

BI4Dynamics Process Automation

How to automatically update data from BC Cloud, Data Lake to Analysis Services

BI server is on Azure VM

Last update: April 2023

Version 2.0

Revision 1.2

Contents

1	Process Automation #1 – Start Container Instance	3
1.1	Introduction	3
1.2	Prerequisites.....	3
1.3	Setup Logic App	4
1.4	Test logic app	7
2	Process Automation #2 – Start and Stop Virtual Machine.....	8
2.1	Start Virtual Machine.....	8
2.2	Deallocate Virtual Machine	9
3	Process Automation #3 – Start SQL server Agent (VM)	11
3.1	Enable SQL Server agent.....	11
3.2	Setup SQL Server Agent	12
4	Process Automation #4 – Start and Stop Azure Analysis Services	13
4.1	Prerequisites	13
4.2	Instructions.....	13
5	Process Automation - Timing Schedule.....	19

1 Process Automation #1 – Start Container Instance

1.1 Introduction

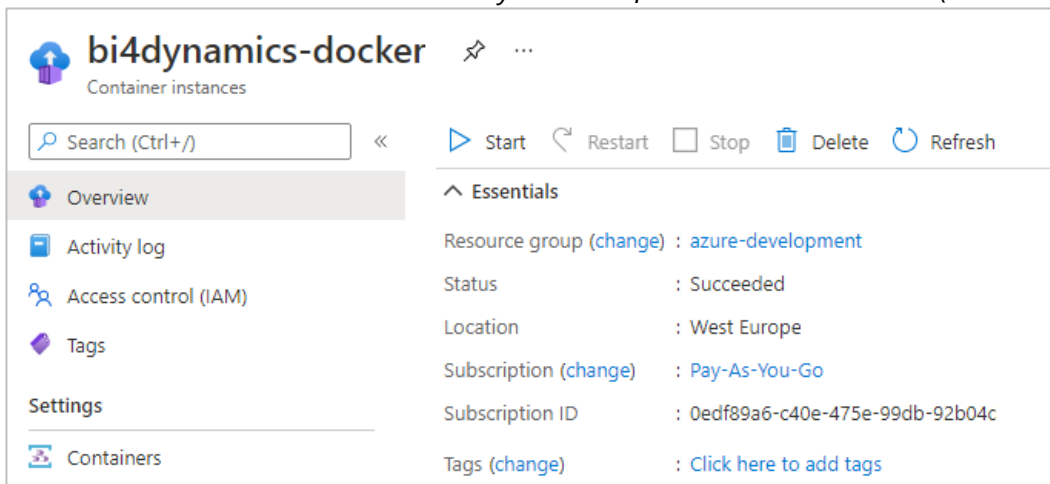
This automation process is for an Azure Container instance (Docker), which is a light virtual machine, based on BI4Dynamics image. These Container instances are used for running table export from BC to Blob storage. Through Logic apps docker will run on a scheduled day and time. Logic app will automatically start and terminate the docker after finishing the export.

1.2 Prerequisites

For this step you will need:

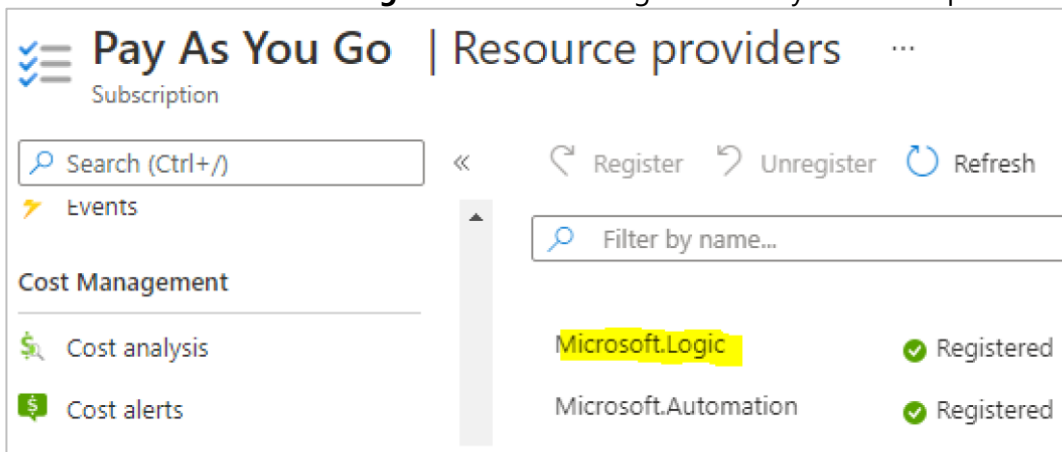
- Container instance

Creation described in document *"BI4Dynamics Infrastructure Installation (Azure VM + Azure resources)"*.



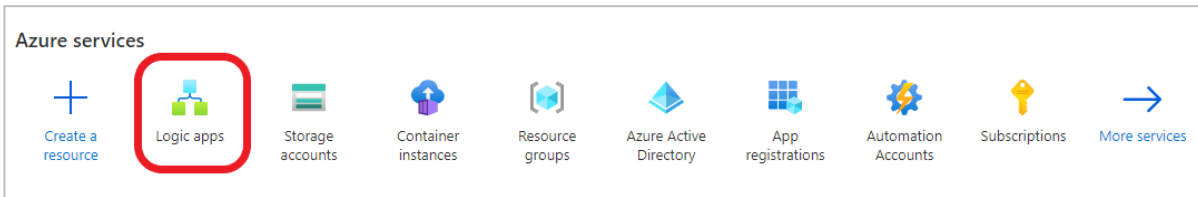
- Logic Apps (to be setup here).

Make sure that **Microsoft.Logic** resources are registered for your subscription.

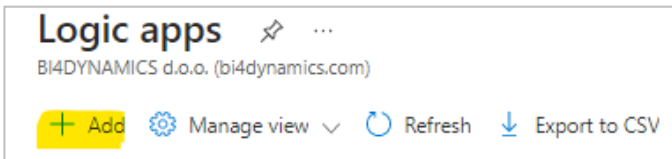


1.3 Setup Logic App

Search for **Logic Apps** in Azure.

