# Hardcoded variables based on your screenshots

$batchAccount = "lvbatchdev"

$resourceGroup = "lv-batch-dev"

$storageAccount = "lvbatchdev"

$appId = "ccdc-batch-jobs"

$appName = "ccdc-batch-jobs" # Your application name

# Login to batch account

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

az batch account login `

--name $batchAccount `

--resource-group $resourceGroup

if ($LASTEXITCODE -ne 0) {

Write-Error "Failed to login to batch account"

exit 1

}

# Get the active application package version

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

$activeVersion = "10978" # I can see this is active in your screenshot

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

# Create the job with auto-pool

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

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

# Create job JSON with Ubuntu configuration

$jobJson = @"

{

"id": "$jobId",

"poolInfo": {

"autoPoolSpecification": {

"autoPoolIdPrefix": "autopool",

"poolLifetimeOption": "job",

"keepAlive": false,

"pool": {

"vmSize": "Standard\_A1\_v2",

"targetDedicatedNodes": 1,

"virtualMachineConfiguration": {

"imageReference": {

"publisher": "canonical",

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

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

"version": "latest"

},

"nodeAgentSKUId": "batch.node.ubuntu 20.04"

},

"applicationPackageReferences": [

{

"applicationId": "$appName",

"version": "$activeVersion"

}

],

"startTask": {

"commandLine": "/bin/bash -c 'echo Pool starting && apt-get update && apt-get install -y unzip && echo Pool ready'",

"userIdentity": {

"autoUser": {

"scope": "pool",

"elevationLevel": "admin"

}

},

"waitForSuccess": true

}

}

}

},

"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 45

# Create task

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

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

# Create a task that explores the app package and runs a simple command

$taskJson = @"

{

"id": "$taskId",

"commandLine": "/bin/bash -c 'echo Task started && echo Application package location: \$AZ\_BATCH\_APP\_PACKAGE\_ccdc-batch-jobs\_10978 && ls -la \$AZ\_BATCH\_APP\_PACKAGE\_ccdc-batch-jobs\_10978/ && echo Task completed'",

"applicationPackageReferences": [

{

"applicationId": "$appName",

"version": "$activeVersion"

}

],

"userIdentity": {

"autoUser": {

"scope": "pool",

"elevationLevel": "nonadmin"

}

},

"constraints": {

"maxWallClockTime": "PT15M",

"maxTaskRetryCount": 1

}

}

"@

# 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 task execution with detailed status

Write-Host "`nMonitoring task execution..."

$timeout = 300 # 5 minutes

$elapsed = 0

$checkInterval = 10

while ($elapsed -lt $timeout) {

Start-Sleep -Seconds $checkInterval

$elapsed += $checkInterval

# Get detailed task information

$taskDetails = az batch task show `

--job-id $jobId `

--task-id $taskId `

-o json | ConvertFrom-Json

$taskState = $taskDetails.state

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

if ($taskState -eq "completed") {

$exitCode = $taskDetails.executionInfo.exitCode

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

# Download and display stdout

Write-Host "`n=== TASK OUTPUT (stdout) ==="

az batch task file download `

--job-id $jobId `

--task-id $taskId `

--file-path "stdout.txt" `

--destination "-" 2>$null

# Download and display stderr

Write-Host "`n=== TASK ERRORS (stderr) ==="

az batch task file download `

--job-id $jobId `

--task-id $taskId `

--file-path "stderr.txt" `

--destination "-" 2>$null

if ($exitCode -eq 0) {

Write-Host "`nSUCCESS: Task completed successfully!"

} else {

Write-Error "Task failed with exit code: $exitCode"

# Show failure info if available

if ($taskDetails.executionInfo.failureInfo) {

Write-Host "`nFailure Information:"

Write-Host "Category: $($taskDetails.executionInfo.failureInfo.category)"

Write-Host "Code: $($taskDetails.executionInfo.failureInfo.code)"

Write-Host "Message: $($taskDetails.executionInfo.failureInfo.message)"

}

exit 1

}

break

}

elseif ($taskState -eq "active") {

if ($taskDetails.nodeInfo.nodeId) {

Write-Host "Task is preparing on node: $($taskDetails.nodeInfo.nodeId)"

}

}

elseif ($taskState -eq "running") {

Write-Host "Task is running on node: $($taskDetails.nodeInfo.nodeId)"

}

elseif ($taskState -eq "preparing") {

Write-Host "Task is in preparing state..."

}

}

if ($taskState -ne "completed") {

Write-Warning "Task did not complete within $timeout seconds"

Write-Host "Final task state: $taskState"

# Try to get any available information

Write-Host "`nGetting pool information..."

$poolId = az batch job show --job-id $jobId --query "poolInfo.poolId" -o tsv

if ($poolId) {

Write-Host "Pool ID: $poolId"

$poolState = az batch pool show --pool-id $poolId --query "state" -o tsv

Write-Host "Pool state: $poolState"

}

exit 1

}

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

Write-Host "EXECUTION COMPLETE"

Write-Host "Job ID: $jobId"

Write-Host "Task ID: $taskId"

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

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