Module 07 - Azure Automation - Integration

Exercise 1: Integrate with Event Grid

Introduction

This scenario will walk through how to trigger Runbooks via a HTTPS Webhook.

Summary

In this lab, we will:

- Create a Webhook to Stop an Azure VM
- Trigger the Webhook

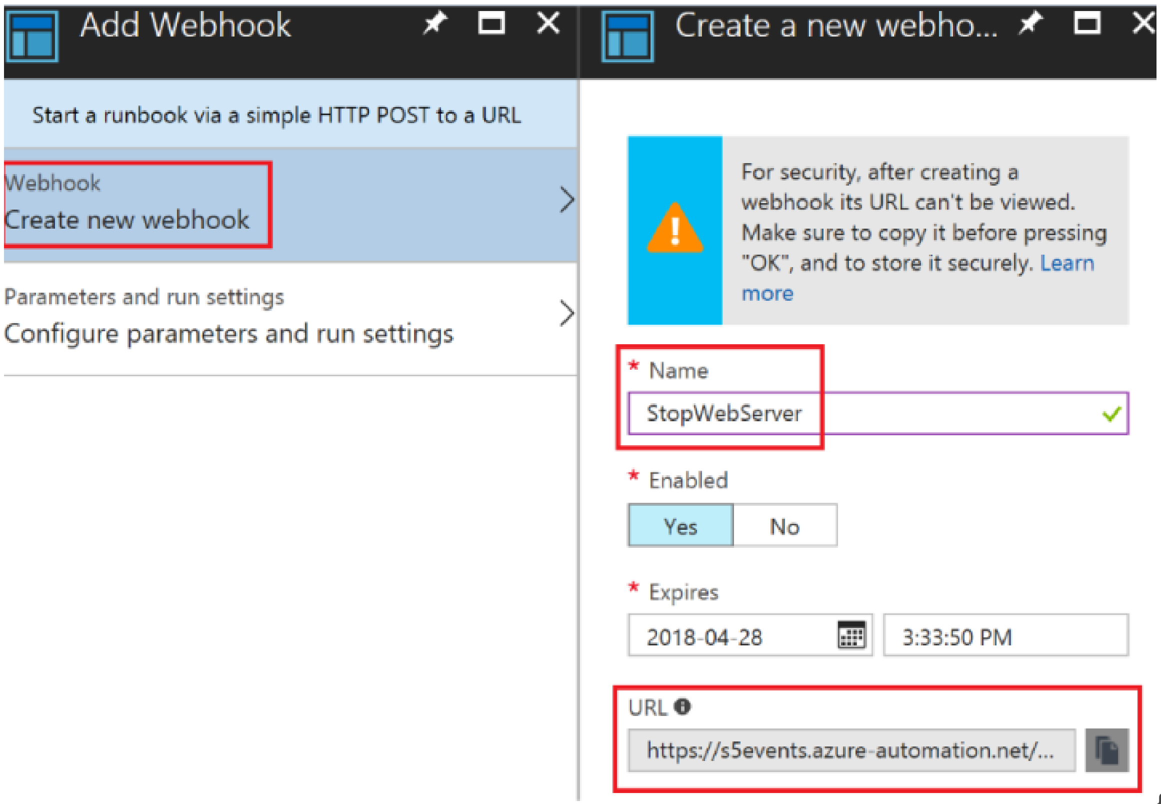
Estimated Time to Complete This Lab

20 minutes

Task Description

1. [] Go to https://portal.azure.com and navigate to your **ContosoAutomationAccount** Automation Account.
2. [] Click **Runbooks** and click the Runbook **StopAzureV2Vm**.
3. [] In the StopAzureV2Vm Runbook, click **Webhooks** under RESOURCES.
4. [] Click **+Add Webhook**, select **Create new webhook** and name the Webhook **StopWebServer**.
5. [] Copy the **URL** of the Webhook and paste it into Notepad.

> IMPORTANT! Copy the URL of the Webhook and make sure you make a note of it. Once you have created the Webhook, you cannot see the URL again!


Add Webhook
Start a runbook via a simple HTTP POST to a URL
Webhook
Create new webhook
Parameters and run settings
Configure parameters and run settings
Create a new webho...

For security, after creating a webhook its URL can't be viewed. Make sure to copy it before pressing "OK", and to store it securely. Learn more

* Name: StopWebServer

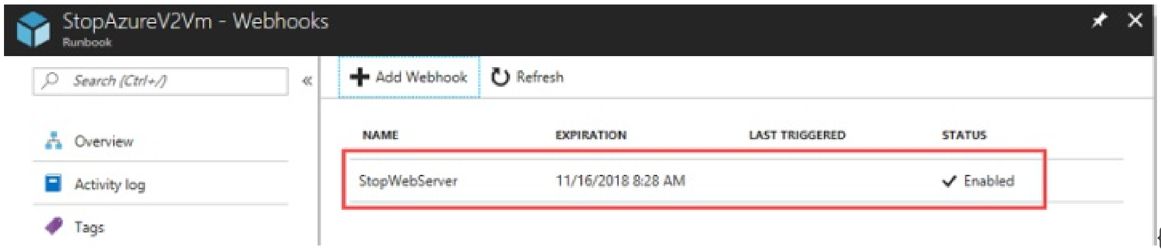
* Enabled: [✓ Yes] No

* Expires: 2018-04-28 3:33:50 PM

URL: https://s5events.azure-automation.net/...

