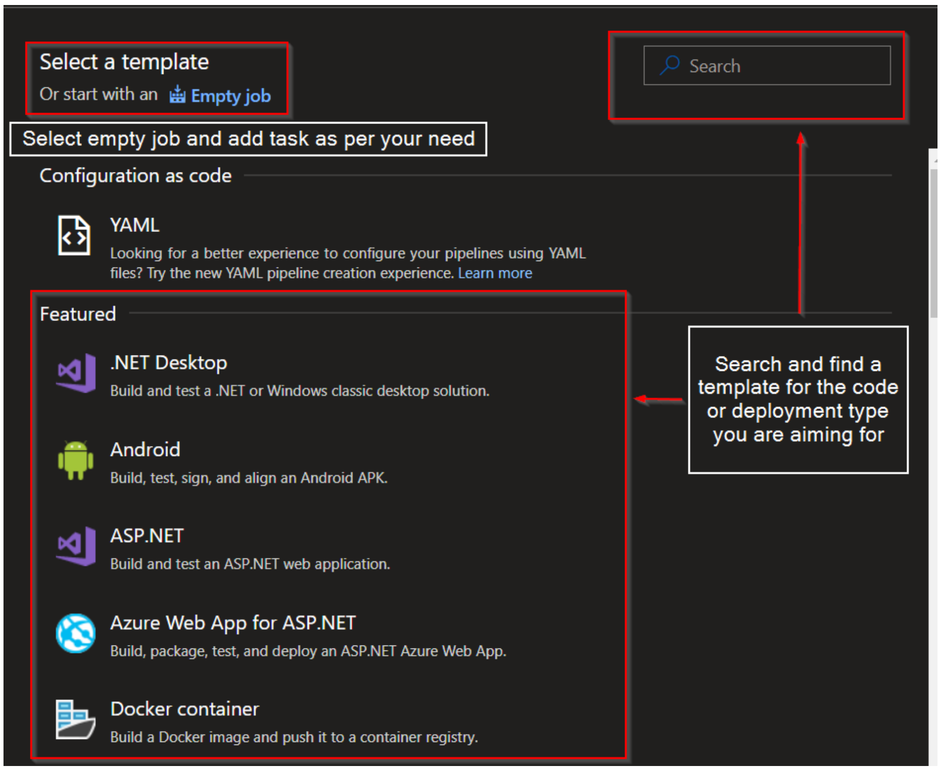
Team project
[redacted]

Repository
[redacted]

Default branch for manual and scheduled builds
main

Continue

iv. You may select a predefined template for building you code if it exists.
Or you may choose Empty Job and add task as per your needs


Select a template
Or start with an Empty job
Search
Select empty job and add task as per your need
Configuration as code
Featured
Search and find a template for the code or deployment type you are aiming for

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

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

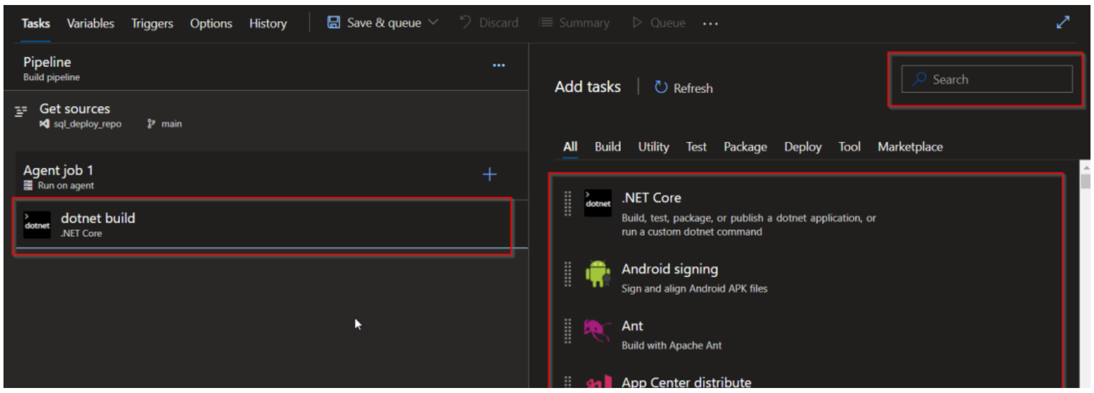
Android
Build, test, sign, and align an Android APK.

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.

