**Explore Azure Automation with DevOps**

**Advanced**

**Administrator**

**Developer**

**DevOps Engineer**

**Security Engineer**

**Security Operations Analyst**

**Service Adoption Specialist**

**Solution Architect**

**Technology Manager**

**Azure**

**Azure Artifacts**

**Azure Boards**

**Azure Cloud Services**

**Azure DevOps**

**Azure Pipelines**

**Azure Repos**

**Azure Test Plans**

**GitHub**

**This module describes Azure Automation with Azure DevOps, using runbooks, webhooks, and PowerShell workflows. You learn how to create and manage automation for your environment.**

**Learning objectives**

**By the end of this module, you're able to:**

* **Implement automation with Azure DevOps.**
* **Create and manage runbooks.**
* **Create webhooks.**
* **Create and run a workflow runbook and PowerShell workflows.**
* **Plan for hybrid management.**

[**Start**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/1-introduction/)**Add**

**Prerequisites**

**None**

**This module is part of these learning paths**

* [**AZ-400: Manage infrastructure as code using Azure and DSC**](https://learn.microsoft.com/training/paths/az-400-manage-infrastructure-as-code-using-azure/)

**Module assessment**

**Assess your understanding of this module. Sign in and answer all questions correctly to earn a pass designation on your profile.**

[**Take the module assessment**](https://learn.microsoft.com/training/modules/explore-azure-automation-devops/12-knowledge-check/)

* [**Introduction**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/1-introduction)**4 min**
* [**Create automation accounts**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/2-create-automation-accounts)**4 min**
* [**What is a runbook?**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/3-what-runbook)**4 min**
* [**Understand automation shared resources**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/4-understand-automation-shared-resources)**3 min**
* [**Explore runbook gallery**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/5-explore-runbook-gallery)**3 min**
* [**Examine webhooks**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/6-examine-webhooks)**4 min**
* [**Explore source control integration**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/7-explore-source-control-integration)**4 min**
* [**Explore PowerShell workflows**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/8-explore-powershell-workflows)**4 min**
* [**Create a workflow**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/9-create-workflow)**3 min**
* [**Explore hybrid management**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/10-explore-hybrid-management)**3 min**
* [**Examine checkpoint and parallel processing**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/11-examine-checkpoint-parallel-processing)**4 min**
* [**Knowledge check**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/12-knowledge-check)**5 min**
* [**Summary**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/13-summary)**1 min**

**Introduction**

Completed100 XP

* 4 minutes

Manually executing environment provisioning and configuration management is both laborious and error-prone.

Microsoft Azure DevOps advocates automation to reduce the probability of errors introduced through manual execution.

Automation also delivers the added advantage of completing the work more quickly without relying on subject experts.

Microsoft Azure is built to support automation from the ground up.

*Azure Automation* is an Azure service that provides a way for users to automate the manual, long-running, error-prone, and frequently repeated tasks commonly done in a cloud and enterprise environment.

Azure Automation saves time and increases the reliability of regular administrative tasks.

You can even schedule the tasks to be executed automatically at regular intervals.

You can automate processes using runbooks or automate configuration management by using Desired State Configuration (DSC).

For more information about Azure Automation, review [An introduction to Azure Automation](https://azure.microsoft.com/documentation/articles/automation-intro/).



Azure Automation isn't the only way to automate within Azure.

You can also use open-source tools to do some of these operations.

However, the integration hooks available to Azure Automation remove much of the integration complexity you would have to manage if you did these operations manually.

Some Azure Automation capabilities are:

* Process automation - Azure Automation provides you with the ability to automate frequent, time-consuming, and error-prone cloud management tasks.
* Azure Automation State Configuration - It's an Azure service that allows you to write, manage, and compile Windows PowerShell DSC configurations, import DSC Resources, and assign configurations to target nodes, all in the cloud. For more information, visit [Azure Automation State Configuration Overview](https://learn.microsoft.com/en-us/azure/automation/automation-dsc-overview).
* Azure Update Manager - Manage operating system updates for Windows and Linux computers in Azure, on-premises environments, or other cloud providers. Get update compliance visibility across Azure, on-premises, and for other cloud services. You can create scheduled deployments to orchestrate update installations within a defined maintenance window. For more information, visit [Azure Update Manager](https://learn.microsoft.com/en-us/azure/update-manager/overview).
* Integration with GitHub, Azure DevOps, Git, or Team Foundation Version Control repositories. For more information, go to [Source control integration in Azure Automation](https://learn.microsoft.com/en-us/azure/automation/source-control-integration).
* Automate Amazon Web Services (AWS) Resources - Automate common tasks with resources in AWS using Automation runbooks in Azure. For more information, go to [Authenticate Runbooks with Amazon Web Services](https://learn.microsoft.com/en-us/azure/automation/automation-config-aws-account).
* Manage Shared resources - Azure Automation consists of a set of shared resources (such as *connections*, *credentials*, *modules*, *schedules*, and *variables*) that make it easier to automate and configure your environments at scale.
* Run backups - Azure Automation allows you to run regular backups of non-database systems, such as backing up Azure Blob Storage at certain intervals.

Azure Automation works across hybrid cloud environments in addition to Windows and Linux operating systems.

This module describes Azure automation with Azure DevOps, using runbooks, webhooks, and PowerShell workflows.

You'll learn how to create and manage automation for your environment.

**Learning objectives**

After completing this module, students and professionals can:

* Implement automation with Azure DevOps.
* Create and manage runbooks.
* Create webhooks.
* Create and run a workflow runbook and PowerShell workflows.

**Prerequisites**

* Understanding of what DevOps is and its concepts.
* Familiarity with version control principles is helpful but isn't necessary.
* Beneficial to have experience in an organization that delivers software.

