You can find the prerequisites and steps to create and configure the pipeline in the document linked below, covering pages 1 through 4:

[Pre-requisites for setting up and creating pipeline](https://docs.google.com/document/d/1d2KAfxHTXUNs4FFbZ0fTIOS1YXW__9Ao4QNub9FRrpE/edit?usp=sharing)

**Below are the roles granted for the service account for running manual sql pipeline:**

Storage Admin

BigQuery Admin

DataCatalog admin

**Pipeline config for initial deployment of sql files to bigquery(datasets, tables, storedprocedures):**-

# Starter pipeline

# Start with a minimal pipeline that you can customize to build and deploy your code.

# Add steps that build, run tests, deploy, and more:

# https://aka.ms/yaml

trigger:

- none

pool:

vmImage: ubuntu-latest

steps:

- checkout: self

clean: true

persistCredentials: true

fetchDepth: 0

- task: DownloadSecureFile@1

name: authkey

displayName: 'Download Service Account Key'

inputs:

secureFile: 'ntta-gcp-poc-6a8a5f2599b4.json'

retryCount: '2'

- script: |

pip install -r requirements.txt

displayName: 'Installing the required libraries'

- script: |

echo $(authkey.secureFilePath)

gcloud config set project ntta-gcp-poc

gcloud auth activate-service-account --key-file $(authkey.secureFilePath)

export GOOGLE\_APPLICATION\_CREDENTIALS=$(authkey.secureFilePath)

displayName: 'Authentication of service account'

- powershell: |

# Get the SQL code files for dataset creation

$datasetFiles = Get-ChildItem -Path Datasets -Filter \*.sql

# Create a separate error log file

$errorLogFile = "$(Build.ArtifactStagingDirectory)/datasets.log"

New-Item -Path $errorLogFile -ItemType File -Force | Out-Null

# Execute SQL files to create datasets in BigQuery

foreach ($file in $datasetFiles) {

$content = Get-Content $file -Raw

$result = $content | bq query --use\_legacy\_sql=false --location=us-south1 2>&1

if ($LASTEXITCODE -ne 0) {

$errorMessage = "##[error] Error occurred while executing SQL script ${file}: $result"

$errorMessage | Out-File -FilePath $errorLogFile -Append

Write-Host "##[error] $errorMessage"

}

}

displayName: 'Creating Datasets'

- bash: |

chmod +x Deployment/deploy\_tables.sh

./Deployment/deploy\_tables.sh > /dev/null 2> "$(Build.ArtifactStagingDirectory)/tables.log"

displayName: 'Execute shell script to create tables'

- bash: |

chmod +x Deployment/deploy\_storedprocedures.sh

./Deployment/deploy\_storedprocedures.sh > /dev/null 2> "$(Build.ArtifactStagingDirectory)/stored\_procedures.log"

displayName: 'Execute shell script to create stored procedures'

- task: PublishBuildArtifacts@1

inputs:

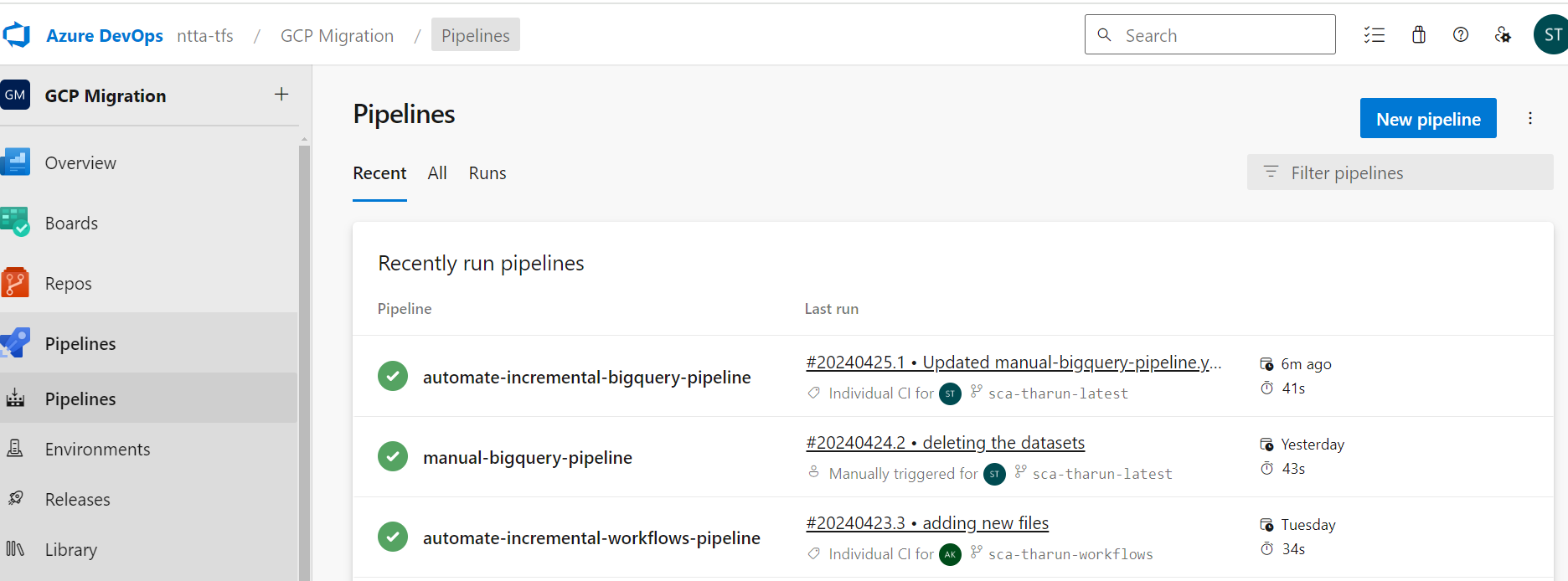
pathtoPublish: '$(Build.ArtifactStagingDirectory)'