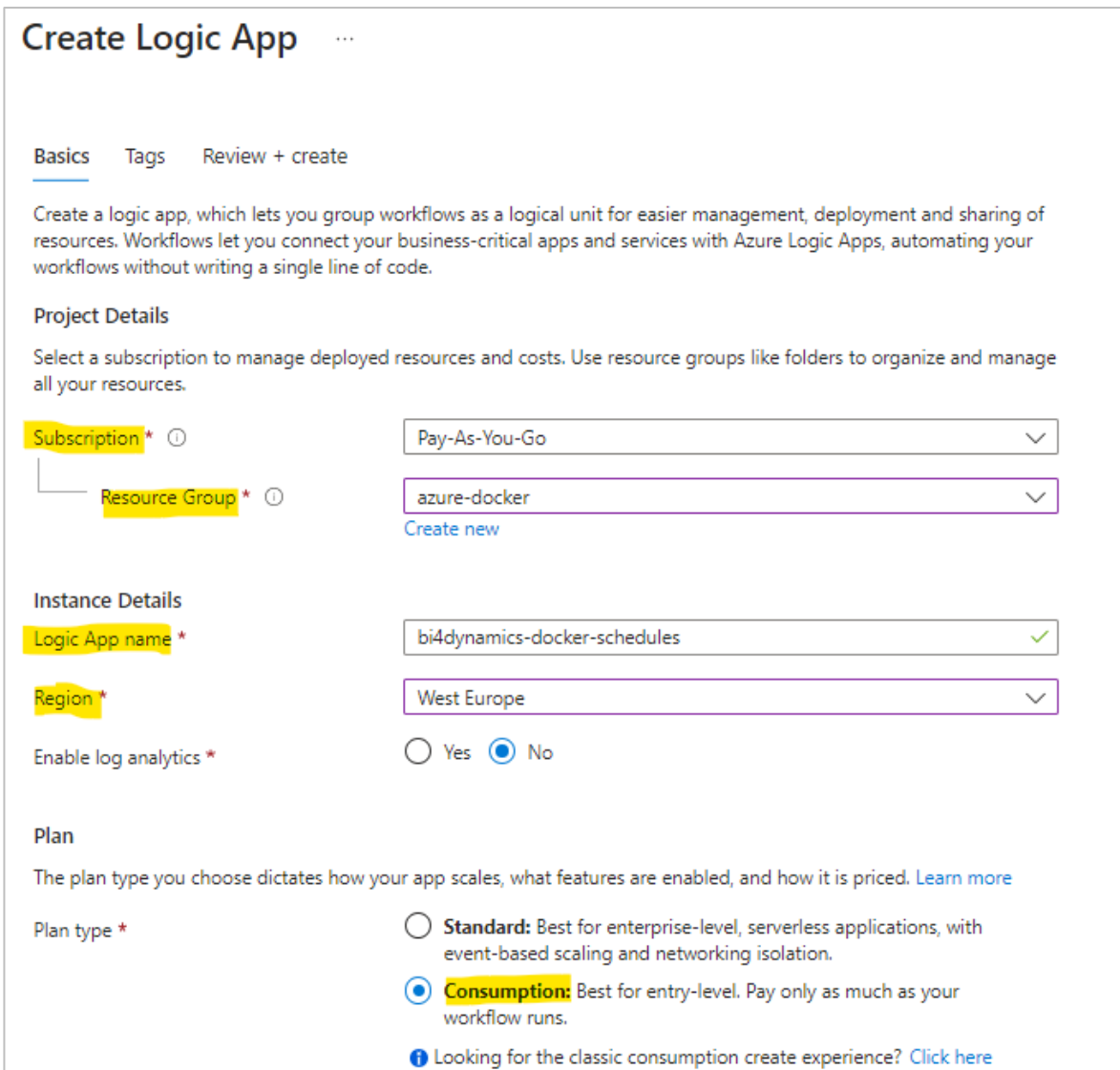


Add a new Logic App.



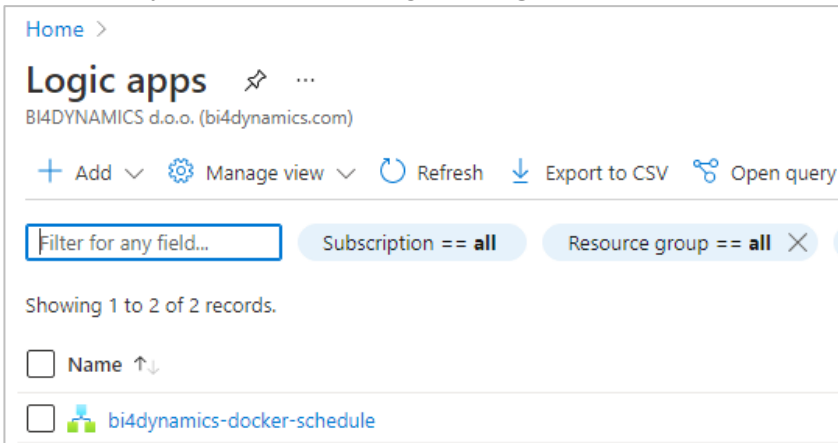
Enter **Subscription**, **Resource Group** and create a meaningful **Name** for your logic app. **Select** the Region and choose **Consumption** as a Plan type.

Click **Review + create** and select **Create** in the next window.

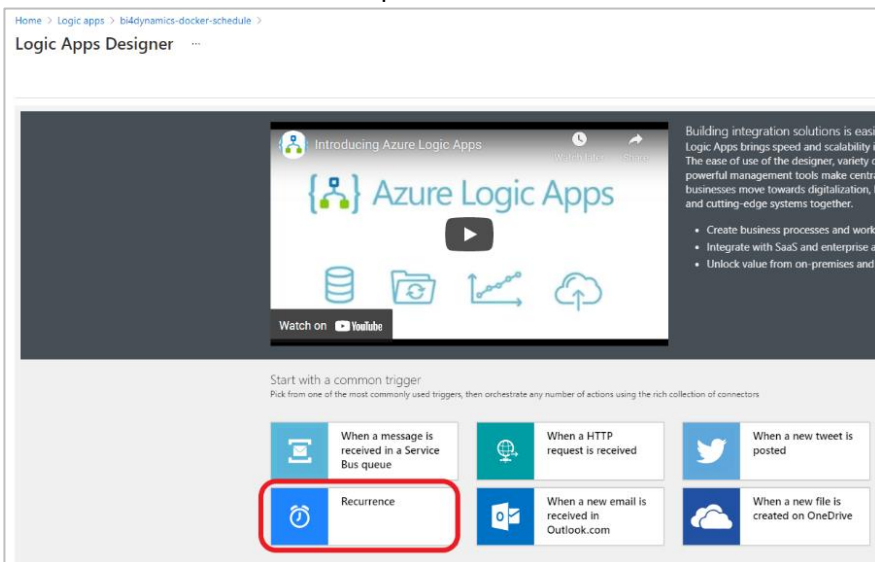
A screenshot of the 'Create Logic App' wizard in the Azure portal. The 'Basics' tab is selected. The page contains the following fields and options:

- Project Details:**
 - Subscription ***: Pay-As-You-Go
 - Resource Group ***: azure-docker (with a 'Create new' link below)
- Instance Details:**
 - Logic App name ***: bi4dynamics-docker-schedules
 - Region ***: West Europe
 - Enable log analytics ***: No (selected)
- Plan:**
 - Plan type ***: Consumption (selected). Description: Best for entry-level. Pay only as much as your workflow runs.
 - Standard: Best for enterprise-level, serverless applications, with event-based scaling and networking isolation.
 - Looking for the classic consumption create experience? [Click here](#)

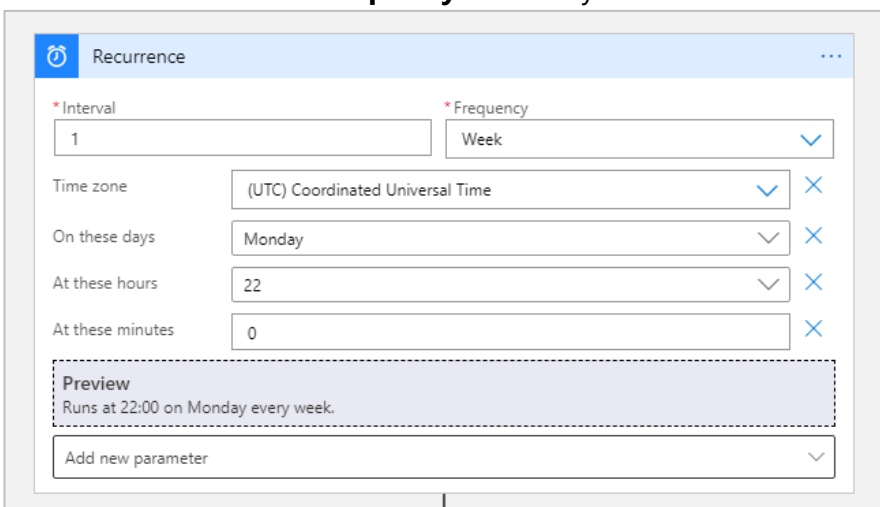
Once deployment is complete, go to **Logic apps** and open the newly created application.



