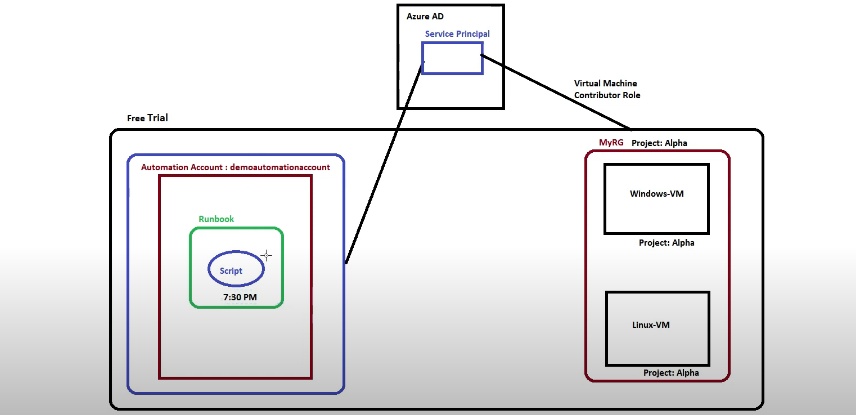
In Azure, a **Runbook** is a set of tasks and processes that you can automate using **Azure Automation**. Runbooks allow you to manage resources efficiently, reduce manual effort, and ensure consistent execution of workflows.

**Scenario: Automated VM Shutdown and Startup**

1. Automate Azure VM Start and Stop Using Azure Runbook  
  
    •    Scenario: You manage a set of Azure virtual machines (VMs) that are used only during business hours (9 AM to 6 PM). To save costs, you want to automate the start of the VMs at 9 AM and the stop at 6 PM.  
    •    Problem Statement: Write an Azure Runbook script that automates the start and stop of a specific set of VMs based on their tags. Schedule this runbook to execute at 9 AM for starting the VMs and 6 PM for stopping them.

**1. Automate Azure VM Start and Stop Using Azure Runbook**

**Step 1: Create an Azure Automation Account**

1. Go to the **Azure Portal**.
2. Search for **Automation Accounts** and click **+ Create**.
3. Provide:
   * Name: VMStartStopAutomation
   * Resource Group: Select/Create a group.
   * Location: Choose your region.
   * Click **Create**.

**Step 2: Create the Runbook**

1. Navigate to the Automation Account.
2. Under **Process Automation**, select **Runbooks** > **+ Create a Runbook**.
3. Enter:
   * Name: StartStopVMsByTags
   * Type: **PowerShell**
   * Description: "Automates starting and stopping VMs based on tags."

**Assign Role to Managed Identity (or Service Principal):** Follow these steps to assign a role to your **Managed Identity** or **Service Principal**:

**Step 3.Assign Role via Azure Portal:**

1. **Navigate to Azure Active Directory > Managed Identities** (if using Managed Identity) or **Azure Active Directory > App Registrations** (if using a Service Principal).
2. **Select the Managed Identity** or **Service Principal** you're using for authentication.
3. Go to the **"Roles and Administrators"** section.
4. Click **"+ Add Assignments"** to assign a role.
5. Choose **Contributor** or **Owner** (depending on your needs) and assign it at the subscription or resource group level.

### **Important Notes:**

* **Role Assignment Scope**: Ensure that the role is assigned to the appropriate scope (either **Subscription** or **Resource Group**). Assigning the role at the **subscription level** ensures that the managed identity has access to all resources within that subscription.
* **Managed Identity**: If you are using a **System-Assigned Managed Identity** (for example, for an Azure Automation Runbook), make sure that the identity has the necessary permissions assigned at the correct scope.

***Script:***

param (

[string]$tagName = "project",

[string]$tagValue = "Alpha"

)

# Authenticate to Azure

Connect-AzAccount -Identity

# Get all the VMs in the subscription

$vms = Get-AzVM

# Get current time

$currentTime = Get-Date

$currentHour = $currentTime.Hour

# Check if it's time to start or stop the VMs

if ($currentHour -ge 9 -and $currentHour -lt 18) {

Write-Output "It's between 9 AM and 6 PM. Starting VMs tagged with $tagName=$tagValue."

