Extending HT Condor onto Azure using the GAHP Interface

Pre-Requisites and installation of GAHP interface

Table of Contents

[Setup and install prerequisites for GAHP 2](#_Toc490120293)

[Python Installation 2](#_Toc490120294)

[Linux 2](#_Toc490120295)

[Windows 2](#_Toc490120296)

[Installation of pip 3](#_Toc490120297)

[Linux 3](#_Toc490120298)

[Windows 3](#_Toc490120299)

[Azure SDK for Python 4](#_Toc490120300)

[Linux or Windows 4](#_Toc490120301)

[Install the full Azure SDK 4](#_Toc490120302)

[Install specific modules of the Azure SDK 4](#_Toc490120303)

[Minimum supported version 4](#_Toc490120304)

[Check modules version 5](#_Toc490120305)

[Upgrade a module 5](#_Toc490120306)

[Service principal for GAHP application and Key vault 6](#_Toc490120307)

[Setup for GAHP key vault functionality 7](#_Toc490120308)

[Assign permissions to the service principal for GAHP key vault functionality 7](#_Toc490120309)

[Shell script to download secret in the post deployment phase 12](#_Toc490120310)

[Setup for GAHP Deletion Job functionality 13](#_Toc490120311)

[Azure Automation Account for GAHP 13](#_Toc490120312)

[Create a standalone Automation account using Azure portal 14](#_Toc490120313)

[Create a runbook in Azure Automation using Azure portal 19](#_Toc490120314)

[Publish a new PowerShell runbook 24](#_Toc490120315)

[Create a webhook using Azure portal to automate the runbook 26](#_Toc490120316)

[Setup AppSettings.txt file 29](#_Toc490120317)

[GAHP solution package 31](#_Toc490120318)

# Setup and Install Prerequisites for GAHP

As GAHP is a Python application, it is platform independent so it can work with Linux and Windows.

Below are the steps to setup, configure and install the software prerequisites in different platforms (Linux and Windows).

We need pip to install a bundle of all the client libraries. This will download the packages from the Python Package Index (PyPI).

You may need administrator rights to perform the installation:

## Python Installation

### Linux

1. Open terminal using Putty or SSH.
2. Check for current installation of Python by running following command.

python --version

1. If not installed, run one of the following commands
   1. Ubuntu or Debian distribution

sudo apt-get install python

* 1. Red Hat distribution like Fedora, CentOS

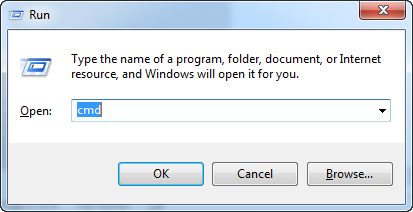
yum install -y python

1. Verify the installation using the following command.

python --version

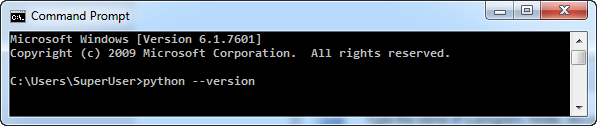
### Windows

1. Open a command prompt.



1. Check for current installation of Python by running the following command

python –-version



1. If not installed, download the Python installer from its official website. <https://www.python.org/downloads/>
2. Run the Python installer.
3. Verify the installation using python –-version command.

## Installation of pip

### Linux

1. Open a terminal and run the following command to check whether pip is installed or not.

pip --version

1. If pip is not installed, run one of the following commands
   1. Ubuntu or any Debian distribution

sudo apt-get install python-dev libxml2-dev libxslt-dev libssl-dev

* 1. Red Hat distributions like Fedora or CentOS

sudo yum install -y python-devel python-lxml openssl-devel gcc libffi libffi-devel

1. Once the tools above are successfully installed, run the following command to get pip which will be used to install Azure SDK for Python.

python get-pip.py

1. Verify that pip is installed using the following command:

pip --version

### Windows

1. Open a command prompt and run the following to check whether pip is installed or not.

pip --version

1. If pip is not installed, download the “get-pip.py” file from the following link.

<https://bootstrap.pypa.io/get-pip.py>

1. In the command prompt navigate to the folder where the “get-pip.py” file was saved.
2. Run the following command to install pip which will be used to install Azure SDK for Python.

python get-pip.py

1. Verify that pip is installed using the following command.

pip --version

## Azure SDK for Python

### Linux or Windows

The GAHP server uses the following Azure SDK, so only these need to be installed. However, all the modules of the SDK can be installed.

* azure-common-credentials
* azure-mgmt-compute
* azure-mgmt-resource
* azure-mgmt-storage
* azure-mgmt-network
* azure-mgmt-scheduler
* azure-storage
* azure-mgmt-keyvault

### Install the full Azure SDK

Install all the modules of Azure SDK for Python using following command.

