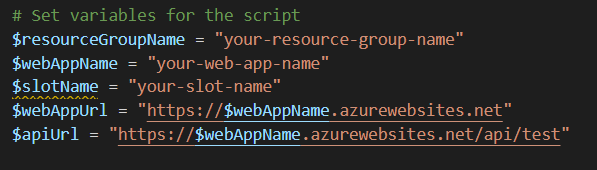
This script performs the following steps:

Text

Description automatically generated

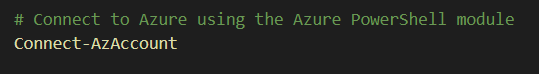
Set variables for the script:



* The first few lines of the script set some variables that will be used later in the script. These variables are:
  + **$resourceGroupName:** 
    - The name of the Azure resource group that the web app is in.
  + **$webAppName:** 
    - The name of the Azure web app.
  + **$slotName:** 
    - The name of the deployment slot for the web app.
  + **$webAppUrl:**
    - The URL of the web app.
  + **$apiUrl:** 
    - The URL of the web app's API endpoint.

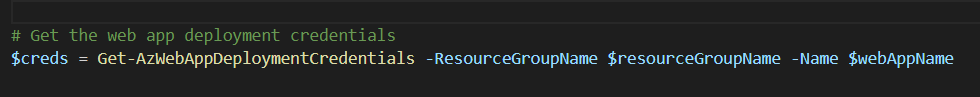
Note: These variable values need to be replaced with the actual resource group, web app, and deployment slot names.

Connect to Azure using the Azure PowerShell module:



* The script then connects to Azure using the Azure PowerShell module. This allows the script to access and manage resources in the Azure subscription.
* **Connect-AzAccount:**
  + This cmdlet prompts the user to log in to their Azure account and then connects to Azure.

Get the web app deployment credentials:



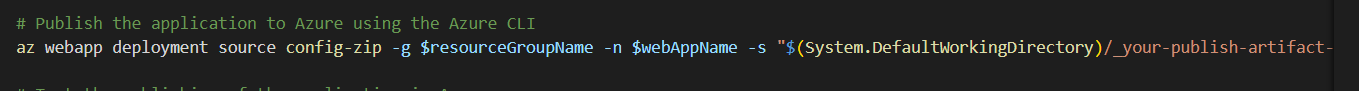
* The next step is to get the deployment credentials for the web app.
  + These credentials will be used later when publishing the application to Azure.

A screenshot of a computer

Description automatically generated with medium confidence

* **$creds = Get-AzWebAppDeploymentCredentials -ResourceGroupName $resourceGroupName -Name $webAppName:**
  + This cmdlet gets the deployment credentials for the web app specified by the **$webAppName** and **$resourceGroupName** variables and stores them in the $creds variable.

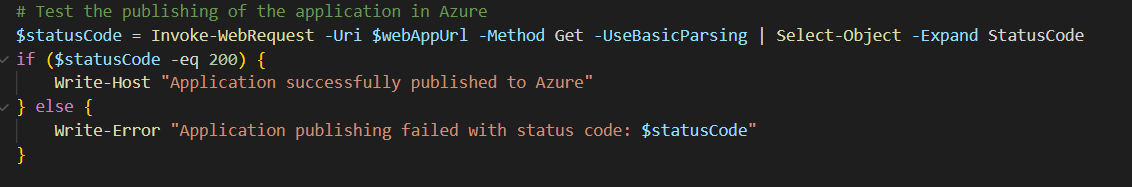
Publish the application to Azure using the Azure CLI:



* The script then publishes the application to Azure using the Azure CLI.
* **az webapp deployment source config-zip -g $resourceGroupName -n $webAppName -s "$(System.DefaultWorkingDirectory)/\_your-publish-artifact-name.zip" -u $creds.PublishingUserName -p $creds.PublishingPassword:** 
  + This cmdlet publishes the application to the Azure web app specified by the **$webAppName** and **$resourceGroupName** variables.
  + The -s parameter specifies the path to the ZIP file containing the application code, and the -u and -p parameters specify the deployment credentials.

Note: The az command is not part of the PowerShell module but is a separate command-line interface that needs to be installed on the machine.

Test the publishing of the application in Azure:



* After publishing the application to Azure, the script tests if the application is accessible through the web app URL.
* **$statusCode = Invoke-WebRequest -Uri $webAppUrl -Method Get -UseBasicParsing | Select-Object -Expand StatusCode:**
  + This cmdlet sends an HTTP GET request to the web app URL and stores the response status code in the $statusCode variable.
* **if ($statusCode -eq 200) { Write-Host "Application successfully published to Azure" } else { Write-Error "Application publishing failed with status code: $statusCode" }:**
  + This code block checks if the status code returned from the web app is 200 (OK). If it is, it prints a success message. If not, it prints an error message.

Test API connectivity to Azure:

Text

Description automatically generated

* The script also tests the connectivity to the web app's API endpoint.
* **$apiResponse = Invoke-WebRequest -Uri $apiUrl -Method Get -UseBasicParsing:**
  + This cmdlet sends an HTTP GET request to the web app's API endpoint and stores the response in the $apiResponse variable.
* **if ($apiResponse.StatusCode -eq 200) { Write-Host "API connectivity to Azure successful" } else { Write-Error "API connectivity to Azure failed with status code: $($apiResponse.StatusCode)" }:**
  + This code block checks if the status code returned from the API endpoint is 200 (OK). If it is, it prints a success message. If not