Provisioning ARM Templates on Azure using Powershell and Github Chat Copilot

A lab/workshop document that teaches how to deploy Azure resources using code and AI

# Introduction

In this lab, you will learn how to provision Azure resources using Azure Resource Manager (ARM) templates and Powershell. You will also use Github Chat Copilot, an AI-powered coding assistant, to help you write and edit your code. By the end of this lab, you will be able to deploy a web app and a database on Azure using code and AI.

# Prerequisites

To complete this lab, you will need the following:

* An Azure subscription. If you don't have one, you can create a free account undefined.
* A Github account. If you don't have one, you can create one undefined.
* A Github Chat Copilot extension for Visual Studio Code. You can install it from the Visual Studio Marketplace undefined.
* A Visual Studio Code editor. You can download it from undefined.
* A Powershell terminal. You can use the integrated terminal in Visual Studio Code or any other terminal of your choice.

# Steps

Follow these steps to complete the lab:

1. Create a Github repository for your project and clone it to your local machine.
2. Create an ARM template file that defines the Azure resources , we will use CopIlot to begin this process:

As a devops developer I need **to** provision an existing resource group with Azure AI Services that will support document intelligence **for** forms processing of pdf documents. I need **to** provision a service **for** sentiment analysis of the document contents, **and** finally, I need **to** translate the findings into Spanish **and** German **for** my affiliates **in** those countries Create an ARM template **to** provision these resources with Powershell. That Powershell script need **to** login into an Azure Account with Service Principal credentials **and** dispatch the ARM template with parameters. Create the procedure **and** scripts

GitHub Copilot: To provision Azure AI services for document intelligence, sentiment analysis, and translation, you will need to use several Azure services: Azure Form Recognizer for forms processing, Azure Text Analytics for sentiment analysis, and Azure Translator Text for translation.

Here's a high-level overview of the steps you'll need to follow:

1. Create an Azure Resource Manager (ARM) template that defines the resources you need.

2. Write a PowerShell script that logs into your Azure account with a service principal and deploys the ARM template.

Here's an example of how you can create the ARM template (`azuredeploy.json`):

{  
 "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",  
 "contentVersion": "1.0.0.0",  
 "parameters": {  
 "formRecognizerName": {  
 "type": "string"  
 },  
 "textAnalyticsName": {  
 "type": "string"  
 },  
 "translatorTextName": {  
 "type": "string"  
 }  
 },  
 "resources": [  
 {  
 "type": "Microsoft.CognitiveServices/accounts",  
 "apiVersion": "2021-04-30",  
 "name": "[parameters('formRecognizerName')]",  
 "location": "[resourceGroup().location]",  
 "kind": "FormRecognizer",  
 "sku": {  
 "name": "F0"  
 },  
 "properties": {  
 "customSubDomainName": "[parameters('formRecognizerName')]"  
 }  
 },  
 {  
 "type": "Microsoft.CognitiveServices/accounts",  
 "apiVersion": "2021-04-30",  
 "name": "[parameters('textAnalyticsName')]",  
 "location": "[resourceGroup().location]",  
 "kind": "TextAnalytics",  
 "sku": {  
 "name": "F0"  
 },  
 "properties": {  
 "customSubDomainName": "[parameters('textAnalyticsName')]"  
 }  
 },  
 {  
 "type": "Microsoft.CognitiveServices/accounts",  
 "apiVersion": "2021-04-30",  
 "name": "[parameters('translatorTextName')]",  
 "location": "[resourceGroup().location]",  
 "kind": "TextTranslation",  
 "sku": {  
 "name": "F0"  
 },  
 "properties": {  
 "customSubDomainName": "[parameters('translatorTextName')]"  
 }  
 }  
 ]  
}

This ARM template creates three Cognitive Services accounts: one for Form Recognizer, one for Text Analytics, and one for Translator Text. The names of these accounts are provided as parameters.

Next, you can create a PowerShell script (`deploy.ps1`) to deploy the ARM template:

param(  
 [Parameter(**Mandatory**=$true)]  
 [string]  
 $SubscriptionId,  
  
 [Parameter(**Mandatory**=$true)]  
 [string]  
 $TenantId,  
  
 [Parameter(**Mandatory**=$true)]  
 [string]  
 $AppId,  
  
 [Parameter(**Mandatory**=$true)]  
 [string]  
 $Password,  
  
 [Parameter(**Mandatory**=$true)]  
 [string]  
 $ResourceGroupName,  
  
 [Parameter(**Mandatory**=$true)]  
 [string]  
 $DeploymentName,  
  
 [Parameter(**Mandatory**=$true)]  
 [string]  
 $FormRecognizerName,  
  
 [Parameter(**Mandatory**=$true)]  
 [string]  
 $TextAnalyticsName,  
  
 [Parameter(**Mandatory**=$true)]  
 [string]  
 $TranslatorTextName  
)  
  
# Login to Azure  
$securePassword = ConvertTo-SecureString -String $Password -AsPlainText -Force  
$credential = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $AppId, $securePassword  
Connect-AzAccount -ServicePrincipal -Credential $credential -Tenant $TenantId -SubscriptionId $SubscriptionId

# Deploy the ARM template

**New**-AzResourceGroupDeployment -ResourceGroupName $ResourceGroupName -TemplateFile "./azuredeploy.json" -TemplateParameterObject @{  
 formRecognizerName = $FormRecognizerName  
 textAnalyticsName = $TextAnalyticsName  
 translatorTextName = $TranslatorTextName  
} -Name $DeploymentName

This PowerShell script logs into your Azure account with a service principal and deploys the ARM template to a specified resource group. The names of the Cognitive Services accounts are provided as parameters.

To run the PowerShell script, you can use the following command:

./deploy.ps1 -SubscriptionId "<subscription-id>" -TenantId "<tenant-id>" -AppId "<app-id>" -Password "<password>" -ResourceGroupName "<resource-group-name>" -DeploymentName "<deployment-name>" -FormRecognizerName "<form-recognizer-name>" -TextAnalyticsName "<text-analytics-name>" -TranslatorTextName "<translator-text-name>"

**Replace the placeholders (`<...>`) with your actual values.**

If you don't have a value for a parameter, you can either:

1. Remove the parameter from the script if it's not necessary for your use case. For example, if you don't need to specify a custom name for the Form Recognizer, Text Analytics, or Translator Text services, you can remove the corresponding parameters from the script and the ARM template.

2. Provide a default value in the script. You can specify a default value for a parameter in the `param` block of the script like this:

param(  
 [Parameter(**Mandatory**=$false)]  
 [string]  
 $FormRecognizerName = "default-form-recognizer-name"  
)

In this example, if you don't provide a value for the `FormRecognizerName` parameter when you run the script, it will use `"default-form-recognizer-name"` as the default value.