Preview release:

pip install --pre azure

Specific version:

pip install azure==2.0.0rc6

pip install azure-mgmt-compute==2.0.0

### Install specific modules of the Azure SDK

* azure-common-credentials
* azure-mgmt-compute
* azure-mgmt-resource
* azure-mgmt-storage
* azure-mgmt-network
* azure-mgmt-scheduler
* azure-storage
* azure-mgmt-keyvault

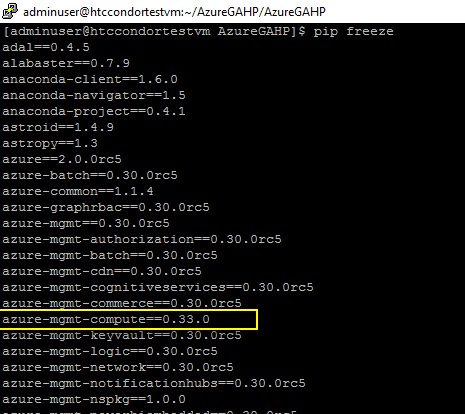
### Minimum supported version

* azure-mgmt-compute – 2.0.0
* azure-mgmt-scheduler – 1.1.2
* azure-mgmt-keyvault – 0.31.0

### Check modules version

The following command is used to check installed version of the modules:

pip freeze



### Upgrade a module

Uninstalling and then installing the module will upgrade that module.

pip uninstall azure-mgmt-compute

pip install azure-mgmt-compute

## Service Principal for GAHP and Key Vault

The GAHP server requires an Azure AD application and a service principal for silent login to access and create Azure artifacts. Create an Azure AD application and service principal by following the steps outlined at:-

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal>

The application must of type **Web app/API**.

After creating a service principal follow the additional steps to obtain the following values that will be used with the GAHP:

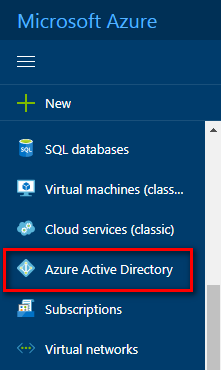
* Application ID
* Authentication Key
* Tenant ID

## Setup for GAHP Key Vault functionality

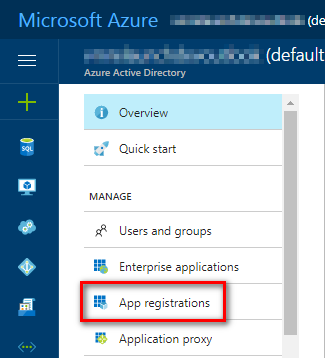
### Assign permissions to service principal for GAHP key vault functionality

Follow the following outlined steps to assign permission to service principal created in [above](#_Service_principal_for) section.

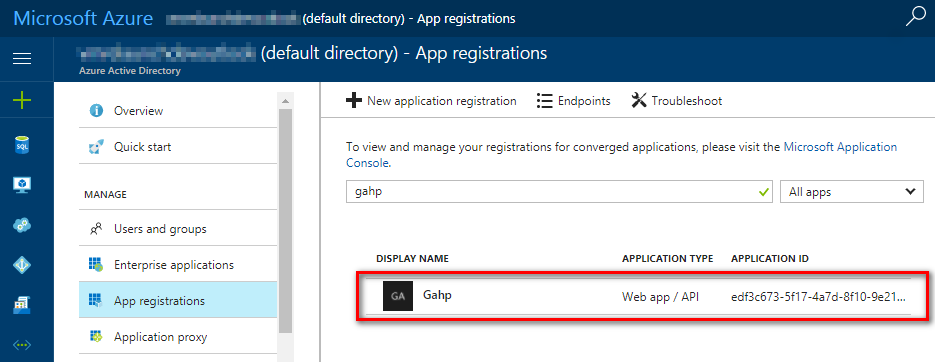
1. Sign in to the Azure portal with an account that is a **Global Administrator** of the Azure Active Directory.
2. In the left navigation pane, select **Azure Active Directory**.



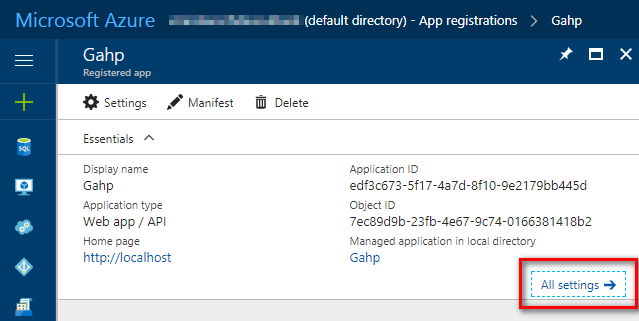
1. Select **App registrations** navigation menu option.