{400}

6. [] You're not done yet. Click **OK** then click the **Configure parameters and run settings**.
7. [] In the Parameters, for **RESOURCEGROUPNAME** enter **ContosoIntegrationRG** Click **OK**.
8. [] Click **Create**.

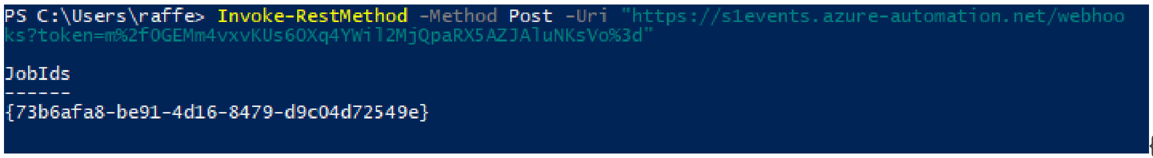


NAME	EXPIRATION	LAST TRIGGERED	STATUS
StopWebServer	11/16/2018 8:28 AM		✓ Enabled

{400}

9. [] Let's make sure the VM is running. In the Azure Portal, go to **Virtual Machines**. Click the VM **IntegrationVM**.
10. [] If the VM is not Running, click **Start**.
11. [] Next, let's start the Webhook. On your lab machine, open **Windows PowerShell**.
12. [] Enter the following command, pasting in your **Webhook URI** as appropriate:

```
Invoke-RestMethod -Method Post -Uri "MYWEBHOOKURI"
```



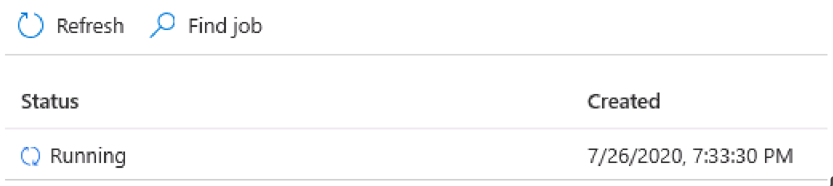
```
PS C:\Users\raffe> Invoke-RestMethod -Method Post -Uri "https://s1events.azure-automation.net/webhoo
ks?token=m%2fOGEMm4vxvKUs6OXq4YWi12MjQpaRX5AZJAluNKsVo%3d"

JobIds
------
{73b6afa8-be91-4d16-8479-d9c04d72549e}
```

{400}

> IMPORTANT! Webhooks are unauthenticated and rely on the privacy of the webhook URL. For this reason it is important to validate the use of a webhook, some ways we can do this are listed at https://docs.microsoft.com/en-us/azure/automation/automation-webhooks#webhook-security. If you want to use Azure AD authentication or some other auth provider consider using an Azure Function (which also supports PowerShell).

13. [] From your Automation account, click **Runbooks** and click the **StopAzureV2Vm** Runbook. Click **Jobs**.

14. [] You should see that the Runbook was successfully triggered by your Webhook:



Status	Created
Running	7/26/2020, 7:33:30 PM

{400}

15. [] In the Azure Portal, navigate to **Virtual machines** validate that the VM is stopped.

16. [] Start **IntegrationVM** so it will be ready for a later exercise.

Task 2: Integrate a runbook webhook with Event Grid

1. [] Go to https://portal.azure.com and navigate to your **ContosoAutomationAccount** Automation Account.

2. [] Create a new, empty runbook and add the following code to it:

```
param(
    [object]$WebhookData
)
$WebhookData.WebhookName | ConvertFrom-Json | Format-List *
$WebhookData.RequestHeaders | ConvertFrom-Json | Format-List *
$WebhookData.RequestBody | ConvertFrom-Json | Format-List *
```

3. [] Save and publish your runbook.

4. [] Click **+Add Webhook**, select **Create new webhook** and name the Webhook **EventGridIntegration**.

5. [] Copy the **URL** of the Webhook and paste it into Notepad.

6. [] In the cloud shell run the following command to ensure the EventGrid provider is registered:

```
Register-AzResourceProvider -ProviderNamespace Microsoft.EventGrid
```

7. [] In the search bar at the top of the screen, search for **Event Grid Subscriptions**

8. [] In your search results, click **Event Grid Subscriptions**

9. [] Click the **+ Event Subscription** and fill out the Create Event Subscription wizard, with the following details:

- **Name**: A friendly name between 3 and 64 characters long Use a combination of your name and the date.
- **Topic Types**: Select **Resource Groups** from the drop down
- **Resource Group**: Select **ContosoIntegrationRG** from the drop down
- **System Topic Name**: Enter a friendly topic name between 3 and 128 characters long
- **Filter to Event Types**: Make sure Resource Write Success is the only box selected.
- **Endpoint Type**: Select **Webhook** from the dropdown
- **Endpoint**: Click on **Select an endpoint** and paste your webhook into the **Subscriber Endpoint** box. Click **Confirm Selection** when done.