Logic apps designer will open with premade templates to use. Select **Recurrence** in the template or search for it in the search dialog.

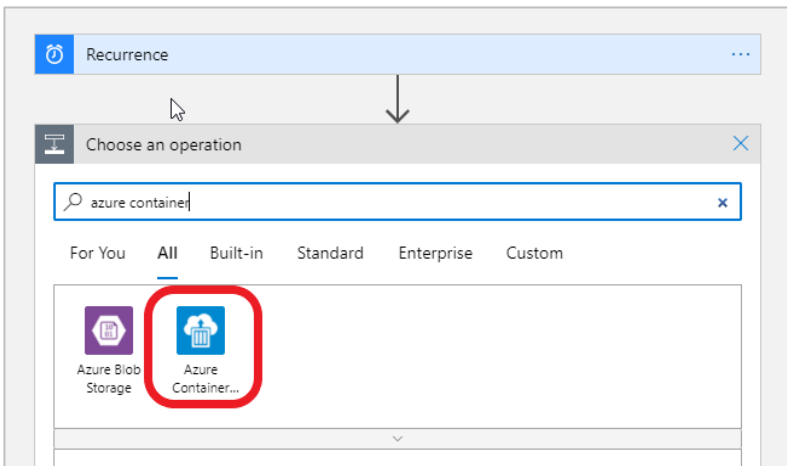


Select the **Interval** and **Frequency** at which you decide the docker should be run.

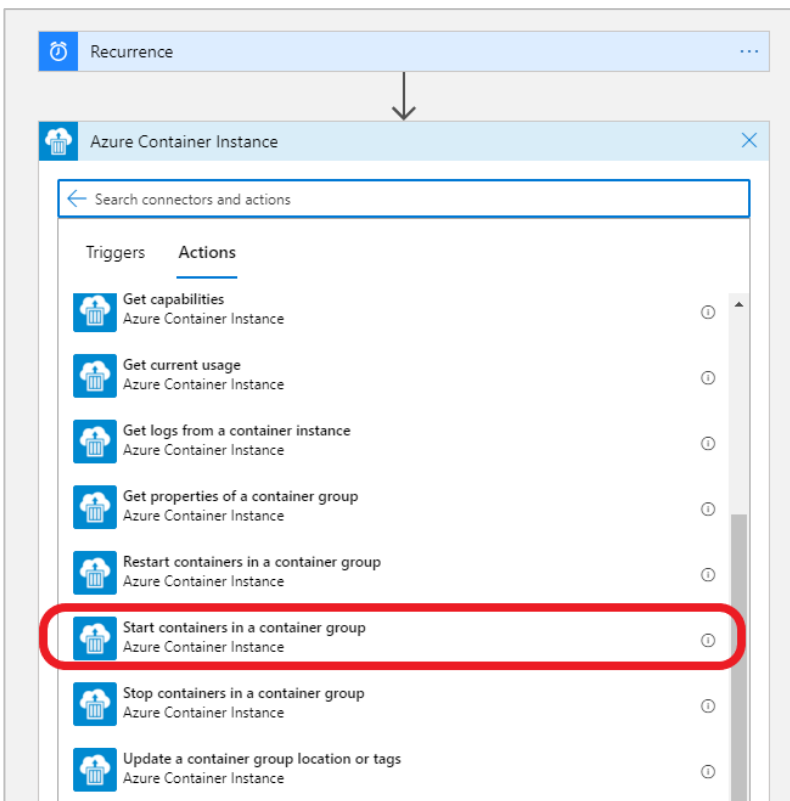


Note: If the selected Frequency is **Week**, you can add new parameters which set the days, hours, and minutes when the Virtual Machine should start.

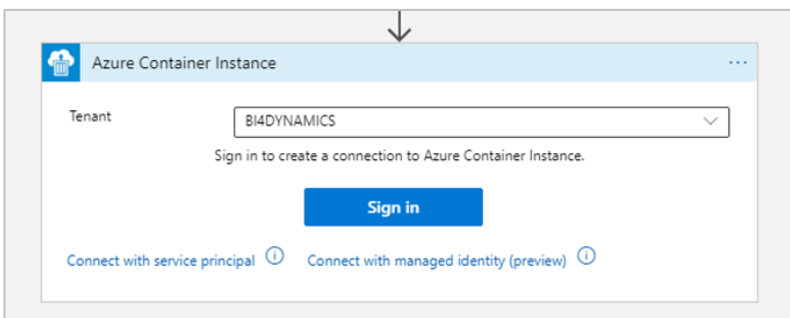
Click + **New step**, search for **Azure Container instance** and select it.



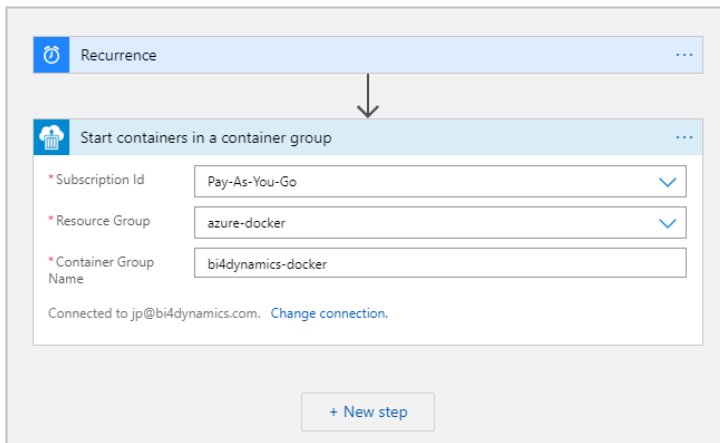
In the drop-down menu select Start containers in a container group.



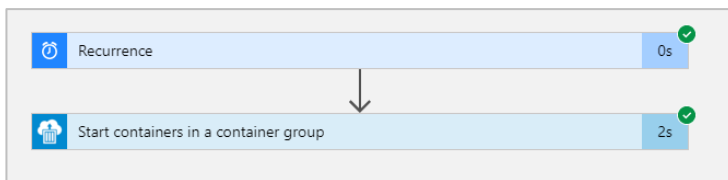
Sign into your **Tenant**.



Enter your **Subscription ID**, **Resource Group** and **Container Group Name** (docker).

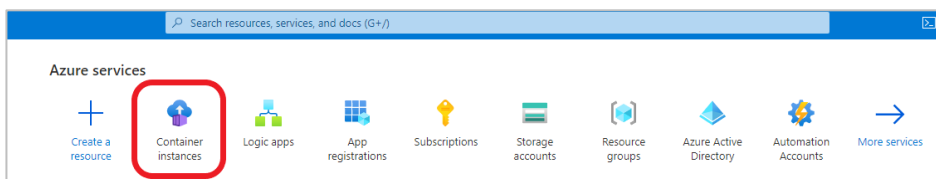


Click **Save** in the top left of the designer and press **Run** to test if the application is working correctly.