Docker container
Build a Docker image and push it to a container registry.

v. For this example, I am using empty job
Search and Add task which are needed for building a project



.NET Core
Build, test, package, or publish a dotnet application, or run a custom dotnet command

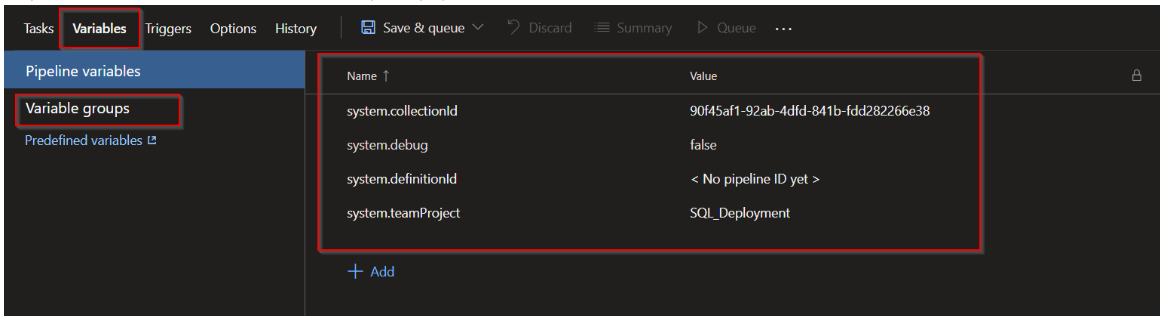
Android signing
Sign and align Android APK files

Ant
Build with Apache Ant

App Center distribute

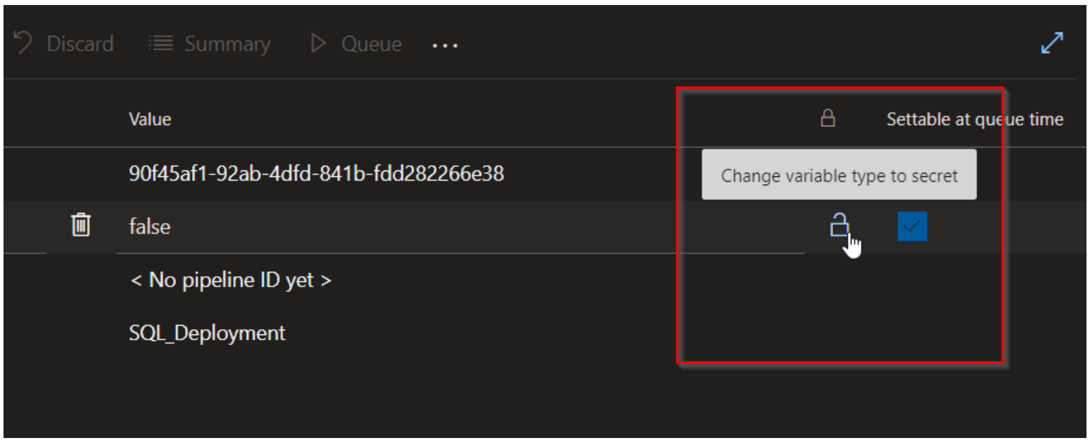
You may optionally choose command line and write scripts for building your project for having more control of what happens

vi. If you want to use variables in you pipeline, add them in variable section.



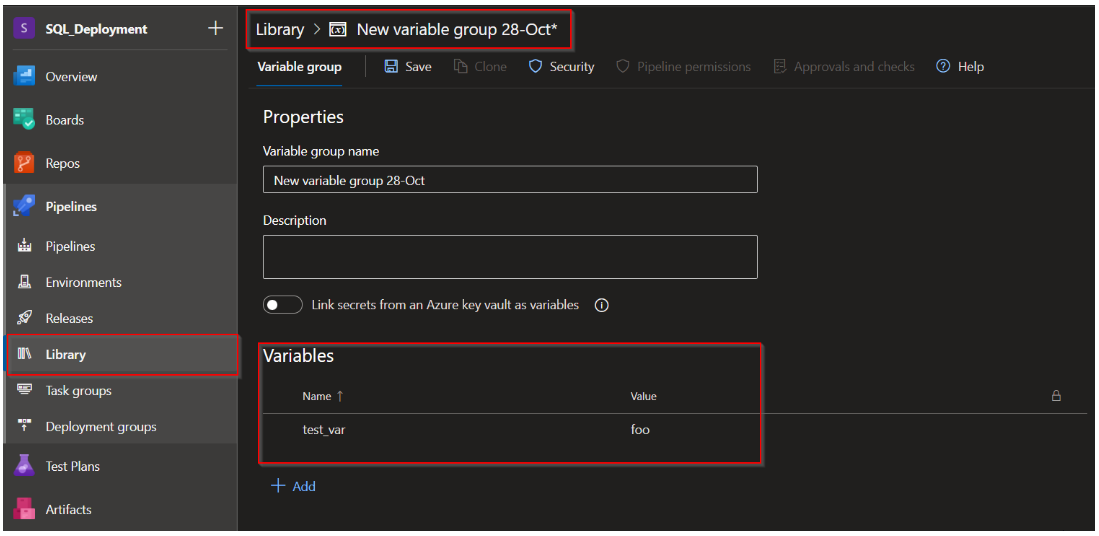
Name ↑	Value
system.collectionId	90f45af1-92ab-4dfd-841b-fdd282266e38
system.debug	false
system.definitionId	< No pipeline ID yet >
system.teamProject	SQL_Deployment

