parameters:

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

secure\_file: 'NOT\_DEFINE'

steps:

- task: Bash@3

displayName: Send Deploy Build Start Notification

#condition: eq(1,2)

continueOnError: true

inputs:

filePath: './devops-scripts/utilities.sh'

workingDirectory: '$(Build.SourcesDirectory)/devops-scripts'

failOnStderr: true

arguments: google\_notification "$(googleChatChannelWebhookUrl)" "\*Started $BUILD\_REASON DEPLOY Build $(Build.BuildId)\* from $(Build.SourceBranch) of $(Build.Repository.Name) in ${{ parameters.environment\_name }} environment. The deploy build was triggered by $(Build.RequestedFor)." "$(Build.SourcesDirectory)"

- bash: |

echo "DELTA\_DEPLOY\_ID: $(DELTA\_DEPLOY\_ID)"

echo "DELTA\_DEPLOY\_STATUS: $(DELTA\_DEPLOY\_STATUS)"

echo "DELTA\_DESTRUCTIVE\_DEPLOY\_ID: $(DELTA\_DESTRUCTIVE\_DEPLOY\_ID)"

echo "DELTA\_DESTRUCTIVE\_DEPLOY\_STATUS: $(DELTA\_DESTRUCTIVE\_DEPLOY\_STATUS)"

displayName: 'List Validation'

- bash: |

export V\_SOURCE\_BRANCH=$(echo $(Build.SourceBranch) | sed -e "s/refs\/heads\///g")

echo "##vso[task.setvariable variable=SOURCE\_BRANCH;isOutput=true]$V\_SOURCE\_BRANCH"

echo "Source (current) branch: $V\_SOURCE\_BRANCH"

export V\_REPO\_NAME=$(echo $(echo $(Build.Repository.Name) | cut -f2 -d"/"))

echo "##vso[task.setvariable variable=REPOSITORY\_NAME;isOutput=true]$V\_REPO\_NAME"

echo "Repo name: $V\_REPO\_NAME"

displayName: 'Set Variables'

name: set\_vars

- bash: |

echo "REPO: $(set\_vars.REPOSITORY\_NAME)"

echo "Salesforce Org branch: ${{ parameters.target\_salesforce\_org\_branch }}"

echo "Source (current) branch: $(set\_vars.SOURCE\_BRANCH)"

displayName: 'Git Repo & branch details'

- bash: |

sfdx --version

sf --version

displayName: 'Check sfdx CLI version'

- bash: |

# Download Key

- task: DownloadSecureFile@1

name: jwtKeyFile

displayName: 'Download key'

inputs:

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

# Login to Salesforce Org

- bash: |

sf org login jwt --client-id $(salesforceClientid) --jwt-key-file $(jwtKeyFile.secureFilePath) --username $(salesforceUsername) --alias $(alias)

displayName: 'Authenticate to Salesforce org'

- bash: |

echo "Check for On-going Salesforce DEPLOY"

readarray execSOQLQuery < <(sfdx force:data:soql:query -q "SELECT Id, Status, StartDate, CompletedDate FROM DeployRequest where status = 'InProgress'" -t -u $(salesforceUsername))

echo "${execSOQLQuery[\*]}"

displayName: 'Check for On-going Salesforce DEPLOY'

# Deploy Delta Package Change - Add/Modify

- bash: |

set -e

echo "Deploy Delta Package Change - Add/Modify ==> DELTA\_DEPLOY\_ID:$(DELTA\_DEPLOY\_ID)"

cd $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)

export SOURCE\_API\_VERSION=$(cat sfdx-project.json | jq -r .sourceApiVersion)

echo "In sfdx-project.json sourceApiVersion=$SOURCE\_API\_VERSION"

export PACKAGE\_DIRECTORIES=$(cat sfdx-project.json | jq -r .packageDirectories[].path)

mkdir -p $PACKAGE\_DIRECTORIES

sfdx force:source:deploy --apiversion $SOURCE\_API\_VERSION -q $(DELTA\_DEPLOY\_ID) -u $(salesforceUsername) --wait 360

displayName: '==> QUICK DEPLOY: DELTA'

condition: and(ne(variables.DELTA\_DEPLOY\_ID, ''),eq(variables.DELTA\_DEPLOY\_STATUS,'true'),ne(variables.TESTLEVEL, 'NoTestRun'))

