Non-interactive login to AKS cluster

Last updated by | Sughasini Karmugilan | Nov 6, 2023 at 11:25 AM GMT+1

Contents

Introduction

This document explains how to connect with your aks cluster via pipeline in a non-interactive way. In FSCP 3.0 by default you would have service connection in Azure DevOps towards Azure on Resource Group level. You can deploy your application on AKS cluster by either using azure cli tasks or by creating Kubernetes service connection.

Use Azure CLI:

When you use azure cli to deploy your application then you need `kubectl` and `kubelogin` to be installed prior to running kubectl commands. In the existing Azure Devops task based approach, kubectl and kubelogin are installed from internet on the agents using bash script which leads to rate limit issues.

As a workaround to the rate limit issue, we tried to use native ADO Kubectl tool installer and Kubelogin tool installer tasks and pin the specific version instead of using latest. The installation is not seamless and native kubelogin ADO tasks installations failed intermittently. So we need a permanent solution to install kubectl and kubelogin. Nexus creates proxy to these URLs and cache it for us in Nexus repo and we could download from Nexus. This eliminates the number of downloads from internet and also avoid intermittent failure while downloading kubelogin.

Tasks on Linux Agents to install kubectl and kubelogin

```
variables:
  - group: Abnamro.Coesd.VariableGroup.GlobalVars
  - name: Kubelogin.Version
    value: #Specify version ex: 0.0.31 or latest
  - name: Kubectl.Version
    value: #Specify version ex: 1.25.5 or latest

Steps:
    - task: Bash@3
      displayName: Install tools
      inputs:
        targetType: 'inline'
        script: |
          TARGETOS=$(echo "$(Agent.OS)" | tr '[:upper:]' '[:lower:]')
          TARGETARCH=$(case "$(Agent.OSArchitecture)" in "X86") echo "x86";; "X64") echo "amd64";; "ARM") e
          mkdir -p "$(Pipeline.Workspace)/.local/bin"
          if [ $(Kubectl.Version) == 'latest' ];then
            latestVersion=$(curl -L -s $(Solo.Nexus3.Repositories.Uri)repository/generic-group/stable.txt)
            echo "kubectl download latest"
            curl -LO $(Solo.Nexus3.Repositories.Uri)repository/generic-group/$latestVersion/bin/$TARGETOS/$
            mv ./kubectl $(Pipeline.Workspace)/.local/bin/
          else
            echo "kubectl download $(Kubectl.Version)"
            curl -LO $(Solo.Nexus3.Repositories.Uri)repository/generic-group/v$(Kubectl.Version)/bin/$TARGE
            mv ./kubectl $(Pipeline.Workspace)/.local/bin/
          fi


          if [ $(Kubelogin.Version) == 'latest' ];then
            echo "kubelogin download latest"
            curl -LO $(Solo.Nexus3.Repositories.Uri)repository/generic-group/latest/download/kubelogin-$TAR
          else
            echo "kubelogin download v$(Kubelogin.Version)"
            curl -LO $(Solo.Nexus3.Repositories.Uri)repository/generic-group/download/v$(Kubelogin.Version)
          fi

          unzip kubelogin-$TARGETOS-$TARGETARCH.zip
          mv ./bin/"$TARGETOS"_"$TARGETARCH"/kubelogin $(Pipeline.Workspace)/.local/bin/
          rm kubelogin-$TARGETOS-$TARGETARCH.zip
          ls -la $(Pipeline.Workspace)/.local/bin/
          chmod +x $(Pipeline.Workspace)/.local/bin/*
          echo "##vso[task.setvariable variable=PATH]${PATH}:$(Pipeline.Workspace)/.local/bin"
```