Any form of credentials/secure objects need to be placed in variables section and marked as Secure



Value
90f45af1-92ab-4dfd-841b-fdd282266e38
false
< No pipeline ID yet >
SQL_Deployment

If you have common variables which are to be used in multiple pipelines, those are required to be added in variables group and select that variable group in our build pipeline



Properties

Variable group name: New variable group 28-Oct

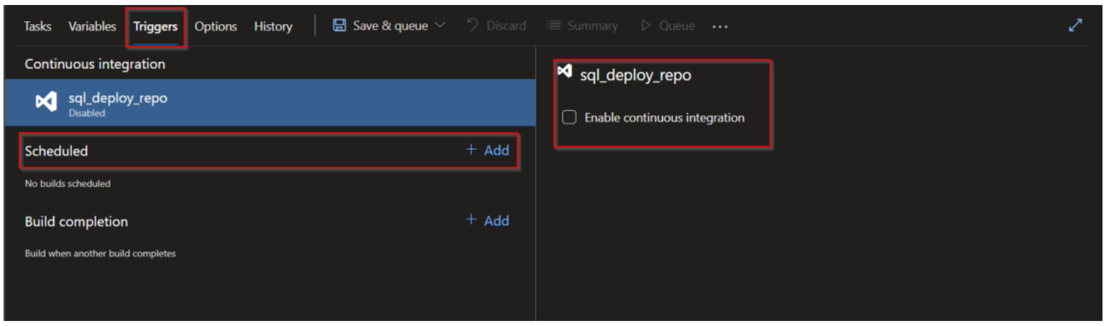
Description:

Link secrets from an Azure key vault as variables

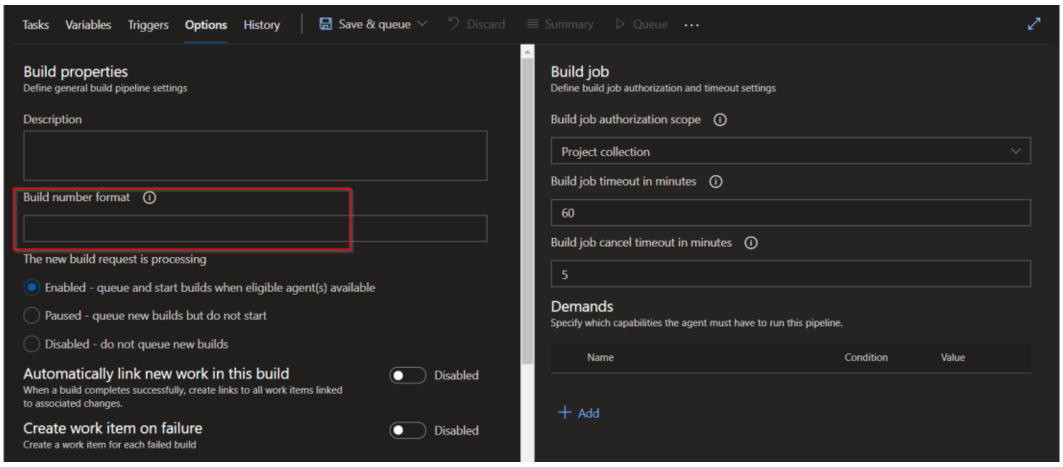
Variables

Name	Value
test_var	foo

vii. Select triggers for build pipeline, which may be set to trigger on commit or on a schedule



viii. Additional settings such as Build Number can be set in the Options setting



Build properties
Define general build pipeline settings

Description

Build number format

The new build request is processing
- [✓ Enabled - queue and start builds when eligible agent(s) available]
- Paused - queue new builds but do not start
- Disabled - do not queue new builds

Automatically link new work in this build — Disabled
When a build completes successfully, create links to all work items linked to associated changes.