**Next unit: Create automation accounts**

**Create automation accounts**

Completed100 XP

* 4 minutes

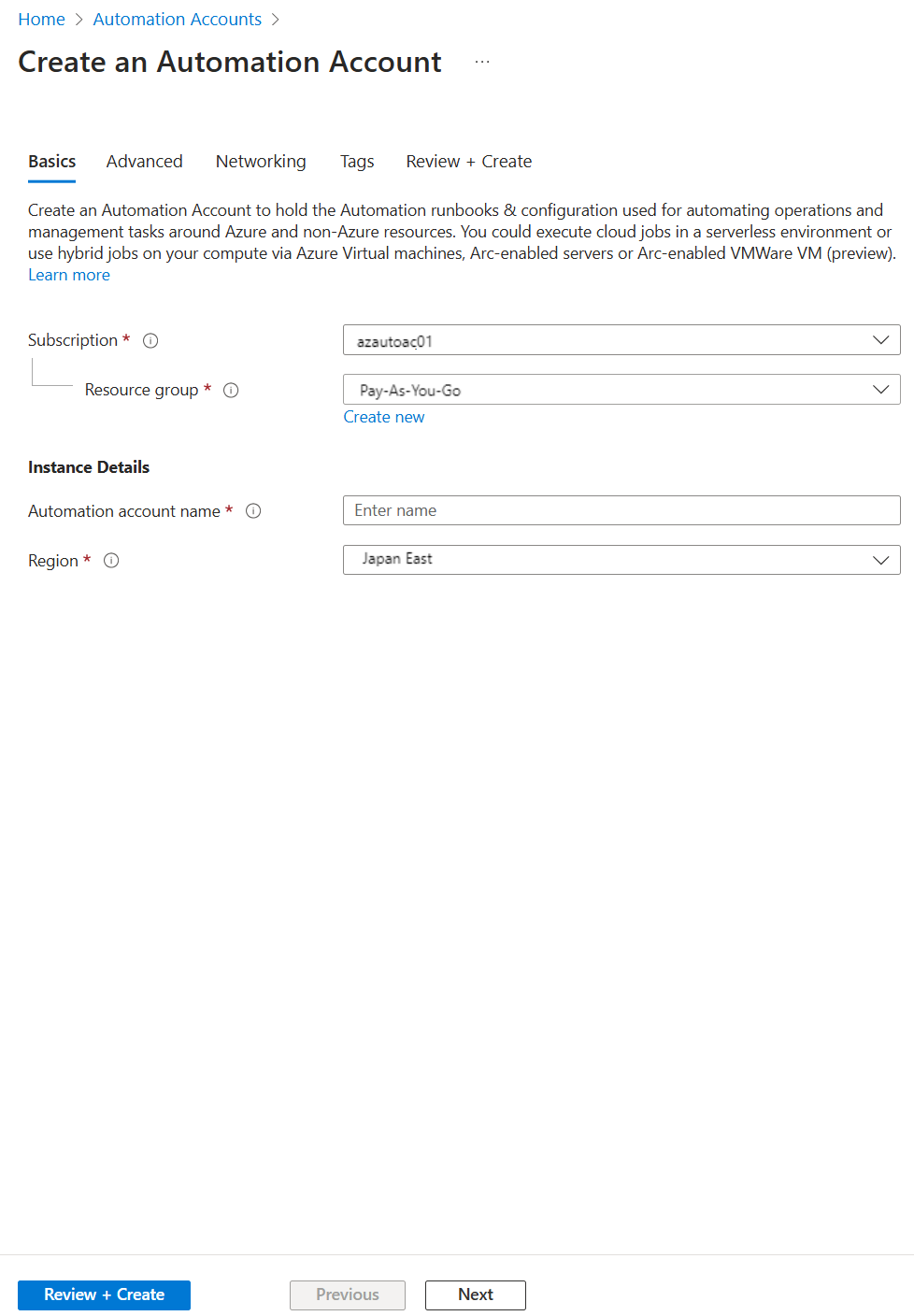
To start using the Microsoft Azure Automation service, you must create an [Automation account](https://azure.microsoft.com/documentation/articles/automation-security-overview/) from within the Azure portal.

Steps to create an Azure Automation account are available on the [Create an Azure Automation account](https://learn.microsoft.com/en-us/azure/automation/quickstarts/create-azure-automation-account-portal) page.

Automation accounts are like Azure Storage accounts, serving as a container to store automation artifacts.

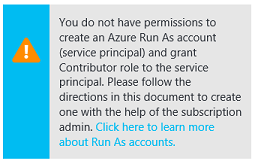
These artifacts could be a container for all your runbooks, runbook executions (*jobs*), and the assets on which your runbooks depend.

An Automation account allows you to manage all Azure resources via an API. To safeguard it, the Automation account creation requires subscription-owner access.



You must be a subscription owner to create the Run As accounts that the service creates.

If you don't have the proper subscription privileges, you'll see the following warning:



You'll need at least one Azure Automation account to use Azure Automation.

However, as a best practice, you should create multiple automation accounts to segregate and limit the scope of access and minimize any risk to your organization.

For example, you might use one account for development, another for production, and another for your on-premises environment. You can have up to 30 Automation accounts.

