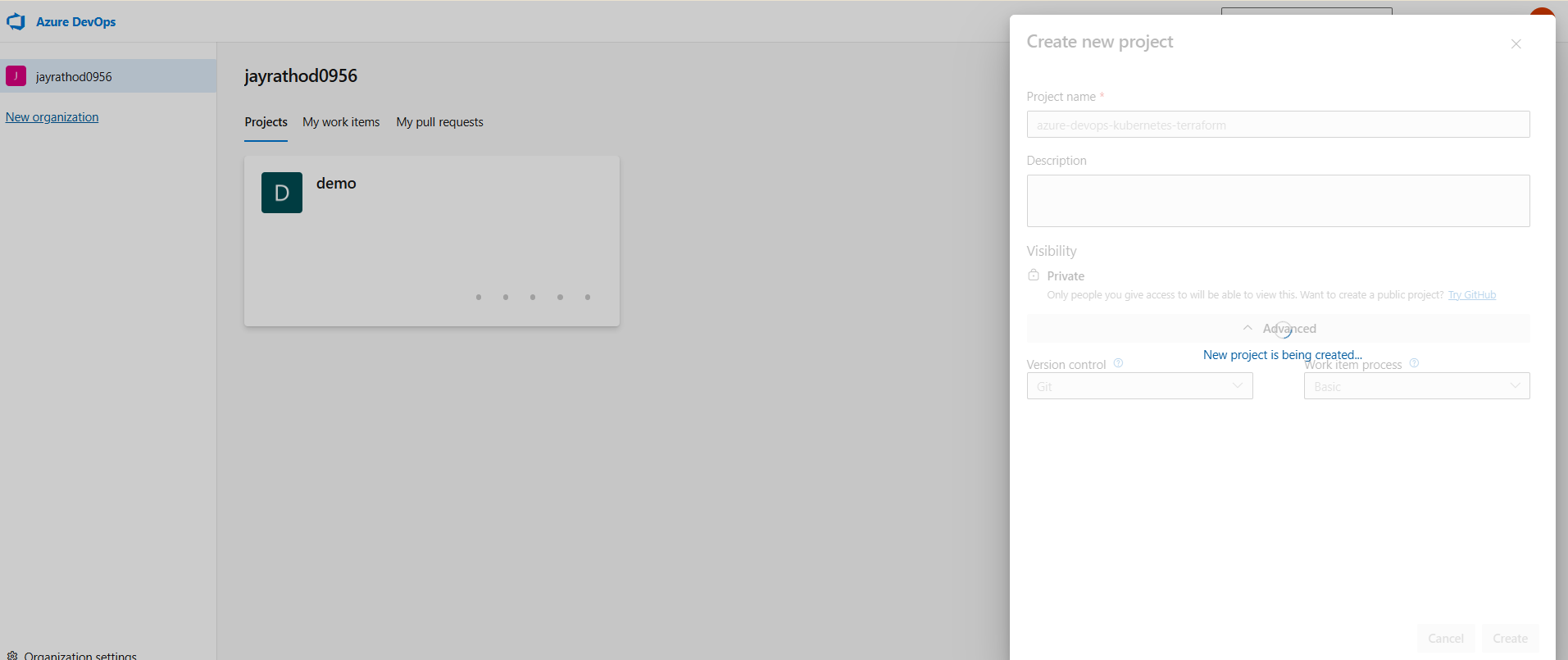
# CICD with Azure DevOps – Kubernetes and Terraform

Create a private project



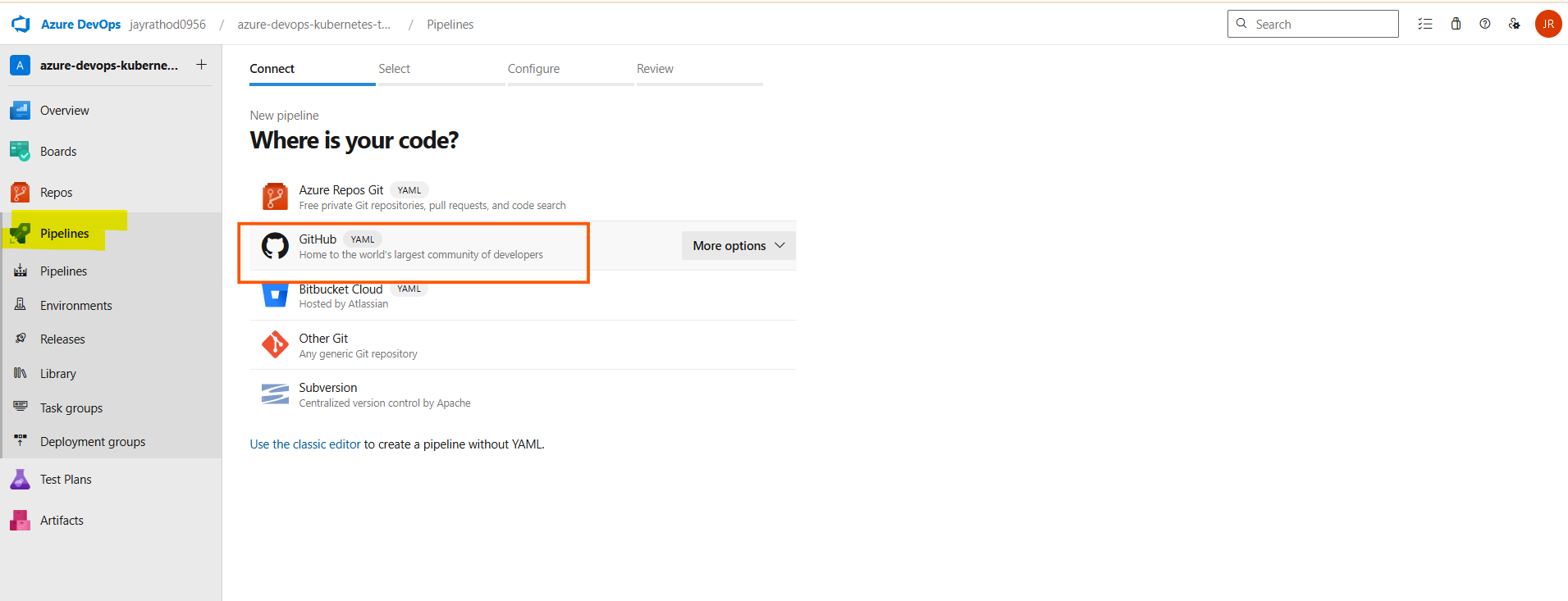
Create a private repo in the github

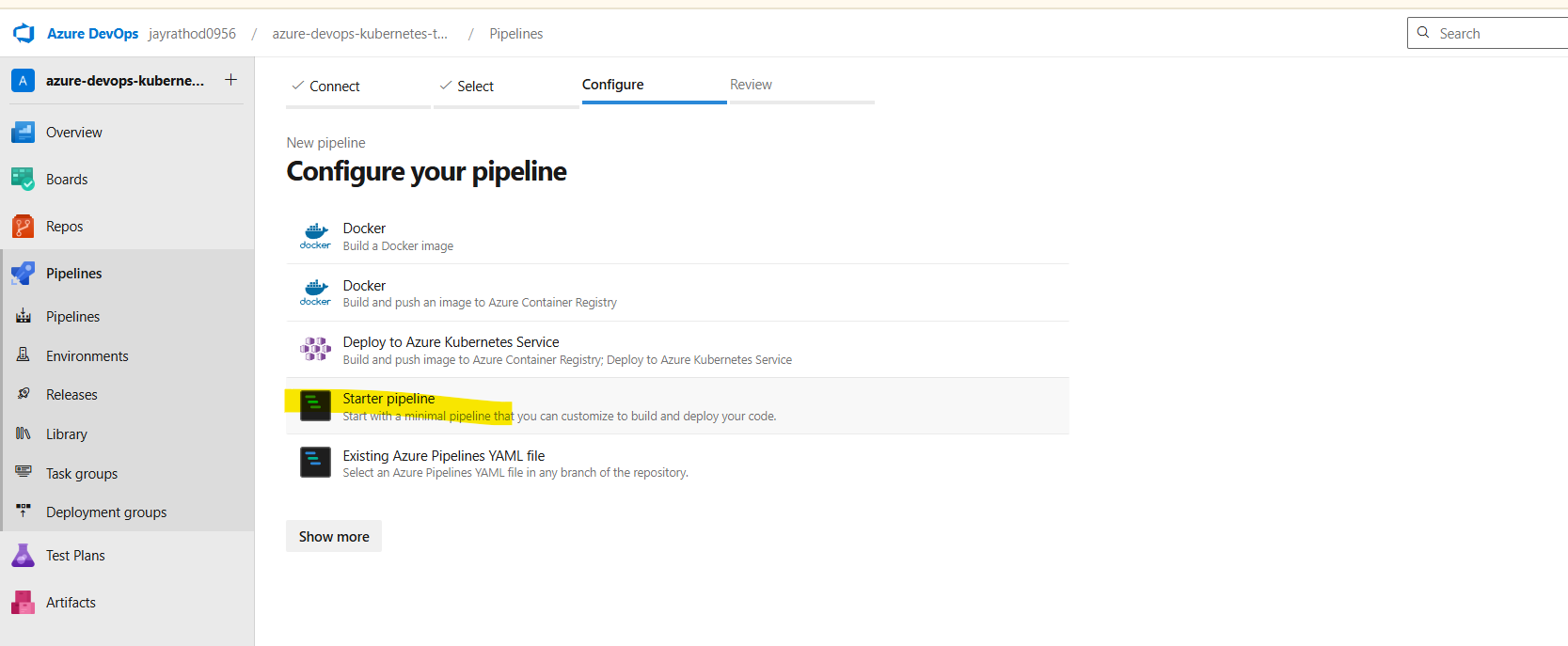
*azure-devops-kubernetes-terraform-pipeline*

Upload java app to git with master branch

Import git to Azure Devops

Go to pipeline > create a pipeline >





trigger:

branches:

include:

- master

stages:

- stage: \_\_default

jobs:

- job: Job

pool:

name: Default

steps:

- task: CmdLine@2

displayName: 'Run a one-line script'

inputs:

script: echo Hello, world!

- task: CmdLine@2

displayName: 'Run a multi-line script'

inputs:

script: |

echo Add other tasks to build, test, and deploy your project.

echo See https://aka.ms/yaml

Run the local agent , replace pool with default and then click on run pipeline

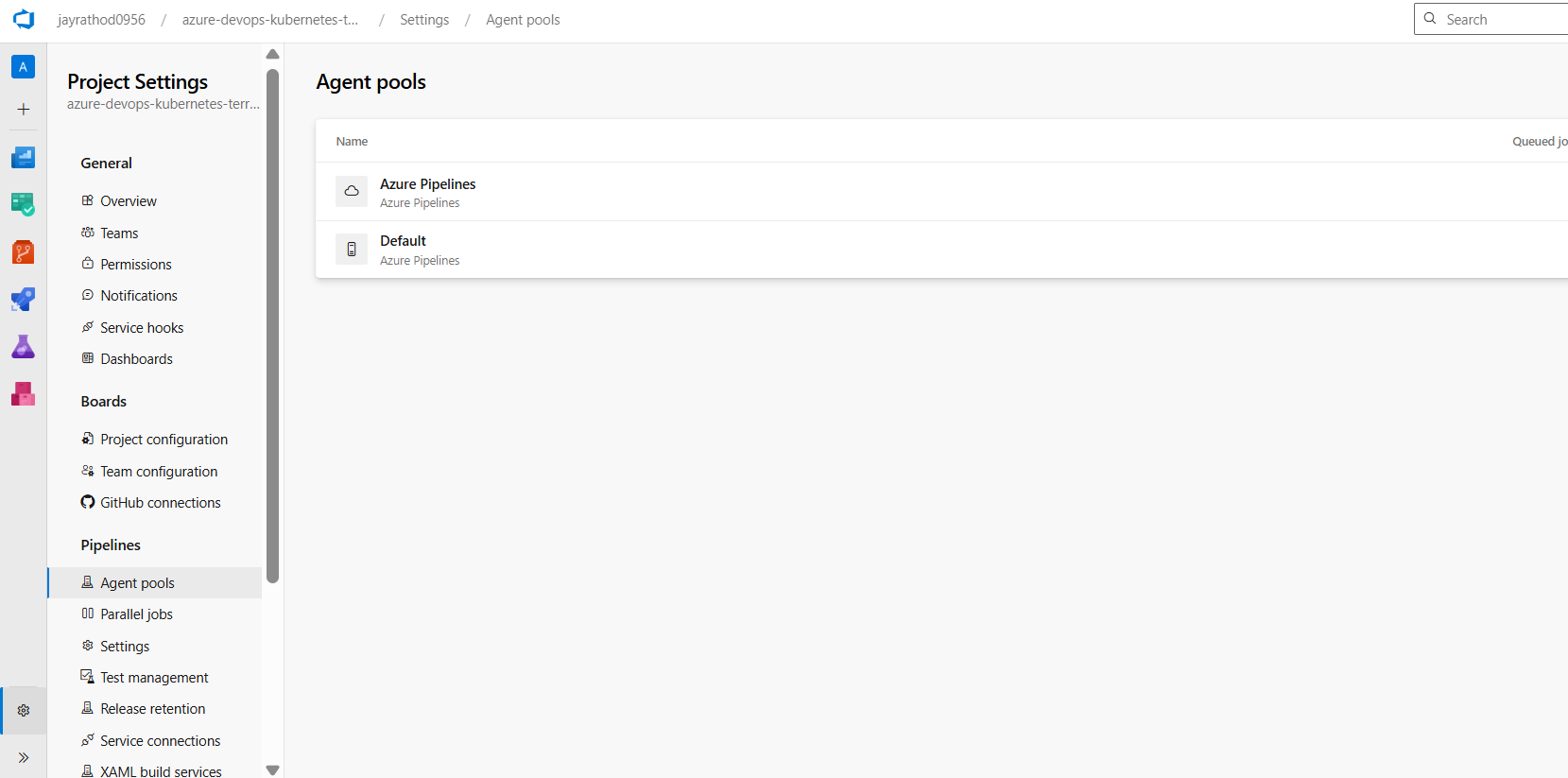
This will create a new file.

Git pull will get that in the local repo.

