Module 14 - Lab 4: Implement Azure Logic Apps integration with Azure Event Grid

Scenario

Adatum Corporation has an extensive set of on-premises network monitoring framework that rely on the combination of agent-based and agentless solutions in order to provide visibility into any changes to its environment. The agentless solutions tend to be relatively inefficient, since they rely on polling in order to determine state changes.

As Adatum is preparing to migrate some of its workloads to Azure, its Enterprise Architecture team wants to address these inefficiencies and evaluate the use of event driven architecture available in the cloud. The notion of using events in a solution or application is not new to the team. In fact, they have been promoting the idea of event-driven programming among its developers. One of the core tenets of an event-driven architecture is to reverse the dependencies that existing services may have with each other. Azure provides this functionality by relying on Event Grid, which is a fully managed service that supports the routing of events by utilizing a publisher-subscriber model. At its core, Event Grid is an event routing service that manages the routing and delivery of events from numerous sources and subscribers.

An event is created by a publisher such as a Blob Storage account, an Azure resource group, or even an Azure subscription. As events occur, they are published to an endpoint called a topic that the Event Grid service manages to digest all incoming messages. Event publishers are not limited to services on Azure. It is possible to use events that originate from custom applications or systems that can run from anywhere. This includes applications that are hosted on-premises, in a datacenter, or even on other clouds, as long as they can post an HTTP request to the Event Grid service.

Event handlers include a number of Azure services, including serverless technologies such as Functions, Logic Apps, or Azure Automation. Handlers are registered with Event Grid by creating an event subscription. If the event handler endpoint is publicly accessible and encrypted by Transport Layer Security, then messages can be pushed to it from Event Grid.

Unlike many other Azure services, there is no Event Grid namespace that needs to be provisioned or managed. Topics for native Azure resources are built in and completely transparent to users while custom topics are provisioned ad hoc and exist in a resource group. Event subscriptions are simply associated with a topic. This model simplifies management of topics as subscriptions and makes Event Grid highly multi-tenant, allowing for massive scale out.

Azure Event Grid is agnostic to any language or platform. While it integrates natively with Azure services, it can just as easily be leveraged by anything that supports the HTTP protocol, which makes it a very clever and innovative service.

To explore this functionality, the Adatum Architecture team wants to test integration of Azure Logic Apps with Event Grid in order to:

- detect when the state of a designated Azure VM is changed
- automatically generate an email notification in response to the event

After completing this lab, you will be able to:

- Integrate Azure Logic Apps with Event Grid
- Trigger execution of Logic Apps in response to an event representing a change to a resource within a resource group

Exercise 0: Prepare the lab environment

Task 1: Deploy an Azure VM by using an Azure Resource Manager template

- [ ] 1. To save time the resources for this lab are already deployed for you.

Exercise 1: Configure authentication and authorization for an Azure logic app

1. Create an Azure Active Directory service principal
2. Assign the Reader role to the Azure AD service principal

Task 1: Create an Azure Active Directory service principal

- [ ] 1. Navigate to **portal.azure.com** and login using username **sheikhnasir5HX1G@gdcs2.com** and password **IEfFCdBSVjOeTy9g** .
- [ ] 2. In the Azure portal, start a **PowerShell** session within the **Cloud Shell**.
- [ ] 3. From the Cloud Shell pane, run the following to create a new Azure AD application that you will associate with the service principal you create in the subsequent steps of this task:

```
Connect-AzureAD
```

```
$primaryDomain = Get-AzureADDomain | Where-Object {$_.Name -like "*onmicrosoft*"} | Select -Expand Name
```

```
$az30304aadapp = New-AzADApplication -DisplayName 'az304aadsp' -HomePage 'http://az304aadsp' -IdentifierUris ('http://az304aadsp' + '.'
```

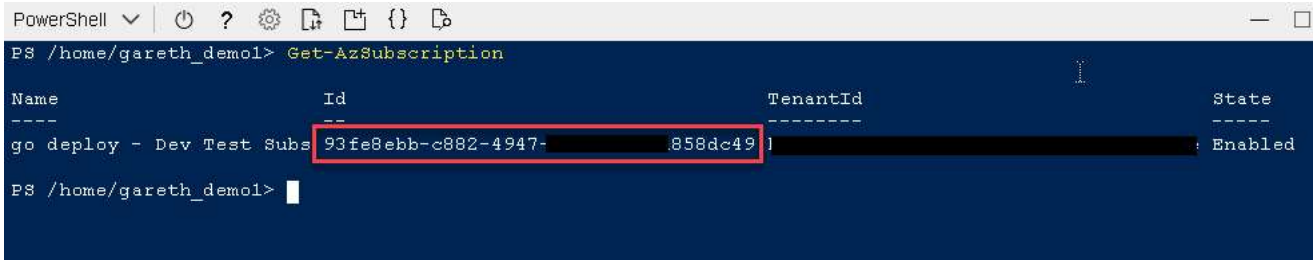
- [ ] 4. From the Cloud Shell pane, run the following to create a new Azure AD service principal associated with the application you created in the previous step:

```
New-AzADServicePrincipal -ApplicationId $az30304aadapp.AppID
```

- [ ] 5. In the output of the **New-AzADServicePrincipal** command, note the value of the **ApplicationId** property. You will need it later in this exercise.