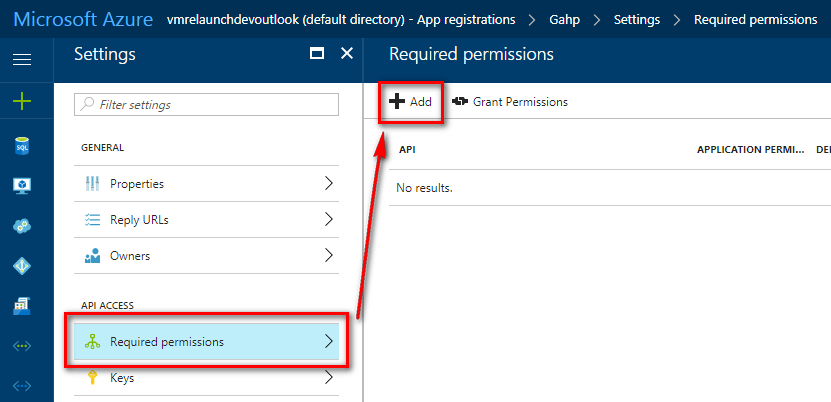
1. Search using name of the service principal and select the service principal.



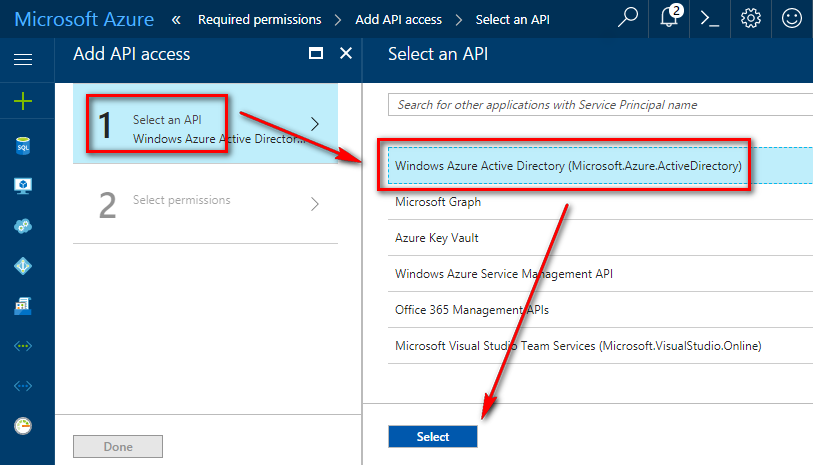
1. Select **All settings** to open settings pane.



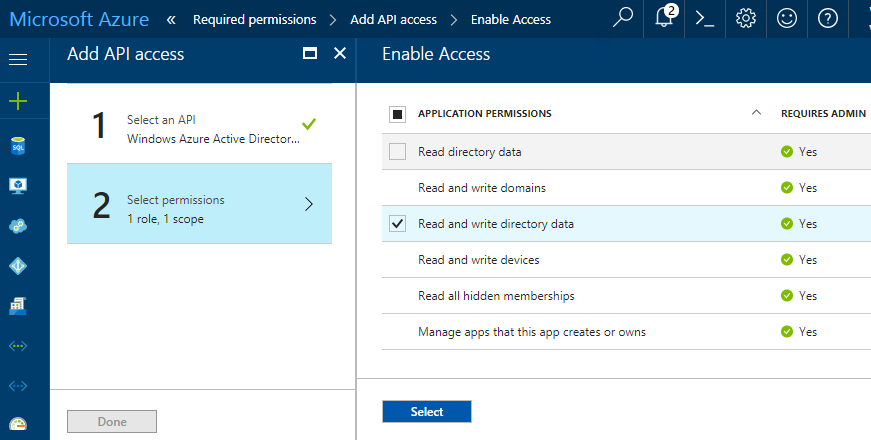
1. In the settings pane, select **Required permissions** and select **Add.**



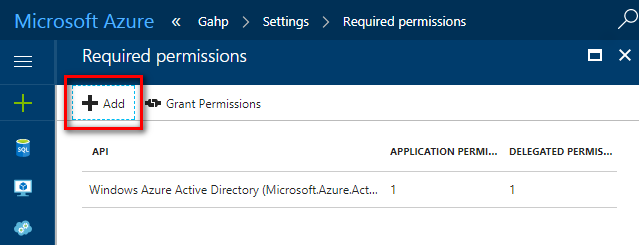
1. In the **Add API access** pane, click **Select an API**, select **Windows Azure Active Directory API** and click **Select** button.



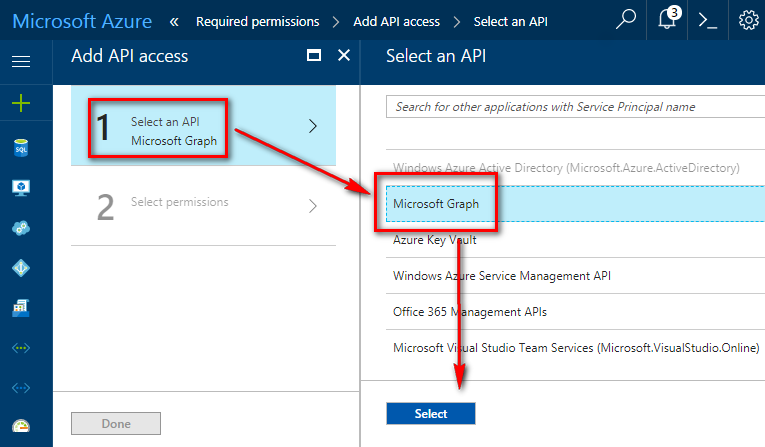
1. Click **Select permissions** in the **Add API access** pane, select following permissions and then click **Select** button:
   1. Read and write directory data (Application permissions)
   2. Read and write all groups. (Delegated permissions)