Change some content , commit and push it on the github

Automatically it will start pipeline

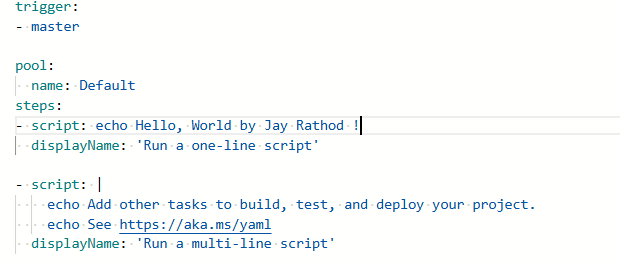
For Agent pool details



## Want to run multiple jobs :

Pipelines > stages > jobs > tasks

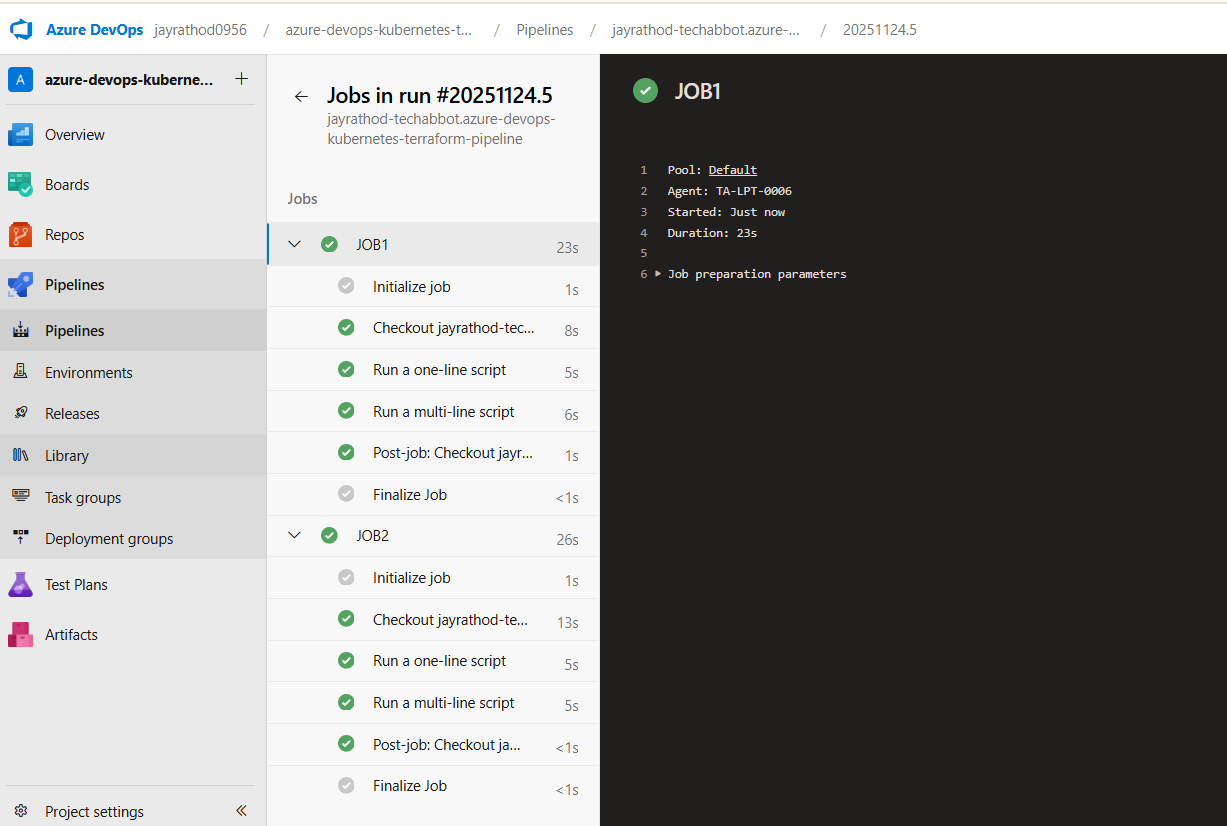
Previously



After



Output

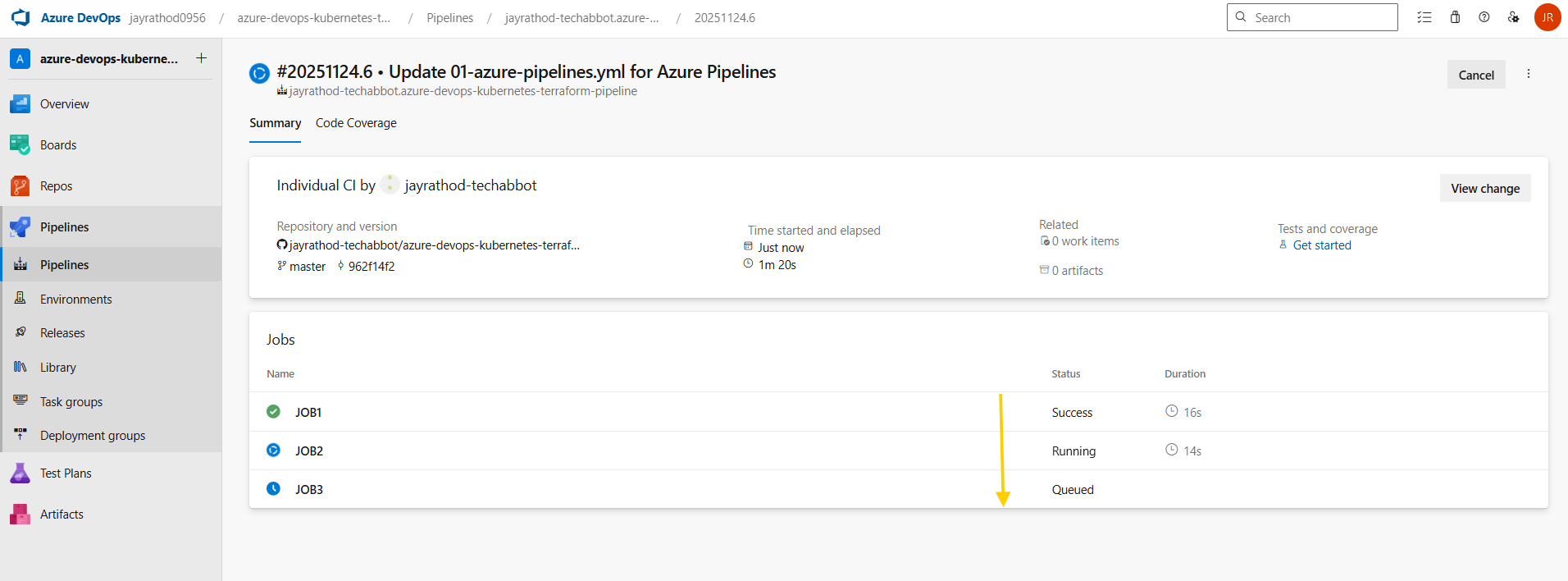


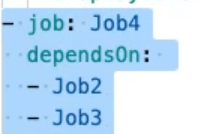
Each job run on agent, parallel if no dependency

By default run on single agent and in random order

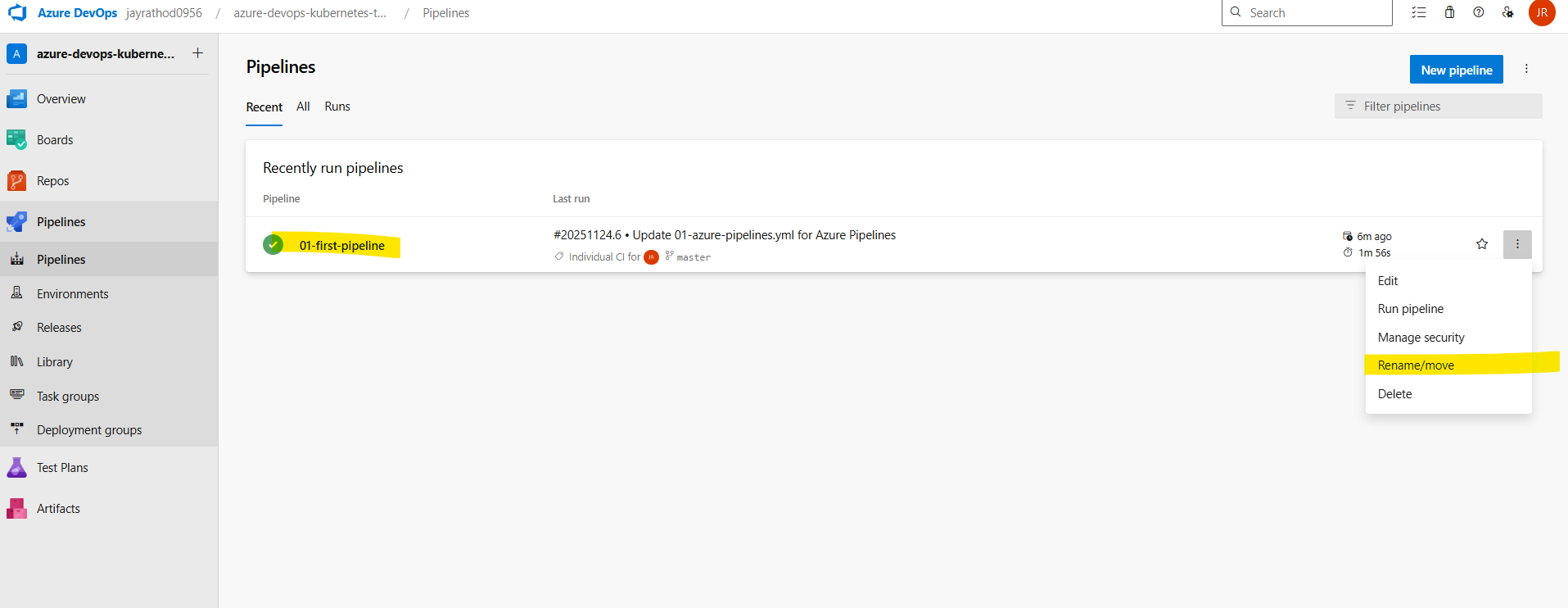
## To make sequence : Job 1 > job 2 > job 3







Rename the pipeline

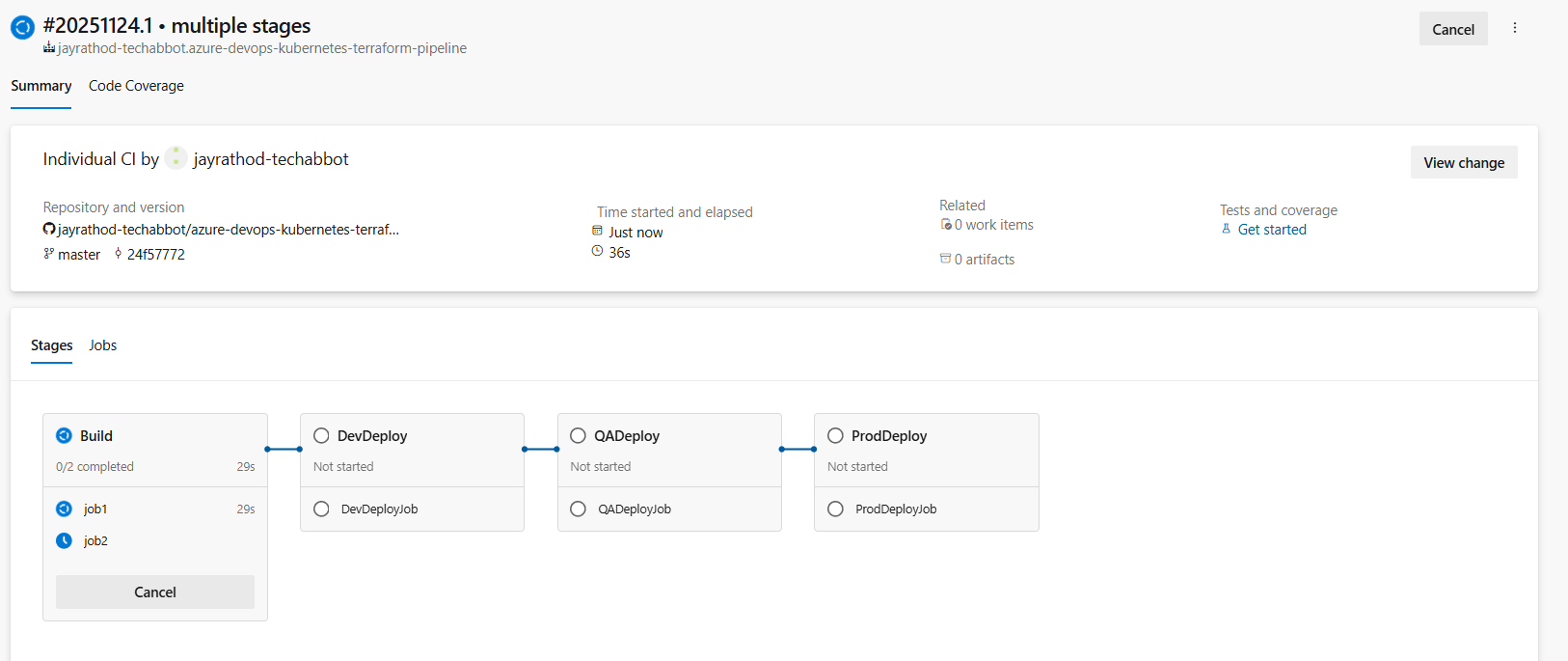


# Multiple stages

All stages are run sequenciary , job inside these stages can run in any order

Create a new pipeline >





## Issue

your pipeline is **running on a Windows self-hosted agent**

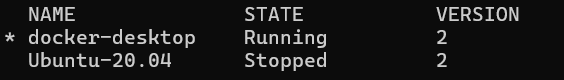
On Windows, Azure Pipelines will try to run Bash using **WSL** (C:\WINDOWS\system32\bash.exe), but WSL is **not installed** — so it fails.



wsl.exe --list –online

wsl.exe --install <Distro>

wsl -l -v



wsl -d Ubuntu-20.04

wsl --unregister Ubuntu-20.04

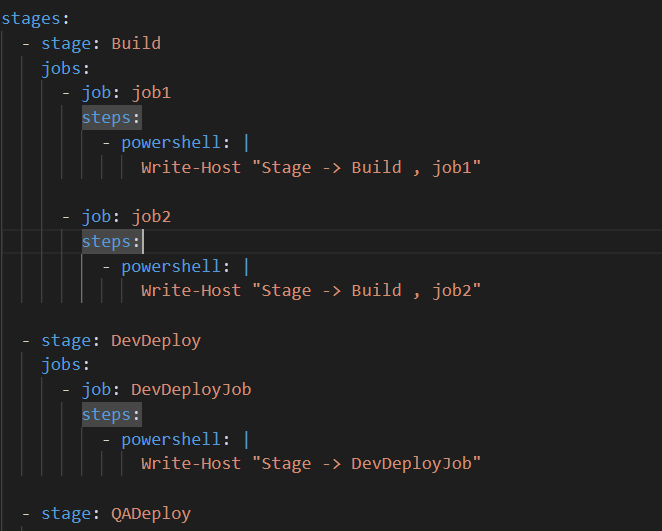
wsl --install -d Ubuntu-20.04

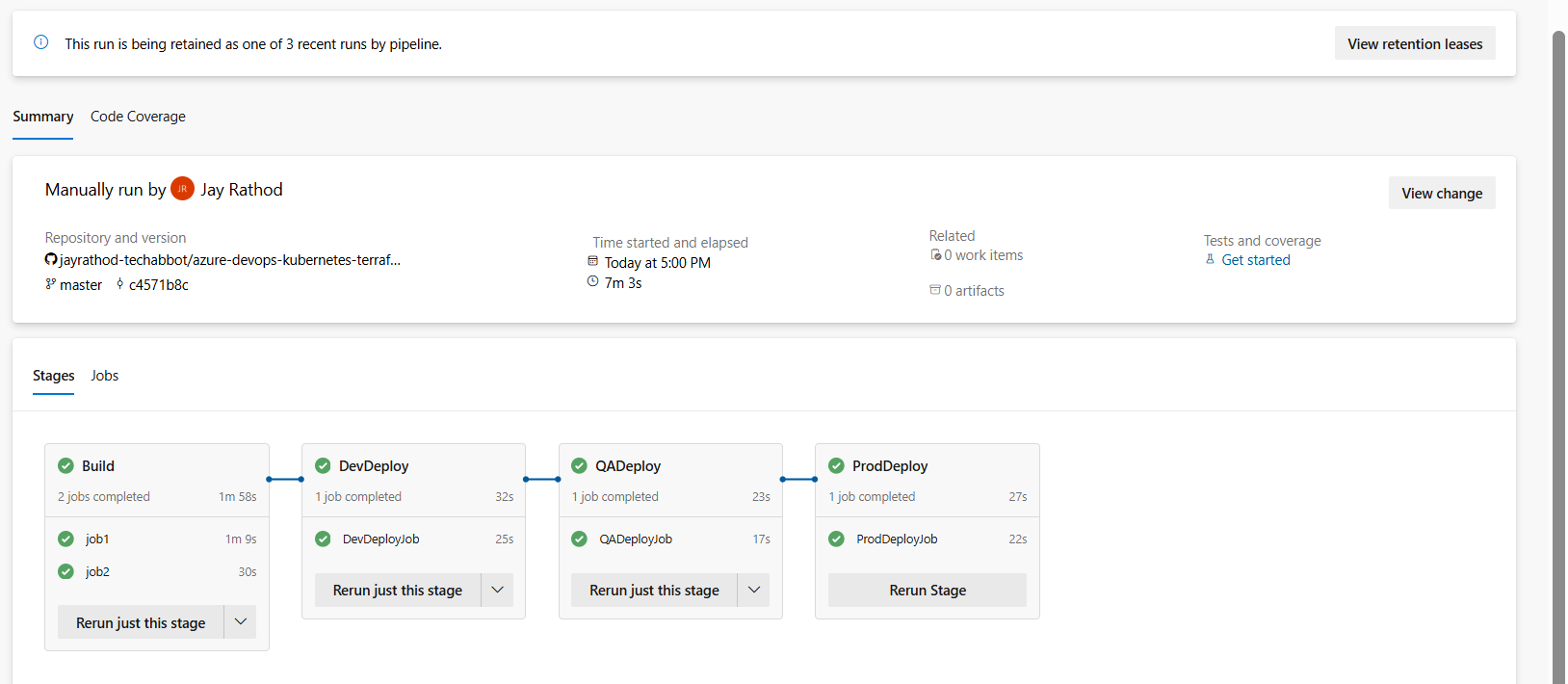
Restart PC → Open Ubuntu → finish setup

Test:

C:\Windows\System32\bash.exe -c "echo Bash OK"

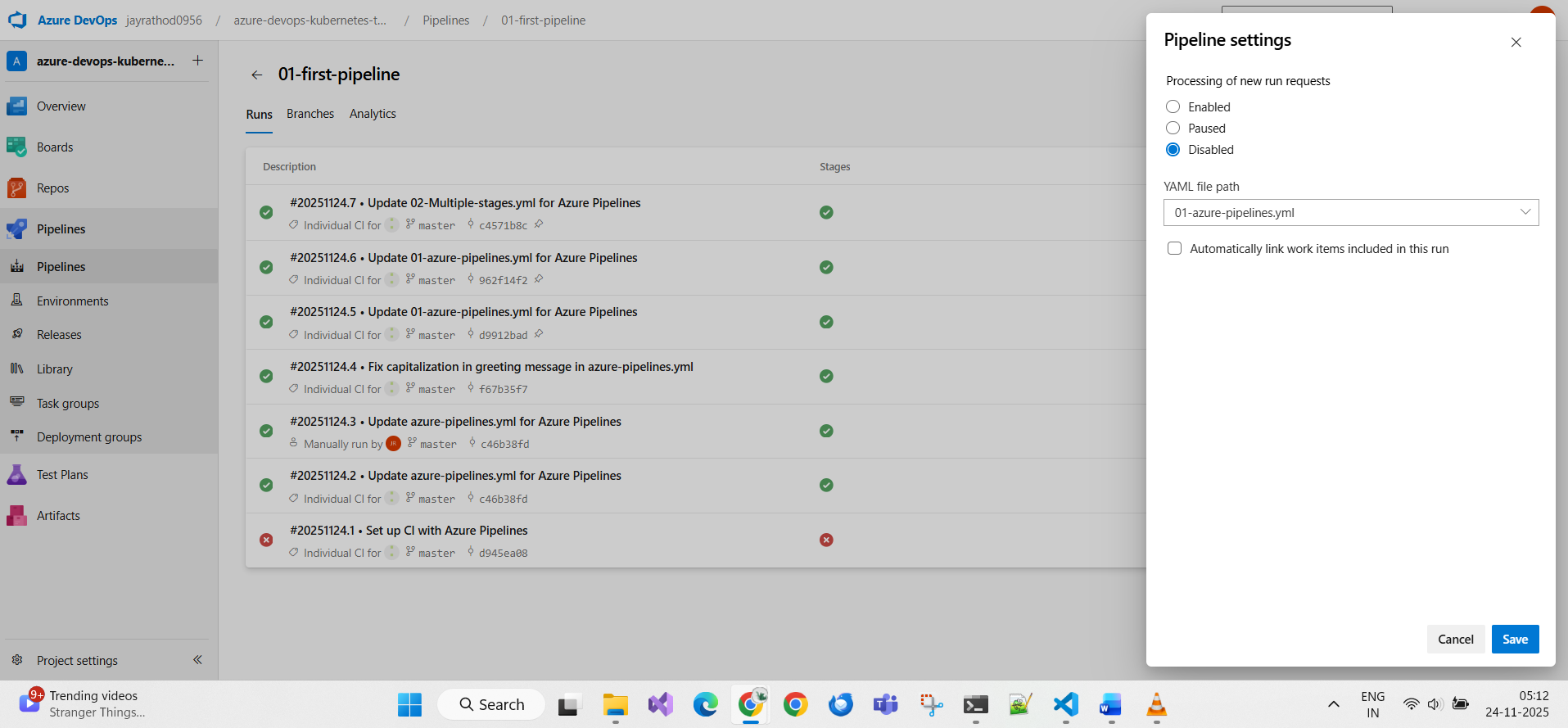
Not working so changed with Powershell



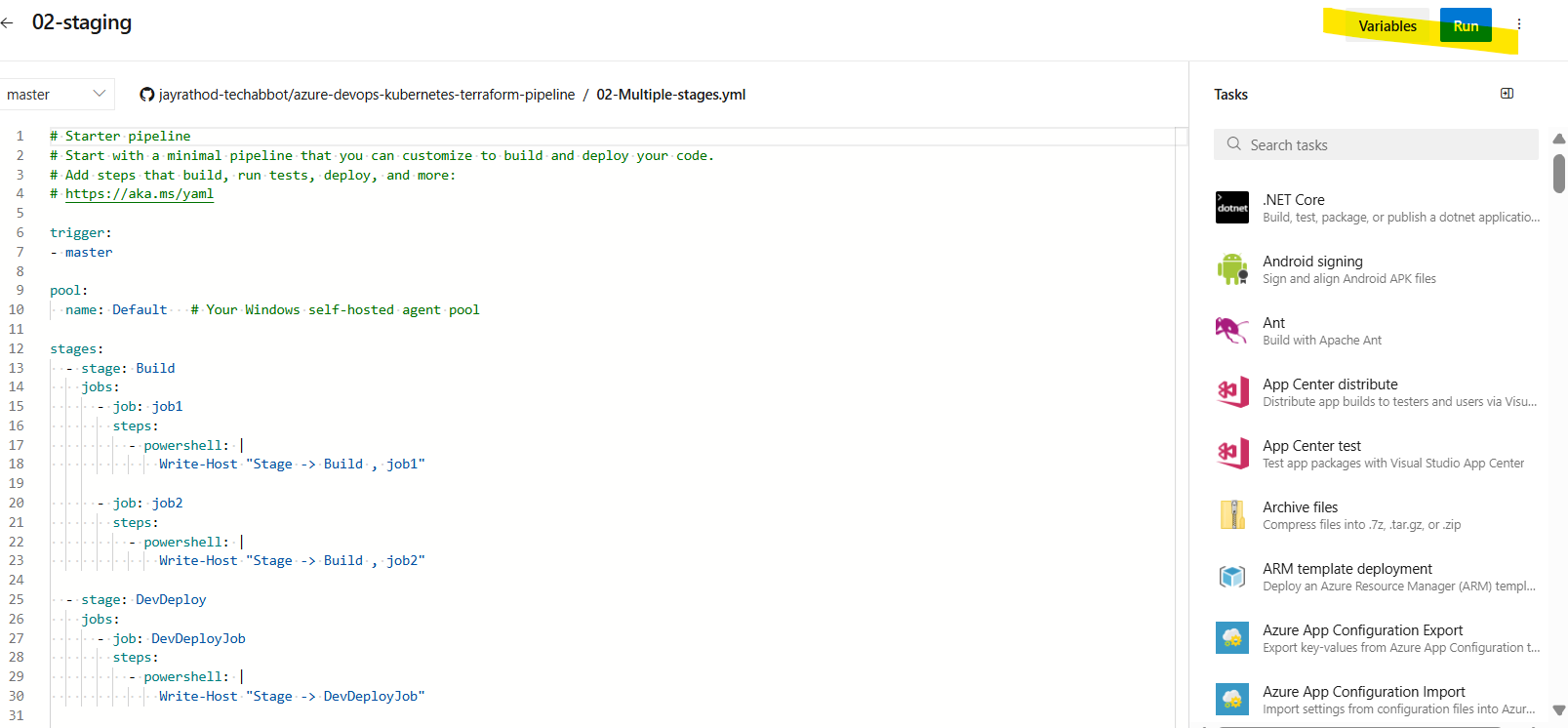


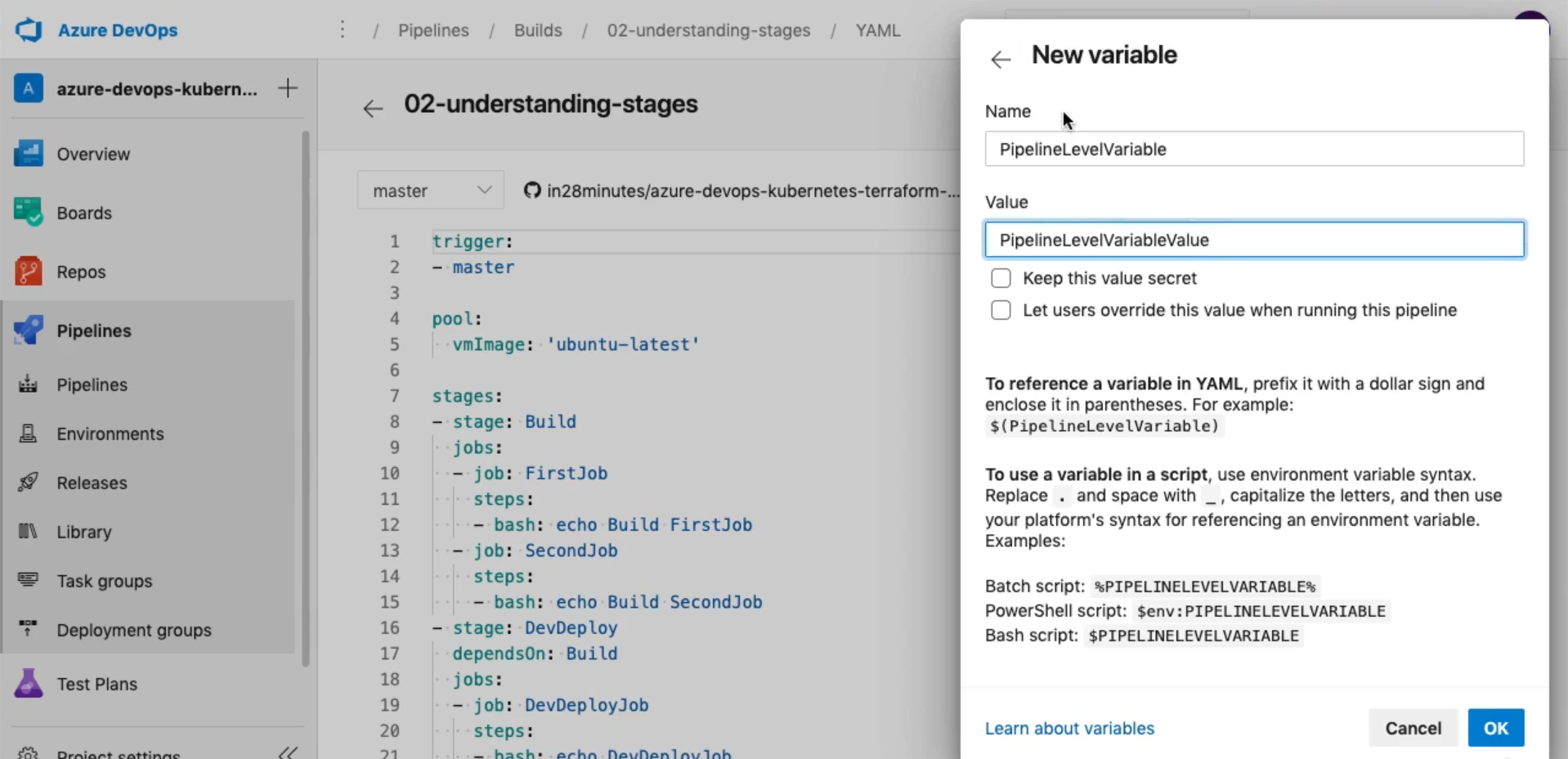
Also we can applied dependsOn at stage level

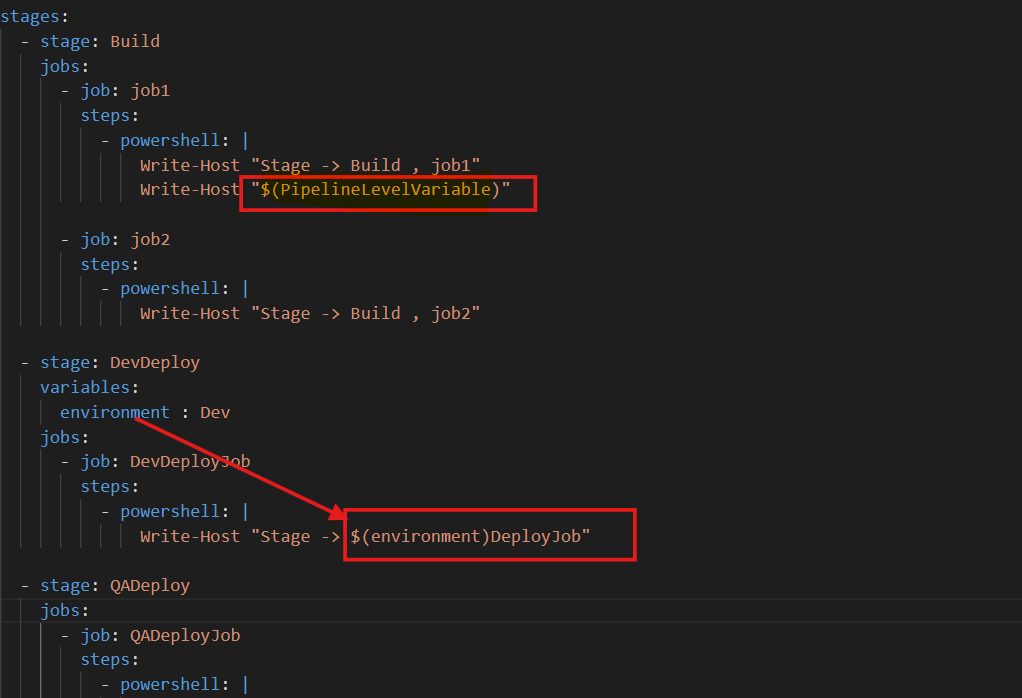
## Do disable other pipeline



## Variable at pipeline level and stage level







# Pre-defined variables

stages:

  - stage: Build

    jobs:

    - job: FirstJob

      displayName: "Print Build Variables"

      steps:

      - powershell: Write-Host "Build\_FirstJob"

        displayName: "Print custom variable"