Create work item on failure — Disabled
Create a work item for each failed build

Build job
Define build job authorization and timeout settings

Build job authorization scope: Project collection

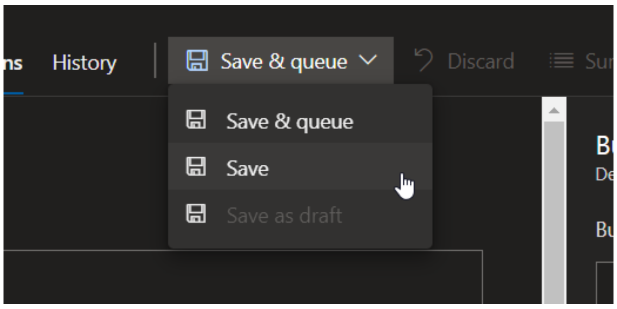
Build job timeout in minutes: 60

Build job cancel timeout in minutes: 5

Demands
Specify which capabilities the agent must have to run this pipeline.

Name	Condition	Value

ix. Hit Save and Queue, to trigger pipeline and check of any failures during building. this would also generate a build artifact which would be used for creating release pipeline



Above settings can set in YAML based template and committed the same git repository.

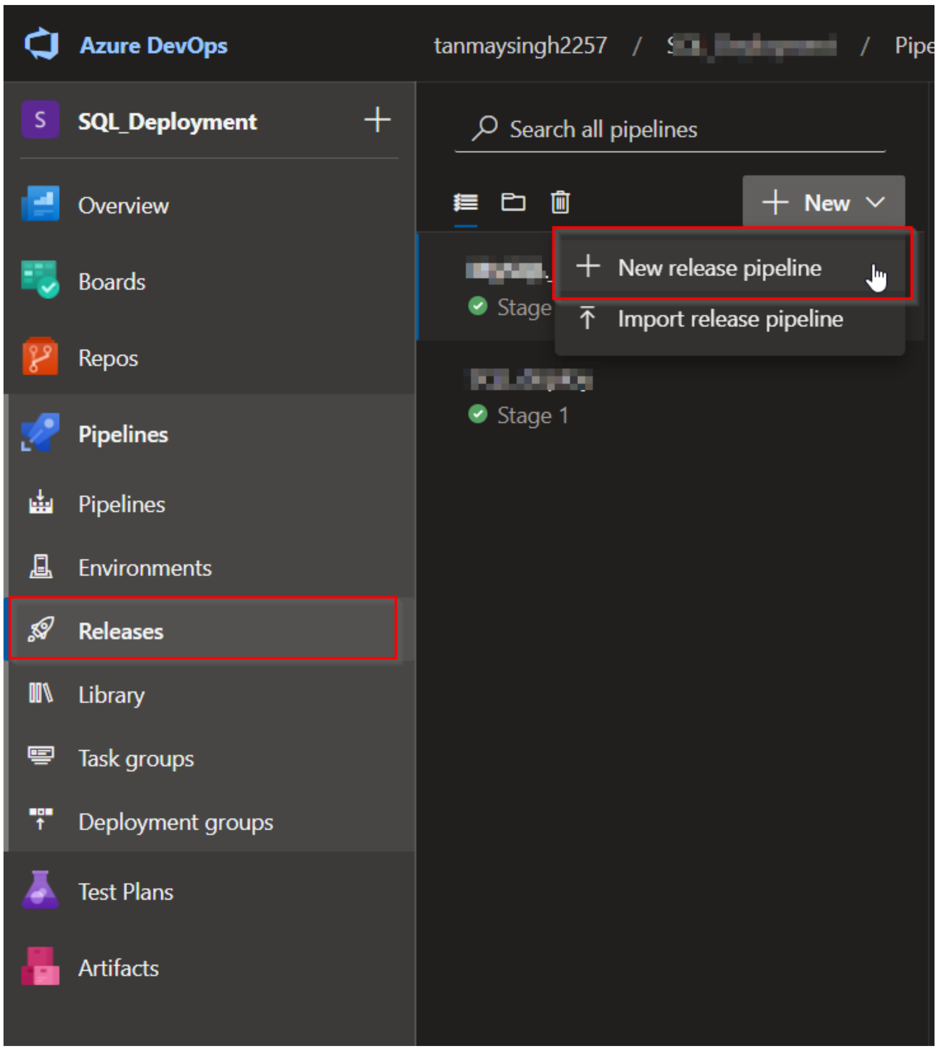
```
# Pipeline Name or Build Version Name
# name: ci_test_dotnet_app-$(Build.SourceBranchName)-$(Date:yyyyMMdd)$(Rev:.r)


# Branch trigger for Current repo
trigger:
  batch: true
  branches:
    include:
      - master
  paths:
    include:
      - donet_app-tf/operations/*
      - terraform/dev/config.tfvars

# Agent Pool to be used for Build Pipeline
pool:
  name: "DevOps $(env)"

# Clean all Previous pipeline files and artifacts from Devops agent
workspace:
  clean: all

steps:
  # Running Bash Script
  - script: |
      eval $(ssh-agent -s)
      ssh-add ~/.ssh/azure-devops-$(env)-agent
      bash operations/prepare.sh $(env)
    displayName: "Prepare Code & Terraform"
  # Running Terraform for getting Current State and Plan. These files will be
present in pipeline artifact
  - script: |
      terraform init -backend-config="./$(env)/operations.tfbackend"
      terraform validate
      terraform state pull > $(env)_state.json
      terraform plan -var-file="./$(env)/config.tfvars" -no-color >
$(env)_plan.txt
      rm -rf .terraform
    workingDirectory: "package/terraform"
    failOnStderr: true
    displayName: "Terraform Plan"
  # Publishing Artifact
  - task: PublishBuildArtifacts@1
    displayName: "Publish Artifact"
    inputs:
      PathtoPublish: "package"
      ArtifactName: "drop"
      publishLocation: "Container"
```

