it has feasibility of providing CI and CD in single pipeline. multi staging is possible

git pull origin master

git add

git commit

git push origin master

YAML file format

name:string #build numbering format

resources:

pipelines:[pipeline resource]

containers:[container resource]

repositories:[repository resource]

variable:[strong:string] | [variable | template reference]

trigger:trigger

pr:pr #pull request

stages:[stage | template reference]

—-> If you have a single stage , you can omit stages and directly specify jobs.

—-> if you have single stage and single job , you can omit those keywords and directly secify steps.

pool:

name: Default

steps:

- task: CopyFiles@2

displayName: 'Copy Files to: $(build.artifactstagingdirectory)'

inputs:

SourceFolder: .

TargetFolder: '$(build.artifactstagingdirectory)'

- task: PublishBuildArtifacts@1

displayName: 'Publish Artifact: drop'

**Stage**

Stage is a logical boundary in a pipeline, it can be used to mark separation concerns(eg ;- QA,stage,dev ..etc) Each stage has one or more **jobs**

A stage is a collection of related jobs. By default, stages run sequentially. Each stage starts only after the preceding stage is complete unless otherwise specified via the dependsOn property.

stages:

- stage: string # name of the stage (A-Z, a-z, 0-9, and underscore)

displayName: string # friendly name to display in the UI

dependsOn: string | [ string ]

condition: string

variables: # several syntaxes, see specific section

jobs: [ job | templateReference]

Job

By default, stages run sequentially, starting only after the stage ahead of them has completed.you can manually control when a stage should run using approval checks.

A [job](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/phases?tabs=yaml&view=azure-devops) is a collection of [steps](https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema%2Cparameter-schema#steps) run by an [agent](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops) or on a [server](https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema%2Cparameter-schema#server). Jobs can run [conditionally](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/phases?tabs=yaml&view=azure-devops#conditions) and might [depend on earlier jobs](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/phases?tabs=yaml&view=azure-devops#dependencies).

example

stages

-stage:build

jobs:

job:build job

steps:

-script:echo building

-stage:Test

jobs:

job:Test job

steps:

-script:echo testing

JOBS

A job is a collection of linear series of steps to be run by an agent or on the server. Its a units of work assignable to particular machine .More than one job can be run parallel.

jobs:

-job:my job

displayname: My first job

continue on error: true

workspace:

clean:outputs

steps:

-step:echo from my job

jobs:

- job: string # name of the job (A-Z, a-z, 0-9, and underscore)

displayName: string # friendly name to display in the UI

dependsOn: string | [ string ]

condition: string

strategy:

parallel: # parallel strategy; see the following "Parallel" topic

matrix: # matrix strategy; see the following "Matrix" topic

maxParallel: number # maximum number of matrix jobs to run simultaneously

continueOnError: boolean # 'true' if future jobs should run even if this job fails; defaults to 'false'

pool: pool # see the following "Pool" schema

workspace:

clean: outputs | resources | all # what to clean up before the job runs

container: containerReference # container to run this job inside of

timeoutInMinutes: number # how long to run the job before automatically cancelling

cancelTimeoutInMinutes: number # how much time to give 'run always even if cancelled tasks' before killing them

variables: # several syntaxes, see specific section

steps: [ script | bash | pwsh | powershell | checkout | task | templateReference ]

services: { string: string | container } # container resources to run as a service container

uses: # Any resources (repos or pools) required by this job that are not already referenced

repositories: [ string ] # Repository references to Azure Git repositories

pools: [ string ] # Pool names, typically when using a matrix strategy for the job

Task

A task is simply a pre-created script offered as a convenience to you. This abstraction makes easier to run common build functions.

example

steps:

-script: echo this runs in the default shield on any machine

-bash:

echo This multiline script runs on Bash

echo Even on Windows machines

-pwsh:

write-host “This multiline script runs in powershell core”

write-host “even on windows machines”

-tas:dotnetcoreCLI

displayname: “build the project”

inputs:

command: “build”

arguments: “-no - restore- configuration”

projects : “ \*\*/\*.csprol”

so on an overall it look like

Stage

job

script/task

Every job requires a agent

Node JS YAML pipeline

https://azuredevopslabs.com/labs/vsts/nodejs/