      - powershell: Write-Host $(PipelineLevelVariable)

        displayName: "Pipeline Level Variable"

      - powershell: Write-Host $(Build.BuildNumber)

        displayName: "Build Number"

      - powershell: Write-Host $(Build.BuildId)

        displayName: "Build ID"

      - powershell: Write-Host $(Build.SourceBranchName)

        displayName: "Source Branch"

      - powershell: Write-Host $(Build.SourcesDirectory)

        displayName: "Sources Directory"

      - powershell: Write-Host $(System.DefaultWorkingDirectory)

        displayName: "Default Working Directory"

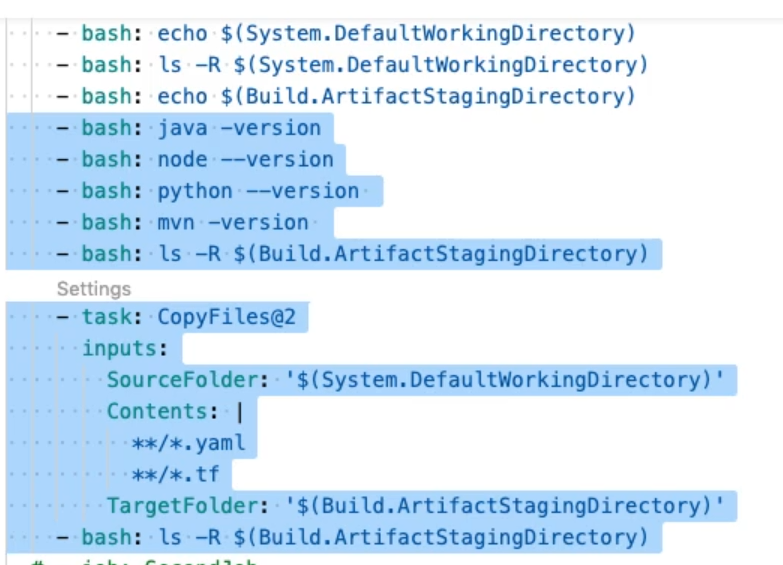
      - powershell: Get-ChildItem -Recurse $(System.DefaultWorkingDirectory)

        displayName: "List Working Directory"

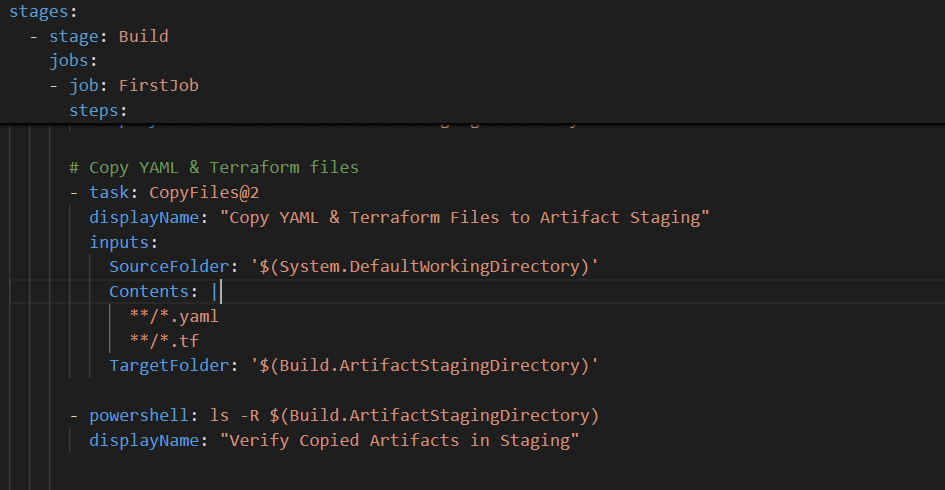
      - powershell: Write-Host $(Build.ArtifactStagingDirectory)

        displayName: "Artifact Staging Directory"

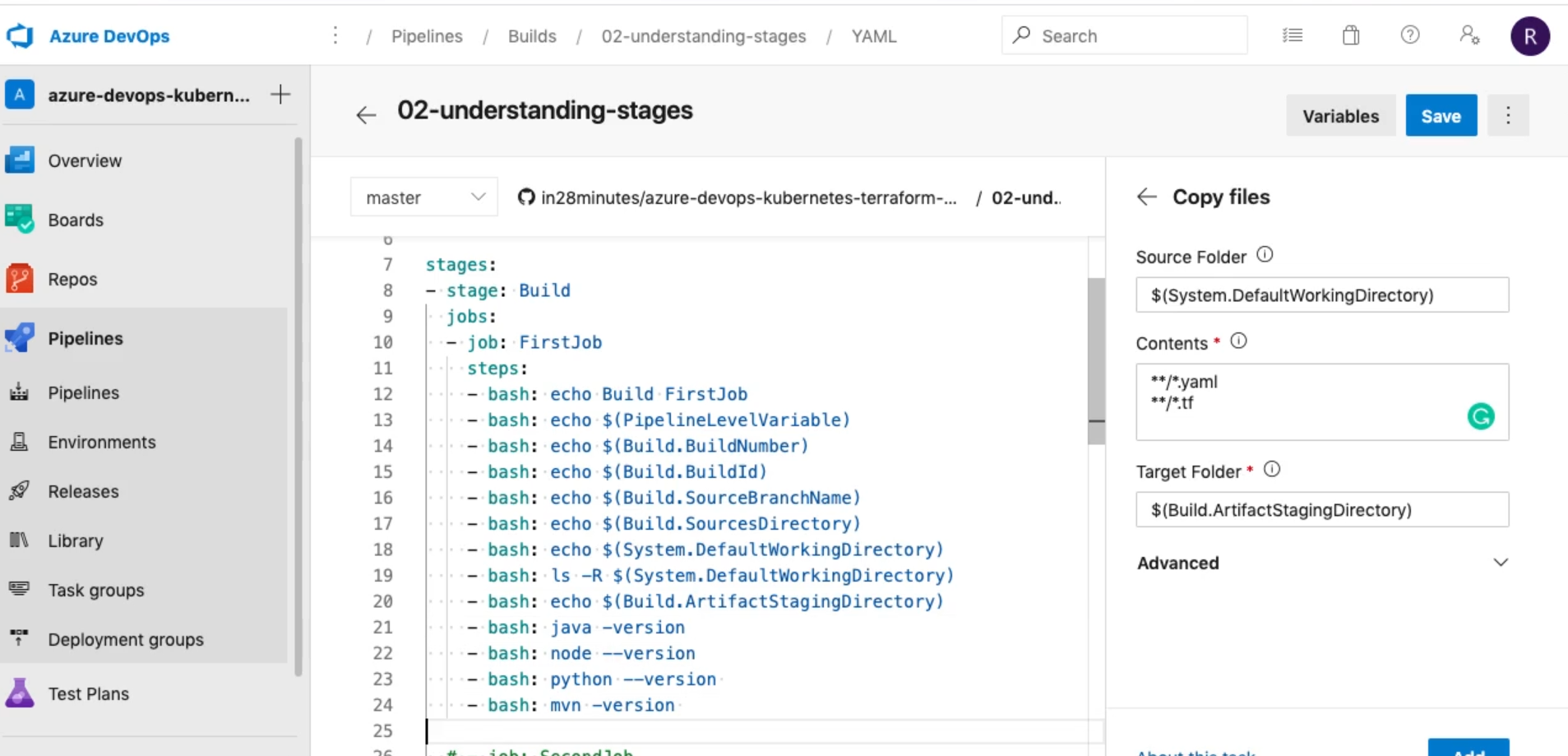
# Copy files from current working to staging



CopyFiles@2 is a built-in Azure DevOps Pipeline task used to **copy files from one directory to another** during a pipeline execution.