**Next unit: What is a runbook?**

**What is a runbook?**

Completed100 XP

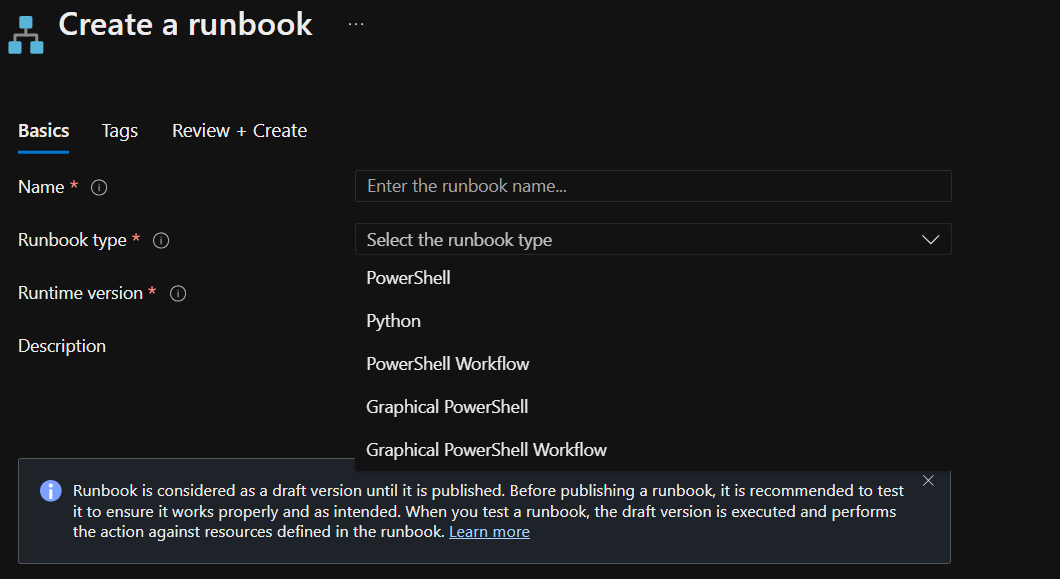
* 4 minutes

**Runbooks** serve as repositories for your custom scripts and workflows.

They also typically reference Automation shared resources such as credentials, variables, connections, and certificates.

Runbooks can also contain other runbooks, allowing you to build more complex workflows.

You can invoke and run runbooks on-demand or according to a schedule using Automation Schedule assets.



**Creating runbooks**

When creating runbooks, you have two options. You can either:

* Create your runbook and import it. For more information about creating or importing a runbook in Azure Automation, go to [Start a runbook in Azure Automation](https://learn.microsoft.com/en-us/azure/automation/start-runbooks).
* Modify runbooks from the runbook gallery. It provides a rich ecosystem of runbooks that are available for your requirements. Visit [Runbook and module galleries for Azure Automation](https://learn.microsoft.com/en-us/azure/automation/automation-runbook-gallery) for more information.

A vibrant open-source community also creates runbooks you can apply directly to your use cases.

You can choose from different runbook types based on your requirements and Windows PowerShell experience.

If you prefer to work directly with Windows PowerShell code, you can use a PowerShell runbook or a PowerShell Workflow runbook.

You can edit offline or with the textual editor in the Azure portal using either of these.

If you prefer to edit a runbook without exposure to the underlying code, you can create a graphical runbook using the Azure portal's graphic editor.

**Graphical runbooks**

Graphical runbooks and Graphical PowerShell Workflow runbooks are created and edited with the graphic editor in the Azure portal.

You can export them to a file and import them into another automation account, but you can't create or edit them with another tool.

**PowerShell runbooks**

PowerShell runbooks are based on Windows PowerShell. You edit the runbook code directly using the text editor in the Azure portal.

You can also use any offline text editor and import the runbook into Azure Automation. PowerShell runbooks don't use parallel processing.

**PowerShell Workflow runbooks**

PowerShell Workflow runbooks are text runbooks based on Windows PowerShell Workflow.

You directly edit the runbook code using the text editor in the Azure portal.

You can also use any offline text editor and import the runbook into Azure Automation.

PowerShell Workflow runbooks use parallel processing to allow for the simultaneous completion of multiple tasks.

Workflow runbooks take longer to start than PowerShell runbooks because they must be compiled before running.

**Python runbooks**

You can directly edit the code of the runbook using the text editor in the Azure portal, or you can use any offline text editor and import the runbook into Azure Automation.

You can also use Python libraries. You must first import the package into the Automation Account to use third-party libraries.

**Note**

You can't convert runbooks from graphical to textual type and vice versa.

For more information on the different types of runbooks, visit [Azure Automation runbook types](https://azure.microsoft.com/documentation/articles/automation-runbook-types).

**Next unit: Understand automation shared resources**

**Understand automation shared resources**

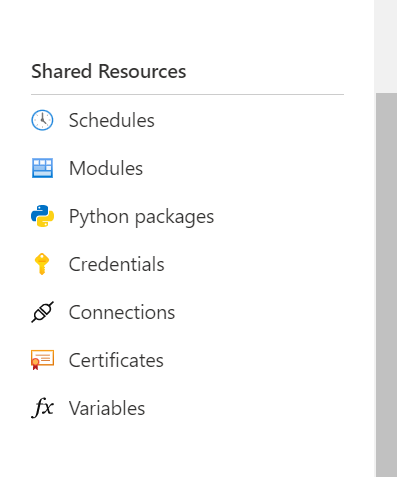
Completed100 XP

* 3 minutes

Azure Automation contains shared resources that are globally available to be associated with or used in a runbook.

There are currently eight shared resources categories:

* **Schedules**: It allows you to define a one-off or recurring schedule.
* **Modules**: Contains Azure PowerShell modules.
* **Modules gallery**: It allows you to identify and import PowerShell modules into your Azure automation account.
* **Python packages**. Allows you to import a Python package by uploading: **.whl** or **tar.gz** packages.
* **Credentials**: It allows you to create username and password credentials.
* **Connections**: It allows you to specify Azure, Azure classic certificate, or Azure Service principal connections.
* **Certificates**: It allows you to upload certificates in .cer or pfx format.
* **Variables**: It allows you to define encrypted or unencrypted variables of types—for example, *String*, *Boolean*, *DateTime*, *Integer*, or no specific type.



As a best practice, always try to create global assets to be used across your runbooks.

It will save time and reduce the number of manual edits within individual runbooks.

**Next unit: Explore runbook gallery**

**Explore runbook gallery**

Completed100 XP

* 3 minutes

Azure Automation runbooks are provided to help eliminate the time it takes to build custom solutions.

The runbooks have already been built by Microsoft and the Microsoft community.

You can use them with or without modification.