artifactName: 'ErrorLogs'

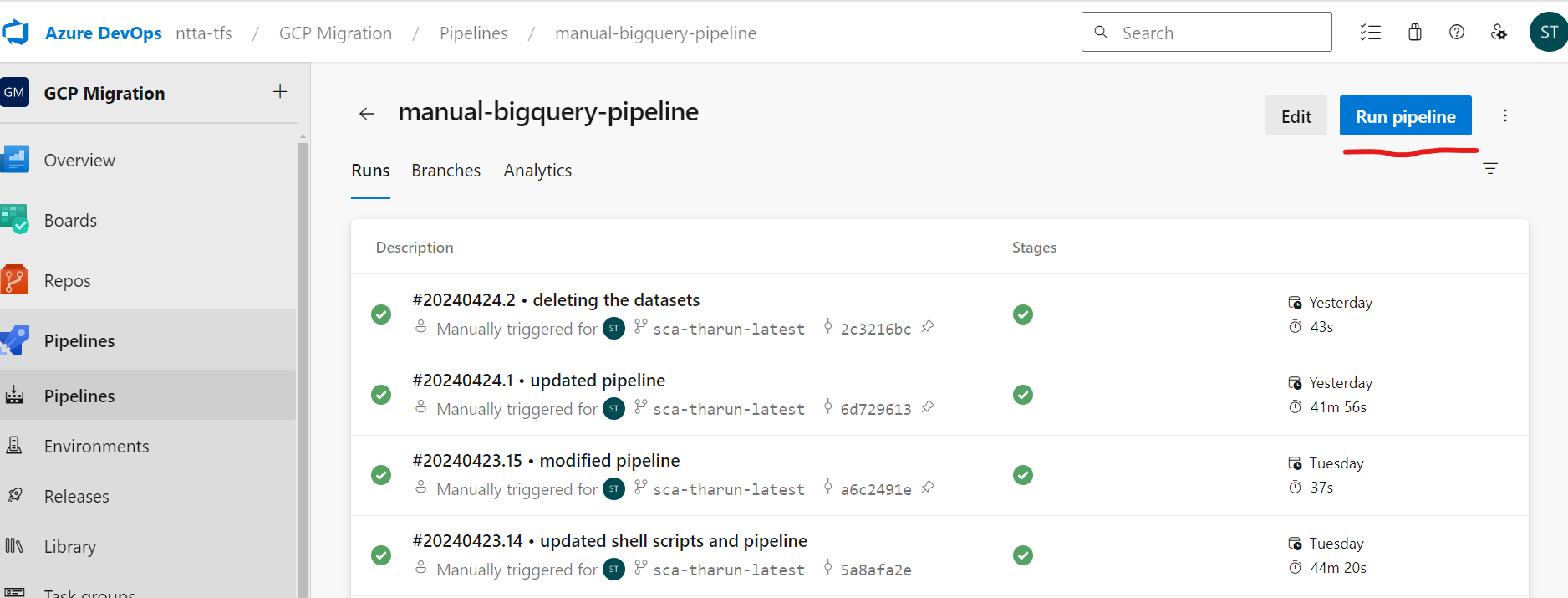
condition: always() # Ensure this task always runs, even if previous steps failed