Through Azure platform

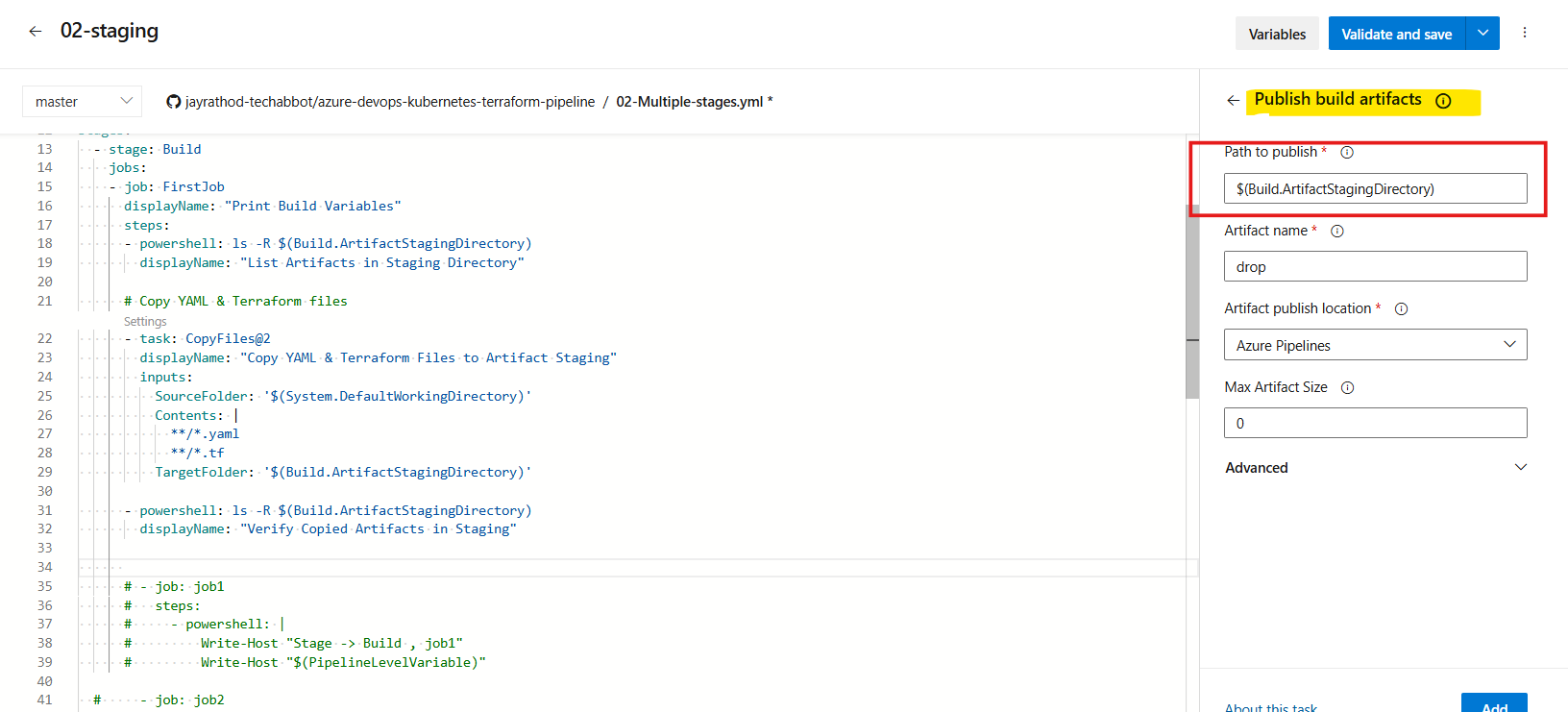


## Issue

Install python, java at the system level

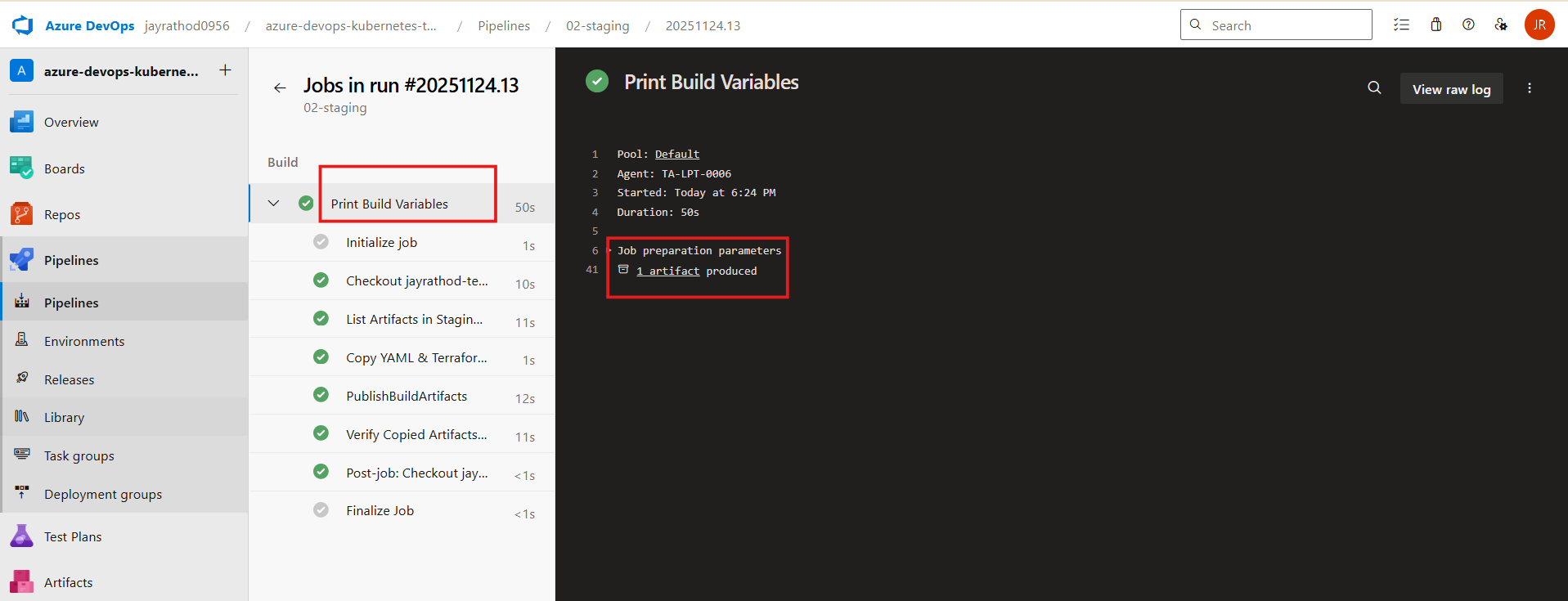
# Publish Artifacts

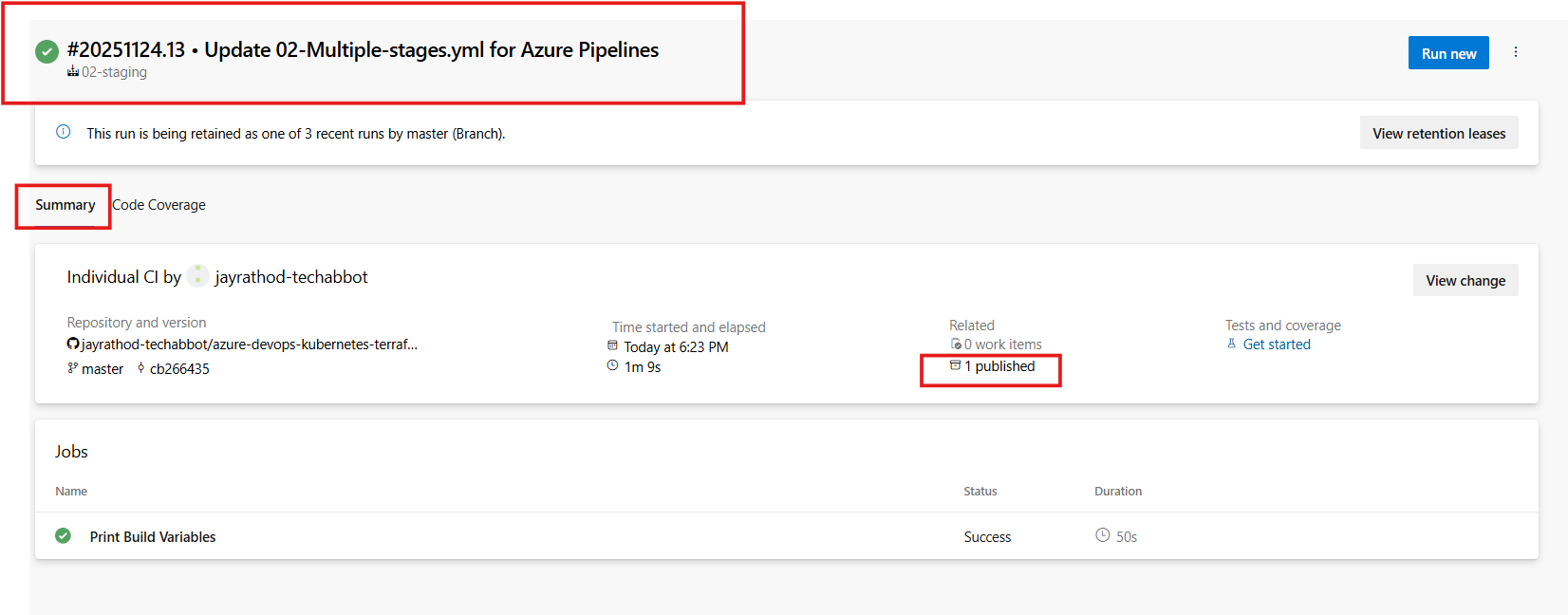
To share content between stages





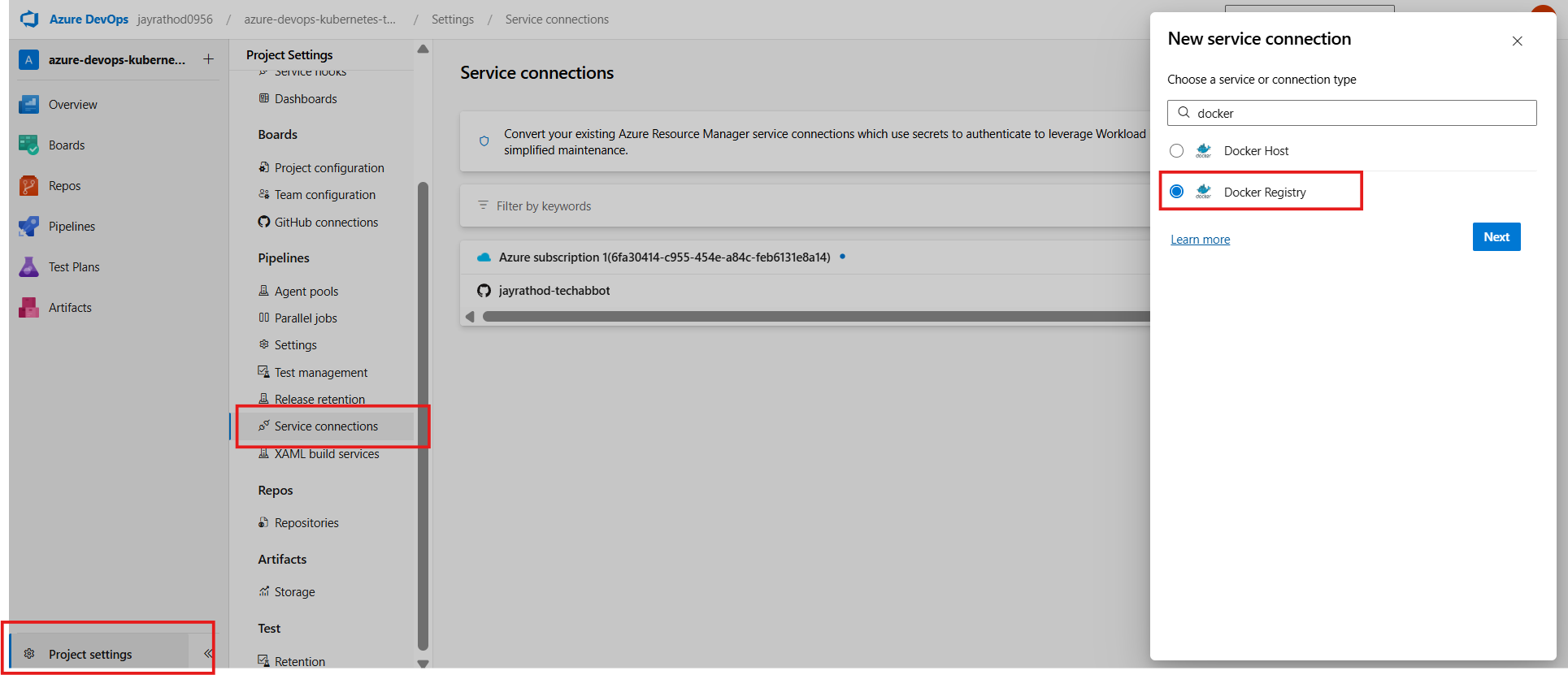
For verify





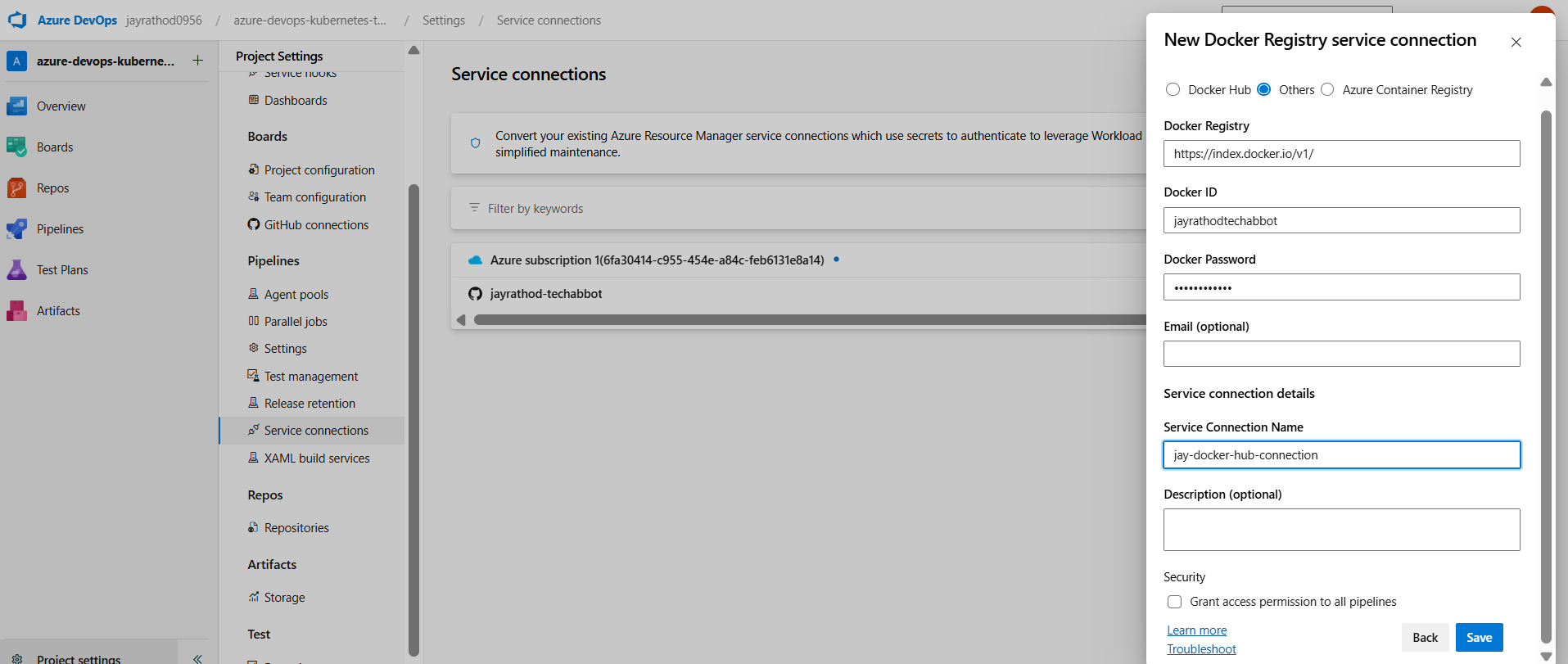
# Build & Push docker image

Make a connect to docker hub from azure devops at project level

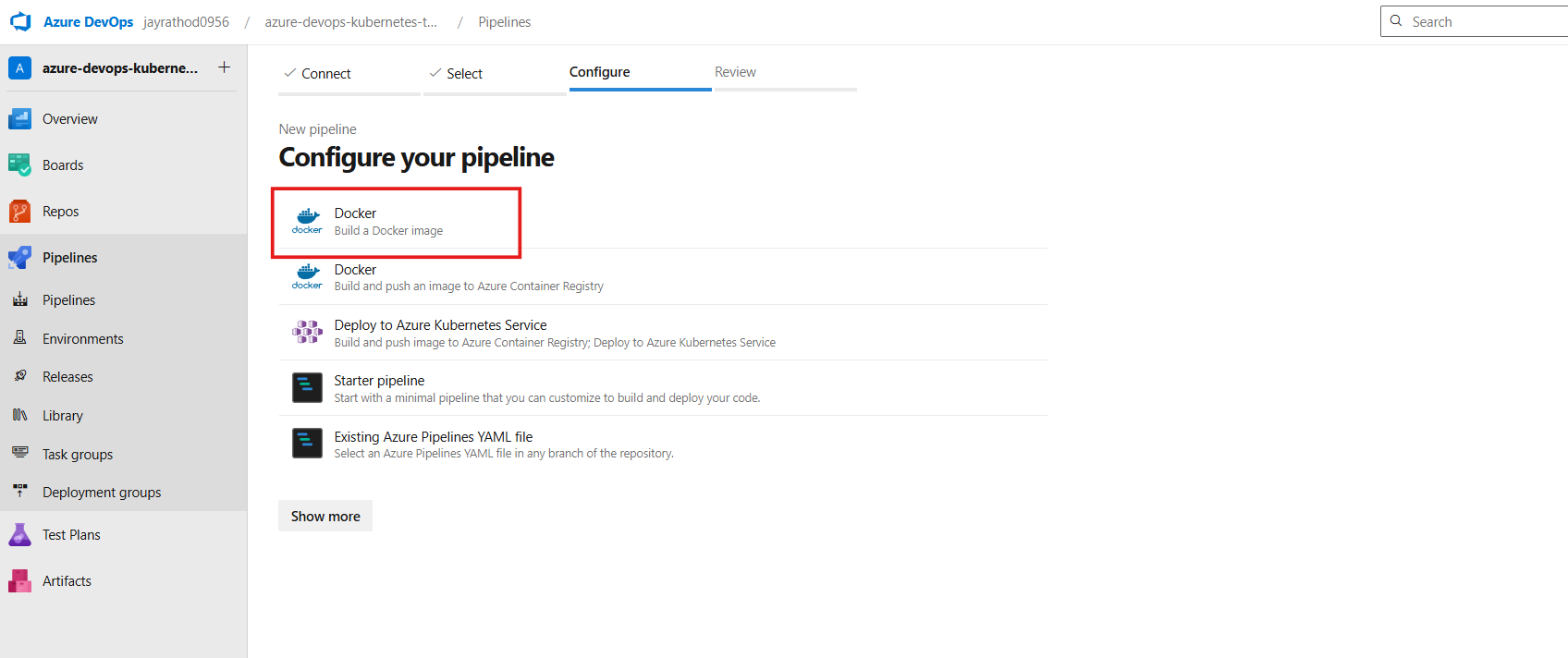


Provide username , password of hub

Connection name = **jay-docker-hub-connection**



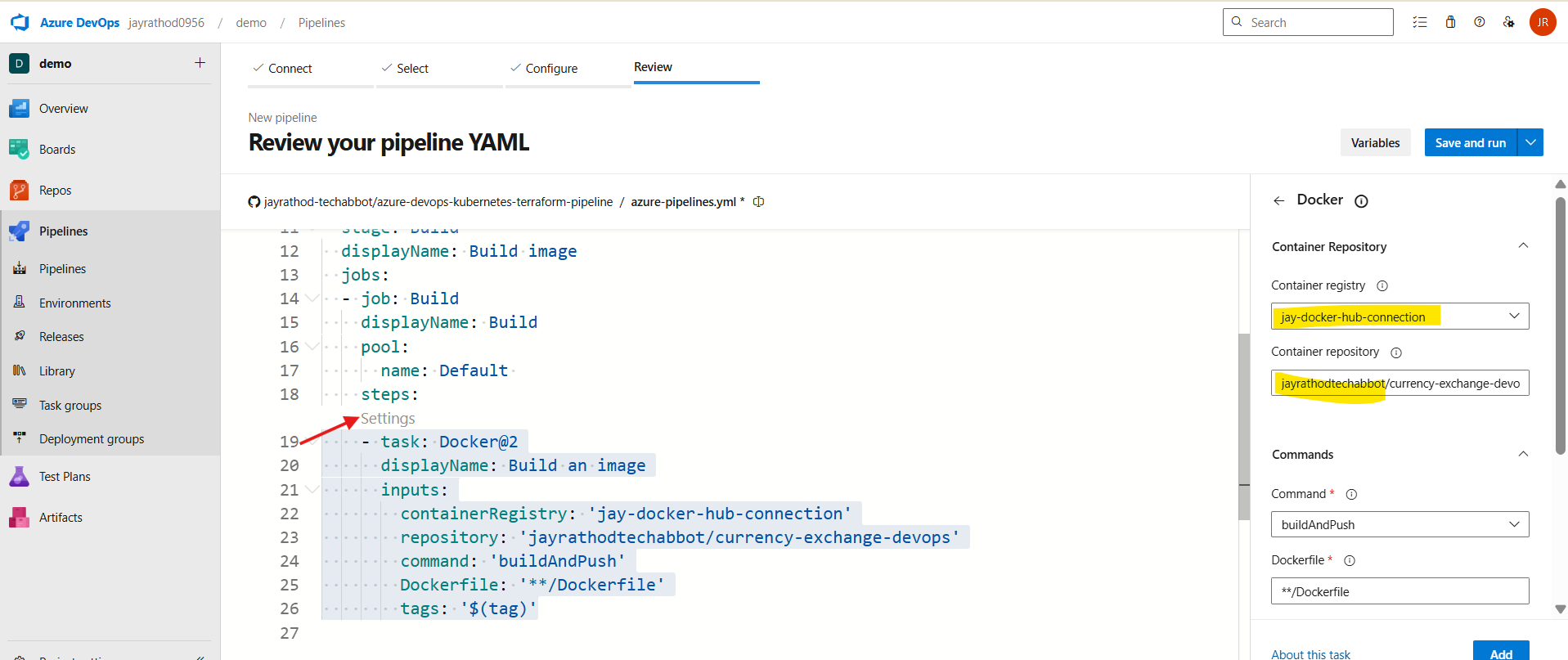
Make a new pipline with github



verify the dockerfile ‘s path

show assistant > docker

select connections



Or click on settings > select registry >

Container repo : <dockerid> / <project\_name>

Command – build and push

Click on add

## Issue - error during connect: in the default daemon configuration on Windows, the docker client must be run with elevated privileges to connect