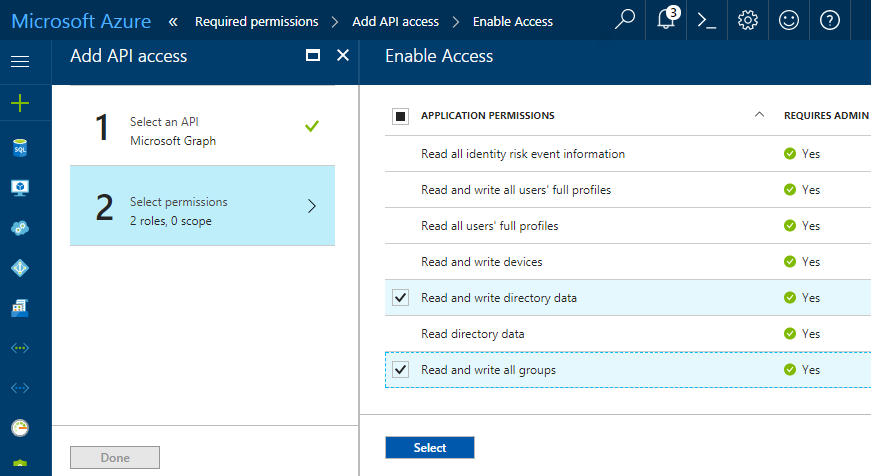
1. Select **Add** to again open **Add API access** pane.



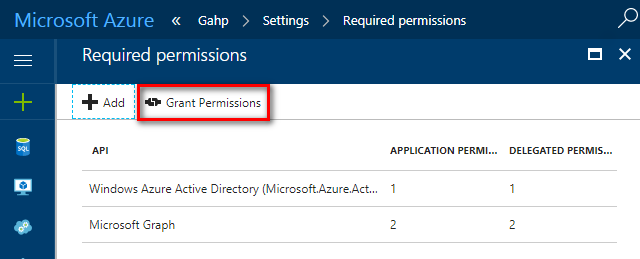
1. In the **Add API access** pane, click **Select an API**, select **Microsoft Graph** and click **Select** button.



1. Click **Select permissions** in the **Add API access** pane, Select following permissions and click **Select** button: (Both Application and Delegated permissions)
   1. Read and write directory data
   2. Read and write all groups



1. Select **Grant Permissions** and press **yes** to confirm.



1. It may take some time to update the permissions of a service principal.

### Shell script to download secret post deployment

Upload following shell script to a private/public repository like GitHub and save that URL. If required, remove the .txt extension.



This shell script enables GAHP to use key vault. It is used for the post deployment configuration of virtual machine instances in VMSS.

## Setup of Deletion Job functionality for GAHP

### Azure Automation Account for GAHP

Microsoft Azure Automation provides a way for users to automate the manual, long-running, error-prone, and frequently repeated tasks that are commonly performed in a cloud and enterprise environment. It saves time and increases the reliability of regular administrative tasks and even schedules them to be automatically performed at regular intervals. You can automate processes using runbooks or automate configuration management using Desired State Configuration.

All the automation tasks you perform against resources using the Azure cmdlets in Azure Automation authenticate to Azure using Azure Active Directory organizational identity credential-based authentication. An Automation account is separate from the account you use to sign in to the portal to configure and use Azure resources.

The Automation resources for each Automation account are associated with a single Azure region, but Automation accounts can manage all the resources in your subscription. Create Automation accounts in different regions if you have policies that require data and resources to be isolated to a specific region.

#### Note

Automation accounts, and the resources they contain that are created in the Azure portal, cannot be accessed in the Azure classic portal. If you want to manage these accounts or their resources with Windows PowerShell, you must use the Azure Resource Manager modules.

When you create an Automation account in the Azure portal, you automatically create two authentication entities:

* A Run As account. This account creates a service principal in Azure Active Directory (Azure AD) and a certificate. It also assigns the Contributor role-based access control (RBAC), which manages Resource Manager resources by using runbooks.
* A Classic Run As account. This account uploads a management certificate, which is used to manage classic resources by using runbooks.

### Create a standalone Automation account using Azure portal

This topic shows you how to create an Automation account from the Azure portal if you want to evaluate and learn Azure Automation without including the additional management solutions or integration with OMS Log Analytics to provide advanced monitoring of runbook jobs. You can add those management solutions or integrate with Log Analytics at any point in the future. With the Automation account, you can authenticate runbooks managing resources in either Azure Resource Manager or Azure classic deployment.

