**Powershell**

**Assignment:**

1. Create a PowerShell script to tag Azure resources and set up a CI/CD pipeline in Azure DevOps to automate the execution of this script.

\* Create few resources in azure portal without tags.

\* Write a PowerShell script that tags an Azure resource. The script should take the resource group name, resource name, resource type, and tags as parameters.

tags:

Purpose and Role, Deployment type, Owner, Environment

2. Set up a yaml based CI/CD pipeline in Azure DevOps to run the PowerShell script. The pipeline will:

Check out the repository containing the PowerShell script.

Execute the script with parameters.

param (

[Parameter(Mandatory = $true)]

[string]$ResourceGroupName,

[Parameter(Mandatory = $true)]

[string]$ResourceName,

[Parameter(Mandatory = $true)]

[string]$ResourceType, # e.g., Microsoft.Storage/storageAccounts

[Parameter(Mandatory = $true)]

[string]$Purpose,

[Parameter(Mandatory = $true)]

[string]$Role,

[Parameter(Mandatory = $true)]

[string]$DeploymentType,

[Parameter(Mandatory = $true)]

[string]$Owner,

[Parameter(Mandatory = $true)]

[string]$Environment

)

# Login to Azure if not already authenticated

if (-not (Get-AzContext)) {

Connect-AzAccount

}

# Get the current resource

$resource = Get-AzResource -ResourceGroupName $ResourceGroupName -ResourceType $ResourceType -ResourceName $ResourceName -ErrorAction Stop

# Prepare the tags

$tags = @{

Purpose = $Purpose

Role = $Role

DeploymentType = $DeploymentType

Owner = $Owner

Environment = $Environment

}

# Merge with existing tags if present

$existingTags = $resource.Tags

if ($existingTags) {

foreach ($key in $tags.Keys) {

$existingTags[$key] = $tags[$key]

}

$tags = $existingTags

}

# Set the updated tags

Set-AzResource -ResourceId $resource.ResourceId -Tag $tags -Force

Write-Host "Tags applied successfully to $ResourceName"