cd D:\Jay\agent\vsts-agent-win-x64-4.264.2

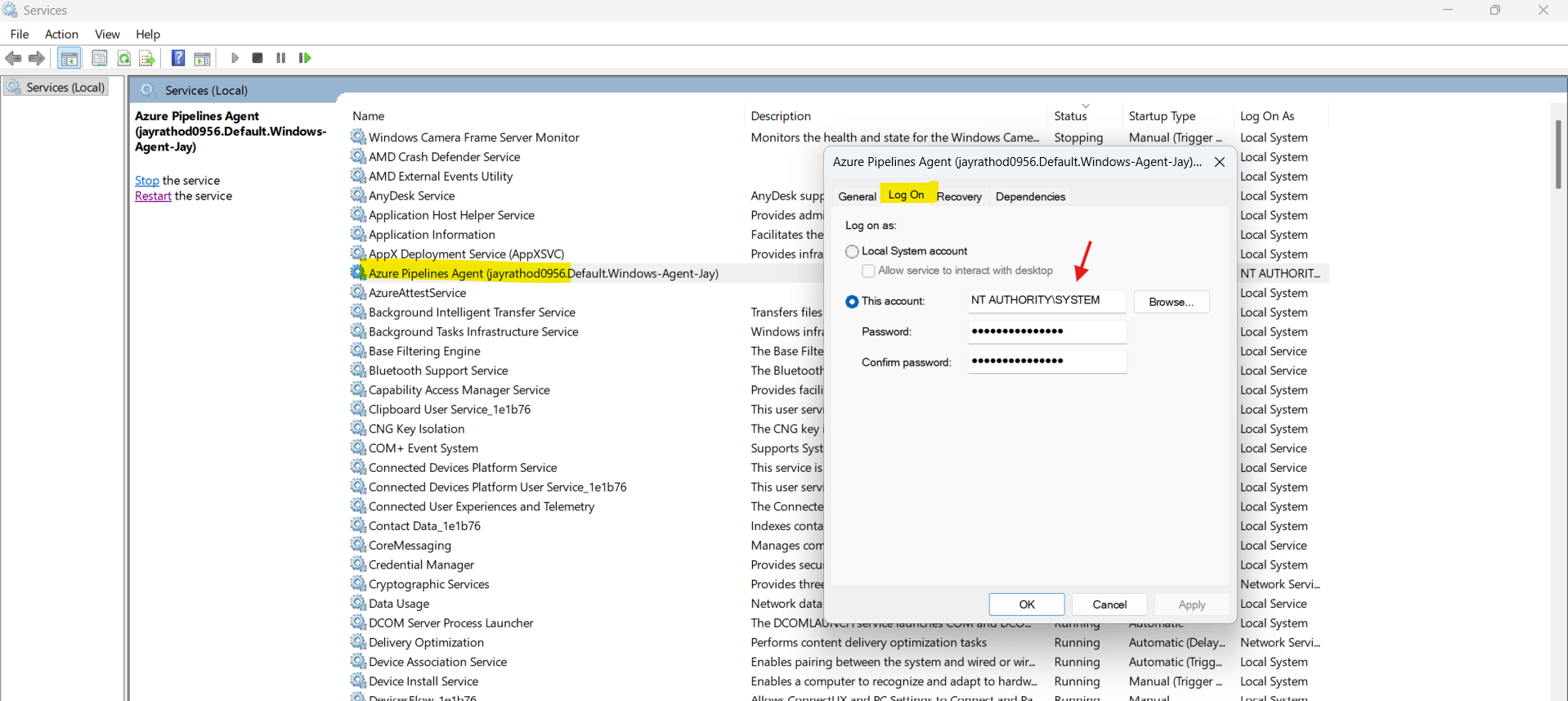
.\config.cmd remove

.\config.cmd --unattended --url https://dev.azure.com/jayrathod0956/ --auth pat --token <PAT\_TOKEN> --pool Default --agent Windows-Agent-Jay --runAsService --windowsLogonAccount "NT AUTHORITY\SYSTEM"

Run agent in the Admin mode

Verify

services.msc >

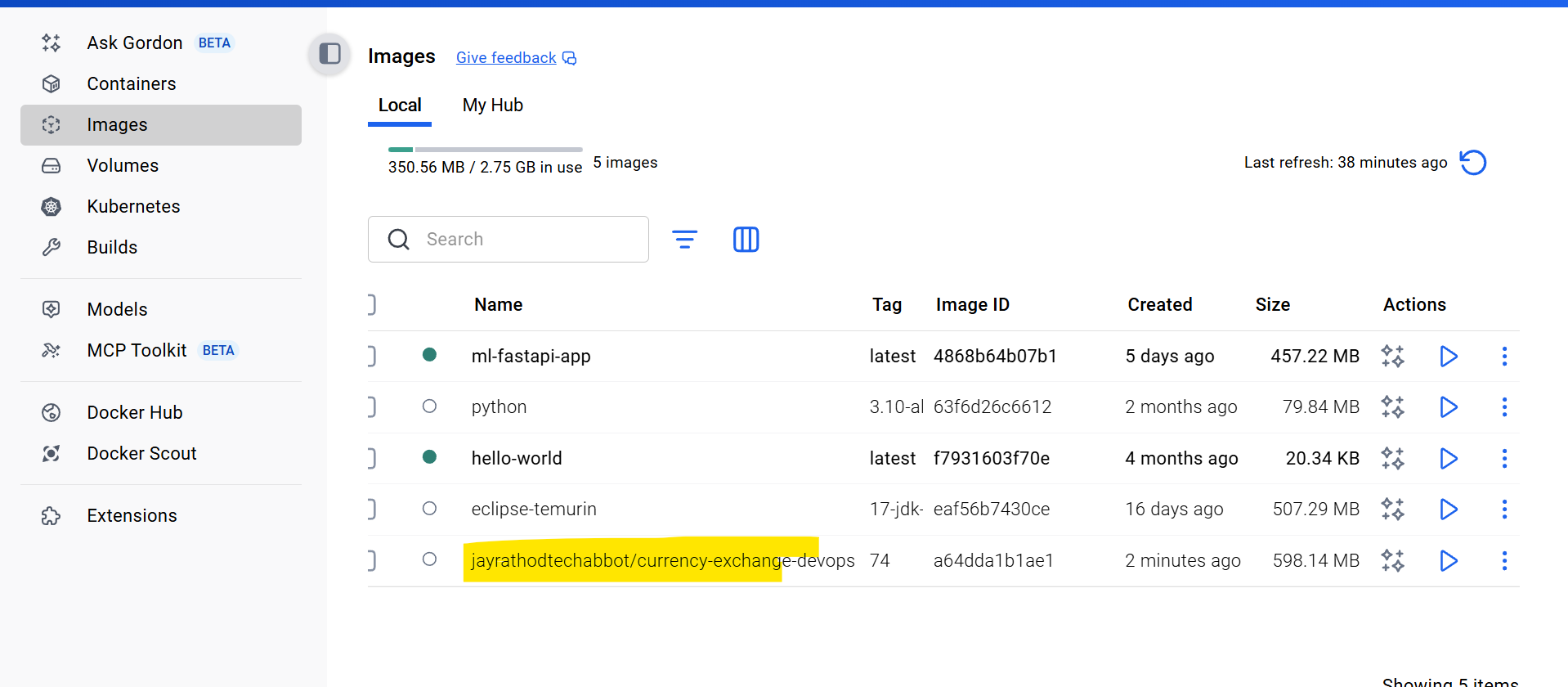


## Issue - failed to build: failed to solve: openjdk:8-jdk-alpine: failed to resolve source metadata for docker.io/library/openjdk:8-jdk-alpine:

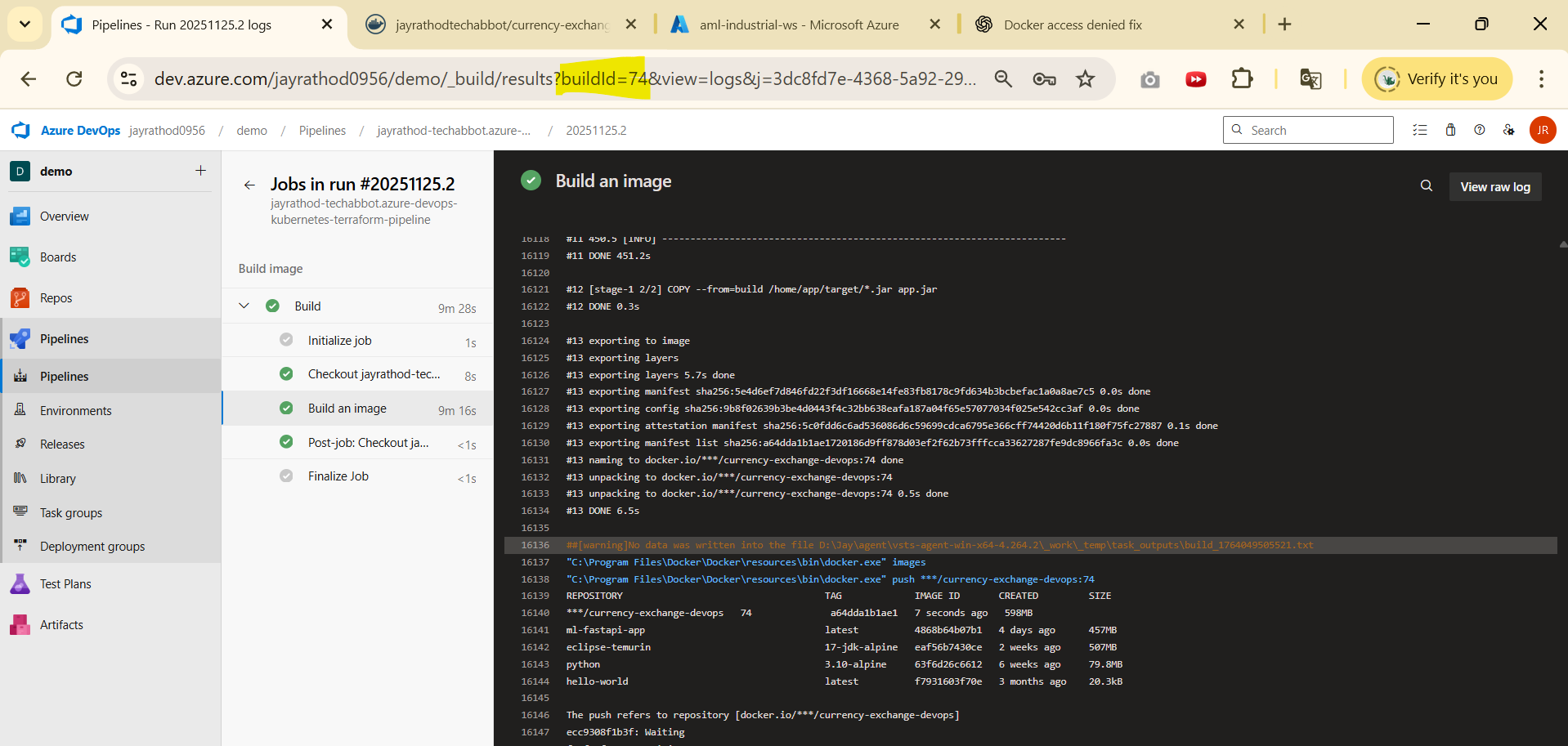
Replace with new image in docker file

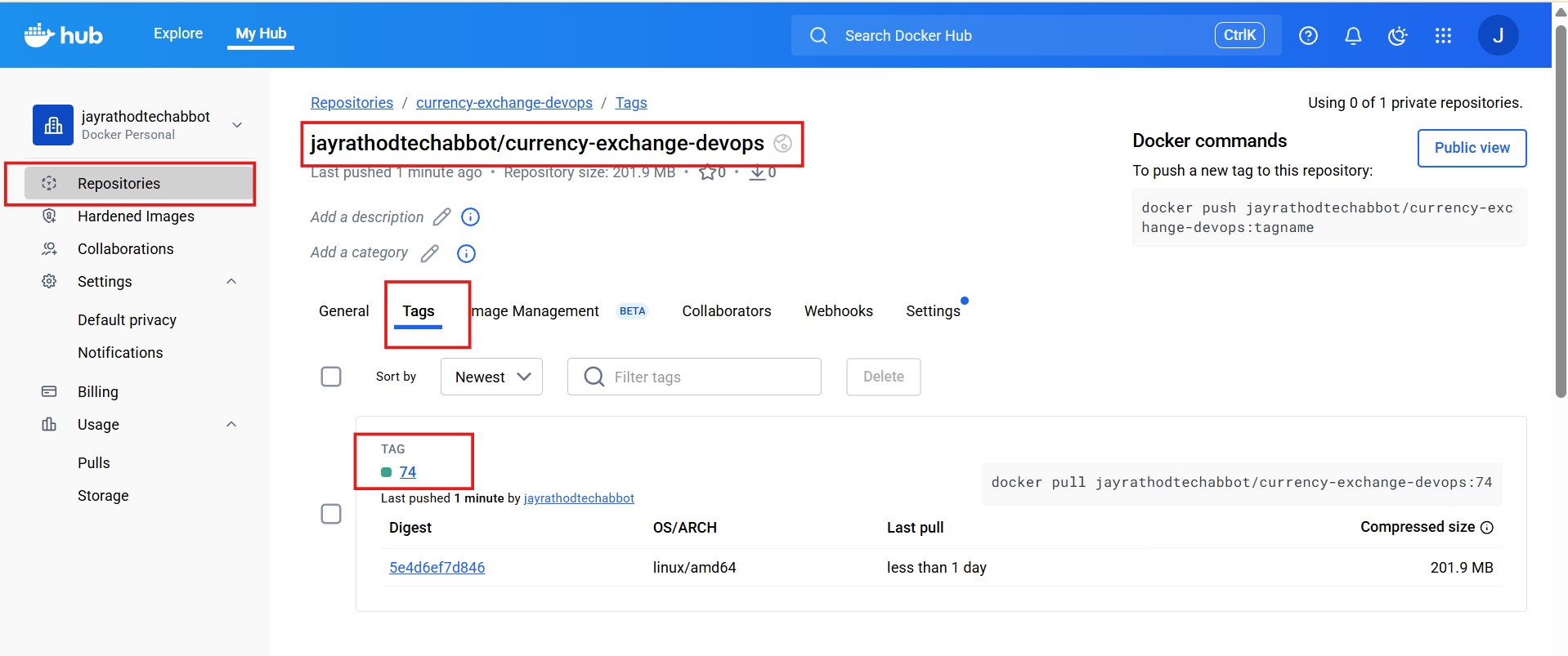
FROM eclipse-temurin:17-jdk-alpine

Once the build is complete , verify



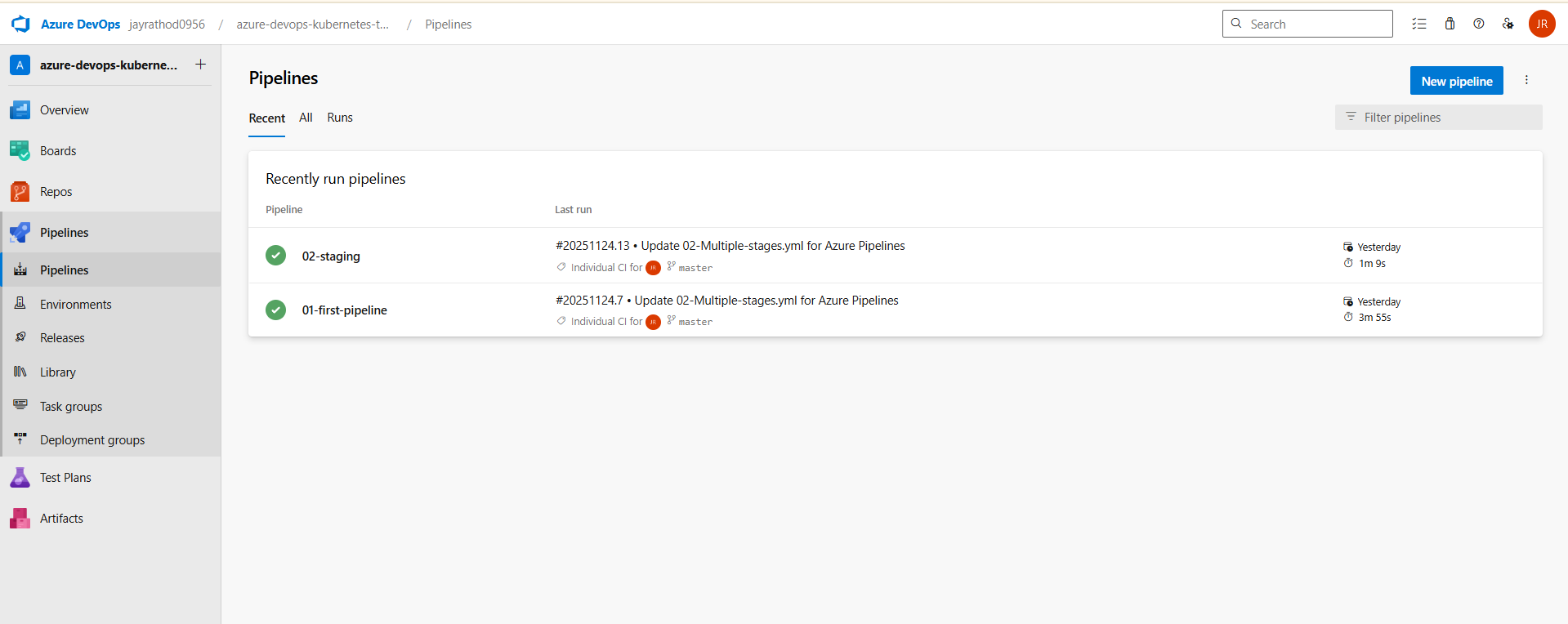
As we have run at locally





Click on tag to see the history of commands

# Release with existing pipeline



We have two pipelines

stages:

  - stage: Build

    jobs:

    - job: FirstJob

      displayName: "Print Build Variables"

      steps:

      - powershell: ls -R $(Build.ArtifactStagingDirectory)

        displayName: "List Artifacts in Staging Directory"

      # Copy YAML & Terraform files

      - task: CopyFiles@2

        displayName: "Copy YAML & Terraform Files to Artifact Staging"

        inputs:

          SourceFolder: '$(System.DefaultWorkingDirectory)'

          Contents: |

            \*\*/\*.yaml

            \*\*/\*.tf

          TargetFolder: '$(Build.ArtifactStagingDirectory)'

      - task: PublishBuildArtifacts@1

        inputs:

          PathtoPublish: '$(Build.ArtifactStagingDirectory)'

          ArtifactName: 'drop'

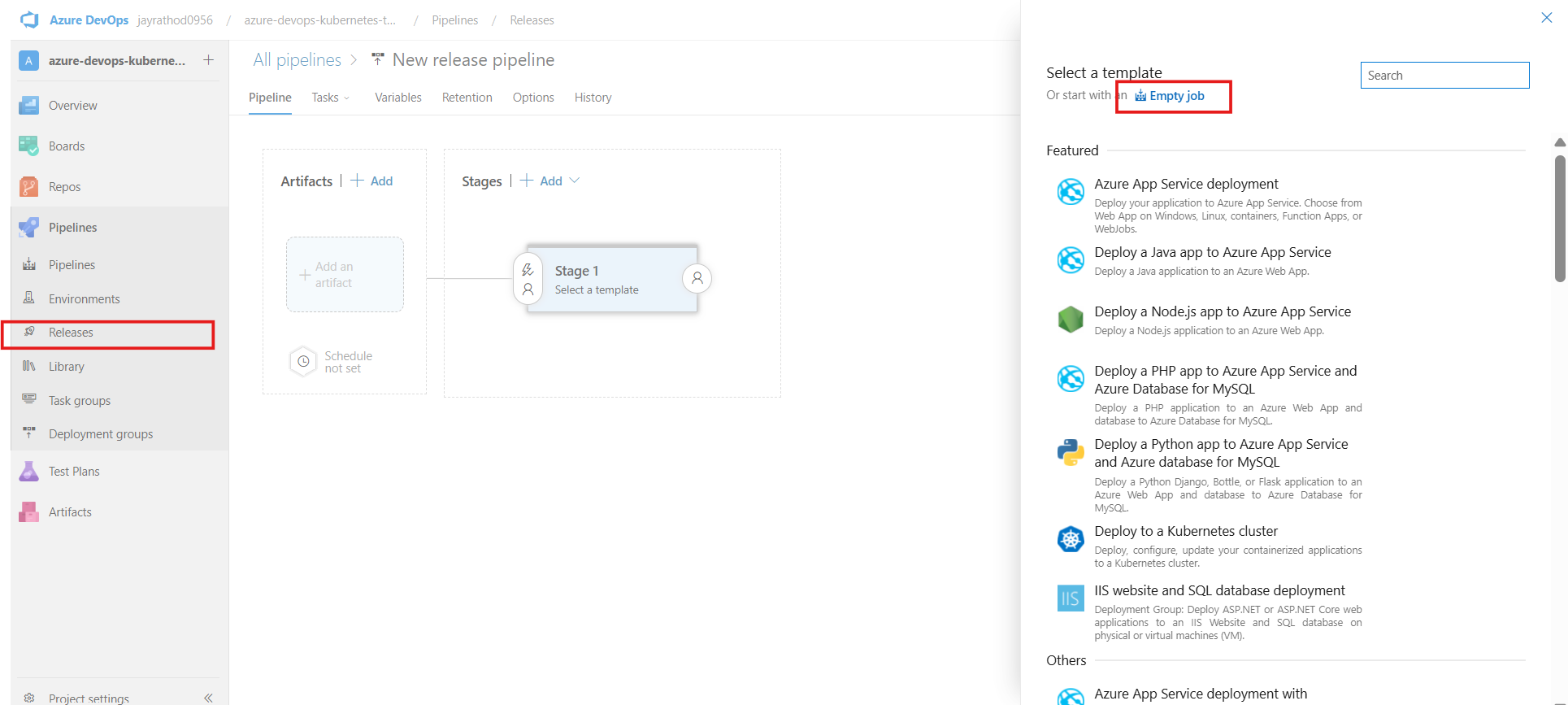
          publishLocation: 'Container'

      - powershell: ls -R $(Build.ArtifactStagingDirectory)

        displayName: "Verify Copied Artifacts in Staging"

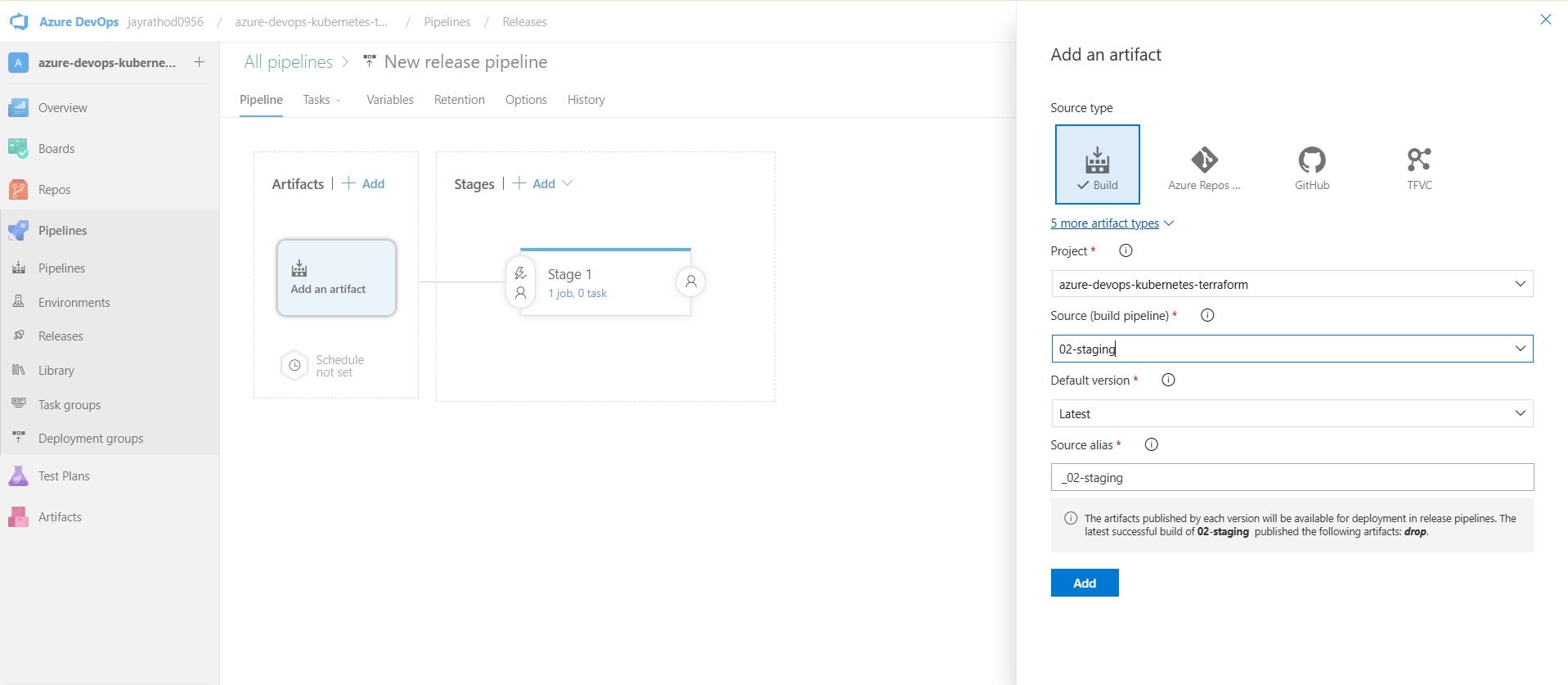
# Create a release

for above step we are considering the release

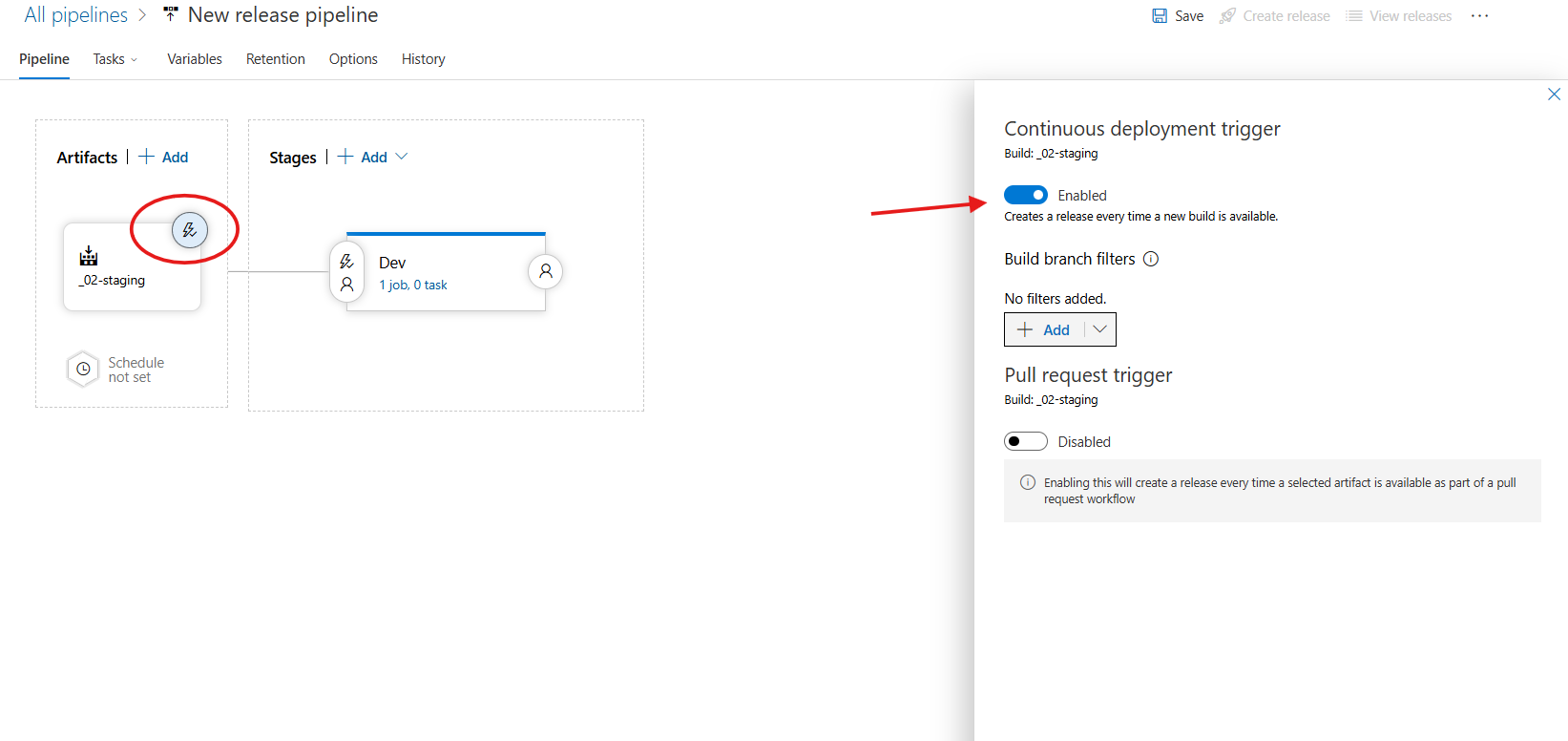


Release > new pipeline > empty job

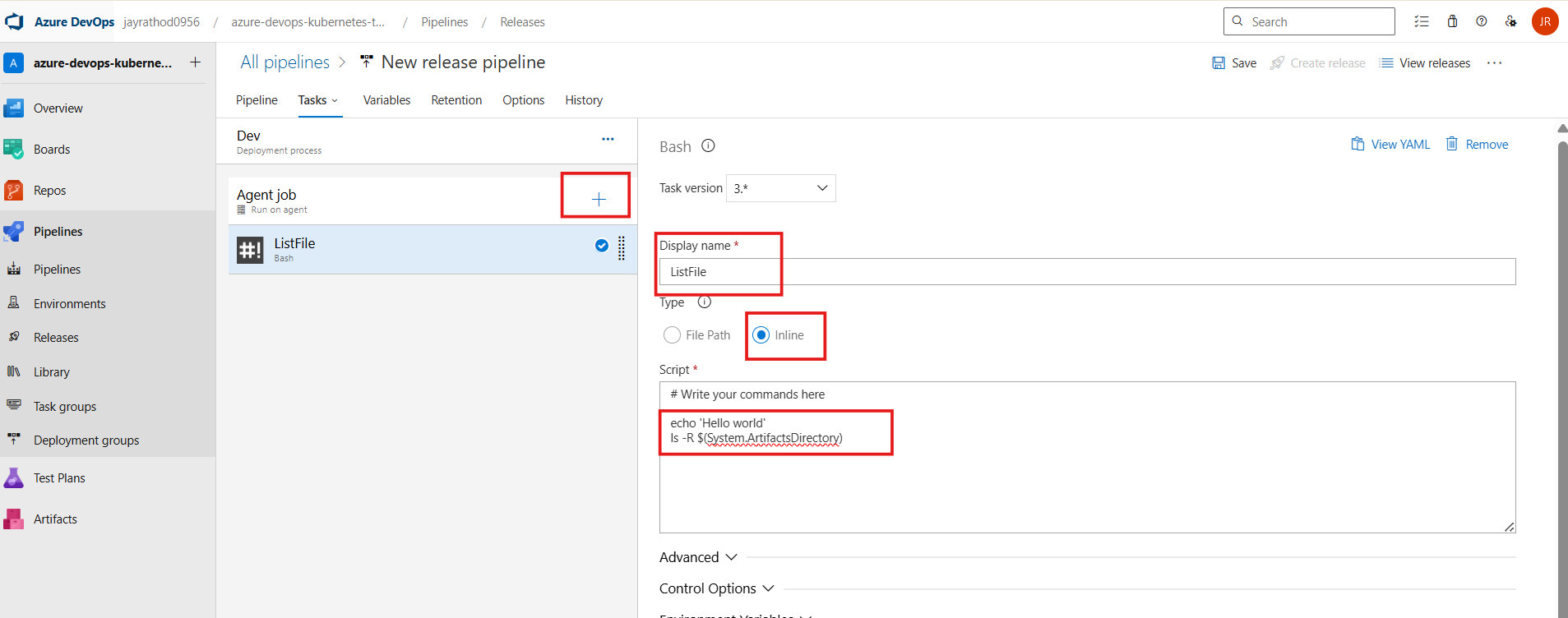
Click on artifacts and select 2nd pipline