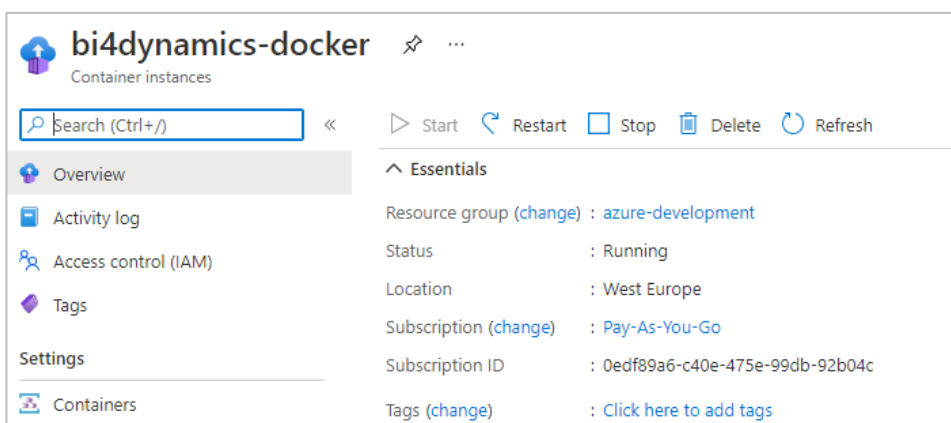


1.4 Test logic app

Go to Container instances.



Check the selected container instance (docker) if it is being **Created** or already **Running**.



You have successfully created and tested a logic app that automatically starts container instance.

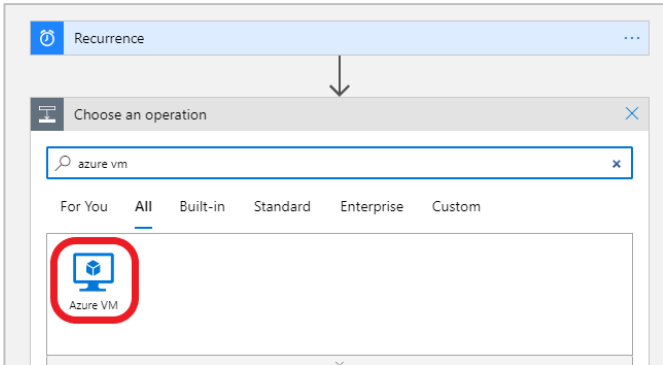
2 Process Automation #2 – Start and Stop Virtual Machine

Process automation for Virtual Machine on azure is very similar to container instance automation

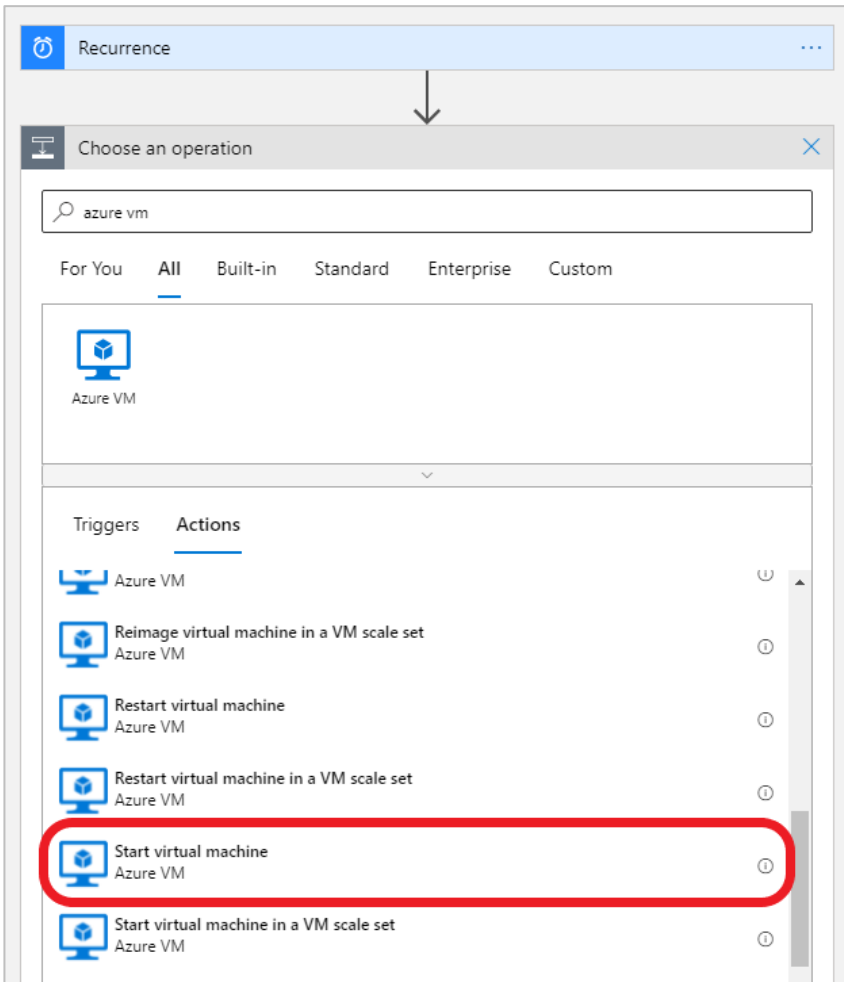
Note: We will create two logic apps, one for starting the VM and one for deallocating(stopping) it.

2.1 Start Virtual Machine

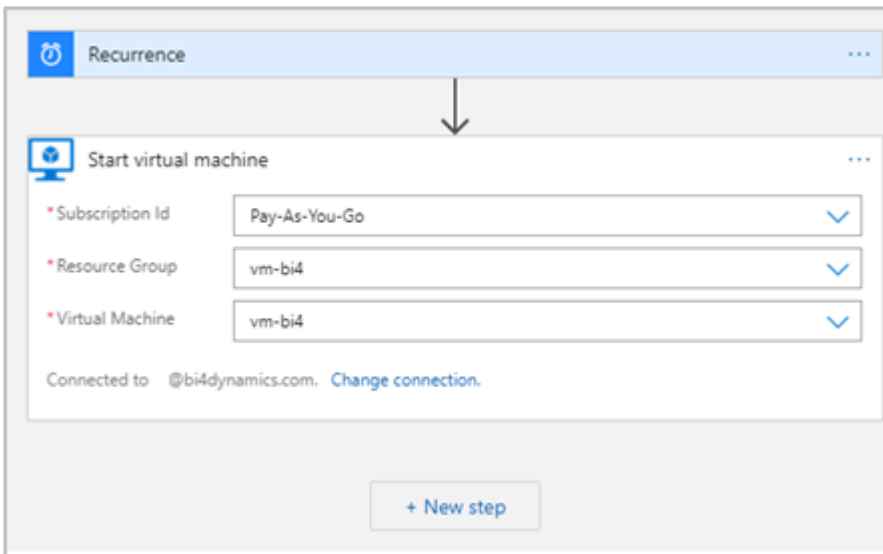
Search for **Azure VM** in search dialog and select it.



