Online Lab - Integrating SaaS Services Available on the Azure Platform

Topic: Deploying Service Instances as Components of Overall Azure Solutions

Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
    - Username: **Admin**
    - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
    - Microsoft Edge
    - File Explorer
    - Visual Studio Code
    - Microsoft Azure Storage Explorer
    - Bash on Ubuntu on Windows
    - Windows PowerShell

      Note: You can also find shortcuts to these applications in the **Start Menu**.

Exercise 1: Deploy Function App and Cognitive Service using ARM Template

Task 1: Open the Azure portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (https://portal.azure.com).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

Task 2: Deploy Cognitive Service using an Azure Resource Manager template

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.

3. On the **Everything** blade, in the search results, click **Template Deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the **F:\Labfiles\Mod10\Starter** folder, select the **cognitive-template.json** file, and click **Open**. This will load the following content into the template editor pane:

```
{ "$schema": "http://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#", "contentVersion": "1.0.0.0", "variables":
{ "serviceName": "[concat('cgnt', uniqueString(resourceGroup().id))]"
}, "resources": [ { "apiVersion": "2017-04-18", "type":
"Microsoft.CognitiveServices/accounts", "name":
"[variables('serviceName')]", "kind": "TextAnalytics", "location":
"[resourceGroup().location]", "sku": { "name": "S1" }, "properties": {}
} ], "outputs": { "cognitiveEndpointUrl": { "type": "string", "value":
"[reference(variables('serviceName')).endpoint]" },
"cognitiveEndpointKey": { "type": "string", "value":
"[listKeys(variables('serviceName'), '2017-04-18').key1]" } } }
```

8. Click the **Save** button to persist the template.
9. Back on the **Custom deployment** blade, perform the following tasks:
    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab1001-RG**.
    - In the **Location** drop-down list, select the Azure region to which you intend to deploy resources in this lab.
    - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
    - Click the **Purchase** button.
10. Wait for the deployment to complete before you proceed to the next step.
11. In the hub menu of the Azure portal, click **Resource groups**.
12. On the **Resource groups** blade, click **AADesignLab1001-RG**.
13. On the **AADesignLab1001-RG** blade, locate the **Deployments** header at the top of the blade and click the below the **Deployments** label, which indicates the number of successful deployments.
14. On the deployments blade, click the name of the most recent deployment.
15. On the **Microsoft.Template- Overview** blade, click **Outputs**.
16. On the **Microsoft.Template - Outputs** blade, identify the values of **COGNITIVEENDPOINTURL** and **COGNITIVEENDPOINTKEY** outputs. Record these values, since you will need them later in the lab.

Task 3: Deploy a function app

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Function App** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Function App**.
4. On the **Function App** blade, click the **Create** button.
5. On the next **Function App** blade, perform the following tasks:
    - In the **App name** text box, type a globally unique name.
    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1001-RG**.
    - In the **OS** section, ensure that the **Windows** button is selected.
    - In the **Hosting Plan** drop-down list, ensure that the **Consumption Plan** entry is selected.
    - In the **Location** drop-down list, select the Azure region to which you deployed an instance of Cognitive Service in the previous task.
    - In the **Runtime Stack** drop-down list, ensure that the .NET entry is selected.
    - In the **Storage** section, ensure that the **Create new** option is selected and accept the default value of the Storage Account name.
    - In the **Application Insights** section, click **Off**.
    - Click the **Create** button.
6. Wait for the provisioning of the function app to complete before you proceed to the next step.
7. In the hub menu of the Azure portal, click **Resource groups**.
8. On the **Resource groups** blade, click **AADesignLab1001-RG**.
9. On the **AADesignLab1001-RG** blade, in the list of resources, click the newly provisioned function app.
10. On the function app blade, click the **Platform features** tab at the top of the blade.
11. On the **Platform features** tab, click the **Application Settings** link in the **GENERAL SETTINGS** section.
12. On the **Application settings** tab, locate the **Application Settings** section. Click the **Add new setting** link and perform the following tasks:
    - In the **Enter a name** text box, type **EndpointUrl**.
    - In the **Enter a value** text box, enter the value of **COGNITIVEENDPOINTURL** you identified earlier.
13. In the **Application Settings** section, click the **Add new setting** link again and pferform the following tasks:
    - In the **Enter a name** text box, type **EndpointKey**.