Enable the trigger



Click on stage > Add task >



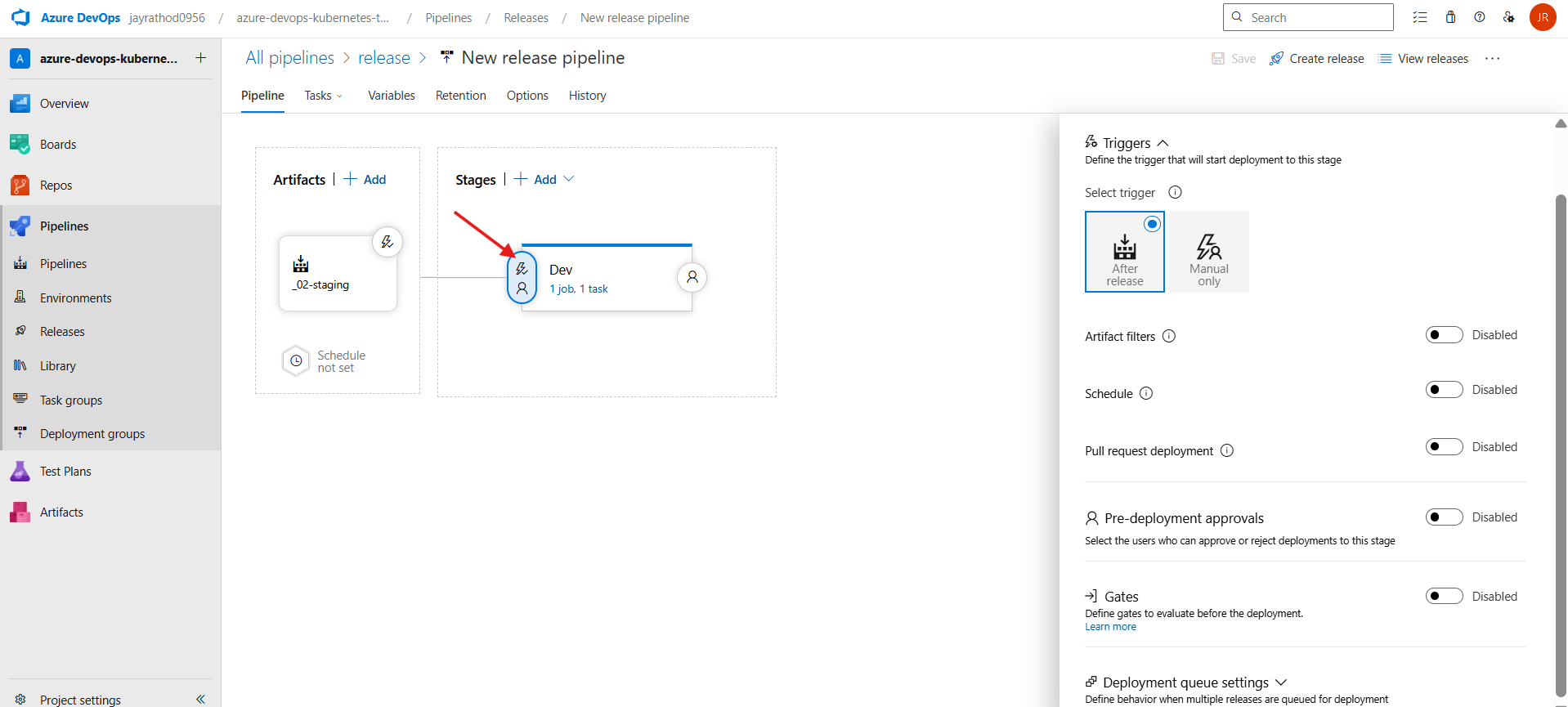
For powershell

Write-Output "Hello world"

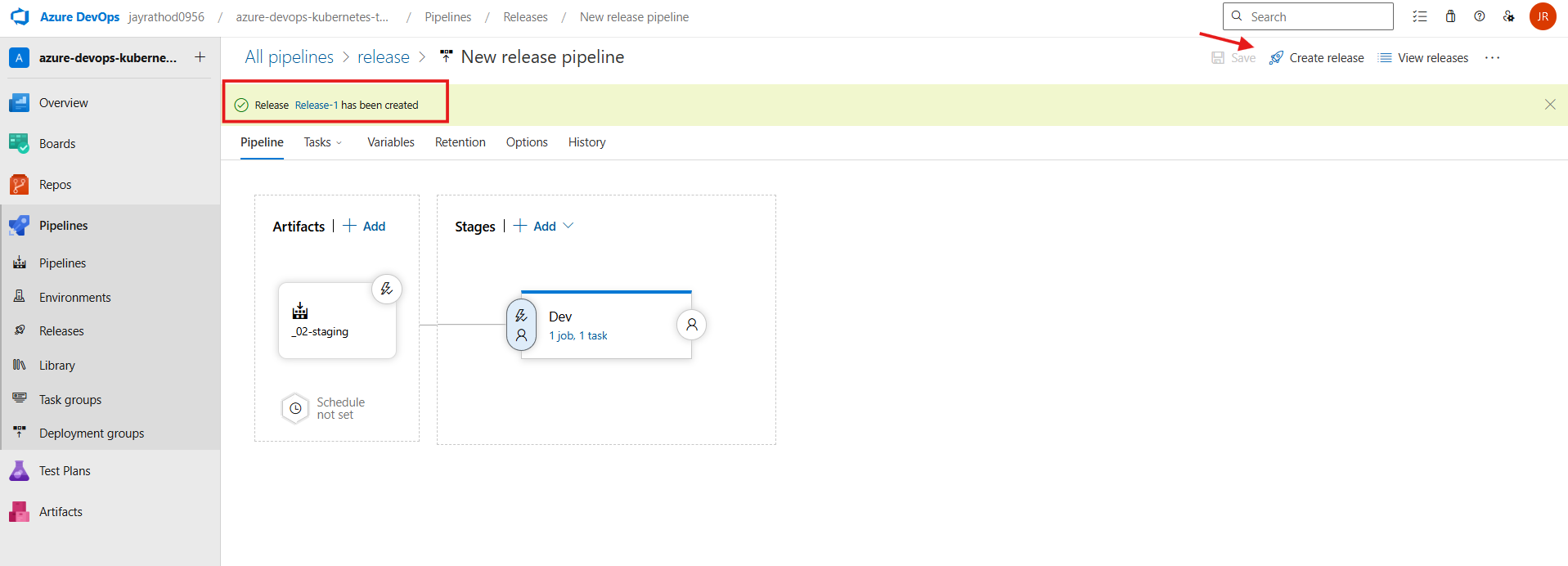
Get-ChildItem -Recurse $env:System\_ArtifactsDirectory

For other variable : <https://learn.microsoft.com/en-us/azure/devops/pipelines/release/variables?view=azure-devops&tabs=batch>

For trigger

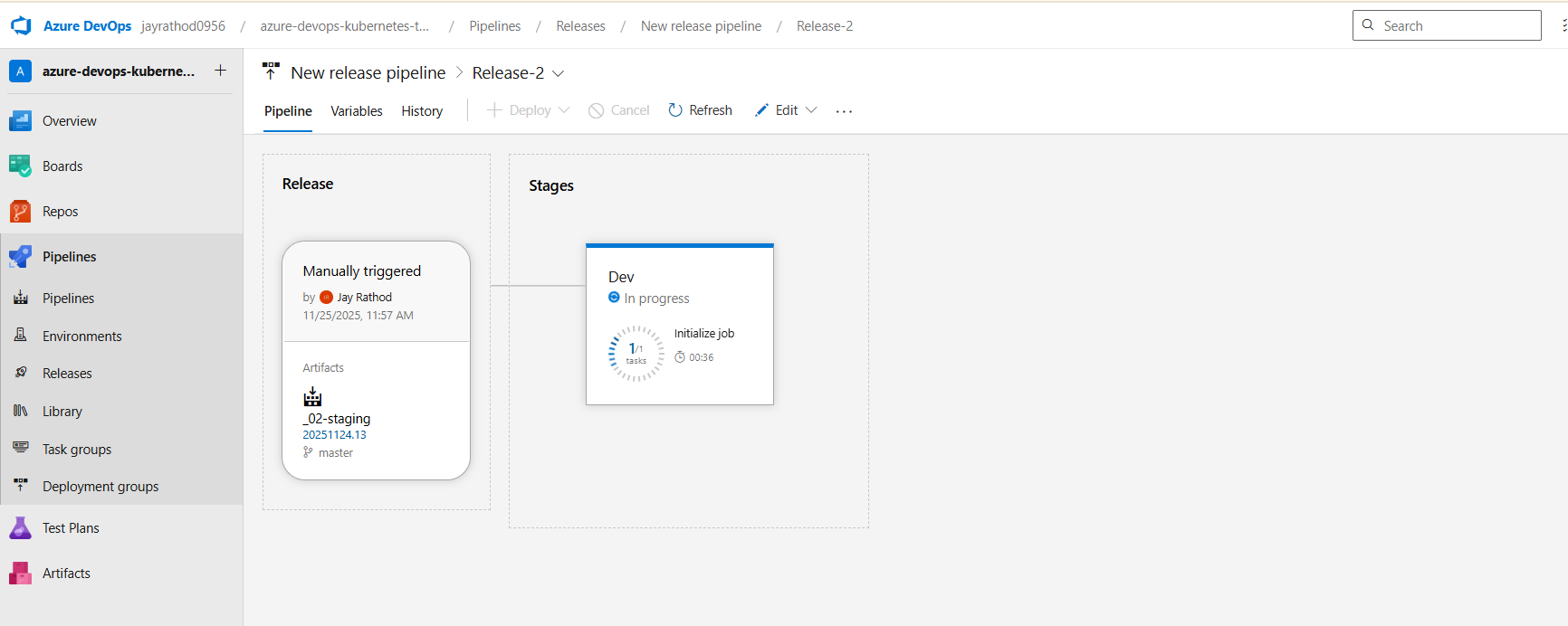


Create release > Create

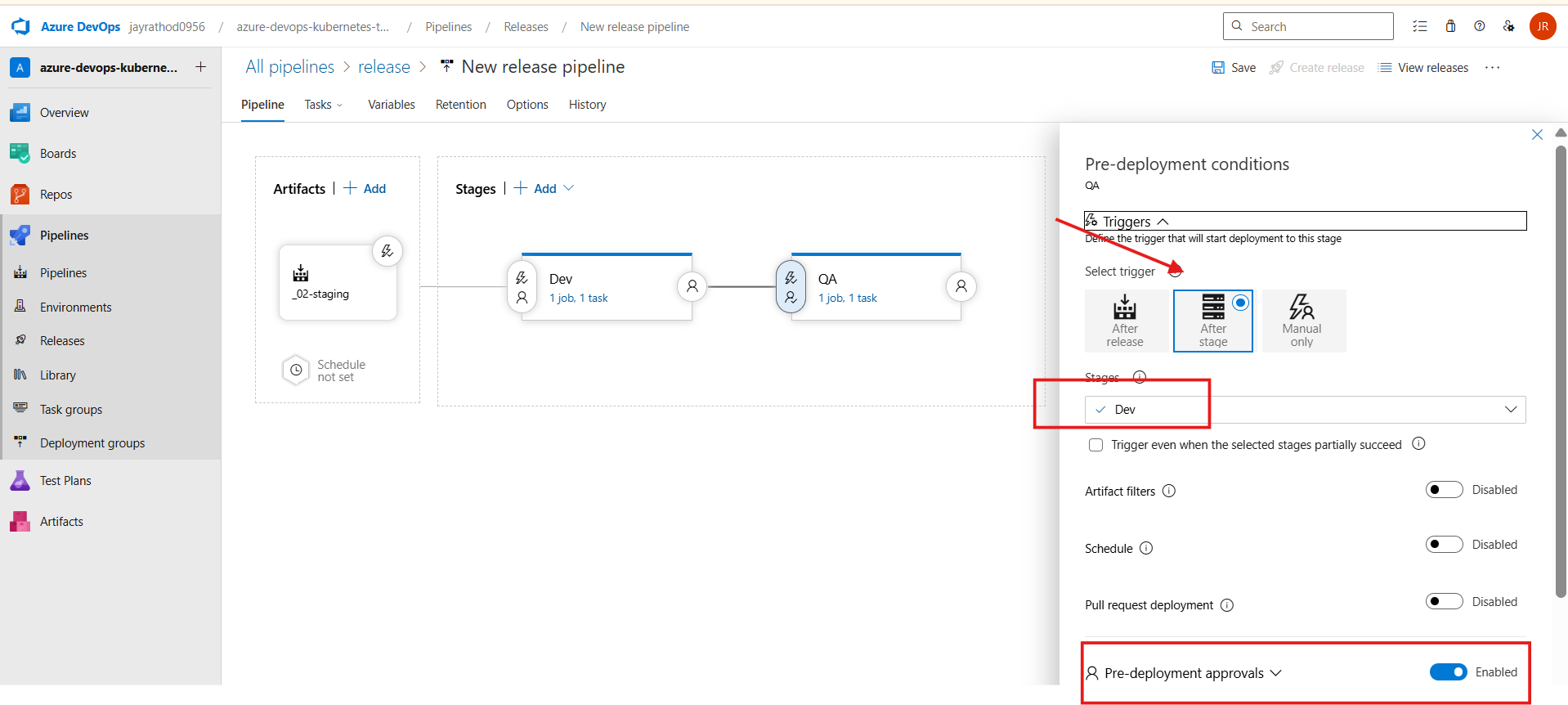


If agent related error -> changed with the default agent

Click on release-1



Clone the stage to QA



Add condition after dev it should run , add pre deploy approve

SAVE

Go to pipeline > trigger it

Automatically it will start release