10. [] Leave the rest of the tabs with their default values. Click **Create**.

Task 3: Modify our runbook to accept the event grid webhook

1. [] navigate to your **ContosoAutomationAccount** Automation Account.
2. [] Select the runbook you created in the previous step and click **edit**
3. [] Edit the runbook so that it looks identical to the code below.

```
param(
     [object]$WebhookData
)

Connect-AzAccount -Identity

$RequestBody = $WebhookData.RequestBody | ConvertFrom-Json

$DateCreated = [datetime]$RequestBody.eventTime
$tag = @{
    MonthCreated = $DateCreated.Month
    DayCreated   = $DateCreated.Day
    YearCreated  = $DateCreated.Year
}


Update-AzTag -ResourceId $RequestBody.Subject -Operation Merge -Tag $tag
```

> [!note] We first connect with our automation account managed identity. After that we isolate the request body and convert it into an object with the ConvertFrom-Json cmdlet. After extracting the data we need, the eventtime in this case, we can use the subject property of our request body to determine which resourceId we want to update.

4. [] Save and publish your runbook.
5. To ensure we are using the latest verison of the AZ Powershell cmdlets in the runbook, go to the **modules** section of the automation account under **Shares Resources**. Click on **Browse gallery** and

search for and import the latest versions of the modules Az.Accounts and Az.Resources. Import them into the Powershell 5.1 runtime.

6. [] Create a new storage account in the **ContosoIntegrationRG** resource group. Leave all of the settings as their default settings.

7. [] Navigate to your **ContosoAutomationAccount** and click on jobs. Periodly refresh until you see a job from your runbook execute.

8. [] Once the job completes, review the job output.

9. [] Navigate to the storage account you created and observe the newly created tags

Exercise 2: Deploy Azure Log Analytics Workspace

Introduction

In this lab, we will look at integrations with Azure Automation.

Summary

During this lab, we will:

- Deploy a New VM for Monitoring
- Create an Azure Log Analytics Workspace
- Deploy Agent to an Azure VM
- Configure Data collection
- Search Collected Data

Estimated Time to Complete This Lab

20 minutes

Task 1: Deploy a New VM for Monitoring

1. Before commencing this lab ensure that any virtual machines you have created are stopped.

2. In the Azure Portal open the Cloud Shell and start PowerShell by entering *pwsh*

3. Run the code below to create a new virtual machine.

```
$location = Read-Host -Prompt "Enter one of the following locations:
southeastasia,westeurope,westus2"
New-AzVM -ResourceGroupName ContosoMonitorRG -Name ContosoMonVM -Location $location -
OpenPorts 3389 -PublicIpAddressName ContosoMonVM -Verbose
```

> **NOTE** Use the username and password below when prompted. The ports supplied in the command allow for RDP access access. **User name:** aa-admin **Password:** R3dDwarf2017

Task 2: Create Azure Log Analytics Workspace

1. Open your browser and navigate to https://portal.azure.com .
2. Expand the top left pane of the Azure Portal, click **+Create a resource**
3. In the marketplace, search for **Log Analytics Workspace**
4. In your search results, click **Log Analytics Workspace**.
5. Click **Create** and fill out the new Log Analytics Workspace wizard, with the following details:
    - **Log Analytics workspace**: This must be a **globally unique** name. Use a combination of your name and the date.
    - **Resource Group**: Select **Use existing** and select **ContosoResourceGroup**.
    - **Location**: Select the same location your Automation Account is in (see note below).
    - **Pricing tier**: Pay-as-you-go

> [!note] When selecting the **Location** of your Azure Log Analytics Workspace, you MUST put it in the same region as your existing Azure Automation account. This is because to link the accounts, they must be in the same region.

> [!note] In situations where you have similar regions, for example East US and East US 2, those may be considered the same region.

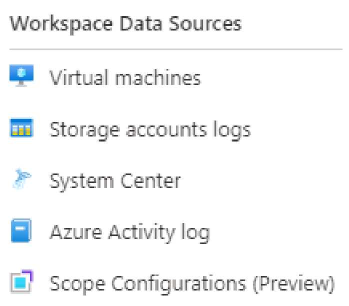
6. Click **Create** to create your Azure Log Analytics workspace.

Task 3: Connect VM to Log Analytics

1. Let's look at your Azure Log Analytics workspace. Expand the menu on the left side of the Azure Portal, click **All Services**. Search for **Log Analytics**, once found click **Log Analytics**.
2. Click on the Azure Log Analytics Workspace, which you just created.

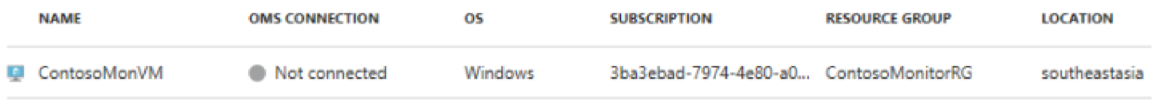
> [!note] You can also pin your Azure Log Analytics Workspace to the Dashboard by clicking the white thumbtack in the top right hand corner of the right pane.