# Get all VMs with the specified tag

$vmsToStart = $vms | Where-Object { $\_.Tags[$tagName] -eq $tagValue -and $\_.PowerState -eq 'VM stopped' }

# Start VMs that are stopped

foreach ($vm in $vmsToStart) {

Write-Output "Starting VM: $($vm.Name)"

Start-AzVM -ResourceGroupName $vm.ResourceGroupName -Name $vm.Name

}

} elseif ($currentHour -ge 18 -or $currentHour -lt 9) {

Write-Output "It's outside business hours (6 PM to 9 AM). Stopping VMs tagged with $tagName=$tagValue."

# Get all VMs with the specified tag

$vmsToStop = $vms | Where-Object { $\_.Tags[$tagName] -eq $tagValue -and $\_.PowerState -eq 'VM running' }

# Stop VMs that are running

foreach ($vm in $vmsToStop) {

Write-Output "Stopping VM: $($vm.Name)"

Stop-AzVM -ResourceGroupName $vm.ResourceGroupName -Name $vm.Name -Force

}

}

Write-Output "Runbook execution completed."

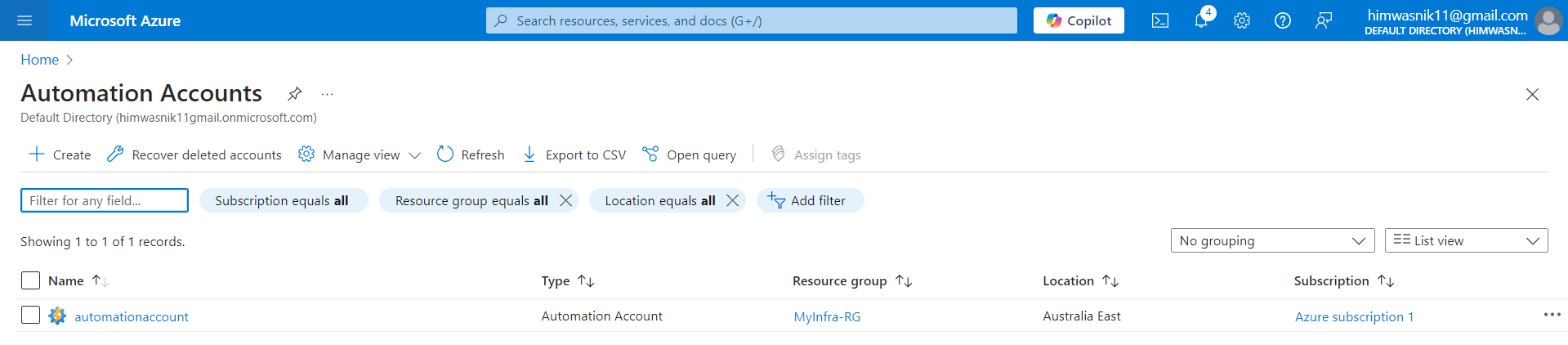
# Save the script and click **Publish**.