When you create an Automation account in the Azure portal, it automatically creates:

* Run As account, which creates a new service principal in Azure Active Directory, a certificate, and assigns the Contributor role-based access control (RBAC), which is used to manage Resource Manager resources using runbooks.
* Classic Run As account by uploading a management certificate, which is used to manage classic resources using runbooks.

This simplifies the process for you and helps you quickly start building and deploying runbooks to support your automation needs.

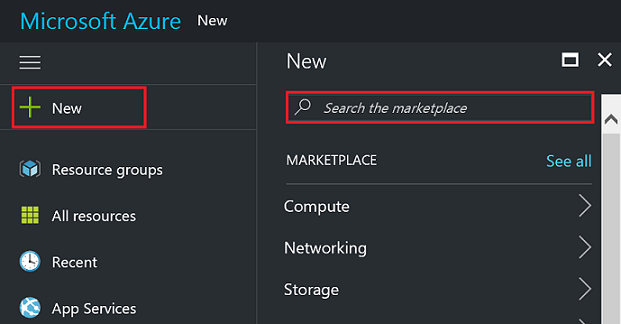
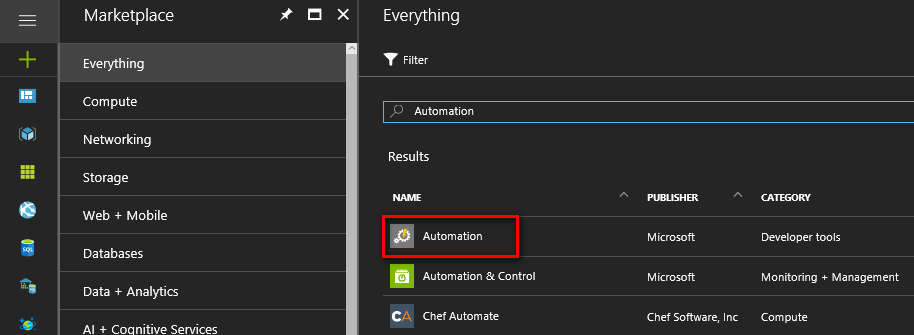
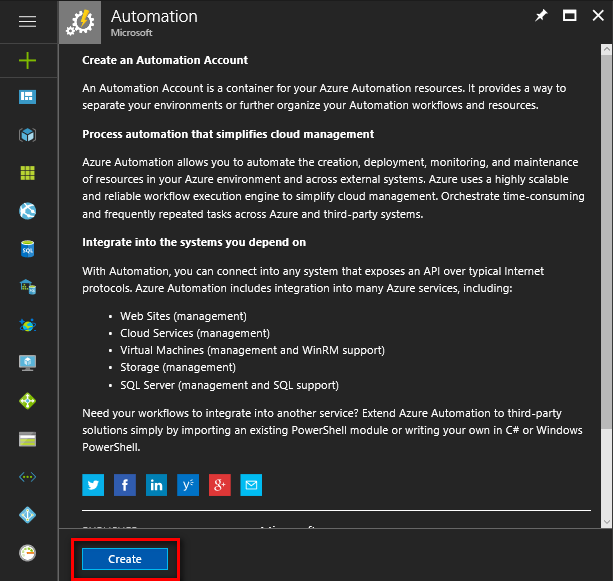
Perform the following steps to create an Azure Automation account in the Azure portal.

#### Note

To create an Automation account, you must be a member of the Service Admins role or co-administrator of the subscription that is granting access to the subscription. You must also be added as a user to that subscription's default Active Directory instance. The account does not need to be assigned a privileged role.

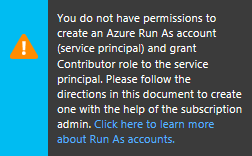
If you are not a member of the subscription’s Active Directory instance before you are added to the co-administrator role of the subscription, you are added to Active Directory as a guest. In this instance, you receive a “You do not have permissions to create…” warning on the Add Automation Account blade.

Users who were added to the co-administrator role first can be removed from the subscription's Active Directory instance and re-added to make them a full User in Active Directory. To verify this situation from the Azure Active Directory pane in the Azure portal by selecting Users and groups, selecting All users and, after you select the specific user, selecting Profile. The value of the User type attribute under the user’s profile should not equal Guest.

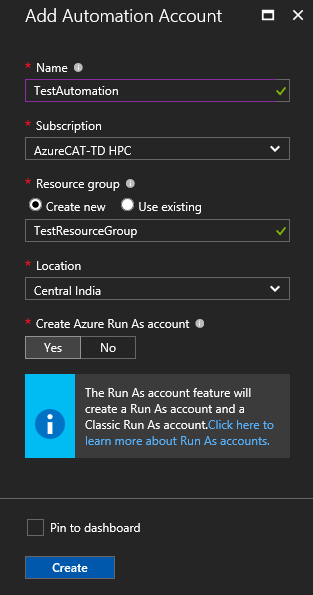
1. Sign in to the Azure portal with an account which is a member of the Subscription Admins role and co-administrator of the subscription.
2. Click **New**.  
   