3. Next, we need to install the agent onto an Azure VM. From Azure Log Analytics, scroll through the menu on the left, to locate **Virtual Machines**.


Workspace Data Sources

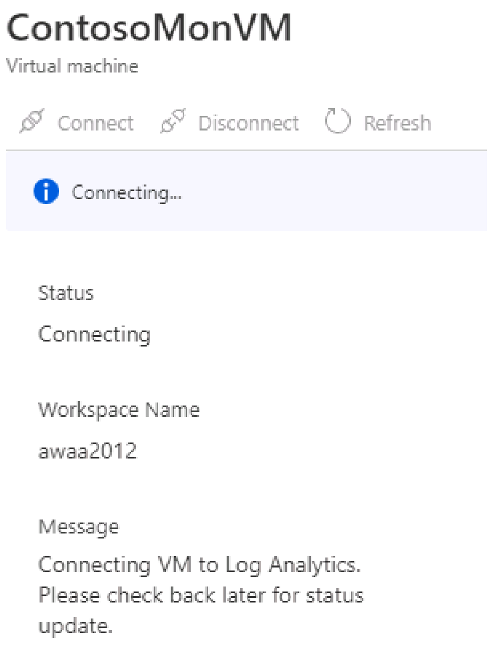
- Virtual machines
- Storage accounts logs
- System Center
- Azure Activity log
- Scope Configurations (Preview)

4. You should be able to see your Virtual Machine in the list. Notice, it is not currently connected to Azure Log Analytics. Click on the **ContosoMonVM** VM:



NAME	OMS CONNECTION	OS	SUBSCRIPTION	RESOURCE GROUP	LOCATION
ContosoMonVM	● Not connected	Windows	3ba3ebad-7974-4e80-a0...	ContosoMonitorRG	southeastasia

5. Click **Connect**. This will use Azure VM extensions to install and configure the Azure Log Analytics agent on the VM. This can take a few minutes to complete.



ContosoMonVM
Virtual machine

Connecting...

Status
Connecting

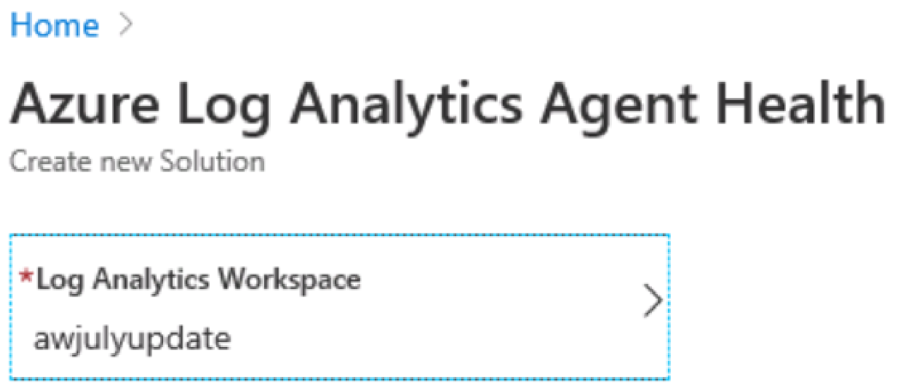
Workspace Name
awaa2012

Message
Connecting VM to Log Analytics. Please check back later for status update.

6. Next, we can configure the Azure Log Analytics workspace to collect data. In the Azure Log Analytics blade click on **Legacy agents management** under settings
7. Click on **Add Windows Event Log**.
8. In the text box enter **Application** and click the **apply** button.
9. Repeat the previous step to add the **System** event log.
10. Click on **Windows Performance Counters**
11. Click on **Add the recommended performance counters**
12. Click on the **Apply** button to persist the changes.

13. We will now add some solutions to the Azure Log Analytics workspace. In the search bar at the top of the portal enter **Agent Health** and click on **Azure Log Analytics Agent Health** under the Marketplace heading.