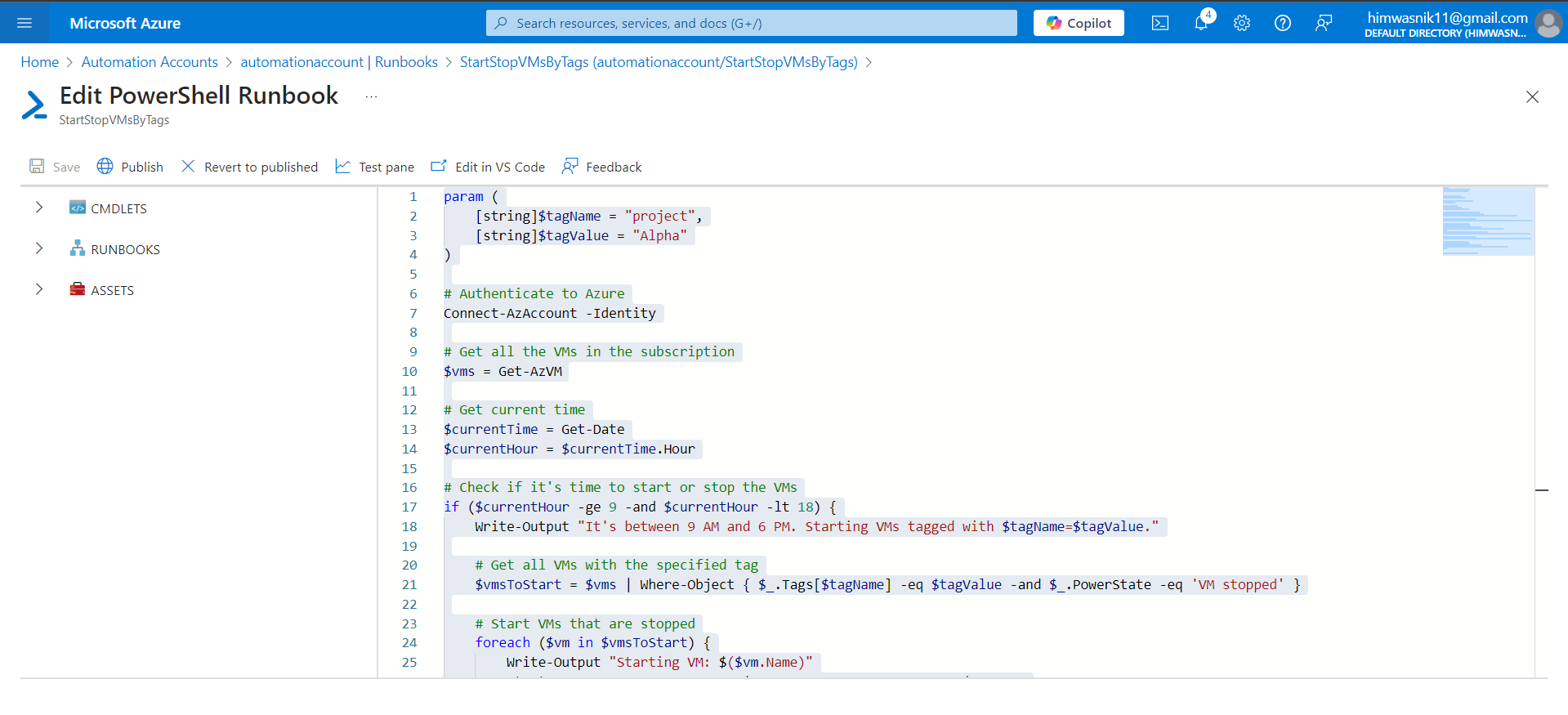
**Step 4.Schedule the automationaccount**

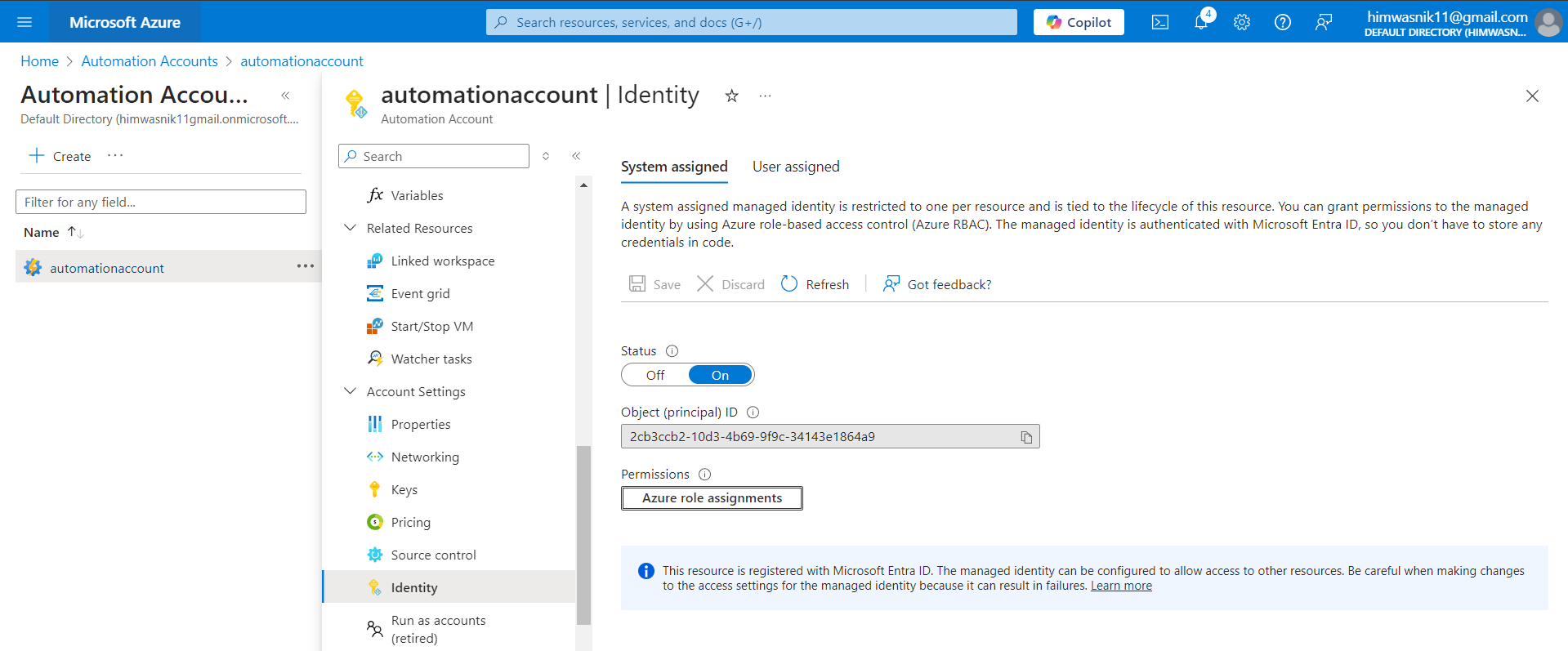
1. Go to the Runbook.
2. Click  **Schedule** > **+ Add a Schedule**.
3. For **Start VMs**:
   * Name: StartVMsSchedule
   * Recurrence: Daily at 9:00 AM.
   * Parameters: Action = Start.
4. For **Stop VMs**:
   * Name: StopVMsSchedule
   * Recurrence: Daily at 6:00 PM.
   * Parameters: Action = Stop.