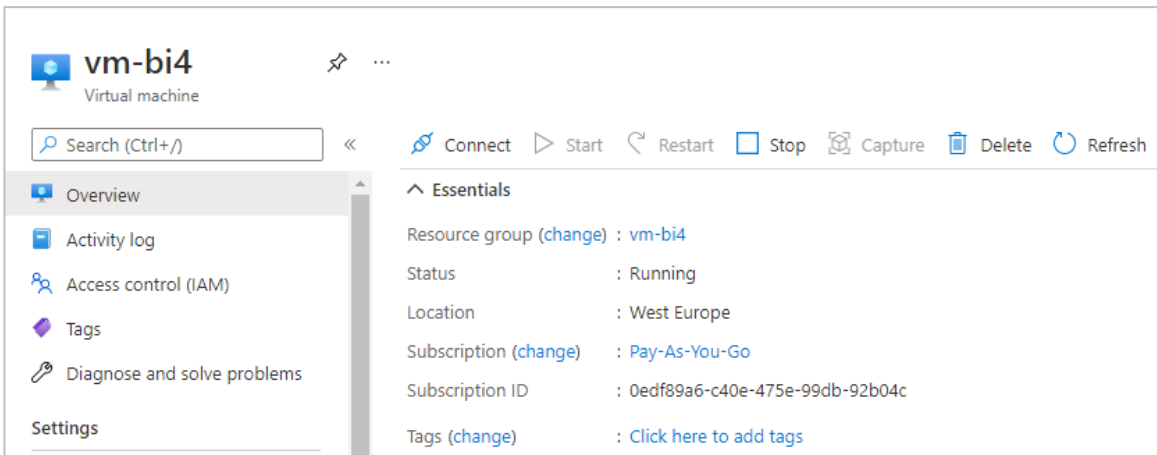
Select Start virtual machine option.



Insert values for **Subscription ID**, **Resource Group** and Virtual Machine name.



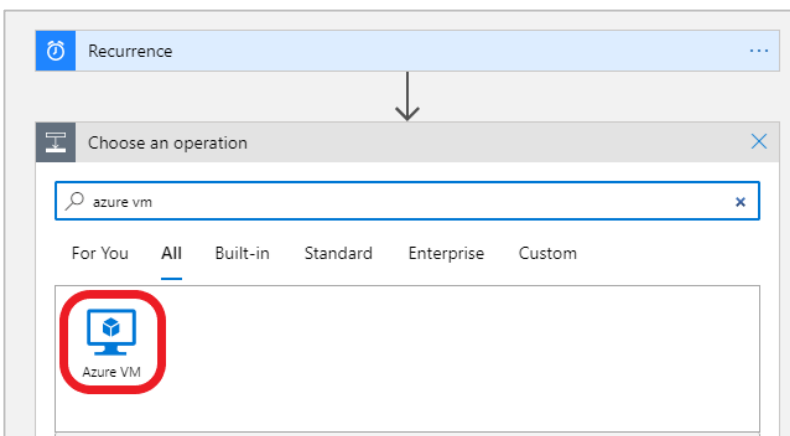
Next step is to **Save** and **Run** the application and go to **Virtual Machines** to check if it is **Running**.



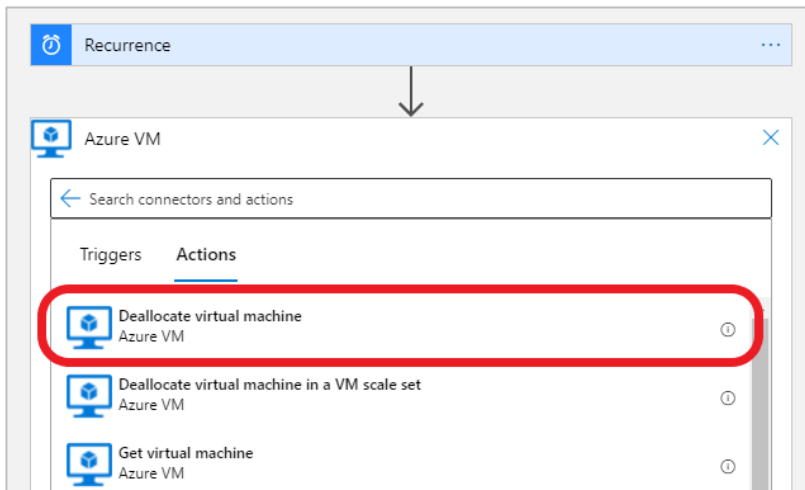
2.2 Deallocate Virtual Machine

Process automation for Virtual Machine Deallocation is almost identical to Start VM Logic app. First select the scheduled time when the VM should stop.

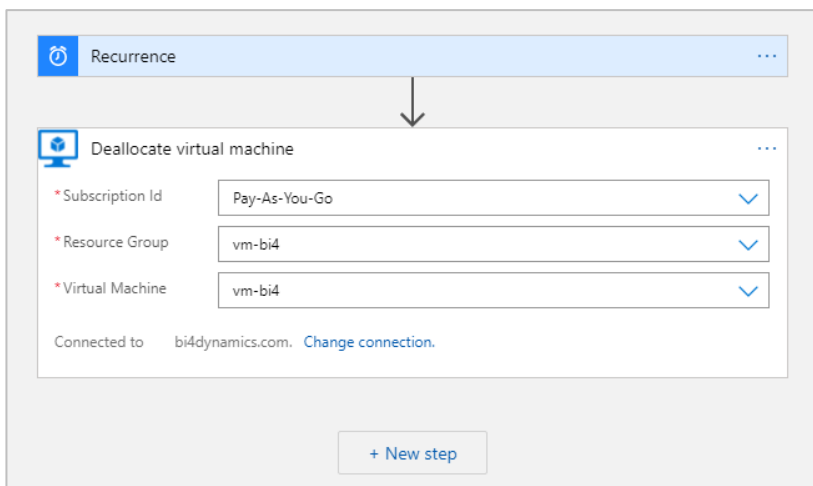
In **Logic Apps** select **Recurrence**, add a **new step** and search for **Azure VM**.



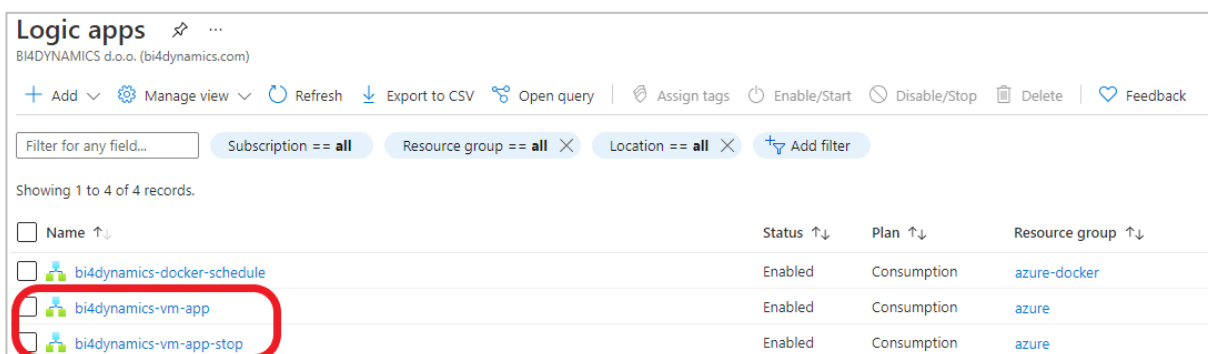
Next select Deallocate virtual machine option.