14. Ensure your Azure Log Analytics workspace is selected and click **Create**



Home >

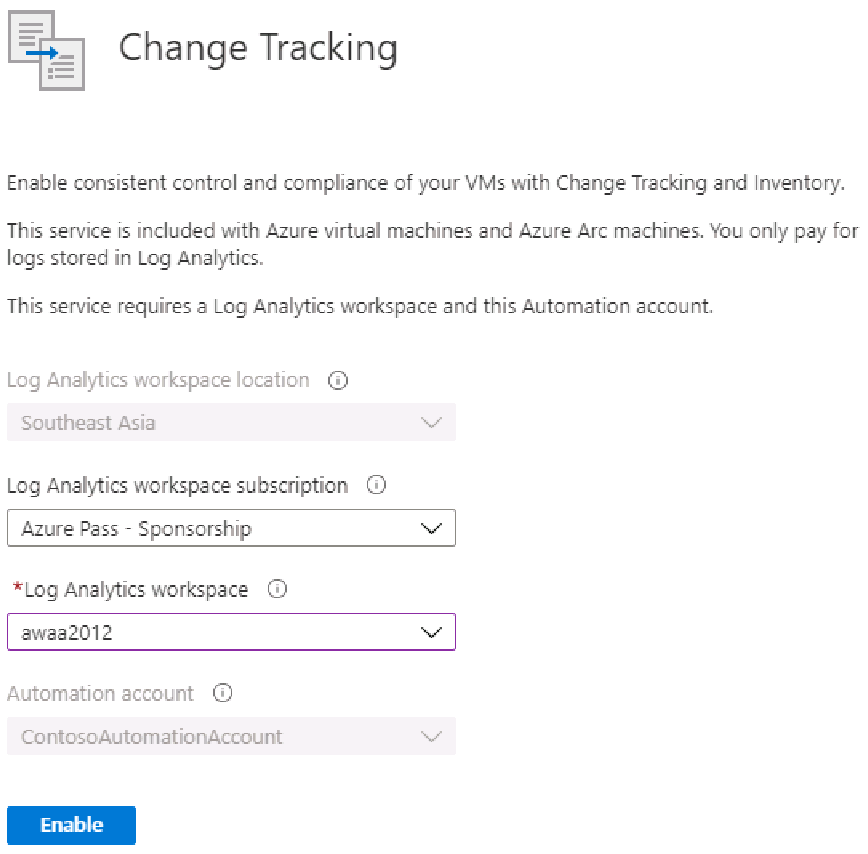
Azure Log Analytics Agent Health

Create new Solution

*Log Analytics Workspace

awjulyupdate

15. Open the **ContosoAutomationAccount** and select **Change Tracking**. Ensure that your Log Analytics workspace is selected and click **Enable**.



Change Tracking

Enable consistent control and compliance of your VMs with Change Tracking and Inventory.

This service is included with Azure virtual machines and Azure Arc machines. You only pay for logs stored in Log Analytics.

This service requires a Log Analytics workspace and this Automation account.

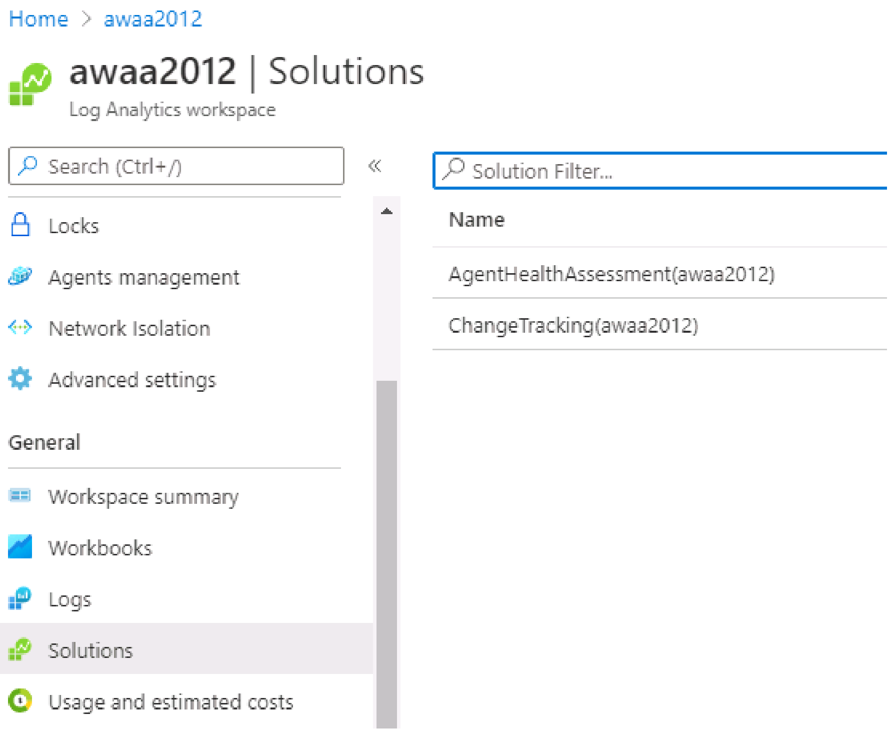
Log Analytics workspace location: Southeast Asia

Log Analytics workspace subscription: Azure Pass - Sponsorship

*Log Analytics workspace: awaa2012

Automation account: ContosoAutomationAccount

16. It may take a few minutes to deploy the solutions - however once complete you can check the **Solutions** entry in the Azure Log Analytics account to confirm both solutions have been deployed.



Name
AgentHealthAssessment(awaa2012)
ChangeTracking(awaa2012)

Task 4: Searching Collected Data

1. In the Azure Log Analytics blade click **Logs**
2. First, we will look at *all* the collected data. In the search box type below query in lower case, and click Run.

```
search *
```

3. On the left of the screen, you can click on **Schema** to see how the data is organised, or **Filter** to narrow down your search query.
4. View just the performance counter entries by entering the query in the box and clicking Run.

```
Perf
```

5. In the **Filters** section - choose one of the entries in the *CounterPath* and click the plus icon. Note how the query is updated.

> [!note] If you do not yet see Perf data wait a few minutes and try again.