3. Search for **Automation** and press enter. In the search results select **Automation.**  
   
4. In the Automation Accounts blade, click **Create**.  
   

#### Note

If you see the following warning in the **Add Automation Account** blade, this is because your account is not a member of the Subscription Admins role and co-admin of the subscription.



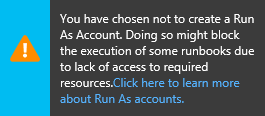
1. In the **Add Automation Account** blade, in the **Name** box enter a name for your new Automation account.



1. If you have more than one subscription, specify one for the new account, a new or existing **Resource group** and an Azure data center **Location**.
2. Verify the value **Yes** is selected for the **Create Azure Run As account** option, and click the **Create** button.

#### Note

If you choose to not create the Run As account by selecting the option **No**, you are presented with a warning message in the **Add Automation Account** blade. While the account is created in the Azure portal, it will not have a corresponding authentication identity within your classic or Resource Manager subscription directory service and therefore, no access to resources in your subscription. This prevents any runbooks referencing this account from being able to authenticate and perform tasks against resources in those deployment models.



When the service principal is not created the Contributor role will not be assigned.

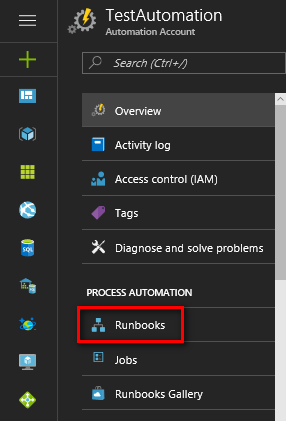
1. During the Automation account creation, the progress is tracked under **Notifications** from the menu.

### Create a runbook in Azure Automation using Azure portal

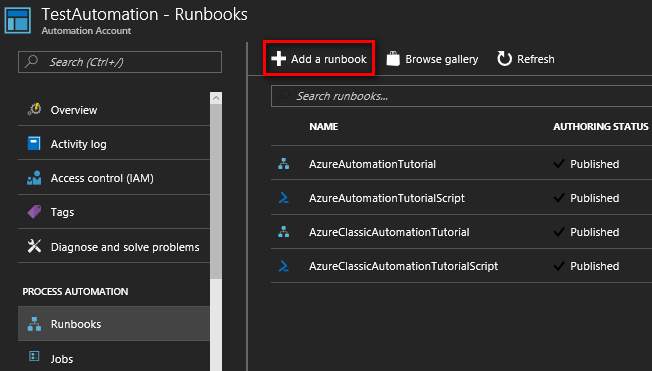
PowerShell runbooks are based on Windows PowerShell. The code of the runbook can be edited using the text editor in the Azure portal.

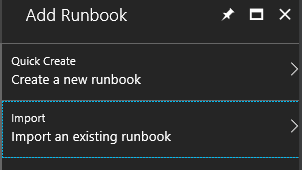
Perform the following steps to create a runbook in the Azure portal.

1. In the Azure portal, open your Automation account.
2. Click on the **Runbooks** tile to open the list of runbooks.



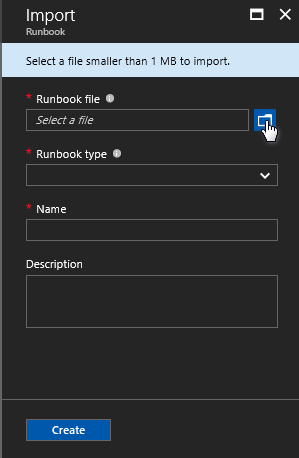
1. Click on the **Add a runbook** button and then **Import an existing new runbook**. Some runbooks are created automatically with Azure Automation account.



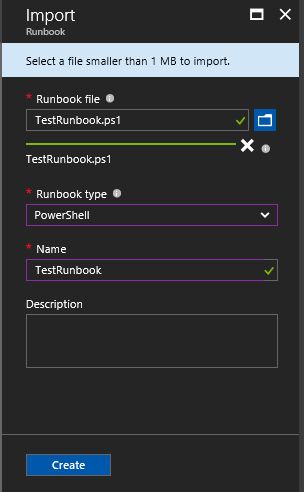


1. In the **Runbook file**, click the import icon to import following attached file. If required, remove the .txt extension.

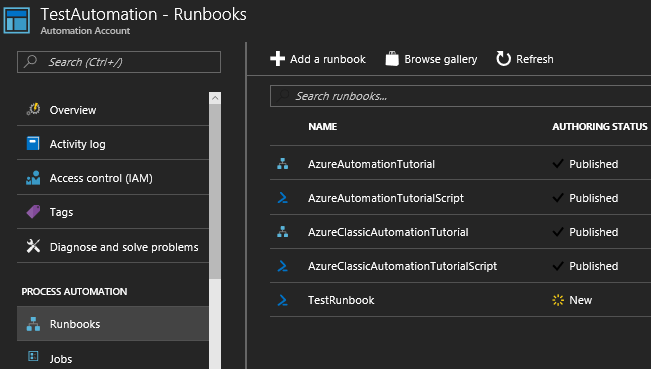




1. If the **Name** field is enabled then it can be changed. The runbook name must start with a letter and can have letters, numbers, underscores, and dashes.
2. The **Runbook type** will be automatically selected as **PowerShell**.