**Steps to trigger the pipeline manually:-**

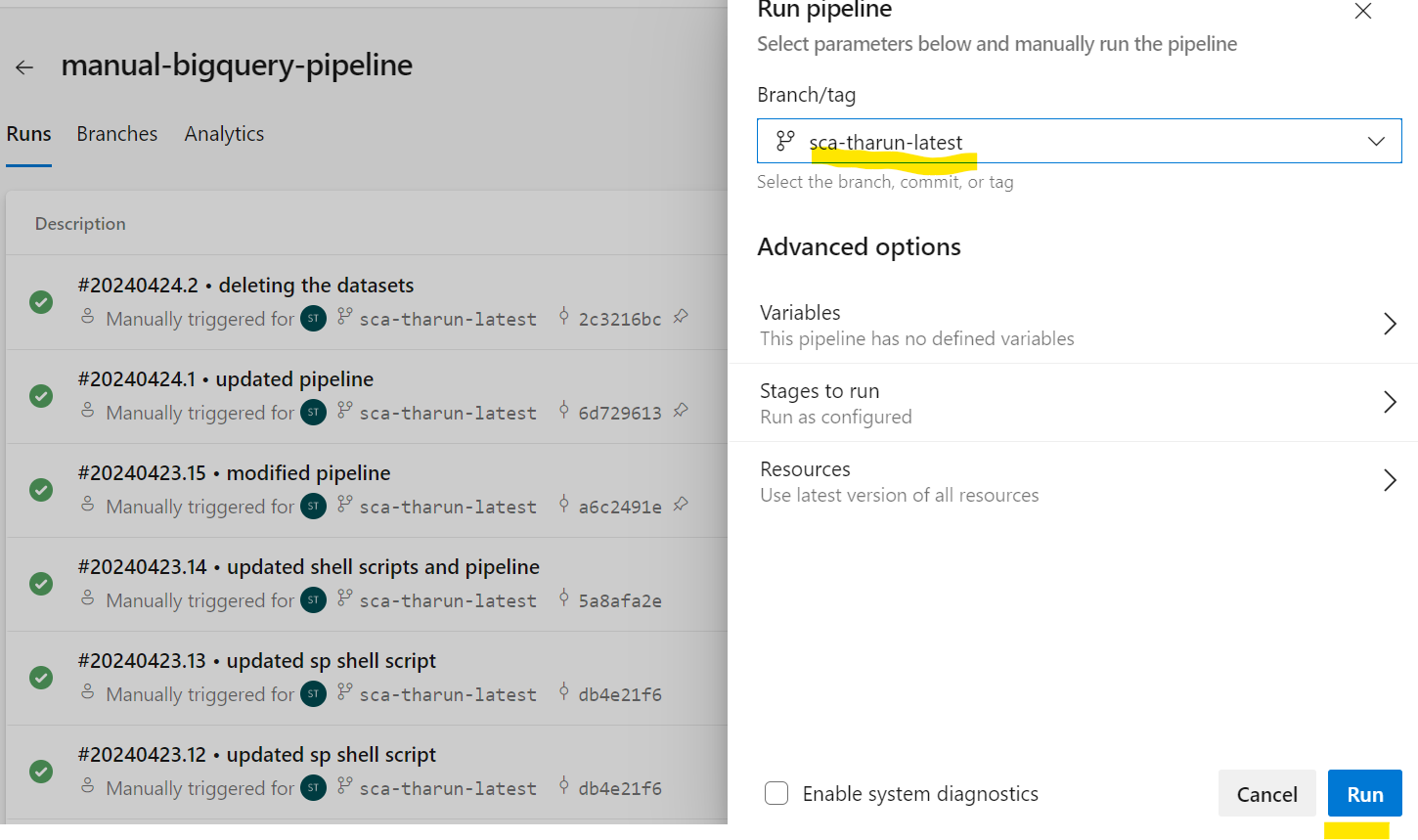
1. In Azure DevOps, go to the Pipelines section and click on it. Here, you'll find a list of all created pipelines, as illustrated in the screenshot below.