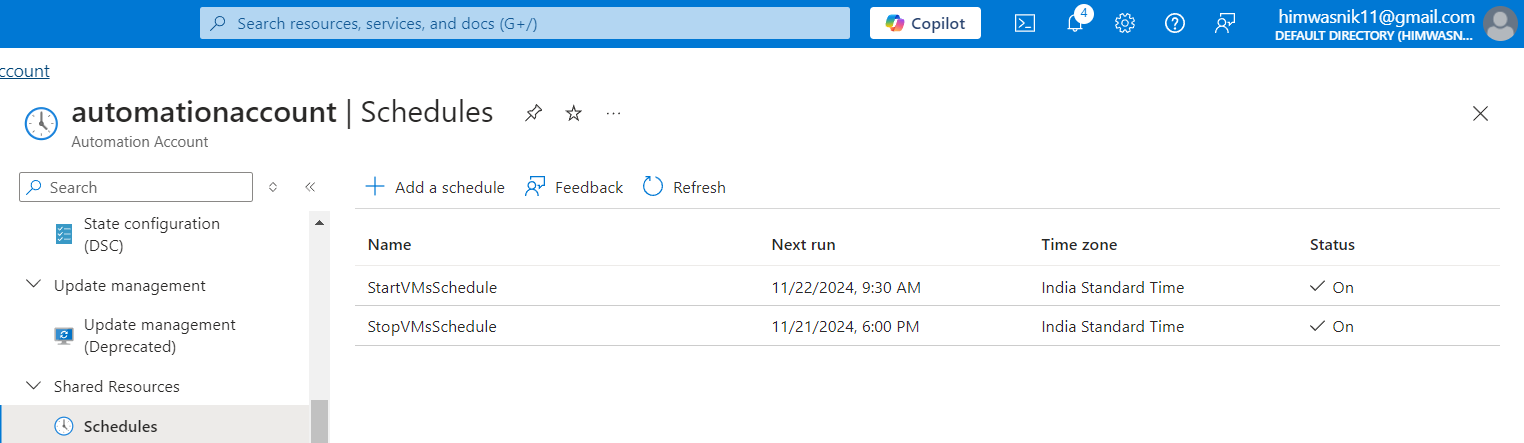
**Step 5.Schedule the Runbook**

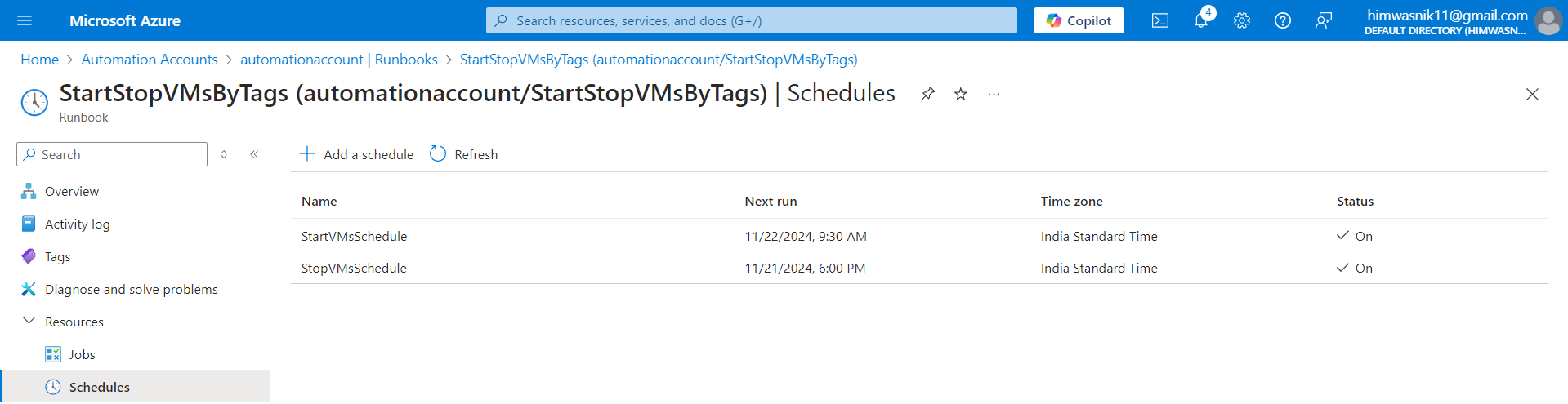
1. Go to the Runbook.
2. Click **Link to Schedule** > **+ Add a Schedule**.
3. For **Start VMs**:
   * Name: StartVMsSchedule
   * Recurrence: Daily at 9:00 AM.
   * Parameters: Action = Start.
4. For **Stop VMs**:
   * Name: StopVMsSchedule
   * Recurrence: Daily at 6:00 PM.
   * Parameters: Action = Stop.











***Automate Resource Group Cleanup Using Runbook***

• Scenario: You have several resource groups in Azure that are no longer in use. To maintain hygiene and reduce costs, you want to automatically delete resource groups that have been inactive for more than 30 days.

• Problem Statement: Write a runbook script to identify resource groups that haven’t been used for the last 30 days and delete them automatically. Ensure the script includes safety checks to avoid deleting important resources.

**Steps:**

1. **Create an Azure Automation Account** (if you don’t already have one).
2. **Create a Runbook** in the Azure Automation account.
3. **Write the Runbook Script** to identify and delete inactive resource groups.

**1. Create an Azure Automation Account:**

1. Go to the Azure Portal.
2. Search for "Automation" and select **Azure Automation**.
3. Click **Create** and follow the prompts to create a new Automation account.

**2. Create a Runbook:**

1. In your **Azure Automation account**, go to **Runbooks**.
2. Click **Create a runbook** and choose **PowerShell** as the runbook type.
3. Give the runbook a meaningful name, such as AutoCleanupResourceGroups.
4. Once created, click on the runbook name to open the editor.

**3. Write the Runbook Script:**

# Authenticate with Azure using managed identity or existing session

Connect-AzAccount -Identity

# Get the current subscription context

$subscription = Get-AzContext

if (-not $subscription) {

Write-Error "No Azure subscription context found. Please log in using Connect-AzAccount."

return

}

Write-Output "Using subscription: $($subscription.Subscription.Name) [$($subscription.Subscription.Id)]"

# Define the threshold for inactivity (30 days)

$currentDate = Get-Date

$thresholdDate = $currentDate.AddDays(-30)

# Get all resource groups in the subscription

Write-Output "Fetching resource groups in the subscription..."

