parameters:

target\_salesforce\_org\_branch: 'NOT\_DEFINE'

secure\_file: 'NOT\_DEFINE'

environment\_name: 'NOT\_DEFINE'

stages:

- stage: Deploy

displayName: Deploy

dependsOn: Build

condition: and ( or (in(dependencies.Build.result, 'Succeeded'),in(dependencies.Build.result, 'SucceededWithIssues')),

ne(variables['Build.Reason'], 'PullRequest') ,

or ( and ( eq(variables['build.sourceBranch'], 'refs/heads/${{ parameters.target\_salesforce\_org\_branch }}'),

notIn( '${{ parameters.environment\_name }}' , 'preprod' ),

notIn( '${{ parameters.environment\_name }}' , 'prod' )),

and ( contains(variables['build.sourceBranch'], 'preprod' ), eq ( '${{ parameters.environment\_name }}' ,'preprod' )),

and ( eq(variables['build.sourceBranch'], 'refs/heads/preprod' ), eq ( '${{ parameters.environment\_name }}' ,'prod' ))))

variables:

- group: '${{ parameters.environment\_name }}'

- name: DELTA\_DEPLOY\_ID

value: $[ stageDependencies.Build.Build.outputs['Build.validate\_delta\_step.DELTA\_DEPLOY\_ID'] ]

- name: DELTA\_DESTRUCTIVE\_DEPLOY\_ID

value: $[ stageDependencies.Build.Build.outputs['Build.validate\_destructive\_delta\_step.DELTA\_DESTRUCTIVE\_DEPLOY\_ID'] ]

- name: DELTA\_DEPLOY\_STATUS

value: $[ stageDependencies.Build.Build.outputs['Build.details\_in\_delta\_step.DELTA\_DEPLOY\_STATUS'] ]

- name: DELTA\_DESTRUCTIVE\_DEPLOY\_STATUS

value: $[ stageDependencies.Build.Build.outputs['Build.details\_in\_delta\_step.DELTA\_DESTRUCTIVE\_DEPLOY\_STATUS'] ]

jobs:

- deployment: Deploy

environment: '${{ parameters.environment\_name }}'

displayName: Deploy

timeoutInMinutes: 360

pool:

vmImage: 'ubuntu-latest'

container: salesforce/salesforcedx:7.191.3-full #salesforce/salesforcedx:7.159.0-full #salesforce/salesforcedx:latest-full

strategy:

runOnce:

deploy:

steps:

- checkout: azure-devops-shared-automation

- template: deploy-steps.yml@azure-devops-shared-automation

parameters:

target\_salesforce\_org\_branch: '${{ parameters.target\_salesforce\_org\_branch }}'

secure\_file: '${{ parameters.secure\_file }}'

environment\_name: '${{ parameters.environment\_name }}'

- stage: Check\_Sanity

- script: '"C:\Program Files (x86)\TRICENTIS\Tosca Testsuite\ToscaCommander\ToscaCI\Client\ToscaCIClient.exe" -m distributed -c "C:\Tosca\_Projects\$(environmentName).xml" -t junit -r "C:\Tosca\_Projects\$(environmentName)\_results\_$(GET\_DATE).xml" -e "http://azwtosdba00001:5007/DistributionServerService/ManagerService.svc"'

displayName: 'Sanity Check - distributed'

#- script: '"C:\Program Files (x86)\TRICENTIS\Tosca Testsuite\ToscaCommander\ToscaCI\Client\ToscaCIClient.exe" -m local -c "C:\Tosca\_Projects\$(environmentName).xml" -t junit -r "C:\Tosca\_Projects\$(environmentName)\_results\_$(GET\_DATE).xml"'

# displayName: 'Sanity Check - local'

- task: PublishTestResults@2

displayName: 'Publish Test Results'

inputs:

testResultsFiles: 'C:\Tosca\_Projects\$(environmentName)\_results\_$(GET\_DATE).xml'

failTaskOnFailedTests: true

- publish: C:\Tosca\_Projects\$(environmentName)\_results\_$(GET\_DATE).xml

artifact: Sanity\_Check\_Results\_$(System.JobAttempt)

condition: succeededOrFailed()