# ONE-TIME POOL CREATION SCRIPT

# Using existing managed identity: lv-batch-dev-UMI

$batchAccount = "lvbatchdev"

$resourceGroup = "lv-batch-dev"

$poolId = "batchpool-dev"

$appName = "ccdc-batch-jobs"

$identityName = "lv-batch-dev-UMI" # Your existing identity

# Login to Azure

az login

az account set --subscription "8d9342e2-96c8-436c-854a-1ad509c7356c"

# Login to batch

az batch account login --name $batchAccount --resource-group $resourceGroup

# Get identity resource ID (using your existing identity)

$identityResourceId = "/subscriptions/8d9342e2-96c8-436c-854a-1ad509c7356c/resourceGroups/lv-batch-dev/providers/Microsoft.ManagedIdentity/userAssignedIdentities/$identityName"

Write-Host "Using existing managed identity: $identityName"

# Check if pool exists and delete if needed

$existingPool = az batch pool show --pool-id $poolId 2>$null

if ($existingPool) {

Write-Host "Deleting existing pool to recreate with managed identity..."

az batch pool delete --pool-id $poolId --yes

Start-Sleep -Seconds 30

}

# Create new pool with the existing managed identity

Write-Host "Creating pool with managed identity..."

# Get latest application version

$latestVersion = az batch application package list `

--application-id $appName `

--query "[?state=='Active'] | sort\_by(@, &version) | [-1].version" -o tsv

if (-not $latestVersion) {

$latestVersion = "1.$(Get-Date -Format 'yyyyMMdd')" # Default version

Write-Host "No active version found, using: $latestVersion"

} else {

Write-Host "Using application version: $latestVersion"

}

$poolJson = @"

{

"id": "$poolId",

"vmSize": "standard\_d2s\_v3",

"targetDedicatedNodes": 1,

"virtualMachineConfiguration": {

"imageReference": {

"publisher": "MicrosoftWindowsServer",

"offer": "WindowsServer",

"sku": "2022-datacenter",

"version": "latest"

},

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

},

"applicationPackageReferences": [

{

"applicationId": "$appName",

"version": "$latestVersion"

}

],

"identity": {

"type": "UserAssigned",

"userAssignedIdentities": {

"$identityResourceId": {}

}

},

"startTask": {

"commandLine": "cmd /c echo Pool bootstrap started && echo App package location: %AZ\_BATCH\_APP\_PACKAGE\_ccdc-batch-jobs% && dir %AZ\_BATCH\_APP\_PACKAGE\_ccdc-batch-jobs%\* && echo Bootstrap complete",

"userIdentity": {

"autoUser": {

"scope": "pool",

"elevationLevel": "admin"

}

},

"waitForSuccess": true,

"maxTaskRetryCount": 3

}

}

"@

$poolJson | Out-File -FilePath "pool.json" -Encoding UTF8

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

if ($LASTEXITCODE -ne 0) {

Write-Error "Failed to create pool"

Remove-Item "pool.json" -Force

exit 1

}

Remove-Item "pool.json" -Force

Write-Host "Pool created successfully with managed identity!"

# Wait for pool to be ready

Write-Host "Waiting for pool to become steady..."

$timeout = 300

$elapsed = 0

while ($elapsed -lt $timeout) {

$poolInfo = az batch pool show --pool-id $poolId --query "{state:allocationState,nodes:currentDedicatedNodes}" -o json | ConvertFrom-Json

Write-Host "Pool state: $($poolInfo.state), Nodes: $($poolInfo.nodes)"

if ($poolInfo.state -eq "steady" -and $poolInfo.nodes -gt 0) {

Write-Host "Pool is ready!"

# Check node state

$nodes = az batch node list --pool-id $poolId -o json | ConvertFrom-Json

if ($nodes.Count -gt 0) {

Write-Host "Node state: $($nodes[0].state)"

}

break

}

Start-Sleep -Seconds 15

$elapsed += 15

}

Write-Host "`nPool setup complete!"

Write-Host "Pool ID: $poolId"

Write-Host "Managed Identity: $identityName"

Write-Host "Status: Ready for jobs"

Last step :

# Complete Azure Batch Pipeline Script

# This replaces your current "Create Job & Task" step

# Uses fixed pool with managed identity as per manager's requirements

# Variables from pipeline

$batchAccount = "$(BatchAccount)"

$resourceGroup = "$(ResourceGroup)"

$storageAccount = "$(StorageAccount)"

$poolId = "$(PoolId)"

$appName = "$(ApplicationId)"

$appVersion = "$(AppVersionFormat)"

$vmSize = "$(VmSize)"

Write-Host "Azure Batch Job Execution"

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

Write-Host "Batch Account: $batchAccount"

Write-Host "Pool: $poolId"

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

# Login

az batch account login --name $batchAccount --resource-group $resourceGroup

# Quick pool check

$pool = az batch pool show --pool-id $poolId 2>$null

if (-not $pool) {

Write-Error "Pool $poolId not found! Run pool creation script first."

exit 1

}

# Create job

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

Write-Host "`nCreating job: $jobId"

az batch job create --id $jobId --pool-id $poolId

if ($LASTEXITCODE -ne 0) {

Write-Error "Failed to create job"

exit 1

}

# Create task with your exact configuration

$taskJson = @"

{

"id": "task-$(Build.BuildId)",

"commandLine": "$(CmdLineWin)",

"applicationPackageReferences": [

{

"applicationId": "$appName",

"version": "$appVersion"

}

],

"constraints": {

"maxWallClockTime": "PT15M",

"maxTaskRetryCount": 1

}

}

"@

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

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

Remove-Item "task.json" -Force

Write-Host "Job and Task created successfully!"

Write-Host "Job ID: $jobId"