Exercise 3: Azure Monitor Alert Auto-Resolution

Introduction

In this lab, we will demonstrate alerts in Azure Monitor and how they can be used to trigger an automation Runbook. This is typically used to automatically resolve issues. For example, a Runbook may resolve an alert by restating a service, clearing space, changing a DB setting, or updating a website, etc.

Summary

During this lab, we will:

- Configure monitoring to detect a stopped service
- Configure Log Analytics Workspace Alert
- Set the Azure Monitor alert to run an Azure Automation Runbook
- Verify the Runbook executed

Estimated Time to Complete This Lab

30 minutes

Task 1: Configure Monitoring

1. Now that our VM is sending data to Azure Log Analytics, we can trigger some events. We will restart a Service on **ContosoMonVM**, to see the events it generates. We will then build automation from this. In the Azure Portal, go to **Virtual Machines** and click on **ContosoMonVM**
2. On **ContosoMonVM**, under **Operations** click on **Run Command** and select **RunPowerShellScript**.
3. Paste or type the following code into the script pane and click **Run**

```
Stop-Service spooler
Start-Sleep -Seconds 10
Start-Service spooler
```

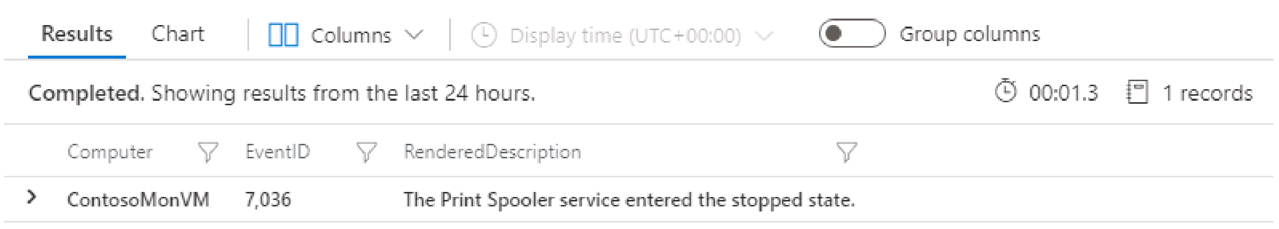
4. Wait for the command to return output - we have stopped and started the service to generate logs in Azure Log Analytics which we will review next.

Task 2: Configure Log Analytics Workspace Alert

1. You will need to wait a few minutes for the events to be sent to the Azure Log Analytics workspace - use the query below in the **Logs** window to search for events related to the spooler service stopping.

```
Event | where EventLog == "System"
| where RenderedDescription contains "spooler" and RenderedDescription contains "Stopped"
| project Computer, EventID, RenderedDescription
```

2. After some time events should be visible.

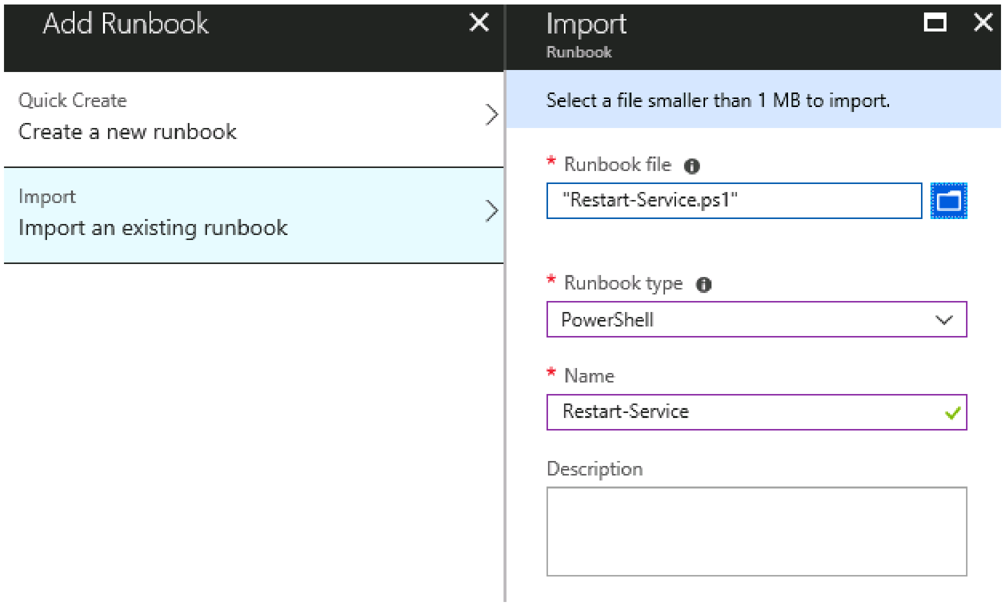


Completed. Showing results from the last 24 hours. 00:01.3 1 records

Computer	EventID	RenderedDescription
ContosoMonVM	7,036	The Print Spooler service entered the stopped state.

3. Open your automation account and click on **Runbooks**
4. Click **Import a runbook** and import the file at C:\Labs\Module7\Restart-Service.ps1 . Click **Create** to add the runbook. When the runbook edit screen opens click on **Publish**.