Insert values for Subscription id, Resource group and Virtual Machine name.



Click **Save** and exit Logic Apps Designer. Go to **Logic Apps** and check for apps VM start and VM stop.



To check if Logic apps are properly working first run the start VM app, after the Virtual machine is running, run the stop VM app and check if it is allocated.

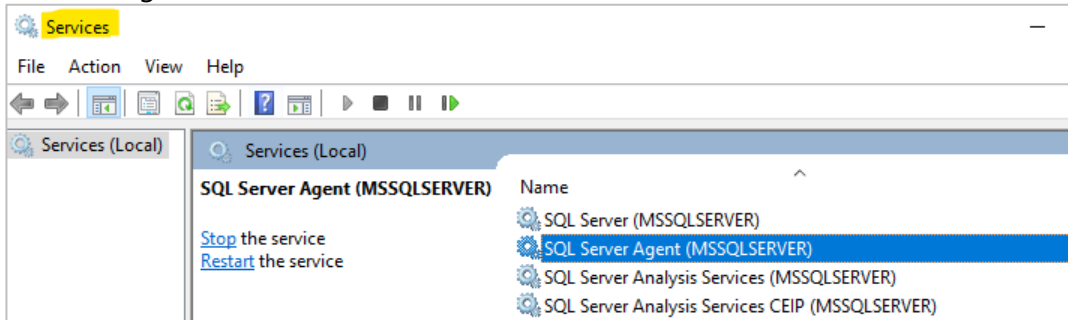
You have now successfully created a logic app that automatically starts the virtual machine at specified times and a logic app that automatically stops(deallocates) the virtual machine at specified times.

3 Process Automation #3 – Start SQL server Agent (VM)

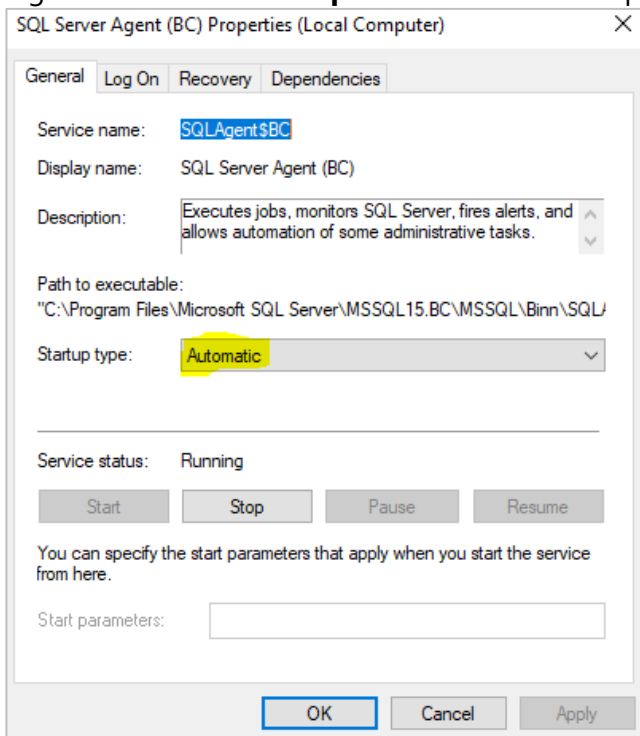
When Virtual machine is running, it is ready to process data. This process is triggered by SQL Server Agent feature, a part of SQL server.

3.1 Enable SQL Server agent

Go to **Services** and find the **SQL Server Agent** service. If you are using newly created Virtual Machine, it will probably be the only SQL Server Agent, but if you are running more SQL server engines, there may be more Agents.



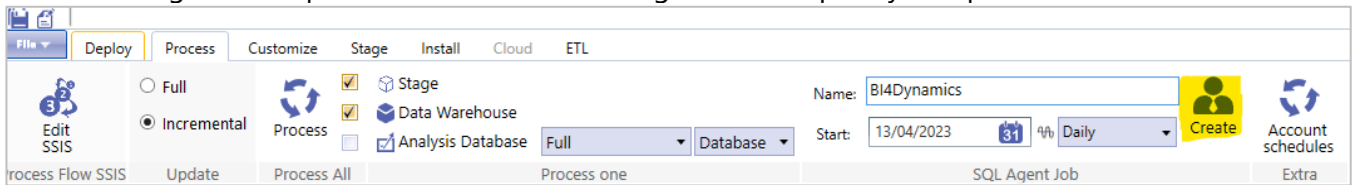
Right click and select **Properties** and set Start-up Type to Automatic.



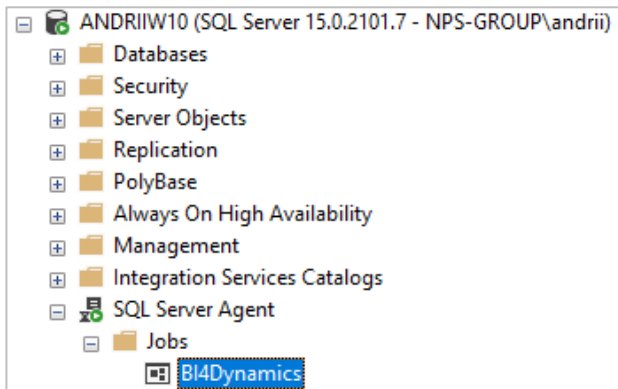
Note: make sure that user running service is a domain admin user (not a service) and has permissions needed to process data warehouse and analysis services. On VM this would be the VM admin user.