- [ ] 6. From the Cloud Shell pane, run the following to identify the value of the **Id** property of the current Azure subscription and the value of the **TenantId** property of the Azure AD tenant associated with that subscription (you will also need them later in this exercise):

```
Get-AzSubscription
```



- [ ] 7. Close the Cloud Shell pane.

Task 2: Authorizing access to the Azure AD service principal

- [ ] 1. In the **Azure portal**, search for and select **Resource groups** and, on the **Resource groups** blade, select **unrecognised token ($gd.com(azure).resourceGroups(az30304a-labRG))**.

- [ ] 2. On the **unrecognised token ($gd.com(azure).resourceGroups(az30304a-labRG))** blade, select **Access control (IAM)**.

- [ ] 3. On the **unrecognised token ($gd.com(azure).resourceGroups(az30304a-labRG)) | Access control (IAM)** blade, select + **Add**, and select **Add role assignment**.

- [ ] 4. On the **Add role assignment** blade, specify the following settings and select **Save**:

Setting	Value
Role	**Reader**
Assign access to	**Azure AD user, group, or service principal**
Select	**az30304aadsp**

Exercise 2: Implement an Azure logic app

The main tasks for this exercise are as follows:

1. Create an Azure logic app
2. Add a trigger to the Azure logic app
3. Add a condition to the Azure logic app
4. Add an action to the Azure logic app

Task 1: Create an Azure logic app

- [ ] 1. In the **Azure portal**, search for and select **Logic App** and, on the **Logic Apps** blade, select **+ Add**.

- [ ] 2. On the **Basics** tab of the **Logic App** blade, specify the following settings (leave others with their default values):

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	the name of a new resource group **unrecognised token ($gd.com(azure).resourceGroups(az30304a-labRG))**
Logic App name	**az30304b-logicapp1**
Region	East US
Log Analytics	**Off**
Plan Type	**Consumption**

- [ ] 3. Select **Review + create** and then select **Create**.

Note: Wait for the logic app to be created. Provisioning should take about 2 minutes.

Task 2: Add a trigger to the Azure logic app

- [ ] 1. In the **Azure portal**, search for and select **Logic App** and, on the **Logic Apps** blade, select **az30304b-logicapp1**.

- [ ] 2. On the **Logic App Designer** blade, select **Blank Logic App**. This will display a blank designer workspace.
- [ ] 3. Use the **Search connectors and triggers** text box, to search for **Event Grid**, in the list of results, in the **Triggers** column, select **When a resource event occurs** Azure Event Grid trigger to add it to the designer workspace.
- [ ] 4. In the **Azure Event Grid** tile, select the **Connect with Service Principal** link, specify the following settings, and select **Create**:

Setting	Value
Connection Name	az30304egconnection
Client ID	the value of the **ApplicationId** property you identified earlier in this exercise
Client Secret	IEfFCdBSVjOeTy9g
Tenant	the value of the **TenantId** property you identified earlier in this exercise

- [ ] 5. In the **When a resource event occurs** tile, specify the following settings:

Setting	Value
Subscription	the value of subscription **Id** property you identified earlier in this exercise
Resource Type	**Microsoft.Resources.resourceGroups**
Resource Name	az30304a-labRG
Event Type Item - 1	**Microsoft.Resources.ResourceWriteSuccess**
Event Type Item - 2	**Microsoft.Resources.ResourceDeleteSuccess**

- [ ] 6. n the **When a resource event occurs** tile, select **Add new parameter** and select **Subscription Name**
- [ ] 7. In the **Subscription Name** text box, type event-subscription-az30304b and select **Save**.

Task 3: Add a condition to the Azure logic app

- [ ] 1. In the the **Azure portal**, on the Logic App Designer blade of the newly provisioned Azure logic app, select **+ New step**.
- [ ] 2. In the choose an action tile, use the **Search connectors and triggers** text box, to search for **Condition**, in the list of results, in the **Actions** column, select **Condition** to add it to the designer workspace.
- [ ] 3. Select the ellipsis symbol in the upper right corner of the **Condition** tile, in the pop-up menu, select **Rename**, and replace **Condition** with the text If a virtual machine in the resource group has changed.
- [ ] 4. Selet the **Choose a value** text box on the left hand side of the condition, in the pop up window, and in the **Dynamic content** tab, select the **Body** entry.
- [ ] 5. Ensure that **is equal to** appears in the middle element of the condition and, in the **Choose a value** text box on the right hand side, type the value representing the opearation you intend to monitor:

```
Microsoft.Compute/virtualMachines/write
```

- [ ] 6. On the **Logic Apps Designer** blade, select **Save**.