Also, you can import runbooks from the runbook gallery at Azure Automation Github in the runbooks repository [Azure Automation - Runbooks](https://github.com/azureautomation/runbooks).

**Note**

The **AzureRM** PowerShell module has been officially deprecated as of February 29, 2024. Users are advised to migrate from AzureRM to the Az PowerShell module to ensure continued support and updates. For more general details on the new Az PowerShell module, go to [**Introducing the new Azure PowerShell Az module**](https://learn.microsoft.com/en-us/powershell/azure/new-azureps-module-az).

**Choosing items from the runbook gallery**

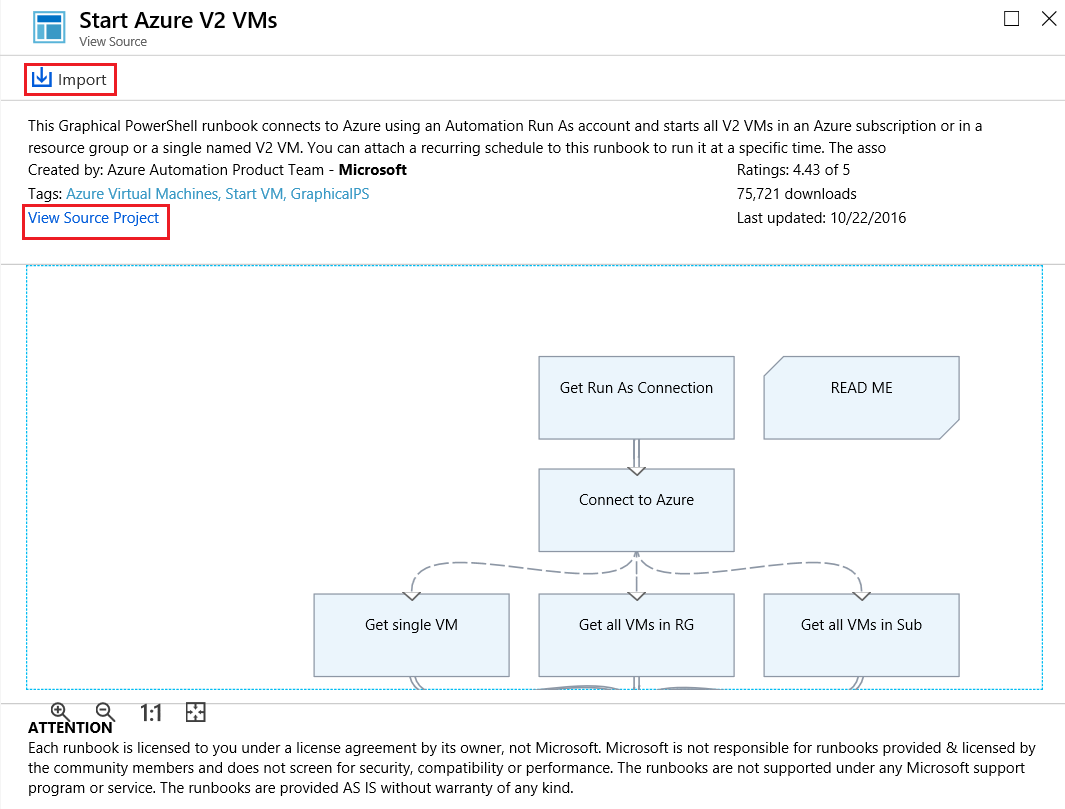
In the Azure portal, you can import directly from the runbook gallery using the following high-level steps:

1. Open your Automation account, and then select **Process Automation** > **Runbooks**.
2. In the runbooks pane, select **Browse gallery**.
3. From the runbook gallery, locate the runbook item you want, select it, and select **Import**.

When browsing through the runbooks in the repository, you can review the code or visualize the code.

You can also check information such as the source project and a detailed description, ratings, and questions and answers.

For more information, see [Azure Automation](https://github.com/azureautomation).



**Note**

Python runbooks are also available from the Azure Automation Github in the runbooks repository. To find them, filter by language and select **Python**.

**Note**

You can't use PowerShell to import directly from the runbook gallery.

**Next unit: Examine webhooks**

**Examine webhooks**

Completed100 XP

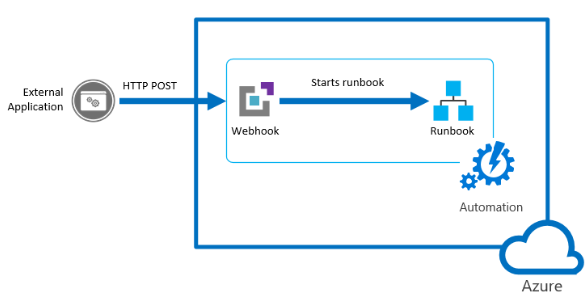
* 4 minutes

You can automate starting a runbook either by scheduling it or by using a webhook.

A **webhook** allows you to start a particular runbook in Azure Automation through a single HTTPS request.

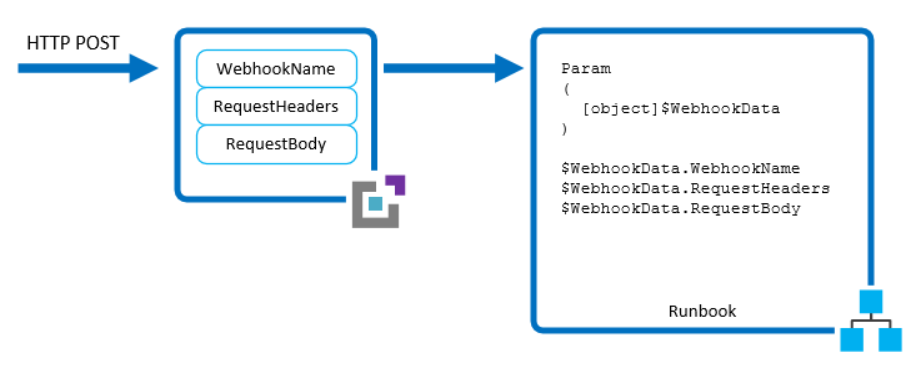
It allows external services such as Azure DevOps, GitHub, or custom applications to start runbooks without implementing more complex solutions using the Azure Automation API.