3.2 Setup SQL Server Agent

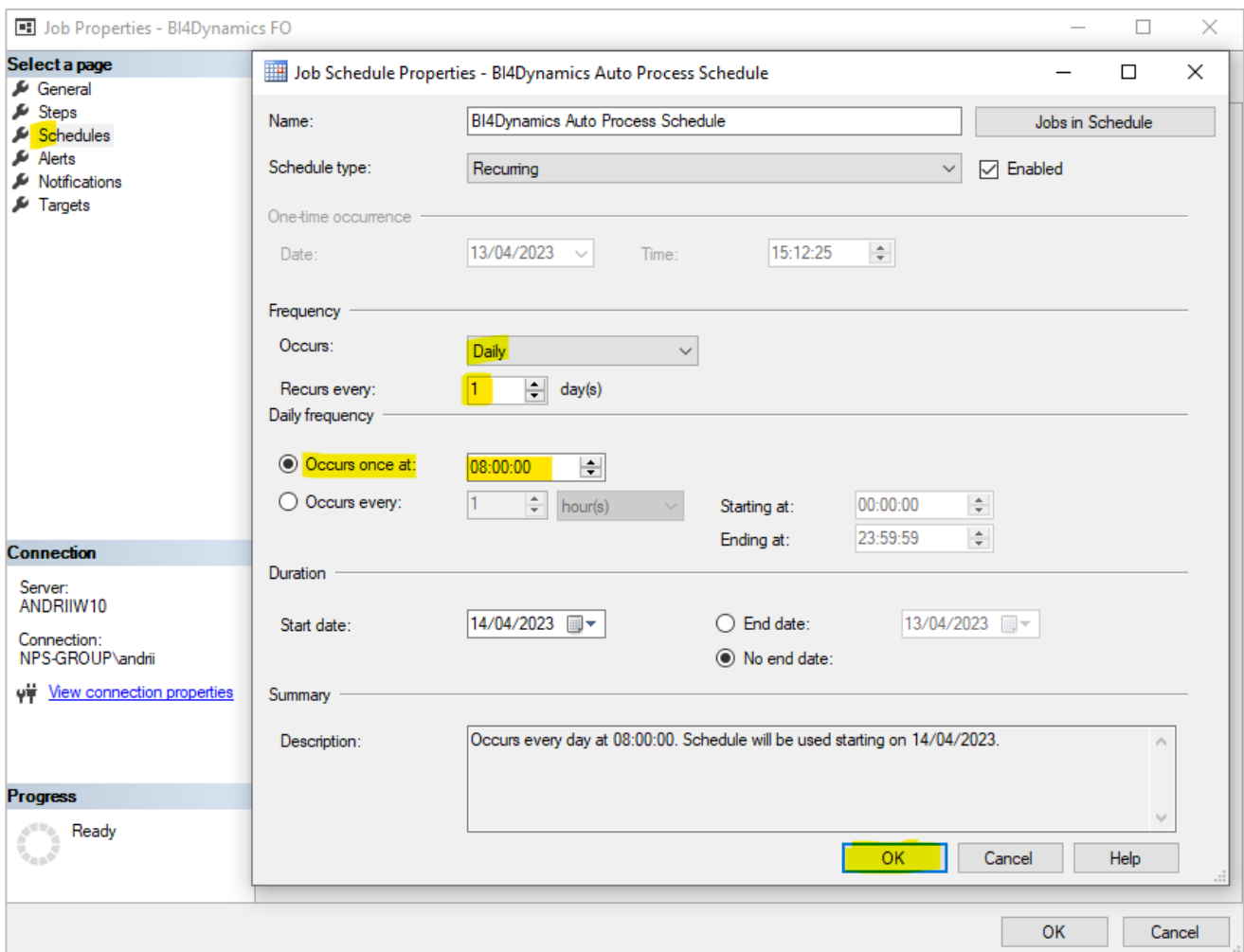
SQL Server agent conducts processing of stage, data warehouse and analytics, bringing new data to users. To set SQL Agent Job open **Process** tab, set SQL Agent Job frequency and press **Create**:



Open SQL Server Management Studio, navigate to **Properties** of the Agent Job that you have created.



Press **Schedules**, click on **Edit**. Set properties for Job Agent and press **OK**.

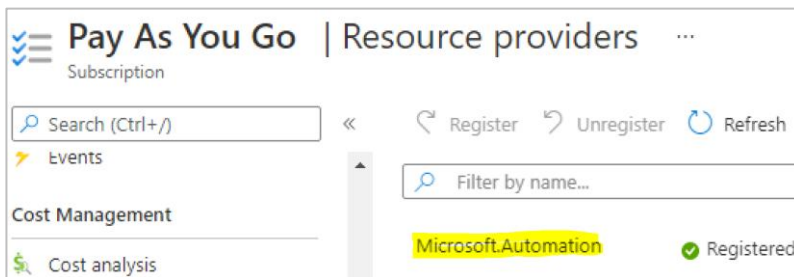


4 Process Automation #4 – Start and Stop Azure Analysis Services

This part of documentation is intended to explain the process of scheduling the work of Azure Analysis Services. It will allow the Analysis Services to start and stop on scheduled days and time based on the business requirements of the end-users.

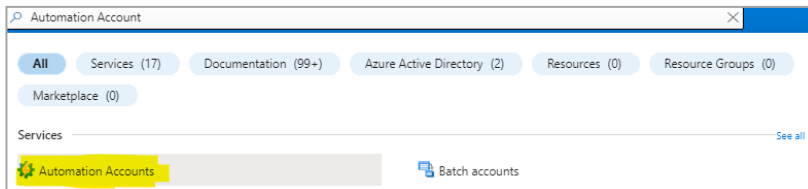
4.1 Prerequisites

- Azure Analysis Services: creation described in document *“Application Installation (Azure VM)”*.
- Registered resource **Microsoft.Automation** for the subscription.



4.2 Instructions

Go to the Azure Portal and search for **Automation Accounts**:



Create a new **Automation Account** under the same **Subscription** and in the same **Resource Group** and **Region** as the **Analysis Services**:

Home > Automation Accounts >

Create an Automation Account

Basics | Advanced | Networking | Tags | Review + Create

Create an Automation Account to hold the Automation runbooks & configuration used for automating operations and management tasks around Azure and non-Azure resources. You could execute cloud jobs in a serverless environment or use hybrid jobs on your compute via Azure Virtual machines, Arc-enabled servers or Arc-enabled VMWare VM (preview). [Learn more](#)

Subscription * ⓘ Pay-As-You-Go

Resource group * ⓘ Select a resource group
[Create new](#)

Instance Details

Automation account name * ⓘ BI4DynamicsAutomation ✓

Region * ⓘ West Europe

In the tab **Advanced**, select **System Assigned Managed Identity**:

Create an Automation Account

Basics Advanced Networking Tags Review + Create

Managed Identities

Use Managed Identities as the recommended method for authenticating with Azure resources from the runbooks. Managed identity would be more secure than Runas account since it doesn't require any credentials to be stored. [Learn more](#)

System assigned

User assigned

Once the **Automation Account** is created, go to **Account Settings > Identity**. In the tab **System Assigned**, make sure that the **Status** is set to **On** and click on **Azure Role Assignments**:

BI4DynamicsAutomation | Identity

Automation Account

Search

- Python packages
- Credentials
- Connections
- Certificates
- Variables

Related Resources

- Linked workspace
- Event grid
- Start/Stop VM

Account Settings

System assigned User assigned

A system assigned managed identity is restricted to one per resource and is tied to the lifecycle in code. [Learn more about Managed identities.](#)

Save Discard Refresh Got feedback?

Status On

Object (principal) ID

Permissions

Add a new role assignment to the **System Managed Identity**:

Azure role assignments

+ Add role assignment (Preview) Refresh

If this identity has role assignments that you don't have permission to read, they won't be shown in the list. [Learn more](#)

Under the option **Scope** select the option **Resource Group**.

Specify the **Subscription** and **Resource Group** in which Analysis Services are located.

Under the option **Role** select **Contributor**.

Add role assignment (Preview) [X]

Scope ⓘ
Resource group [v]

Subscription [v]

Resource group ⓘ
BI4Dynamics [v]

Role ⓘ
Contributor [v]

[Learn more about RBAC](#)

Once role assignment is added, navigate to **Process Automation > Runbooks** and create a new one.

Give to a new **Runbook** a meaningful name such as **Start_Stop_AAS**.

In the **Runbook Type** select the option **PowerShell**.

In the **Runtime Version** select the option **5.1**.

Create a runbook ...

Name * ⓘ Start_Stop_AAS ✓

Runbook type * ⓘ PowerShell [v]

Runtime version * ⓘ 5.1 [v]

Description [v]

i During runbook execution, PowerShell modules targeting 5.1 runtime version will be used. Please make sure the required PowerShell modules are present in 5.1 runtime version.

Important: Different **Runtime Version** might lead to the error in the execution of the script. Syntax for authentication might differ between PowerShell versions.

Once the Runbook is created, you will be navigated to the edit view of the Runbook:

Edit PowerShell Runbook ...