    - In the **Enter a value** text box, type the value of **COGNITIVEENDPOINTKEY** you identified earlier.
14. Click the **Save** button at the top of the **Application settings** tab.
15. Back on the function app blade, click the **Platform features** tab at the top of the blade.
16. In the **Platform features** tab, click the **Deployment options** link in the **Code Deployment** section.
17. On the **Platform features** tab, click the **Application settings** link in the **General Settings** section.
18. On the **Application settings** tab, scroll down to the **Application settings** section and add the following entry:

    •APP SETTING NAME: **EndpointUrl**

    •VALUE: the value of the **COGNITIVEENDPOINTURL** you recorded earlier in this lab

19. On the **Application settings** tab, in the **Application settings** section, add the following entry:

    •APP SETTING NAME: **EndpointKey**

    •VALUE: the value of the **COGNITIVEENDPOINTKEY** you recorded earlier in this lab

20. Save the changes.

21. On the **Deployments** blade that appears, click the **Setup** button at the top of the blade.

22. On the **Deployment option** blade, click **Choose Source**.

23. On the **Choose source** blade, click **External Repository**.

24. On the **Deployment option** blade, perform the following tasks:

- In the **Repository URL** text box, type **https://github.com/azure-labs/cognitive-services-function**
- In the **Branch** drop-down list, ensure that **master** entry is selected.
- In the **Repository Type** section, ensure that the **Git** option is selected.
- Click the **OK** button.

25. Wait for the deployment to complete before you proceed to the next task.

Note: You will be able to determine that the first deployment has completed by monitoring the **Deployments** tab. This tab updates automatically.

Task 4: Test a function app using Cognitive Services

1. Back on the function app blade, click **Functions** to expand the list of functions.

   > **Note**: You may need to click **Functions** twice to refresh the list of functions.

2. Select the **DetermineLanguage** function from the list of functions.
3. In the **run.csx** pane that opens, click **Test** on the right side of the pane.
4. In the **Test** pane, perform the following tasks:
   - In the **Request body** text box, type the following:

```
{ "text": "I stuffed a shirt or two into my old carpet-bag, tucked it under my arm, and started for Cape Horn and the Pacific." }
```

   - Click the **Run** button.
   - Review the output in the **Output** section. The output should identify the language as **en** (English).

Review: In this exercise, you created a function app that uses Azure Cognitive Services.

Exercise 2: Create a Logic App that uses a Function App

Task 1: Create a logic app

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Logic App** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Logic App**.
4. On the **Logic App** blade, click the **Create** button.
5. On the **Create logic app** blade, perform the following tasks:
   - In the **Name** text box, enter the value **CognitiveWorkflow**.
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1001-RG**.
   - In the **Location** drop-down list, select the same Azure region you chose in the previous exercise of this lab.

    - In the **Log Analytics** section, ensure that the **Off** button is selected.
    - Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next task.

Task 2: Configure logic app steps

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1001-RG**.
3. On the **AADesignLab1001-RG** blade, click the entry representing the logic app you created in the previous task.
4. On the **Logic Apps Designer** blade, scroll down and click the **Blank Logic App** tile in the **Templates** section.
5. On the **Logic Apps Designer** blade, click the **Code view** button at the top of the pane.
6. On the **Logic Apps Designer** blade, review the blank Logic App JSON template:

```
{ "definition": { "$schema":
"https://schema.management.azure.com/providers/Microsoft.Logic/schemas/
2016-06-01/workflowdefinition.json#", "actions": {}, "contentVersion":
"1.0.0.0", "outputs": {}, "parameters": {}, "triggers": {} } }
```