When creating a pipeline set the repo and this YAML file to create pipeline

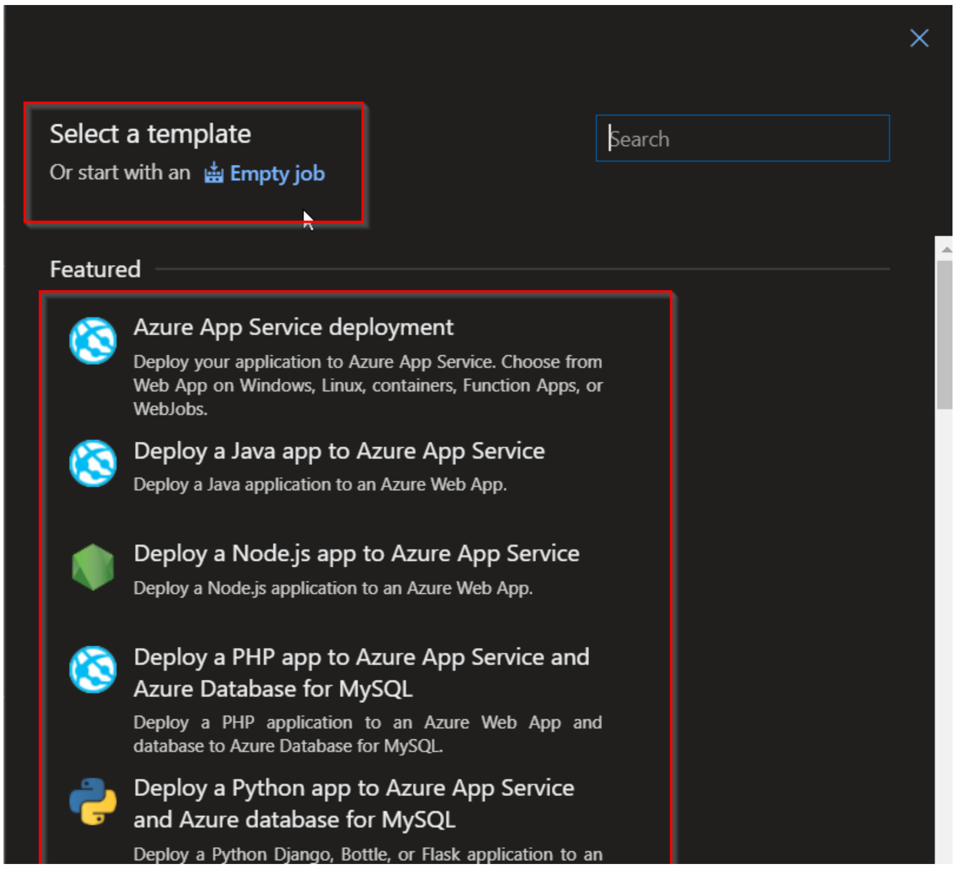
b. Creating Release Pipeline

In Azure DevOps Release pipeline can only be created with classic model.

i. Go to Release and click Create Release



ii. Similar to Build pipeline you can select an Empty Job and Add task as per your need, or select a predefined template for the deployment type



Select a template
Or start with an Empty job

Search

Featured

Azure App Service deployment
Deploy your application to Azure App Service. Choose from Web App on Windows, Linux, containers, Function Apps, or WebJobs.