More information about webhooks is available at [Starting an Azure Automation runbook with a webhook](https://learn.microsoft.com/en-us/azure/automation/automation-webhooks).



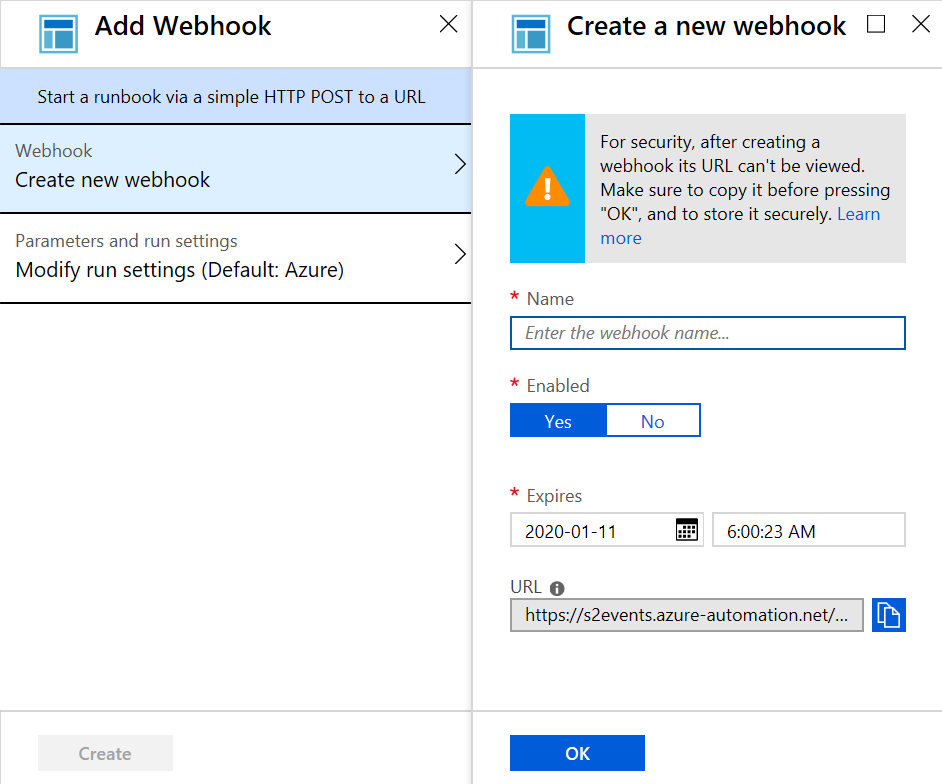
**Create a webhook**

You create a webhook linked to a runbook using the following steps:

1. In the Azure portal, open the runbook that you want to create the webhook.
2. In the runbook pane, under Resources, select **Webhooks**, and then choose **Add webhook**.
3. Select **Create new webhook**.
4. In the **Create new webhook** dialog, there are several values you need to configure. After you configure them, select **Create**:
   * **Name**. Specify any name you want for a webhook because the name isn't exposed to the client. It's only used for you to identify the runbook in Azure Automation.
   * **Enabled**. A webhook is enabled by default when it's created. If you set it to Disabled, then no client can use it.
   * **Expires**. Each webhook has an expiration date, at which time it can no longer be used. You can continue to modify the date after creating the webhook providing the webhook isn't expired.
   * **URL**. The webhook URL is the unique address that a client calls with an HTTP POST to start the runbook linked to the webhook. It's automatically generated when you create the webhook, and you can't specify a custom URL. The URL contains a security token that allows the runbook to be invoked by a third-party system with no further authentication. For this reason, treat it like a password. You can only view the URL in the Azure portal for security reasons when the webhook is created. Make a note of the URL in a secure location for future use. 

**Note**

When creating it, make sure you copy the webhook URL and then store it in a safe place. After you create the webhook, you can't retrieve the URL again.

1. Select the **Parameters run settings (Default: Azure)** option. This option has the following characteristics, which allows you to complete the following actions:
   * If the runbook has mandatory parameters, you'll need to provide these required parameters during creation. You aren't able to create the webhook unless values are provided.
   * If there are no mandatory parameters in the runbook, there's no configuration required here.
   * The webhook must include values for any mandatory parameters of the runbook and include values for optional parameters.
   * When a client starts a runbook using a webhook, it can't override the parameter values defined.
   * To receive data from the client, the runbook can accept a single parameter called $*WebhookData* of type [object] that contains data that the client includes in the POST request.
   * There's no required webhook configuration to support the *$WebhookData* parameter. 
2. When finished, select **Create**.

**Using a webhook**

To use a webhook after it has been created, your client application must issue an HTTP POST with the URL for the webhook.

* The syntax of the webhook is in the following format:

Copy

http://< Webhook Server >/token?=< Token Value >

* The client receives one of the following return codes from the POST request.

Expand table

| **Code** | **Test** | **Description** |
| --- | --- | --- |
| 202 | Accepted | The request was accepted, and the runbook was successfully queued. |
| 400 | Bad request | The request wasn't accepted because the runbook has expired, been disabled, or the token in the URL is invalid. |
| 404 | Not found | The request wasn't accepted because the webhook, runbook, or account wasn't found. |
| 500 | Internal Server Error |  |

* If successful, the webhook response contains the job ID in JSON format as follows:

JSONCopy

{"JobIds":["< JobId >"]}

The response will contain a single job ID, but the JSON format allows for potential future enhancements.

* You can't determine when the runbook job completes or determine its completion status from the webhook. You can only choose this information using the job ID with another method such as PowerShell or the Azure Automation API.