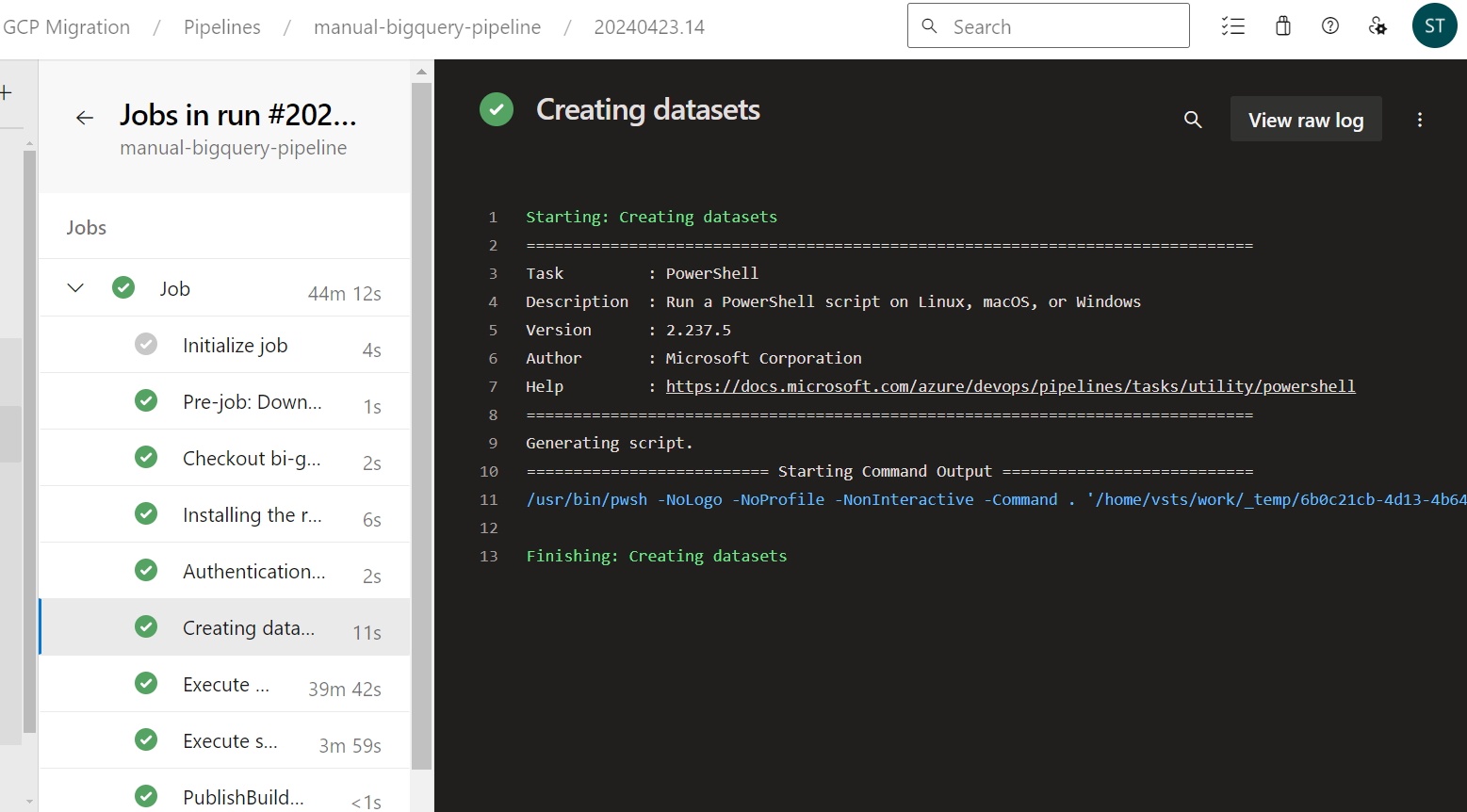
1. Next, select the pipeline you wish to trigger (for our case we use manual-bigquery-pipeline). Here, you'll find comprehensive details about this pipeline, including all its runs, branches, and analytics.
2. Look to the top right corner of the page, where you'll find a button labeled "Run Pipeline." Click on this button to manually initiate the pipeline as illustrated in the screenshot below



1. Upon clicking "Run Pipeline," a pop-up window will appear prompting you to select the branch for which the pipeline was configured, as depicted in the screenshot below. Finally click on the “Run” button to trigger the pipeline.



**Screenshot of successful job for sql files deployment to bigquery:**

****

**Error logs:-**

Error logs are captured in the log files in case of any errors during the execution of the pipeline job. These logs are accessible within the "Artifacts" generated by the job run, as demonstrated in the screenshot below.