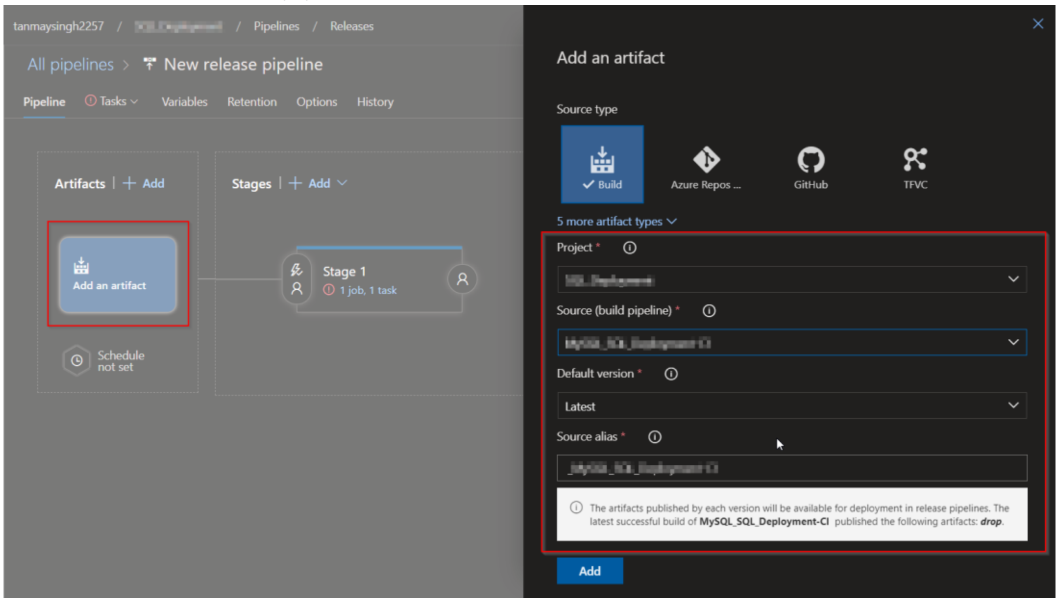
Deploy a Java app to Azure App Service
Deploy a Java application to an Azure Web App.

Deploy a Node.js app to Azure App Service
Deploy a Node.js application to an Azure Web App.

Deploy a PHP app to Azure App Service and Azure Database for MySQL
Deploy a PHP application to an Azure Web App and database to Azure Database for MySQL.

Deploy a Python app to Azure App Service and Azure database for MySQL
Deploy a Python Django, Bottle, or Flask application to an

iii. Add an Artifact for release pipeline
Select the source build pipeline



tanmaysingh2257 / [redacted] / Pipelines / Releases

All pipelines > New release pipeline

Pipeline | Tasks | Variables | Retention | Options | History

Artifacts | + Add
Add an artifact
Schedule not set

Stages | + Add
Stage 1
1 job, 1 task

Add an artifact

Source type
Build ✓ | Azure Repos ... | GitHub | TFVC

5 more artifact types

Project * : [redacted]

Source (build pipeline) * : [redacted]

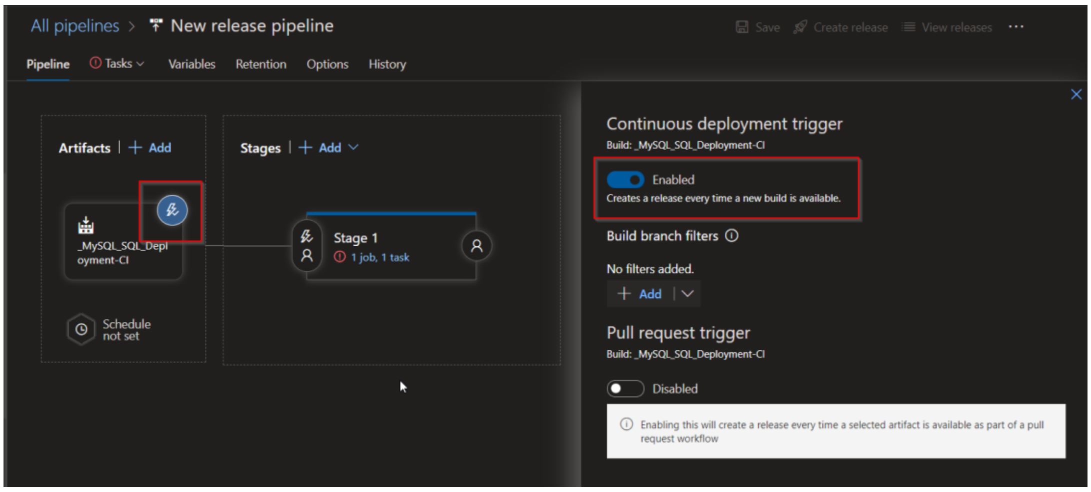
Default version * : Latest

Source alias * : [redacted]

