# Variables

$batchAccount = "lvbatchdev"

$resourceGroup = "lv-batch-dev"

$storageAccount = "lvbatchdev"

$appId = "ccdc-batch-jobs"

$outputContainer = "ccdcbatchjobs-output"

$appName = "ccdcbatchjobs" # Application name from your register step

# Login to batch account

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

az batch account login `

--name $batchAccount `

--resource-group $resourceGroup

# Get the latest version that was just registered

Write-Host "Getting latest application package version..."

$latestVersion = az batch application package list `

--application-id $appName `

--query "[?state=='Active'] | [0].version" -o tsv

if (-not $latestVersion) {

Write-Error "No active application package found for $appName"

exit 1

}

Write-Host "Using application package: $appName version $latestVersion"

# Create the job with auto-pool that includes the application package

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

Write-Host "Creating job: $jobId with auto-pool"

# Create job JSON with application package reference

$jobJson = @"

{

"id": "$jobId",

"poolInfo": {

"autoPoolSpecification": {

"autoPoolIdPrefix": "autopool",

"poolLifetimeOption": "job",

"keepAlive": false,

"pool": {

"vmSize": "Standard\_D2s\_v3",

"targetDedicatedNodes": 1,

"virtualMachineConfiguration": {

"imageReference": {

"publisher": "MicrosoftWindowsServer",

"offer": "WindowsServer",

"sku": "2019-datacenter",

"version": "latest"

},

"nodeAgentSKUId": "batch.node.windows amd64"

},

"applicationPackageReferences": [

{

"applicationId": "$appName",

"version": "$latestVersion"

}

]

}

}

},

"onAllTasksComplete": "terminatejob"

}

"@

# Save and create job

$jobJson | Out-File -FilePath "job.json" -Encoding UTF8

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

if ($LASTEXITCODE -ne 0) {

Write-Error "Failed to create job"

Remove-Item "job.json" -Force

exit 1

}

Write-Host "Job created successfully!"

Remove-Item "job.json" -Force

# Wait for pool to be ready

Write-Host "Waiting for auto-pool to be ready..."

Start-Sleep -Seconds 30

# Create task that uses the application package

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

Write-Host "Creating task: $taskId"

# The application will be available at %AZ\_BATCH\_APP\_PACKAGE\_ccdcbatchjobs\_<version>%

$taskJson = @"

{

"id": "$taskId",

"commandLine": "cmd /c echo Application package deployed at %AZ\_BATCH\_APP\_PACKAGE\_${appName}\_${latestVersion}% && dir %AZ\_BATCH\_APP\_PACKAGE\_${appName}\_${latestVersion}% && echo Task completed successfully",

"applicationPackageReferences": [

{

"applicationId": "$appName",

"version": "$latestVersion"

}

],

"constraints": {

"maxWallClockTime": "PT30M",

"maxTaskRetryCount": 0

}

}

"@

# If you have a specific exe or script in your package, you can run it like:

# "commandLine": "cmd /c %AZ\_BATCH\_APP\_PACKAGE\_${appName}\_${latestVersion}%\\YourApp.exe"

$taskJson | Out-File -FilePath "task.json" -Encoding UTF8

az batch task create --job-id $jobId --json-file "task.json"

if ($LASTEXITCODE -ne 0) {

Write-Error "Failed to create task"

Remove-Item "task.json" -Force

exit 1

}

Write-Host "Task created successfully!"

Remove-Item "task.json" -Force

# Monitor task

Write-Host "Monitoring task execution..."

$timeout = 300

$elapsed = 0

while ($elapsed -lt $timeout) {

Start-Sleep -Seconds 10

$elapsed += 10

$taskState = az batch task show `

--job-id $jobId `

--task-id $taskId `

--query "state" -o tsv

Write-Host "[$elapsed/$timeout sec] Task state: $taskState"

if ($taskState -eq "completed") {

$exitCode = az batch task show `

--job-id $jobId `

--task-id $taskId `

--query "executionInfo.exitCode" -o tsv

Write-Host "Task completed with exit code: $exitCode"

# Show task output

Write-Host "`nTask stdout:"

az batch task file download `

--job-id $jobId `

--task-id $taskId `

--file-path "stdout.txt" `

--destination "-" 2>$null

Write-Host "`nTask stderr:"

az batch task file download `

--job-id $jobId `

--task-id $taskId `

--file-path "stderr.txt" `

--destination "-" 2>$null

break

}

}

Write-Host "`n=================="

Write-Host "Summary:"

Write-Host "Job ID: $jobId"

Write-Host "Task ID: $taskId"

Write-Host "Application: $appName v$latestVersion"

Write-Host "Final State: $taskState"

Write-Host "=================="