$resourceGroups = Get-AzResourceGroup

if (-not $resourceGroups) {

Write-Output "No resource groups found in the subscription."

return

}

# Loop through each resource group and check activity

foreach ($rg in $resourceGroups) {

Write-Output "Checking resource group: $($rg.ResourceGroupName)"

# Get the activity logs for the resource group from the last 30 days

$activityLogs = Get-AzLog -ResourceGroupName $rg.ResourceGroupName -StartTime $thresholdDate -EndTime $currentDate -WarningAction SilentlyContinue

# If no activity logs are found

if (-not $activityLogs -or $activityLogs.Count -eq 0) {

# Safety check: Verify that the resource group is not critical

if ($rg.ResourceGroupName -notlike "\*critical\*") {

try {

# Delete the resource group (uncomment the line below to enable actual deletion)

# Remove-AzResourceGroup -Name $rg.ResourceGroupName -Force -WarningAction SilentlyContinue

Write-Output "Deleted resource group: $($rg.ResourceGroupName)"

} catch {

Write-Error "Failed to delete resource group: $($rg.ResourceGroupName). Error: $\_"

}

} else {

Write-Output "Skipping critical resource group: $($rg.ResourceGroupName)"

}

} else {

Write-Output "Resource group $($rg.ResourceGroupName) is active."

}

}

Write-Output "Resource group cleanup process completed."

**4.Publish Your Runbook**

**5.Create a Schedule for the Runbook**

Once your runbook is published, you can schedule it to run automatically:

1. **Go to your Runbook:**
   * Under Process Automation in your Automation Account, click on Runbooks.
   * Select the runbook you want to schedule (e.g., "DeleteInactiveResourceGroups").
2. **Schedule the Runbook:**
   * Click Start in the top menu, and then select Schedule.
   * Click + Add a schedule to create a new schedule.

Here, you'll define:

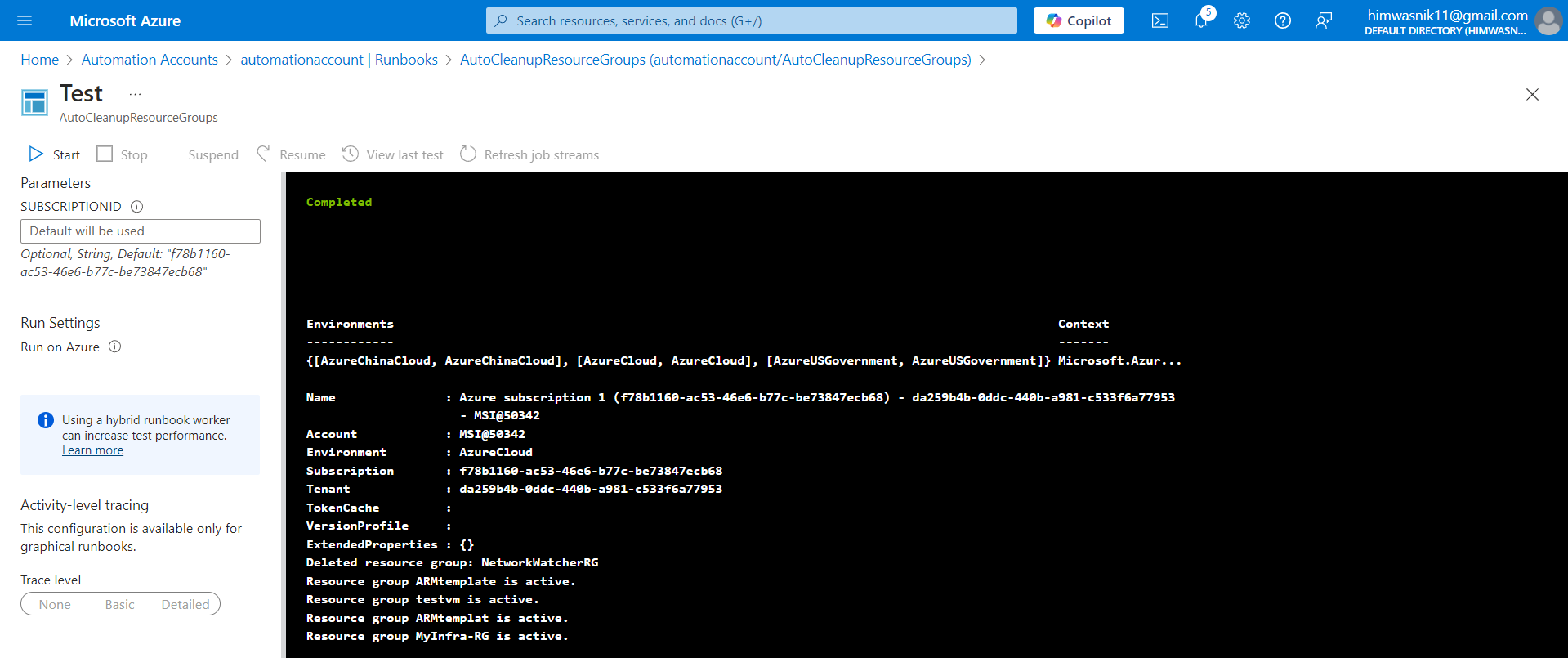
* + Name: Name your schedule (e.g., "Cleanup Schedule").
  + Start Time: Select when you want the automation to start.
  + Recurrence: Define how often it should run (daily, weekly, etc.).
    - For example, you can set it to run daily or weekly, depending on your requirements.