Task 4: Add an action to the Azure logic app

- [ ] 1. In the the **Azure portal**, on the Logic App Designer blade of the newly provisioned Azure logic app, in the **If true** tile, select **Add an action**.
- [ ] 2. In the **Choose an action** pane, in the **Search connectors and actions** text box, type **Outlook**.
- [ ] 3. In the list of results, select **Outlook.com**.
- [ ] 4. In the list of actions for **Outlook.com**, select **Send an email (V2)**.
- [ ] 5. In the **Outlook.com** pane, select **Sign in**.
- [ ] 6. When prompted, authenticate by using the Microsoft Account you are using in this lab.
- [ ] 7. When prompted for the consent to grant Azure Logic App permissions to access Outlook resources, select **Yes**.
- [ ] 8. In the **Outlook.com** pane, select the ellipsis symbol in the upper right corner of the **Send an email (V2)** tile, in the pop-up menu, select **Rename**, and replace **Send an email (v2)** with the text **Send an email**.
- [ ] 9. In the **Send an email** pane, specify the following settings and select **Save**:

Setting	Value
To	the primary e-mail address of your Microsoft Account
Subject	type **Resource updated:** and, in the **Dynamic Content** column to the right of the **Send an email** pane, select **Subject**

Setting	Value
Body	Type **Resource group:**, in the search text box under the **Dynamic Content** column to the right of the **Send an email** pane, type and select **Topic**. Back in the **Body** text box, on a new line, type **Event type:**. In the search text box under the **Dynamic Content** column to the right of the **Send an email** pane, type and select **Event Type**. Back in the **Body** text box, on a new line type **Event ID:**, in the search text box under the **Dynamic Content** column to the right of the **Send an email** pane, type and select **ID**. back in the **Body** text box, on a new line, type **Event Time:**, and in the search text box under the **Dynamic Content** column to the right of the **Send an email** pane, type and select **Event Time**.

- [ ] 10. On the **Logic Apps Designer** blade, select **Save**.

Exercise 3: Implement an event subscription

The main tasks for this exercise are as follows:

1. Configure event subscription
2. Review the functionality of the Azure logic app
3. Remove Azure resources deployed in the lab

Task 1: Configure event subscription

- [ ] 1. In the **Azure portal**, navigate to the **az30304b-logicapp1** blade, in the **Summary** section, select **See trigger history**.
- [ ] 2. On the **When_a_resource_event_occurs** blade, copy the value of the **Callback url [POST]** text box.
- [ ] 3. In the Azure portal, navigate to the **unrecognised token ($gd.com(azure).resourceGroups(az30304a-labRG))** resource group and, in the vertical menu, select **Events**.
- [ ] 4. On the **unrecognised token ($gd.com(azure).resourceGroups(az30304a-labRG)) | Events** blade, select **+ Event subscription**.
- [ ] 5. On the **Create Event Subscription** blade, specify the following settings and select **Create**:

Setting	Value
Name	event-subscription-az30304a-LabRG
Event Schema	**Event Grid Schema**
Filter to Event Types	**Resource Write Success**, **Resource Delete Success**, **Resource Action Success**
Endpoint Type	**Web Hook**
Endpoint	the URL string you copied at the beginning of this task

- [ ] 6. Select **Create**.

Task 2: Review the functionality of the Azure logic app

- [ ] 1. In the **Azure portal**, navigate to the **unrecognised token ($gd.com(azure).resourceGroups(az30304a-labRG))** resource group and, in the list of resources, select the entry representing the **az30304a-vm0** Azure VM.
- [ ] 2. On the **az30304a-vm0** blade, in the **Settings** section, select **Size**.
- [ ] 3. On the **az30304a-vm0 | Size** blade, select a size different from the one currently set and select **Resize** and verify that the resize operation completed successfully.
- [ ] 4. Navigate back to the **az30304b-logicapp1** blade, select **Refresh**, and note that the **Runs history** includes entries corresponding to changes of the state of the Azure VM.
- [ ] 5. In the **Runs history** listing, select an entry with the longest duration, representing the successful resizing on the Azure VM.
- [ ] 6. On the **Logic app run** blade, review the diagram representing the workflow of the logic app run.
- [ ] 7. On the **Logic app run** blade, select the **When a resource event occurs** rectangle to expand it and, in the **OUTPUTS** section, select **Show raw outputs**.
- [ ] 8. On the **Outputs** blade, review the details of the event and note that includes such details as the identity of your user account and the IP address from which the request to resize the Azure VM originated.
- [ ] 9. Navigate to the inbox of the email account you specified in the previous exercise and verify that includes an email generated by the logic app.