Tasks on Windows Agents to install kubectl and kubelogin

```
variables:
  - group: Abnamro.Coesd.VariableGroup.GlobalVars
  - name: Kubelogin.Version
    value: #Specify version ex: 0.0.31 or latest
  - name: Kubectl.Version
    value: #Specify version ex: 1.25.5 or latest

steps:
    - bash: |
        TARGETOS=$(echo "$(Agent.OS)" | sed 's/Windows_NT/win/' )
        TARGETARCH=$(case "$(Agent.OSArchitecture)" in "X86") echo "x86";; "X64") echo "amd64";; "ARM") ech

        if [ $(Kubectl.Version) == 'latest' ];then
          latestVersion=$(curl -L -s $(Solo.Nexus3.Repositories.Uri)repository/generic-group/stable.txt)
          echo "kubectl download latest"
          curl -LO $(Solo.Nexus3.Repositories.Uri)repository/generic-group/$latestVersion/bin/$TARGETOS/$TA
        else
          echo "kubectl download $(Kubectl.Version)"
          curl -LO $(Solo.Nexus3.Repositories.Uri)repository/generic-group/v$(Kubectl.Version)/bin/$TARGETC
        fi

        if [ $(Kubelogin.Version) == 'latest' ];then
          echo "kubelogin download latest"
          curl -LO $(Solo.Nexus3.Repositories.Uri)repository/generic-group/latest/download/kubelogin-$TARGE
        else
          echo "kubelogin download v$(Kubelogin.Version)"
          curl -LO $(Solo.Nexus3.Repositories.Uri)repository/generic-group/download/v$(Kubelogin.Version)/k
        fi

        echo "kubelogin unzip"
        unzip kubelogin-$TARGETOS-$TARGETARCH.zip
        rm kubelogin-$TARGETOS-$TARGETARCH.zip
      displayName: 'Download kubectl and kubelogin'


    - task: PowerShell@2
      displayName: 'Install kubectl and kubelogin'
      inputs:
        targetType: 'inline'
        script: |
          $TARGETOS = if ($env:AGENT_OS -eq "Windows_NT") {"windows"}; Write-Host "TARGETOS=$TARGETOS"
          $TARGETARCH = switch ($env:AGENT_OSARCHITECTURE) { "X86" { "x86" } "X64" { "amd64" } "ARM" { "arm

          New-Item -Path "$($env:PIPELINE_WORKSPACE)\.local\bin" -Type Directory -Force
          Move-Item -Path ".\kubectl" -Destination "$($env:PIPELINE_WORKSPACE)\.local\bin" -Force
          Move-Item -Path "./bin/$($TARGETOS)_$($TARGETARCH)/*" -Destination "$($env:PIPELINE_WORKSPACE)\.l
          Get-ChildItem -Path "$($env:PIPELINE_WORKSPACE)\.local\bin" -Force
          Write-Host "##vso[task.setvariable variable=PATH]$($env:PATH);$($env:PIPELINE_WORKSPACE)\.local\b
```

Using Active Directory Cluster user credentials tagged to Service Principal

It is also possible to use Service principal based Authentication for Kubelogin. Following is the example of same which can be achieved using Pipeline task by running using 'Private Pool Deployment Docker' pool.

```
- task: AzureCLI@2
  displayName: 'Install k8s resources'
  inputs:
    azureSubscription: $(resourceGroup)
    addSpnToEnvironment: true
    scriptType: bash
    scriptLocation: inlineScript
    failOnStandardError: false
      # Get AKS credentials
      az aks get-credentials -g $(RESOURCE_GROUP_NAME) \
        -n $(CLUSTER_NAME) \
        --subscription $(SUBSCRIPTION_ID) \
        --overwrite-existing
      # Convert AKS credentials to kubelogin-style
      kubelogin convert-kubeconfig -l spn \
        --client-id $servicePrincipalId \
        --client-secret $servicePrincipalKey \
        --tenant-id $tenantId
      # export AAD_SERVICE_PRINCIPAL_CLIENT_ID=$servicePrincipalId
      # export AAD_SERVICE_PRINCIPAL_CLIENT_SECRET=$servicePrincipalKey

      # Test use of kubectl
      kubectl get pods
      kubectl cluster-info
```

Using Active Directory Cluster user credentials tagged to Managed Identity

This can be used as only when you access from local machine or VDI, if pipeline is used then service principal is the way to go for now...

```
# Get AKS credentials
az aks get-credentials -g $(RESOURCE_GROUP_NAME) \
   -n $(CLUSTER_NAME) \
   --subscription $(SUBSCRIPTION_ID) \
   --overwrite-existing

# Convert AKS credentials to kubelogin-style
kubelogin convert-kubeconfig -l msi --client-id <msi-client-id>

# Test use of kubectl
kubectl get pods
kubectl cluster-info
```

Use Kubernetes Service Connection:

First service connection should be created. Refer this document .

Once the service connection is created you can use kubernetes task in Azure Devops (ADO) to deploy your application. Kubernetes tasks provided by ADO has Kubectl bundled with it. So it will automatically takes care of running Kubectl commands to you. No need to install Kubectl and Kubelogin unless you use Kubectl commands explicitly via shell command.