# 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 Linux VM configuration (matching your existing pools)

$jobJson = @"

{

"id": "$jobId",

"poolInfo": {

"autoPoolSpecification": {

"autoPoolIdPrefix": "autopool",

"poolLifetimeOption": "job",

"keepAlive": false,

"pool": {

"vmSize": "Standard\_DS1\_v2",

"targetDedicatedNodes": 1,

"virtualMachineConfiguration": {

"imageReference": {

"publisher": "canonical",

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

"sku": "22\_04-lts",

"version": "latest"

},

"nodeAgentSKUId": "batch.node.ubuntu 22.04"

},

"startTask": {

"commandLine": "/bin/bash -c 'echo Pool started && apt-get update'",

"userIdentity": {

"autoUser": {

"scope": "pool",

"elevationLevel": "admin"

}

},

"waitForSuccess": true

}

}

}

}

}

"@

# 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 20

# Create the task

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

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

# Create simple task for Linux

$taskJson = @"

{

"id": "$taskId",

"commandLine": "/bin/bash -c 'echo Hello from Azure Batch > output.txt && ls -la > directory.txt && echo Task completed'",

"userIdentity": {

"autoUser": {

"scope": "pool",

"elevationLevel": "nonadmin"

}

},

"constraints": {

"maxWallClockTime": "PT1H",

"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 -eq 0) {

Write-Host "Task created successfully!"

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

Write-Host "Job ID: $jobId"

Write-Host "Task ID: $taskId"

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

# Check task status after a few seconds

Start-Sleep -Seconds 10

$taskInfo = az batch task show `

--job-id $jobId `

--task-id $taskId `

--query "{state:state,nodeId:nodeInfo.nodeId,exitCode:executionInfo.exitCode}" -o json | ConvertFrom-Json

Write-Host "Task State: $($taskInfo.state)"

Write-Host "Running on Node: $($taskInfo.nodeId)"

# Show how to monitor the task

Write-Host ""

Write-Host "To monitor task status run:"

Write-Host "az batch task show --job-id $jobId --task-id $taskId"

Write-Host ""

Write-Host "To view task output files:"

Write-Host "az batch task file list --job-id $jobId --task-id $taskId"

Write-Host ""

Write-Host "To download stdout:"

Write-Host "az batch task file download --job-id $jobId --task-id $taskId --file-path stdout.txt --destination ./stdout.txt"

} else {

Write-Error "Failed to create task"

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

exit 1

}

# Clean up

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