More details are available on the [Starting an Azure Automation runbook with a webhook](https://learn.microsoft.com/en-us/azure/automation/automation-webhooks) page.

**Next unit: Explore source control integration**

**Explore source control integration**

Completed100 XP

* 4 minutes

Azure Automation supports source control integration that enables you to keep your runbooks in your Automation account up to date with your scripts in your GitHub or Azure DevOps source control repository.

Source control allows you to collaborate with your team more efficiently, track changes, and roll back to earlier versions of your runbooks.

For example, source control will enable you to sync different branches in source control to your development, test, or production Automation accounts.

It makes it easier to promote code you've tested in your development environment to your production Automation account.

Azure Automation supports three types of source control:

* GitHub.
* Azure DevOps (Git).
* Azure DevOps (TFVC).

Source control allows you to push code from Azure Automation to source control or pull your runbooks from source control to Azure Automation.

Source control sync jobs run under the user's Automation Account and are billed at the same rate as other Automation jobs.

**Integrate source control with Azure Automation**

You integrate source control with Azure Automation using the following steps:

1. In the Azure portal, access your Automation account.
2. Under Account Settings, select **Source control**, and then choose **+ Add**.
3. In the **Source Control type**, select **GitHub** as source control type and then select **Authenticate**.

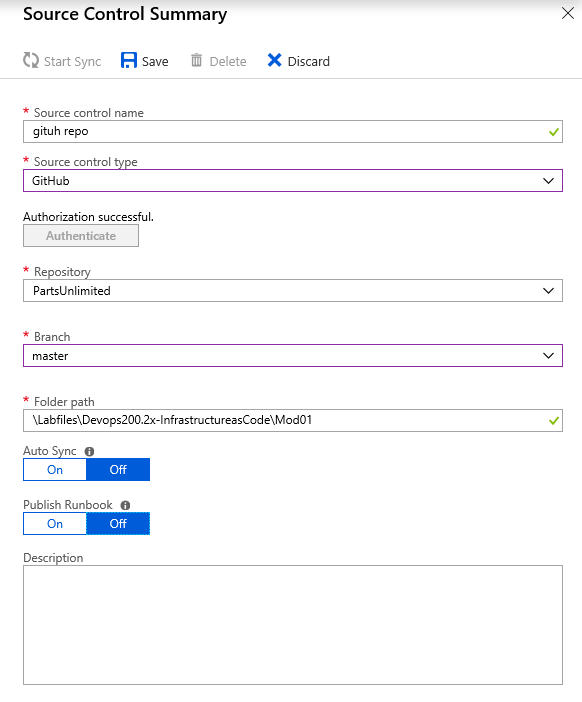
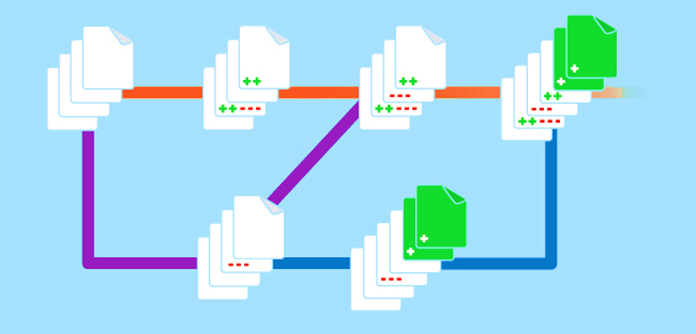
**Note**

Note: You'll require a GitHub account to complete the next step.

1. When the browser page opens, prompting you to authenticate to [https://www.github.com](https://www.github.com/), select **Authorize azureautomation** and enter your GitHub account password. If successful, you should receive an email notification from GitHub stating that *A third-party OAuth Application (Automation Source Control) with repo scope was recently authorized to access your account.*
2. After authentication completes, fill in the details based on the following table, and then select **Save**.

Expand table

| **Property** | **Description** |
| --- | --- |
| Name | Friendly name |
| Source control type | GitHub, Azure DevOps Git, or Azure DevOps TFVC |
| Repository | The name of the repository or project |
| Branch | The branch from which to pull the source files. Branch targeting isn't available for the TFVC source control type. |
| Folder Path | The folder that contains the runbooks to sync. |
| Autosync | Turns on or off automatic sync when a commit is made in the source control repository. |
| Publish Runbook. | If set to **On**, after runbooks are synced from source control, they'll be automatically published. |
| Description | A text field to provide more details. |

1. If you set **Autosync** to **Yes**, full sync will start. If you set **Autosync** to **No**, open the **Source Control Summary** blade again by selecting your repository in Azure Automation and then selecting **Start Sync**.  
     
   
2. Verify that your source control is listed on the **Azure Automation Source control** page for you to use.  
     
   

**Next unit: Explore PowerShell workflows**

**Explore PowerShell workflows**

Completed100 XP

* 4 minutes

IT pros often automate management tasks for their multi-device environments by running sequences of long-running tasks or workflows.

These tasks can affect multiple managed computers or devices at the same time.

PowerShell Workflow lets IT pros and developers apply the benefits of Windows Workflow Foundation with the automation capabilities and ease of using Windows PowerShell.

**Tip**

Refer to [**A Developer's Introduction to Windows Workflow Foundation (WF) in .NET 4**](https://learn.microsoft.com/en-us/previous-versions/dotnet/articles/ee342461%28v=msdn.10%29) for more information.

Windows PowerShell Workflow functionality was introduced in Windows Server 2012 and Windows 8 and is part of Windows PowerShell 3.0 and later.

Windows PowerShell Workflow helps automate distribution, orchestration, and completion of multi-device tasks, freeing users and administrators to focus on higher-level tasks.

**Activities**

An **activity** is a specific task that you want a workflow to do. Just as a script is composed of one or more commands, a workflow is composed of activities carried out in sequence.

You can also use a script as a single command in another script and use a workflow as an activity within another workflow.