5. Repeat the above step and import the file at C:\Labs\Module7\Update-AutomationAzureModulesForAccount.ps1 - again ensure you click **Publish** when the runbook opens.



Select a file smaller than 1 MB to import.

Runbook file: "Restart-Service.ps1"

Runbook type: PowerShell

Name: Restart-Service

Description:

6. Under **Shared Resources**, click **Credentials**, and click **Add a Credential**. Fill in the following, then click **Create**:
   - **Name:** aa-admin
   - **User name:** aa-admin
   - **Password:** R3dDwarf2017
7. Click on **Modules → Browse Gallery** and add the **Az.Network**, **Az.OperationalInsights**, **As.Compute** and **Az.Resources** modules to the automation account in the **Powershell 5.1 runtime**.
8. In the case that you didn't complete previous labs, run the following in the cloud shell to grant the managed identity for the automation account contributor rights to your Azure subscription:

```
$scope = "/subscriptions/$((Get-AzContext).Subscription)"
$identity = (Get-AzResource -Name ContosoAutomationAccount).Identity.PrincipalId
New-AzRoleAssignment -Scope $scope -ObjectId $identity -RoleDefinitionName 'Contributor'
```

9. In the portal search for **Monitor** and click on **Monitor**. Click on **Alerts**
10. Click on **New Alert Rule**
11. In the resource section - search for the Log Analytics workspace you have created and select it. We scope this alert rule to that workspace rather than the virtual machine as we are going to perform a log search.
12. In the **Condition** section click on **Select condition**.
13. In the blade that open select **Log Search → Custom Log Search** as the signal. A new blade will appear prompting for a search query. Enter the query below and click out of the window.

> When you do this a query is performed with a time range of 5 minutes. If an event has occurred within this range it should appear.

```
Event | where EventLog == "System"
| where RenderedDescription contains "spooler" and RenderedDescription contains "Stopped"
| project Computer, EventID, RenderedDescription
```

13. Scroll down and enter **0** in the Threshold box.

14. Click on **Done** and the blade will close.

15. In the **Action Groups** section click on **Add action groups**

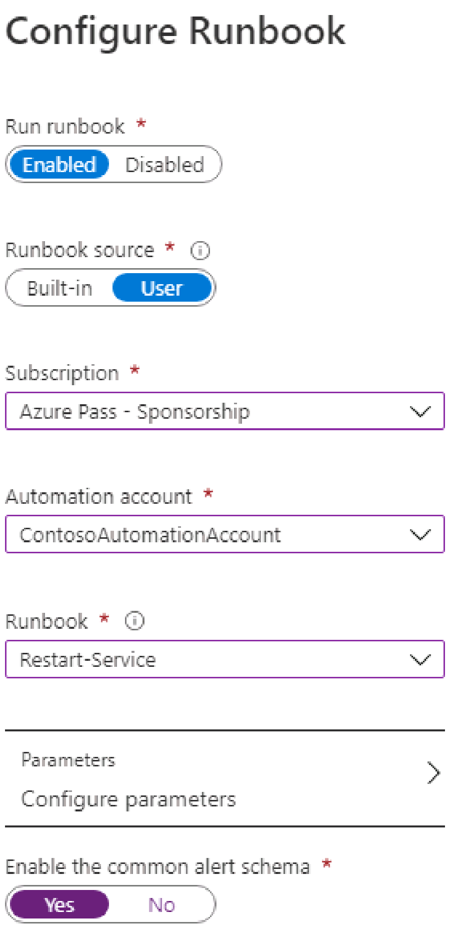
16. In the new blade that opens - enter the details as below.

    - **Action group name**: Default Action Group
    - **Short name**: DAG
    - **Resource group**: ContosoMonitorRG

17. In the **Actions** section add an **Action Name** called *Runbook* and in **Action Type** select **Automation Runbook**

18. In the blade that opens click on **User** under **Runbook source**.

19. Select the Subscription, Automation Account and Runbook as indicated below.



Configure Runbook

Run runbook *: [✓ Enabled] Disabled

Runbook source *: Built-in [✓ User]

Subscription *: Azure Pass - Sponsorship

Automation account *: ContosoAutomationAccount

Runbook *: Restart-Service

Parameters: Configure parameters

Enable the common alert schema *: [✓ Yes] No

20. Click on Ok twice to close both blades.

21. In the **Define Alert Details** section enter the following information.

- **Alert Rule Name:**: Print Spooler Stopped
- **Description**: Detects a stoppage of the print spooler Leave the other settings as they are.

22. Click on **Create alert rule** to create the alert.

Task 3: Test the Alert.

1. Use the **Run Command** feature as before on the virtual machine - and execute the code below to stop the spooler service.

```
Stop-Service spooler -Verbose
Get-Service spooler
```

2. Wait for a few minutes and then in the Automation Account you should see a job for the Restart-Service runbook.

3. If it has completed you can check the status of the service by running the code below. The service should return as started.

```
Get-Service -Name spooler
```

Exercise 4: Use Logic Apps to send custom data to log analytics

Task 1: Setup the pre-requisities

1. Search storage acocunts in the top search bar, click **Storage Accounts** in the dropdown list of services.

2. Click **Create** to create a new storage account.

   - Choose **ContosoResourceGroup** as the resource group.
   - Pick an unique suffix for your storage account, you can use a combination of your initials and date again as a suffix. Prefix storage account with **st** and use your unique suffix, such as **stko24122021**.
   - Choose region, you can use the same region as your resource group.
   - Click on **Review + create** followed by **Create** again.