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

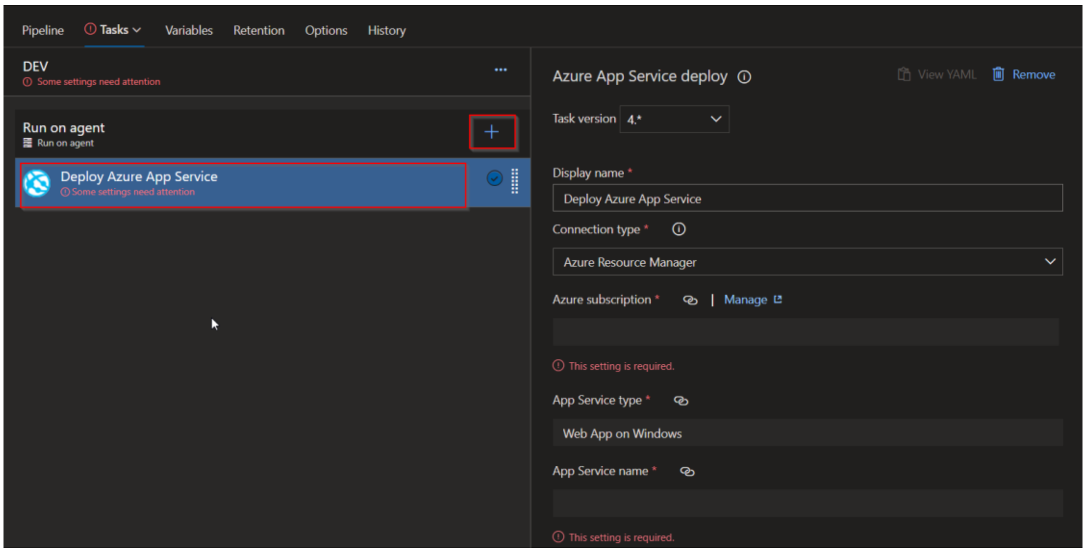
Add

iv. Select the trigger on artifact and create release when a build is available.



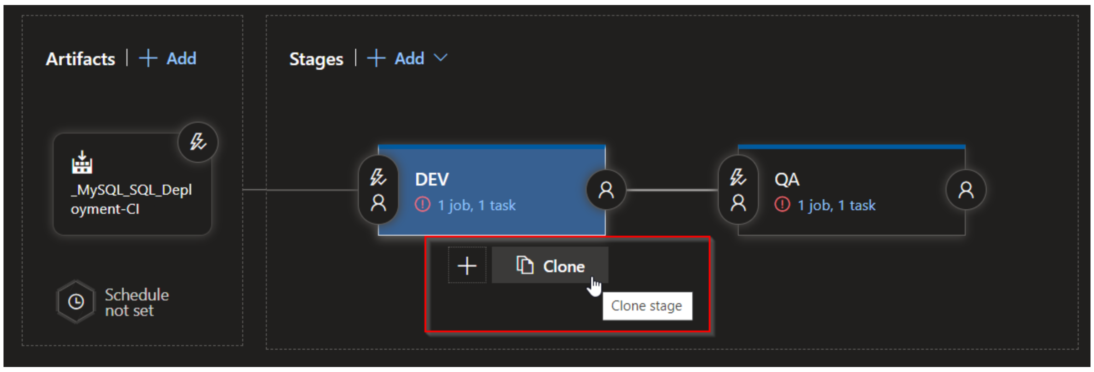
Continuous deployment trigger
Build: _MySQL_SQL_Deployment-CI
Enabled
Creates a release every time a new build is available.
Build branch filters
No filters added.
Pull request trigger
Build: _MySQL_SQL_Deployment-CI
Disabled
Enabling this will create a release every time a selected artifact is available as part of a pull request workflow

v. Select a Stage, rename it as per environment it has to be deployed on
Add tasks which are required for deployment