**Workflow characteristics**

A workflow can:

* Be long-running.
* Be repeated over and over.
* Run tasks in parallel.
* Be interrupted—can be stopped and restarted, suspended, and resumed.
* Continue after an unexpected interruption, such as a network outage or computer/server restart.

**Workflow benefits**

A workflow offers many benefits, including:

* Windows PowerShell scripting syntax. Is built on PowerShell.
* Multidevice management. Simultaneously apply workflow tasks to hundreds of managed nodes.
* Single task runs multiple scripts and commands. Combine related scripts and commands into a single task. Then run the single task on multiple computes. The activity status and progress within the workflow are visible at any time.
* Automated failure recovery.
  + Workflows survive both planned and unplanned interruptions, such as computer restarts.
  + You can suspend a workflow operation, then restart or resume the workflow from the point it was suspended.
  + You can author checkpoints as part of your workflow so that you can resume the workflow from the last persisted task (or checkpoint) instead of restarting the workflow from the beginning.
* Connection and activity retries. You can retry connections to managed nodes if network-connection failures occur. Workflow authors can also specify activities that must run again if the activity cannot be completed on one or more managed nodes (for example, if a target computer was offline while the activity was running).
* Connect and disconnect from workflows. Users can connect and disconnect from the computer running the workflow, but the workflow will remain running. For example, suppose you're running the workflow and managing the workflow on two different computers. In that case, you can sign out of or restart the computer from which you're managing the workflow and continue to monitor workflow operations from another computer without interrupting the workflow.
* Task scheduling. You can schedule a task to start when specific conditions are met, as with any other Windows PowerShell cmdlet or script.

**Next unit: Create a workflow**

**Create a workflow**

Completed100 XP

* 3 minutes

Use a script editor such as the Windows PowerShell Integrated Scripting Environment (ISE) to write the workflow.

It enforces workflow syntax and highlights syntax errors. For more information, review the tutorial [Tutorial - Create a PowerShell Workflow runbook in Azure Automation](https://learn.microsoft.com/en-us/azure/automation/learn/automation-tutorial-runbook-textual).

A benefit of using PowerShell ISE is that it automatically compiles your code and allows you to save the artifact.

Because the syntactic differences between scripts and workflows are significant, a tool that knows both workflows and scripts will save you considerable coding and testing time.

**Syntax**

When you create your workflow, begin with the **workflow** keyword, which identifies a workflow command to PowerShell.

A script workflow requires the **workflow** keyword. Next, name the workflow, and have it follow the **workflow** keyword.

The body of the workflow will be enclosed in braces.

1. A workflow is a Windows command type, so select a name with a verb-noun format:

PowerShellCopy

workflow Test-Workflow

{

...

}

1. To add parameters to a workflow, use the **Param** keyword. It's the same techniques that you use to add parameters to a function.
2. Finally, add your standard PowerShell commands.

PowerShellCopy

workflow MyFirstRunbook-Workflow

{

Param(

[string]$VMName,

[string]$ResourceGroupName

)

....

Start-AzureRmVM -Name $VMName -ResourceGroupName $ResourceGroupName

}

**Next unit: Explore hybrid management**

**Explore hybrid management**

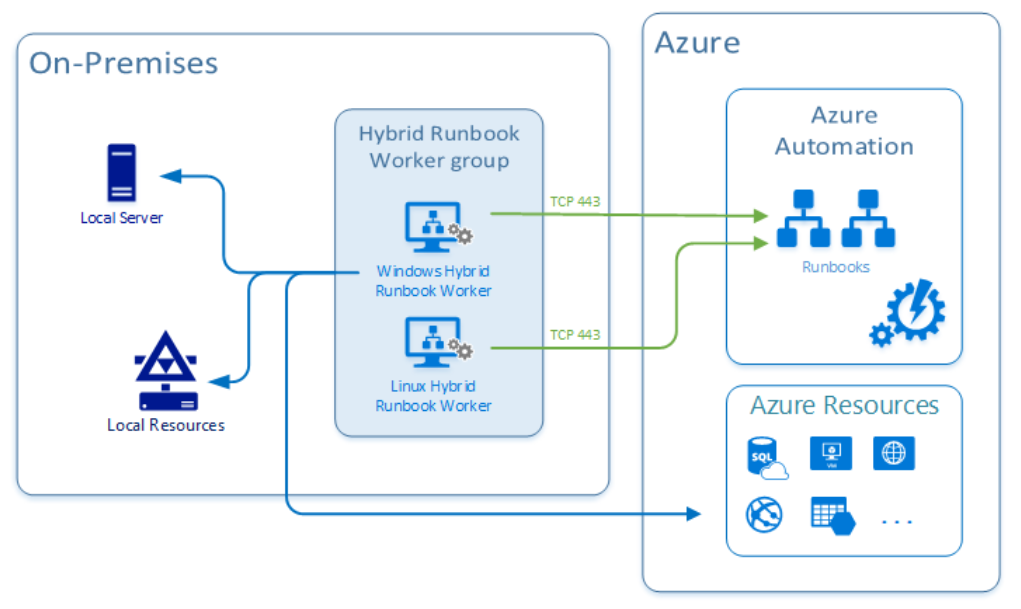
Completed100 XP

* 3 minutes

The Hybrid Runbook Worker feature of Azure Automation allows you to run runbooks that manage local resources in your private data center on machines located in your data center.

Azure Automation stores and manages the runbooks and then delivers them to one or more on-premises machines.

The Hybrid Runbook Worker functionality is presented in the following graphic:



**Hybrid Runbook Worker workflow and characteristics**

The following list is characteristics of the Hybrid Runbook Worker workflow:

* You can select one or more computers in your data center to act as a Hybrid Runbook Worker and then run runbooks from Azure Automation.
* Each Hybrid Runbook Worker is a member of a Hybrid Runbook Worker group, which you specify when you install the agent.
* A group can include a single agent, but you can install multiple agents in a group for high availability.
* There are no inbound firewall requirements to support Hybrid Runbook Workers, only Transmission Control Protocol (TCP) 443 is required for outbound internet access.
* The agent on the local computer starts all communication with Azure Automation in the cloud.
* When a runbook is started, Azure Automation creates an instruction that the agent retrieves. The agent then pulls down the runbook and any parameters before running it.