Start_Stop_AAS

Save Publish Revert to published Test pane Feedback

> CMDLETS 1

Insert the following script to the command lines:

```

# Parameters
[CmdletBinding()]
param(
    [Parameter(Mandatory=$True,Position=0)]
    [ValidateSet('Start','Stop')]
    [string]$AasAction,

    [Parameter(Mandatory=$True,Position=1)]
    [ValidateLength(1,100)]
    [string]$ResourceGroupName,

    [Parameter(Mandatory=$True,Position=2)]
    [ValidateLength(1,100)]
    [string]$AnalysisServerName
)
# Keep track of time
$StartDate=(GET-DATE)
# Log in to Azure with AZ (standard code)
Write-Verbose -Message 'Connecting to Azure'
# Name of the Azure Run As connection
$ConnectionName = 'AzureRunAsConnection'
try {
    $AzureContext = (Connect-AzAccount -Identity).context
}
catch{
    Write-Output "There is no system-assigned user identity. Aborting.";
    exit
}
# Getting the AAS for testing and logging purposes
$myAzureAnalysisServer = Get-AzAnalysisServicesServer -ResourceGroupName $ResourceGroupName -
Name $AnalysisServerName
if (!$myAzureAnalysisServer)
{
    Write-Error "$($AnalysisServerName) not found in $($ResourceGroupName)"
    return
}
else
{
    Write-Output "Current status of $($AnalysisServerName): $($myAzureAnalysisServer.State)"
}
# Check for incompatible actions
if (($AasAction -eq "Start" -And $myAzureAnalysisServer.State -eq "Succeeded") -Or ($AasAction -
eq "Stop" -And $myAzureAnalysisServer.State -eq "Paused"))
{
    Write-
Error "Cannot $($AasAction) $($AnalysisServerName) while the status is $($myAzureAnalysisServer.State
)"
    return
}
# Resume Azure Analysis Services
elseif ($AasAction -eq "Start")
{
    Write-Output "Now starting $($AnalysisServerName)"
    $null = Resume-AzAnalysisServicesServer -ResourceGroupName $ResourceGroupName -
Name $AnalysisServerName
}
# Pause Azure Analysis Services
else
{
    Write-Output "Now stopping $($AnalysisServerName)"
    $null = Suspend-AzAnalysisServicesServer -ResourceGroupName $ResourceGroupName -
Name $AnalysisServerName
}
# Show when finished
$Duration = NEW-TIMESPAN -Start $StartDate -End (GET-DATE)
Write-
Output "Done in $($[int]$Duration.TotalMinutes) minute(s) and $($[int]$Duration.Seconds) second(s)"

```


After that got to the **Test Pane** and fill in the required parameters:

- Under the option **AASACTION** write **Start** (if the Analysis Services are turned on - **Stop**).
- Under the option **RESOURCEGROUPNAME** insert the name of the **Resource Group**.
- Under the option **ANALYSSERVERNAME** insert the name of **Azure Analysis Services**.

Once the parameters are inserted, click on **Start**:

The screenshot shows the 'Test Pane' interface. At the top, there are buttons for 'Start' (highlighted in yellow), 'Stop', 'Suspend', 'Resume', 'View last test', and 'Refresh job streams'. Below these are three parameter input fields, each with a yellow highlight on the label: 'AASACTION *', 'RESOURCEGROUPNAME *', and 'ANALYSSERVERNAME *'. Each field contains the text 'Enter a value' and is labeled as 'Mandatory, String'. To the right of the input fields, there is a black box with blue text that reads: 'Click 'Start' to begin the test run. Streams will display when the test completes.'

Once the command was executed successfully, go to the Edit panel, **Save** and **Publish** the runbook:

The screenshot shows the 'Edit PowerShell Runbook*' interface. At the top, there are buttons for 'Save' (highlighted in yellow), 'Publish' (highlighted in yellow), 'Revert to published', 'Test pane', and 'Feedback'. Below these are three navigation options: 'CMDLETS', 'RUNBOOKS', and 'ASSETS'. On the right side, there is a code editor showing the following PowerShell script:

```

1 # Parameters
2 [CmdletBinding()]
3 param(
4     [Parameter(Mandatory=$True,Position=0)]
5     [ValidateSet('Start','Stop')]
6     [string]$AasAction,
7

```

Once it is published, click on the option **Link to the schedule**.

There you will need to set up the schedule and parameters for the Runbook Execution:

The screenshot shows the 'Schedule Runbook' interface. At the top, there is a clock icon and the text 'Schedule Runbook' with a sub-label 'Start_Stop_AAS'. Below this, there are three sections: 'Schedule' with the text 'Link a schedule to your runbook', 'Parameters and run settings' with the text 'Configure parameters and run settings', and a third section that is currently empty.

Parameters and run settings for the schedule should be configured similarly to the previous step. Only **AASACTION** will differ based on the command of the schedule (Start or Stop).

Note:
The automation schedules for Start and Stop commands must be created separately.

In the schedule settings provide a name to the new schedule. We suggest giving it a name corresponding to the executed command (Start or Stop). Also, you can provide additional details in the Description.

Important:
Do not forget to specify the correct **Time Zone** according to which the time of automation will be scheduled.

As a next step, change **Recurrence** from Once to Recurring and set up **Recur every** option to once a Day or Week. In case of week, the schedule can be set up at the specific days of the week so that Saturday and Sunday could be excluded from the automation as on the screenshot on the right.

The final schedules should look as following:

Name	Next run	Time zone	Status
Start	4/12/2023, 7:00 AM	Central European Time	✓ On
Stop	4/12/2023, 5:00 PM	Central European Time	✓ On

New Schedule ✕

Name *

Description

Starts *

Time zone

Recurrence

Recur every *

On these days Monday
 Tuesday
 Wednesday
 Thursday
 Friday
 Saturday
 Sunday

Set expiration

The execution of the schedules can be monitored in the **Process Automation > Jobs** tab:

Runbook	Job created	Status	Ran on	Last status update
Start_Stop_AS	4/11/2023, 6:00:17 AM	✓ Completed	Azure	4/11/2023, 6:03:12 AM
Start_Stop_AS	4/10/2023, 6:00:17 PM	✓ Completed	Azure	4/10/2023, 6:01:52 PM
Start_Stop_AS	4/10/2023, 6:00:30 AM	✓ Completed	Azure	4/10/2023, 6:03:53 AM
Start_Stop_AS	4/9/2023, 6:00:21 PM	✓ Completed	Azure	4/9/2023, 6:03:31 PM
Start_Stop_AS	4/9/2023, 6:00:20 AM	✓ Completed	Azure	4/9/2023, 6:04:04 AM

5 Process Automation - Timing Schedule

Here is an example of processing schedule for daily update:

Step	Step description	Start Time	Duration	Comment
1	Start Container instance	22:00	45 min	<p>BC export to data lake can run any time after BC users are finishing their daily work.</p> <p>This process time can vary 30% (!) day by day, exporting same amount of data, in the after-office hours when no-one is using BC.</p> <p>Keep enough buffer time for next step.</p>
2	Start Virtual Machine	07:00	2-3 min	VM hosts data warehouse that must be ready when DW processing start
3	Start Azure Analysis Services	07:00	2-3 min	Azure AS must be ready when DW processing start
4	Start SQL Server Agent	07:15	20 min	DW processing (data are in Azure AS)
5	Stop Virtual Machine	08:00		Leave some buffer time after DW is processed and then stop VM.
6	Stop Azure Analysis Services	17:00		AAS will run during business hours when users are querying data.