# Variables

$batchAccount = "lvbatchdev"

$resourceGroup = "lv-batch-dev"

$storageAccount = "lvbatchdev"

$appId = "ccdc-batch-jobs"

$outputContainer = "ccdcbatchjobs-output"

# Login to batch account

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

az batch account login `

--name $batchAccount `

--resource-group $resourceGroup

# Create the job with auto-pool

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

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

# Create job JSON with WINDOWS configuration (since your first screenshots showed Windows)

$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"

}

}

}

},

"onAllTasksComplete": "terminatejob"

}

"@

# Save JSON to file

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

# Create job

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

if ($LASTEXITCODE -ne 0) {

Write-Error "Failed to create job"

Remove-Item "job.json" -Force -ErrorAction SilentlyContinue

exit 1

}

Write-Host "Job created successfully!"

Remove-Item "job.json" -Force -ErrorAction SilentlyContinue

# Wait for pool to be ready

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

Start-Sleep -Seconds 30

# Create a VERY SIMPLE task that will complete immediately

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

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

# Super simple task that just echoes and exits

$taskJson = @"

{

"id": "$taskId",

"commandLine": "cmd /c echo Hello Azure Batch > output.txt && echo Task Completed && exit 0",

"constraints": {

"maxWallClockTime": "PT5M",

"maxTaskRetryCount": 0

}

}

"@

# Save task JSON to file

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

# Create task

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 -ErrorAction SilentlyContinue

exit 1

}

Write-Host "Task created successfully!"

Remove-Item "task.json" -Force -ErrorAction SilentlyContinue

# Monitor the task for completion

Write-Host "Monitoring task execution..."

$maxWaitTime = 180 # 3 minutes

$startTime = Get-Date

$checkInterval = 10

while ((Get-Date) -lt $startTime.AddSeconds($maxWaitTime)) {

Start-Sleep -Seconds $checkInterval

# Get task state

$taskState = az batch task show `

--job-id $jobId `

--task-id $taskId `

--query "state" -o tsv

Write-Host "Task state: $taskState"

if ($taskState -eq "completed") {

# Get exit code

$exitCode = az batch task show `

--job-id $jobId `

--task-id $taskId `

--query "executionInfo.exitCode" -o tsv

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

# List output files

Write-Host "Task output files:"

az batch task file list `

--job-id $jobId `

--task-id $taskId `

--output table

# Try to get stdout

Write-Host "`nTask stdout:"

az batch task file download `

--job-id $jobId `

--task-id $taskId `

--file-path "stdout.txt" `

--destination "-" 2>$null

# Try to get stderr

Write-Host "`nTask stderr:"

az batch task file download `

--job-id $jobId `

--task-id $taskId `

--file-path "stderr.txt" `

--destination "-" 2>$null

break

}

elseif ($taskState -eq "running") {

Write-Host "Task is running..."

}

else {

Write-Host "Task is in state: $taskState"

}

}

if ($taskState -ne "completed") {

Write-Warning "Task did not complete within the timeout period"

# Get more details about why it's stuck

$taskDetails = az batch task show `

--job-id $jobId `

--task-id $taskId `

--query "executionInfo" -o json | ConvertFrom-Json

Write-Host "Task execution details:"

Write-Host ($taskDetails | ConvertTo-Json -Depth 10)

}

Write-Host ""

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

Write-Host "Summary:"

Write-Host "Job ID: $jobId"

Write-Host "Task ID: $taskId"

Write-Host "Final State: $taskState"

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