3. Navigate to the sotrage account that is created.

   - Inside the storage account view, find **Containers** in the left pane.
   - Add a container called **reports**.
   - Inside the same storage account view, find **Access keys** on the left pane.
   - Click on **Show keys**, copy one of the keys and record it to somewhere that you can copy from later on.
   - Similarly, copy the storage account name and record it to the same place.

4. Search **ContosoAutomationAccount** in the top search bar, navigate to your automation account.
5. Find **Runbooks** on the left inside the automation account view, and click on it. Click **Create a runbook**.
   - Name your runbook **Export-UntaggedRGsAsJson**
   - Select **PowerShell** as the runbook type.
   - Select **7** as the runtime version.
   - Click **Create**
6. Copy the below code into the runbook authoring view:

```
Connect-AzAccount -Identity | Out-Null

$untaggedRgs = Get-AzResourceGroup | Where-Object { $_.Tags.Count -eq 0 }

$output = @{
    Found = $false
    UntaggedRGs = @()
    TimeStamp = (Get-Date)
}

if ($unTaggedRgs) {
    $output.UntaggedRGs = $untaggedRgs | Select-Object -Property
ResourceGroupName,ResourceId,Tags
    $output.Found = $true
}
else {
    Write-Host 'Resource groups without tags were not found.'
}
Write-Output ($output | ConvertTo-Json -Depth 5)
```

7. You can test this code by going to **Test pane** and clicking **Start**.
8. **Save** and **Publish** the runbook.

Task 2: Create the Logic App

1. Search **Logic Apps** in the top search bar, click **Logic apps** in the list of available services.
2. Click **Add** to create a new logic app.
   - Choose **ContosoResourceGroup** as the resource group.
   - Select **Type** as **Consumption**
   - Name your logic app **ContosoLogicApp**.
   - Choose region, you can use the same region as your resource group.
   - Click on **Review + create** followed by **Create** again.

3. Navigate to the newly created **ContosoLogicApp**.
    - Click on **Logic app designer** on the left pane.
    - Scroll down to find **Blank Logic App** template in the logic app template browser view. Click on it once you find it.
4. In the logic app designer view, we will now design the **workflow**. Workflow will consist of few **actions** to execute the script **Export-UntaggedRGsAsJson** we have added to the automation account and put the results to the storage account container **reports** that we have created in the task below.
5. First add a schedule based trigger, search for **Recurrence** in the logic app designer search bar. Click on **Schedule** to reveal the available actions and triggers and select **Recurrence** trigger.

   Set the interval to what you desire, or keep it as it is.

6. CLick on **+ New step** in the designer, search for **Azure Automation**, click on **Azure Automation** to view the list of avaialable actions.
7. Add **Create job** action. Set the paramaters accordingly:
    - **Subscription**: Choose the subscription that your automatiion account is in.
    - **Resource Group**: Choose **ContosoResourceGroup** resource group that your automation is in.
    - **Automation Account**: Choose **ContosoAutomationAccount** from the drop down.
    - **Wait for job**: Set to **Yes**.
    - Click on **Add new parameter** dropdown and select **Runbook Name** parameter.
    - Find the **Export-UntaggedRGsAsJson** and set it as the **Runbook Name** parameter value.

8. CLick on **+ New step** in the designer, search for **Azure Automation**, click on **Azure Automation** to view the list of avaialable actions.
9. Add **Get job output** action. Set the paramaters accordingly:
    - **Subscription**: Choose the subscription that your automatiion account is in.
    - **Resource Group**: Choose **ContosoResourceGroup** resource group that your automation is in.
    - **Automation Account**: Choose **ContosoAutomationAccount** from the drop down.
    - Click on the **Job ID** field, from the pop-up menu choose **Job ID** variable. You can trigger the pop-up menu by clicking on **Add dynamic content**.

10. CLick on **+ New step** in the designer to add the last step, search for **Azure Blob Storage**, add the

Create blob action with the following parameters

- **Storage account name**: Use the name of the storage account that you have created before in this exercise, the one where we added **reports** container.
- **Folder path**: Set it to **reports**.
- **Blob name**: Click on **Add dynamic content**, in the expression tab search for **utcNow()**. Click **OK** to add the function to the text field, append the name with **.json**.
- **Blob content**: Click on **Add dynamic content**, search for **Content** to add the data from the previous action that gets the automation job output.

11. Click on the **Save** button in the logic app designer to save your logic app.
12. Click on **Run Trigger->Run** to kick-off your logic app on-demand.
13. If your logic app executes successfuly, you will have green ticks on all actions. If it fails, you can click on actions to further drill down to view error details.
14. Click on the last action **Create blob** to view the **Blob content** field in the logic app designer. You should see the JSON output from the runbook.