7. Replace the default JSON template with the following template that includes an HTTP trigger (**F:\Labfiles\Mod10\Starter\logic-app.json**):

```
{ "definition": { "$schema":
"https://schema.management.azure.com/providers/Microsoft.Logic/schemas/
2016-06-01/workflowdefinition.json#", "actions": {}, "contentVersion":
"1.0.0.0", "outputs": {}, "parameters": {}, "triggers": { "manual": {
"inputs": { "method": "POST", "schema": { "properties": { "text": {
"type": "string" } }, "type": "object" } }, "kind": "Http", "type":
"Request" } } } }
```

8. On the **Logic Apps Designer** blade, click the **Designer** button.

    Note: At this point, you should see a single step in the designer. This is the "trigger" step that begins a workflow.

9. Click the **+ New Step** button in the designer.
10. In the **Choose an action** section, perform the following tasks:
    - In the search text box, type **Azure Functions**.
    - In the search results, select the action named **Choose an Azure function**.
    - In the next set of search results, select the Azure Function instance you created in the previous exercise of this lab.

    - In the final set of search results, select the **DetermineLanguage** function that will be used for the action.
11. In the **DetermineLanguage** step, perform the following tasks:
    - Click the **Show advanced options** link to display all options.
    - In the **Request Body** text box, type **@triggerBody()**.
    - In the **Method** drop-down list, select the **POST** option.
12. Click the **+ New Step** button in the designer.
13. In the **Choose an action** dialog that displays, perform the following tasks:
    - In the search text box, type **Azure Functions**.
    - In the search results, select the action named **Choose an Azure function**.
    - In the next set of search results, select the Azure Function instance you created in the previous exercise of this lab.
    - In the final set of search results, select the **DetermineKeyPhrases** function that will be used for the action.
14. In the **DetermineKeyPhrases** step, perform the following tasks:
    - Click the **Show advanced options** link to display all options.
    - In the **Request Body** text box, enter the value **@body('DetermineLanguage')**.
    - In the **Method** drop-down list, select the **POST** option.
15. Click the **+ New Step** button in the designer.
16. In the **Choose an action** dialog that displays, perform the following tasks:
    - In the search text box, type **Response**.
    - In the search results, select the **Action** named **Response Request**.
17. In the **Response** step, perform the following tasks:
    - In the **Status Code** text box, ensure that the value **200** is specified.
    - In the **Body** text box, type **@body('DetermineKeyPhrases')**.
18. At the top of the **Logic Apps Designer** blade, click the **Save** button to persist your workflow.
19. Scroll to the top of the **Logic Apps Designer** area and click the **When a HTTP request is received** step.
20. Copy the value of the **HTTP POST URL** text box. This URL will be used later in this lab.

Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.

2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.

   > **Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   - Leave the **Subscription** drop-down list entry set to its default value.
   - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you deployedf resources in this lab
   - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1001-RG**.
   - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   - Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

Task 3: Validate Logic App using Python

1. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to open the interactive **python** terminal:

   ```
   python
   ```

2. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to import the **requests** library:

   ```
   import requests
   ```

3. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command (replacing the placeholder `<logic app POST Url>` with the value of your url recorded earlier in this lab) and press **Enter** to create a variable containing the value of your logic app's url :

   ```
   url = "<logic app POST Url>"
   ```

4. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to send an HTTP POST request to trigger your logic app workflow:

```
response = requests.post(url, json={'text': 'Circumambulate the city of a dreamy Sabbath afternoon. Go from Corlears Hook to Coenties Slip, and from thence, by Whitehall, northward.'})
```

5. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to display the output of the Logic App workflow:

```
print(response.status_code, response.reason, response.text)
```

6. Close the **Cloud Shell** pane.

Review: In this exercise, you created a logic app that leverages the function app created in the previous exercise of this lab.

Exercise 3: Remove lab resources

Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

```
az group list --query "[?starts_with(name,'AADesignLab10')]".name --output tsv
```

3. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

```
az group list --query "[?starts_with(name,'AADesignLab10')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'
```

2. Close the **Cloud Shell** prompt at the bottom of the portal.

Review: In this exercise, you removed the resources used in this lab.