You're absolutely right! Now I understand the confusion. Looking at your pipeline structure:

1. \*\*"Create auto Pool"\*\* - Creates just a pool (not a job)

2. \*\*"Create Job & Task"\*\* - Should create a job using that pool and submit tasks

## The Problem:

Your "Create auto Pool" step is creating an \*\*auto-pool\*\* but your "Create Job & Task" step is trying to find a \*\*job\*\* that doesn't exist yet.

## The Solution:

Your "Create Job & Task" step needs to:

1. \*\*Create a job\*\* that uses the pool created in the previous step

2. \*\*Then create a task\*\* in that job

Here's the corrected "Create Job & Task" step:

```powershell

# Create Job & Task step - CORRECTED

Write-Host ">>> Step 5: Create Job & Task"

# Variables

$batchAccount = "lvbatchdev"

$resourceGroup = "lv-batch-dev"

$storageAccount = "lvbatchdev"

$appId = "ccdc-batch-jobs"

# Get the pool ID that was created in "Create auto Pool" step

$poolId = "autopool\_5634C2BC-79EB-431..." # This should match the pool created in previous step

# OR use a pattern to find the pool

$poolId = az batch pool list --query "[?starts\_with(id, 'autopool\_')].id | [0]" -o tsv

Write-Host "Using pool: $poolId"

# Create output container

Write-Host "Creating output container..."

az storage container create `

--account-name $storageAccount `

--name "ccdcbatchjobs-output" `

--auth-mode login 2>$null

# Login to batch account

Write-Host "Logging into batch account..."

az batch account login `

--name $batchAccount `

--resource-group $resourceGroup

# Step 1: CREATE THE JOB (using the existing pool)

$jobId = "$appId-$(Build.BuildId)-$(Release.ReleaseId)"

Write-Host "Creating job: $jobId using pool: $poolId"

az batch job create `

--id $jobId `

--pool-id $poolId

if ($LASTEXITCODE -ne 0) {

Write-Error "Failed to create job"

exit 1

}

Write-Host "Job created successfully!"

# Step 2: CREATE THE TASK in the job

$taskId = "task-$(Build.BuildId)"

Write-Host "Creating task: $taskId in job: $jobId"

$commandLine = @"

/bin/bash -c "echo 'Starting Python application...' && python `$AZ\_BATCH\_APP\_PACKAGE\_ccdc-batch-jobs/main.py > output.txt 2>&1 && echo 'Uploading output...' && az storage blob upload --account-name $storageAccount --container-name ccdcbatchjobs-output --name $(Release.ReleaseId)/task-output.txt --file output.txt --auth-mode login --overwrite"

"@

az batch task create `

--job-id $jobId `

--task-id $taskId `

--command-line "$commandLine" `

--application-package-references '[{"applicationId":"ccdc-batch-jobs","version":"10978"}]'

if ($LASTEXITCODE -eq 0) {

Write-Host "Task created successfully!"

Write-Host "Job ID: $jobId"

Write-Host "Task ID: $taskId"

Write-Host "Pool ID: $poolId"

} else {

Write-Error "Failed to create task"

exit 1

}

```

## Alternative: Change "Create auto Pool" to create a Job with Auto-Pool

If you want to keep the original design (job with auto-pool), change your "Create auto Pool" step to create a \*\*job\*\* with auto-pool configuration instead of just a pool:

```powershell

# Create auto Pool step - Should create JOB with auto-pool

$jobId = "ccdc-batch-jobs-$(Build.BuildId)-$(Release.ReleaseId)-$(Release.AttemptNumber)"

$jobJson = @"

{

"id": "$jobId",

"poolInfo": {

"autoPoolSpecification": {

"autoPoolIdPrefix": "autopool",

"poolLifetimeOption": "job",

"pool": {

"vmSize": "Standard\_D2\_v2",

"virtualMachineConfiguration": {

"imageReference": {

"publisher": "canonical",

"offer": "0001-com-ubuntu-server-focal",

"sku": "20\_04-lts",

"version": "latest"

},

"nodeAgentSkuId": "batch.node.ubuntu 20.04"

},

"targetDedicatedNodes": 1

}

}

}

}

"@

# This creates a JOB with auto-pool

az batch job create --json-file job.json

```

Which approach do you prefer?

1. Keep "Create auto Pool" as-is and fix "Create Job & Task" to create job first

2. Change "Create auto Pool" to create a job with auto-pool configuration​​​​​​​​​​​​​​​​