To configure your on-premises servers that support the Hybrid Runbook Worker role with DSC, you must add them as DSC nodes.

For more information about onboarding them for management with DSC, see [Onboarding machines for management by Azure Automation State Configuration](https://learn.microsoft.com/en-us/azure/automation/automation-dsc-onboarding).

For more information on installing and removing Hybrid Runbook Workers and groups, see:

* [Automate resources in your datacenter or cloud by using Hybrid Runbook Worker.](https://learn.microsoft.com/en-us/azure/automation/automation-hybrid-runbook-worker#installing-hybrid-runbook-worker)
* [Hybrid Management in Azure Automation](https://azure.microsoft.com/blog/hybrid-management-in-azure-automation/)

**Next unit: Examine checkpoint and parallel processing**

[**Previous**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/9-create-workflow/)

**Examine checkpoint and parallel processing**

Completed100 XP

* 4 minutes

Workflows let you implement complex logic within your code. Two features available with workflows are checkpoints and parallel processing.

**Checkpoints**

A **checkpoint** is a snapshot of the current state of the workflow.

Checkpoints include the current value for variables and any output generated up to that point. (For more information on what a checkpoint is, read the [checkpoint](https://learn.microsoft.com/en-us/azure/automation/automation-powershell-workflow) webpage.)

If a workflow ends in an error or is suspended, the next time it runs, it will start from its last checkpoint instead of at the beginning of the workflow.

You can set a checkpoint in a workflow with the **Checkpoint-Workflow** activity.

For example, if an exception occurs after Activity2, the workflow will end in the following sample code.

When the workflow is rerun, it starts with Activity2, followed just after the last checkpoint set.

PowerShellCopy

<Activity1>

Checkpoint-Workflow

<Activity2>

<Exception>

<Activity3>

**Parallel processing**

A script block has multiple commands that run concurrently (or *in parallel*) instead of sequentially, as for a typical script.

It's referred to as *parallel processing*. (More information about parallel processing is available on the [Parallel processing](https://learn.microsoft.com/en-us/azure/automation/automation-powershell-workflow) webpage.)

In the following example, two *vm0* and *vm1* VMs will be started concurrently, and *vm2* will only start after *vm0* and *vm1* have started.

PowerShellCopy

Parallel

{

Start-AzureRmVM -Name $vm0 -ResourceGroupName $rg

Start-AzureRmVM -Name $vm1 -ResourceGroupName $rg

}

Start-AzureRmVM -Name $vm2 -ResourceGroupName $rg

Another parallel processing example would be the following constructs that introduce some extra options:

* **ForEach -Parallel**. You can use the **ForEach -Parallel** construct to concurrently process commands for each item in a collection. The items in the collection are processed in parallel while the commands in the script block run sequentially.

In the following example, *Activity1* starts at the same time for all items in the collection.

For each item, *Activity2* starts after *Activity1* completes. *Activity3* starts only after both *Activity1* and *Activity2* have been completed for all items.

* *ThrottleLimit* - We use the *ThrottleLimit* parameter to limit parallelism. Too high of a *ThrottleLimit* can cause problems. The ideal value for the *ThrottleLimit* parameter depends on several environmental factors. Try starting with a low *ThrottleLimit* value, and then increase the value until you find one that works for your specific circumstances:

PowerShellCopy

ForEach -Parallel -ThrottleLimit 10 ($<item> in $<collection>)

{

<Activity1>

<Activity2>

}

<Activity3>

A real-world example of it could be similar to the following code: a message displays for each file after it's copied. Only after all files are copied does the completion message display.

PowerShellCopy

Workflow Copy-Files

{

$files = @("C:\LocalPath\File1.txt","C:\LocalPath\File2.txt","C:\LocalPath\File3.txt")

ForEach -Parallel -ThrottleLimit 10 ($File in $Files)

{

Copy-Item -Path $File -Destination \\NetworkPath

Write-Output "$File copied."

}

Write-Output "All files copied."

}

**Next unit: Knowledge check**

**Knowledge check**

Completed200 XP

* **Module assessment**
* 5 minutes

 Answer 100% of questions correctly in order to pass. [**Retake**](https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/12-knowledge-check)

Dismiss alert

Choose the best response for each question.

**Check your knowledge**

Top of Form

**1.**

**Which of the following choices isn't an option when creating runbooks?**

Import runbooks from Azure Artifacts.

**Correct. When creating runbooks, you have two options. You can either create your runbook and import it or modify runbooks from the runbook gallery.**

Create your runbook and import it.

Modify runbooks from the runbook gallery.

**Incorrect. When creating runbooks, you have two options. You can either create your runbook and import it or modify runbooks from the runbook gallery.**

**2.**

**Which of the following choices isn't a supported source control by Azure Automation?**

Azure DevOps (Git or Team Foundation Version Control).

**Incorrect. Azure Automation supports three types of source control: GitHub, Azure DevOps (Git), and Azure DevOps (Team Foundation Version Control).**

BitBucket.

**Correct. Azure Automation supports three types of source control: GitHub, Azure DevOps (Git), and Azure DevOps (Team Foundation Version Control).**

GitHub.

**3.**

**Which of the following choices is the required keyword at the beginning when creating your workflow to PowerShell?**

ResourceGroupName.

Param.

Workflow.

**Correct. When you create your workflow, begin with the workflow keyword, which identifies a workflow command to PowerShell.**

Bottom of Form

“”” <https://learn.microsoft.com/en-us/training/modules/explore-azure-automation-devops/13-summary#completion> “”