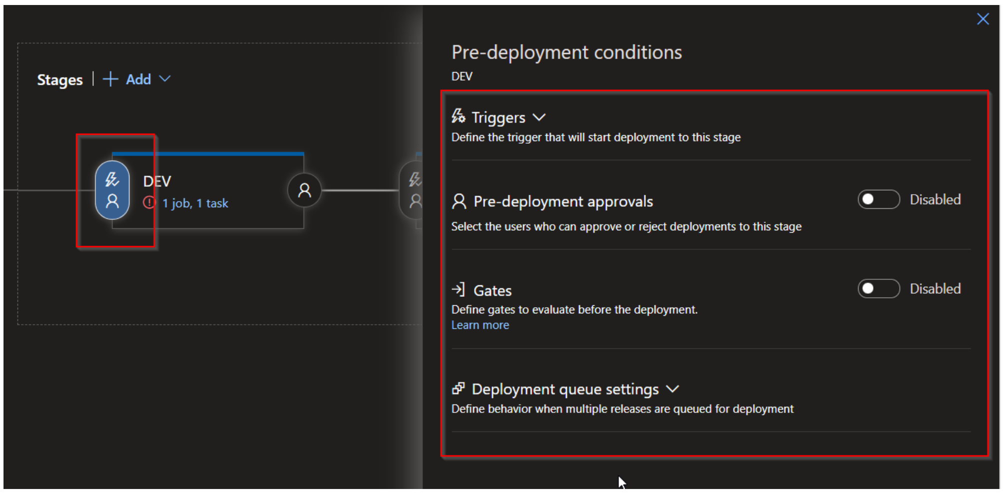
Azure App Service deploy
Task version 4.*
Display name * Deploy Azure App Service
Connection type * Azure Resource Manager
Azure subscription * | Manage
This setting is required.
App Service type * Web App on Windows
App Service name *
This setting is required.

vi. Clone the stage and create stages for new environments



Change parameter/connection/destination in new stage as per environment.
You may optionally create blank stage and add another set of tasks which may be specific to that environment.

vii. Optionally add Post or Pre-Deployment conditions such as Approvals, Gates, Auto-Redeploy Trigger



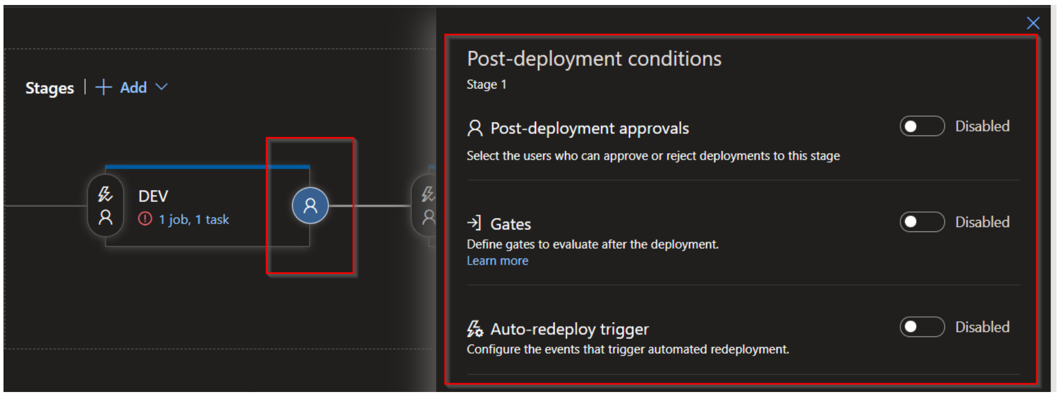
Pre-deployment conditions
DEV

Triggers
Define the trigger that will start deployment to this stage

Pre-deployment approvals — Disabled
Select the users who can approve or reject deployments to this stage

Gates — Disabled
Define gates to evaluate before the deployment.
Learn more

Deployment queue settings
Define behavior when multiple releases are queued for deployment



Post-deployment conditions
Stage 1

Post-deployment approvals — Disabled
Select the users who can approve or reject deployments to this stage

Gates — Disabled
Define gates to evaluate after the deployment.
Learn more

Auto-redeploy trigger — Disabled
Configure the events that trigger automated redeployment.

viii. Save the pipeline and run the release to test for errors and troubleshoot as needed.

4) Mentioned in github repo “terraform_sample”
5) Considering that keyvault & secret is created manually, we can reference that resource with “data” type resource block and pass the value over to VM.
   This has been presented as an example in the above sample terraform, please refer.