- bash: |

set -e

cd $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)

export SOURCE\_API\_VERSION=$(cat sfdx-project.json | jq -r .sourceApiVersion)

echo "In sfdx-project.json sourceApiVersion=$SOURCE\_API\_VERSION"

export PACKAGE\_DIRECTORIES=$(cat sfdx-project.json | jq -r .packageDirectories[].path)

mkdir -p $PACKAGE\_DIRECTORIES

if [[ ( "$(DELTA\_DEPLOY\_STATUS)" == 'true' ) && ( "$(DELTA\_DESTRUCTIVE\_DEPLOY\_STATUS)" == 'false' ) ]]; then

echo "\*\*\* Only ADD/MODIFY CHANGES DETECTED to DEPLOY... \*\*\*"

sfdx force:source:deploy --apiversion $SOURCE\_API\_VERSION -x $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)/package/package.xml -u $(salesforceUsername) --verbose --wait 360

elif [[ ( "$(DELTA\_DEPLOY\_STATUS)" == 'true' ) && ( "$(DELTA\_DESTRUCTIVE\_DEPLOY\_STATUS)" == 'true' ) ]]; then

echo "\*\*\* BOTH ADD/MODIFY & DESTRUCTIVE CHANGES DETECTED to DEPLOY... \*\*\*"

cp $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)/destructiveChanges/destructiveChanges.xml $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)/package/destructiveChangesPost.xml

sfdx force:source:deploy --apiversion $SOURCE\_API\_VERSION -x $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)/package/package.xml --postdestructivechanges $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)/package/destructiveChangesPost.xml -u $(salesforceUsername) --verbose --wait 360

fi

displayName: '==> DEPLOY: DELTA'

condition: and(eq(variables.TESTLEVEL, 'NoTestRun'),eq(variables.DELTA\_DEPLOY\_STATUS,'true'))

# Deploy Delta Destructive Package Change - Delete

- bash: |

set -e

echo "Deploy Delta Destructive Package Change - Delete ==> DELTA\_DESTRUCTIVE\_DEPLOY\_ID:$(DELTA\_DESTRUCTIVE\_DEPLOY\_ID)"

cd $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)

export SOURCE\_API\_VERSION=$(cat sfdx-project.json | jq -r .sourceApiVersion)

echo "In sfdx-project.json sourceApiVersion=$SOURCE\_API\_VERSION"

export PACKAGE\_DIRECTORIES=$(cat sfdx-project.json | jq -r .packageDirectories[].path)

mkdir -p $PACKAGE\_DIRECTORIES

sfdx force:mdapi:deploy --apiversion $SOURCE\_API\_VERSION -q $(DELTA\_DESTRUCTIVE\_DEPLOY\_ID) -u $(salesforceUsername) --wait -1

displayName: '==> QUICK DEPLOY: ONLY DESTRUCTIVE'

condition: and(ne(variables.DELTA\_DESTRUCTIVE\_DEPLOY\_ID, ''),ne(variables.DELTA\_DEPLOY\_STATUS,'true'),eq(variables.DELTA\_DESTRUCTIVE\_DEPLOY\_STATUS,'true'),ne(variables.TESTLEVEL, 'NoTestRun'))

- bash: |

set -e

cd $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)

export SOURCE\_API\_VERSION=$(cat sfdx-project.json | jq -r .sourceApiVersion)

echo "In sfdx-project.json sourceApiVersion=$SOURCE\_API\_VERSION"

export PACKAGE\_DIRECTORIES=$(cat sfdx-project.json | jq -r .packageDirectories[].path)

mkdir -p $PACKAGE\_DIRECTORIES

sfdx force:mdapi:deploy --apiversion $SOURCE\_API\_VERSION -d $(Pipeline.Workspace)/download\_artifacts/artifact\_$(Build.BuildId)/destructiveChanges -u $(salesforceUsername) --wait 360 --verbose

displayName: '==> DEPLOY: ONLY DESTRUCTIVE'

condition: and(eq(variables.TESTLEVEL, 'NoTestRun'),ne(variables.DELTA\_DEPLOY\_STATUS,'true'),eq(variables.DELTA\_DESTRUCTIVE\_DEPLOY\_STATUS,'true'))

- bash: |

echo "Check for On-going Salesforce DEPLOY"

readarray execSOQLQuery < <(sfdx force:data:soql:query -q "SELECT Id, Status, StartDate, CompletedDate FROM DeployRequest where status = 'InProgress'" -t -u $(salesforceUsername))

echo "${execSOQLQuery[\*]}"

displayName: 'Check for On-going Salesforce DEPLOY'

- task: Bash@3

displayName: Send Deploy Build Status Notification

condition: succeededOrFailed()

continueOnError: true