1. Click **Create** to create the runbook.
2. The new runbook will appear in the list of runbooks for the Automation Account. The newly imported runbook has Authoring Status as **New.**



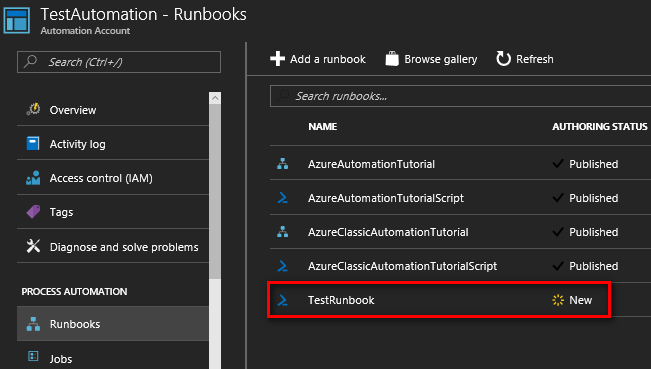
1. Publish the runbook before running it.

### Publish a new PowerShell runbook

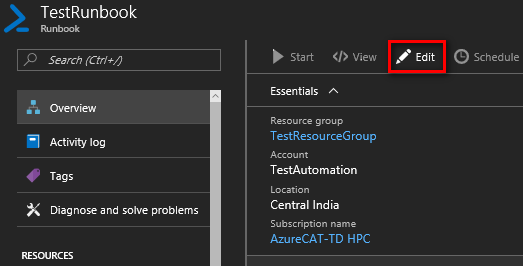
When a runbook is imported or created, publish it before it can be run. Each runbook in Automation has a Draft and a Published version. Only the Published version can be run, and only the Draft version can be edited. The Published version is unaffected by any change in the Draft version. When the Draft version should be made available, then publish it which overwrites the Published version with the Draft version.

Perform the following steps to publish new runbook in the Azure portal.

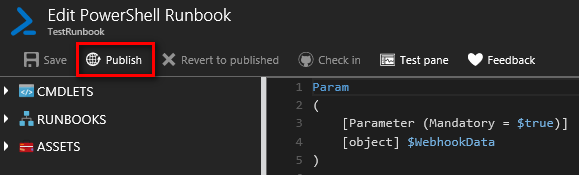
1. Open the new runbook in the Azure portal.



1. Click the **Edit** button.



1. Click the **Publish** button and then **Yes** to the verification message.



### Create a WebHook using Azure portal to automate the runbook

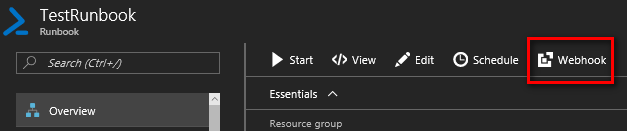
A webhook is used to start a runbook in Azure Automation through a single HTTP request. This allows GAHP server to start runbook using the Azure Automation API.

A webhook can define values for runbook parameters which are used when the runbook is started by that webhook. The webhook must include values of mandatory parameters for the runbook and may include values for optional parameters. A parameter value configured in a webhook can be modified even after creating the webhook.

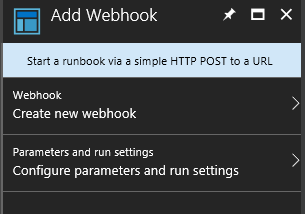
When a client starts a runbook using a webhook, it cannot override the parameter values defined in the webhook. To receive data from the client, the runbook can accept a single parameter called **$WebhookData** of type [object] which will contain data that client includes in the POST request.

Perform the following steps to create a webhook in the Azure portal.

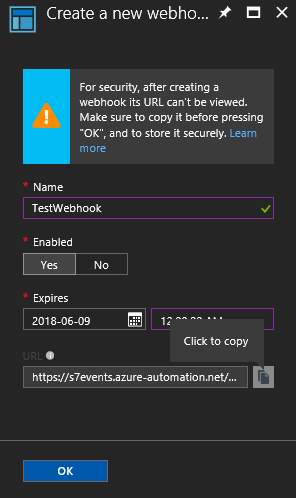
1. From the **Runbooks blade** in the Azure portal, click the runbook that the webhook will start.
2. Click **Webhook** at the top of the blade to open the **Add Webhook** blade.



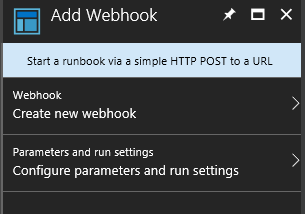
1. Click **Create new webhook** to open the **Create webhook blade**.