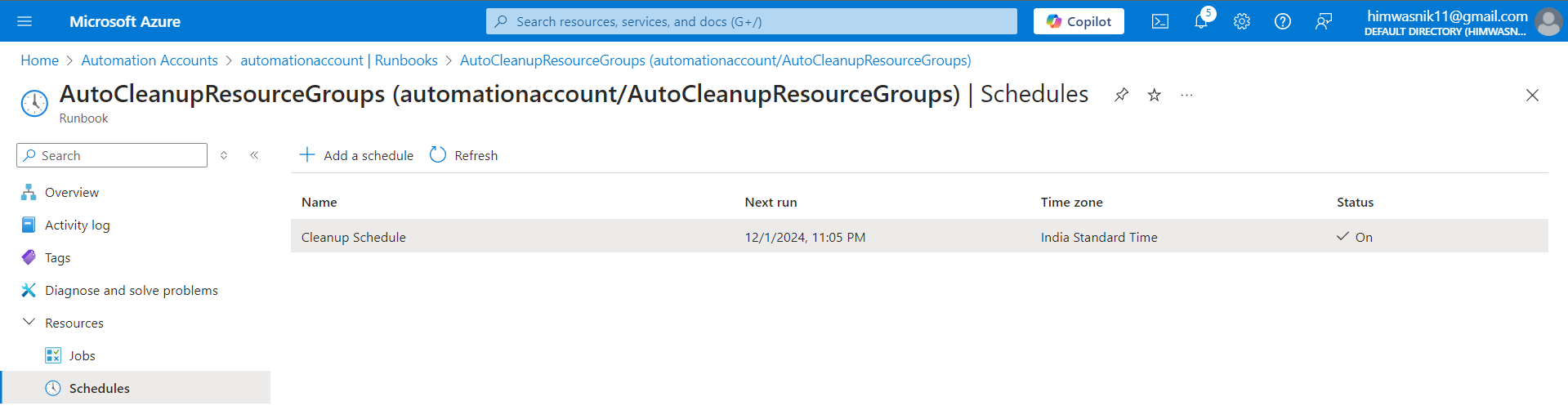
Click Create once you're done.

**4. Link the Schedule to the Runbook**

* After creating the schedule, link it to your runbook:
  + You'll be prompted to choose the Runbook you want to schedule (it will be listed under Runbook).
  + Choose the schedule you just created.

Now, your runbook will execute automatically based on the schedule you've set**.**

****

****

Create a Backup of Azure SQL Database using Runbook  
  
    •    Scenario: As a database administrator, you want to create regular backups of your Azure SQL Database. Instead of doing it manually, you need to automate this task using Azure Runbooks.  
    •    Problem Statement: Create an Azure Runbook that automates the backup of an Azure SQL Database to a storage account at regular intervals. Schedule this runbook to run daily.  
    •    Key Focus Areas: Azure SQL Database, Backup strategies, Azure Runbooks, Automation, PowerShell.

**1. Create an Automation Account**

* **Step 1:** Go to the **Azure portal** (<https://portal.azure.com>).
* **Step 2:** In the left sidebar, click on **"Create a resource"** and search for **"Automation"**.
* **Step 3:** Click on **"Automation"**, then click **"Create"**.
* **Step 4:** Fill in the required information:
  + **Name:** Choose a name for the automation account (e.g., SQLBackupAutomation).
  + **Subscription:** Choose your subscription.
  + **Resource Group:** Select an existing resource group or create a new one.
  + **Location:** Select the location for the Automation Account.
* **Step 5:** Click **"Create"**.