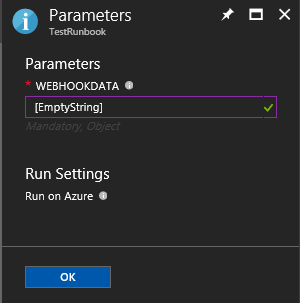
1. Specify a **Name**, **Expiration Date** for the webhook and if it should be enabled.
2. Copy the URL of the webhook then save it in a safe place. **Once you create the webhook, you cannot retrieve the URL again. Click OK for the next step.**



1. Click **Configure parameters and run settings** to provide value for the runbook parameter.



1. Set **WEBHOOKDATA** parameter value to [EmptyString] as shown in the following image.



1. Click **OK** to complete the setup of webhook.
2. Click **Create** to create the webhook.

# Setup AppSettings.txt file

The GAHP server creates a scheduler job to execute the PowerShell runbook to delete a VMSS. It creates a scheduler job collection that hosts a cleaner job and deletion jobs. The cleaner job frequently monitors all the jobs in the collection and delete them once they are complete.

The GAHP server user need to setup AppSettings.txt file with required information for deletion job and key vault functionality.

Open the AppSettings.txt in a text editor and add following information.

1. Application ID of the service principal created in Azure active directory.

Client\_id <Application id>

1. Authentication Key of the service principal.

secret <Authentication key value>

1. Tenant ID of the azure subscription.

tenant\_id <Authentication key value>

1. Number of VMs to process in each thread in AZURE\_VM\_LIST command

max\_vm\_count\_in\_thread <number of VMs in a thread>

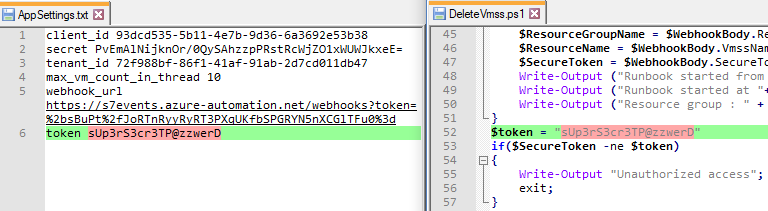
1. Webhook URL to invoke the PowerShell runbook. Paste the webhook URL copied in the [above](#_Create_a_webhook_1) setup in the below format.

webhook\_url <URL>

1. A secure token to authenticate the request to execute the commands in PowerShell runbook. In the format shown in following image, add the token which is used for authentication in the Runbook PowerShell script, the script which was used in the PowerShell script to create the Runbook in #4 of [this section](#_Create_a_PowerShell).

token <value>

So, the token value should be same in both files, AppSettings.txt and the PowerShell scripts.



1. Resource group name of scheduler job collection.

jobs\_rg <Resource group name>

1. Scheduler job collection name where all the scheduler jobs will be created.

job\_collection <Scheduler job collection name>

1. SKU for the scheduler job collection. Following values are allowed:
   1. Free
   2. Standard
   3. P10Premium
   4. P20Premium

job\_collection\_sku <sku value>

1. Frequency of the cleaner job. Following values are allowed:
   1. Minute
   2. Hour
   3. Day
   4. Week
   5. Month

job\_frequency\_type <frequency type>

1. Interval value for the Job\_frequency\_type (number value).

job\_interval <value>

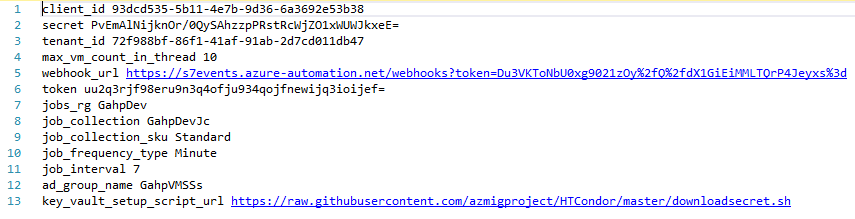
1. Name of the Azure AD group where all the VMSS will be added for key vault functionality of GAHP.

ad\_group\_name <Azure AD group name>

1. key\_vault\_setup\_script\_url <URL from [above setup](#_Shell_script_to) of the shell script to setup the artifacts for downloading the secret on VM>

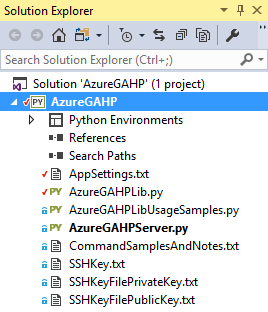
key\_vault\_setup\_script\_url <URL>

Configured AppSettings.txt file should look like below.



# GAHP Solution Package

The GAHP solution package contains the following files:



* **AppSettings.txt**
* **AzureGAHPServer.py**
  + User will run/execute this Python file using Python command to start execution of GAHP
* **AzureGAHPLib.py**
  + This Python file contains the main logic of GAHP for each command.
* **AzureGAHPLibUsageSamples.py**
  + This Python file is used to create dummy VM using hard-coded values.
* **CommandSamplesAndNotes.txt**
  + This text file includes sample commands and rules for the development purposes.
* **SSHKey.txt, SSHKeyFilePrivateKey.txt and SSHKeyFilePublicKey.txt**