**2. Create a Runbook**

* **Step 1:** After the Automation Account is created, go to the Automation Account and select **"Runbooks"** from the left sidebar.
* **Step 2:** Click **"Create a runbook"**.
* **Step 3:** Provide the following information:
  + **Name:** Choose a name (e.g., BackupSQLDatabase).
  + **Runbook type:** Select **PowerShell**.
  + **Description:** Optional description.
* **Step 4:** Click **"Create"**.

**3. Write the PowerShell Script for SQL Database Backup**

* **Step 1:** In the newly created Runbook, click on **"Edit"**.
* **Step 2:** Write the PowerShell script to back up the SQL Database to a storage account. Below is an example script for backing up an Azure SQL Database:

powershell

# Connect to Azure with system-assigned managed identity

Connect-AzAccount -Identity

# Set and store context

$AzureContext = Set-AzContext –SubscriptionId "f78b1160-ac53-46e6-b77c-be73847ecb68" # Replace with your Subscription ID

# Resource group name

$resourceGroup = "Myinfra-RG" # Replace with your resource group name

# Storage account name that will have the backups

$storageAccountName = "webstoring"

# Storage account access key that will have the backups

$storageKey = "iI3twPgv/XaVs2N8P4/WIyr9/YS5b+FaqDh7vm3jC1wjG3RjMgwfh/ZKv4lVuOwNH7M8KyvCc4uj+AStcB8qzQ==" # Replace with your storage account key

# Container name that will have the backups

$containerName = "backup" # Replace with your container name

# Storage blob URI with the datetime

$storageUri = "https://$storageAccountName.blob.core.windows.net/$containerName/db-$(Get-Date -UFormat "%Y-%m-%d\_%H-%M-%S").bacpac"

# Storage access key type

$storageKeyType = "StorageAccessKey" # Correct type for the key being used

# SQL server name

$server\_name = "mysqlserve1" # Replace with your SQL Server name

# Database name to be exported

$SQL\_db = "Mydatabase" # Replace with your database name

# SQL Auth Username

$SQL\_username = "sysadmin" # Replace with your SQL Admin Username

# SQL Auth Password

$SQL\_secure\_secret = ConvertTo-SecureString -String "Himanshu@2001" -AsPlainText -Force # Replace with your SQL Password

# Run the Export job with the required parameters

New-AzSqlDatabaseExport -ResourceGroupName $resourceGroup `

-ServerName $server\_name `

-DatabaseName $SQL\_db `

-StorageKeyType $storageKeyType `

-StorageKey $storageKey `

-StorageUri $storageUri `

-AdministratorLogin $SQL\_username `

-AdministratorLoginPassword $SQL\_secure\_secret

* **Step 3:** Save the script.

**4. Test the Runbook**

* **Step 1:** In the Runbook page, click **"Test pane"**.
* **Step 2:** Test the script by clicking **"Start"**. Ensure that the script runs without errors and the backup is successfully stored in the Azure Storage Account.

**5. Assign Managed Identity (Optional, Recommended for Security)**

If you're using **Managed Identity** for the Automation Runbook to access Azure resources (such as SQL Database or Storage Account), follow these steps:

* **Step 1:** In the **Automation Account** settings, go to **"Identity"**.
* **Step 2:** Enable the **System-assigned Managed Identity**.
* **Step 3:** Go to your **SQL Server** and **Storage Account**. Assign the necessary permissions (e.g., Contributor or Backup Operator) to the managed identity.
  + For SQL Database: Assign the identity the **SQL Server Contributor** role.
  + For Storage Account: Assign the **Storage Blob Data Contributor** role.

**6. Publish the Runbook**

* After testing the Runbook, click **"Publish"** to make it available for scheduling.

**7. Create a Schedule for the Runbook**

* **Step 1:** In the Automation Account, go to **"Schedules"**.
* **Step 2:** Click **"Add a schedule"**.
* **Step 3:** Set the schedule to run daily at the desired time.
* **Step 4:** After creating the schedule, link it to the Runbook:
  + Go to the **Runbook**, and under **"Schedules"**, click **"Link to schedule"**.
  + Select the schedule you just created.

**8. Verify the Backup and Monitor**

* **Step 1:** After the scheduled time passes, check the **Azure Storage Account** to confirm that the backup file is created.
* **Step 2:** Go to **"Runbooks"** in the Automation Account and click on the **Runbook** to see the execution history. Ensure the backup process completed successfully.

