# My first PowerShell Workflow runbook

This tutorial walks you through the creation of a [PowerShell Workflow runbook](/documentation/articles/automation-runbook-types#powerShell-workflow-runbooks) in Azure Automation. We’ll start with a simple runbook that we’ll test and publish while we explain how to track the status of the runbook job. Then we’ll modify the runbook to actually manage Azure resources, in this case starting an Azure virtual machine. We’ll then make the runbook more robust by adding runbook parameters.

## Prerequisites

To complete this tutorial, you will need the following.

* Azure subscription. If you don’t have one yet, you can [sign up for a trial](/pricing/1rmb-trial/).
* [Automation account](/documentation/articles/automation-configuring) to hold the runbook.
* An Azure virtual machine. We will stop and start this machine so it should not be production.
* [Azure Active Directory user and Automation Credential asset](/documentation/articles/automation-configuring) to authenticate to Azure resources. This user must have permission to start and stop the virtual machine.

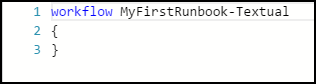
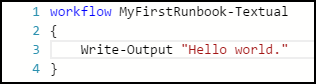
## Step 1 - Create new runbook

We’ll start by creating a simple runbook that outputs the text *Hello World*.

1. In the Azure Management Portal, click **NEW** > **APP SERVICES** > **AUTOMATION** > **RUNBOOK** > **QUICK CREATE**
2. Enter the **RUNBOOK NAME**, **DESCRIPTION**, and choose an **AUTOMATION ACCOUNT**. If you don’t have one, you can create one by choosing **Create a new automation account** and entering the **ACCOUNT NAME**
3. After the runbook being created, you can find it under the **RUNBOOKS** tile of you automation account.
4. Click the **AUTHOR** tile of you runbook to enable the textual editor.

## Step 2 - Add code to the runbook

You can either type code directly into the runbook, or you can select cmdlets, runbooks, and assets from the Library control and have them added to the runbook with any related parameters. For this walkthrough, we’ll type directly into the runbook.

1. Our runbook is currently empty with only the required *workflow* keyword, the name of our runbook, and the braces that will encase the entire workflow. 
2. Type *Write-Output “Hello World.”* between the braces. 
3. Save the runbook by clicking **Save**.

## Step 3 - Test the runbook

Before we publish the runbook to make it available in production, we want to test it to make sure that it works properly. When you test a runbook, you run its **Draft** version and view its output interactively.

1. Click **Test** to open the Test pane. And then, click **Yes** to confirm.
2. After a few seconds you will see output pane printing the message *“Hello World.”*

## Step 4 - Publish and start the runbook

The runbook that we just created is still in Draft mode. We need to publish it before we can run it in production. When you publish a runbook, you overwrite the existing Published version with the Draft version. In our case, we don’t have a Published version yet because we just created the runbook.

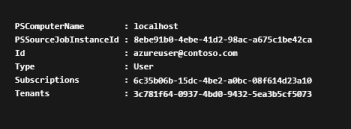
1. Click **Publish** to publish the runbook and then **Yes** when prompted.
2. If you view the runbook in the **Runbooks** tile of you automation account now, it will show an **Authoring Status** of **Published**.
3. Click **Start** to start the runbook and then **Yes** when prompted.
4. Click **View Job** to see the summary of the job you just started.
5. The job status is shown in **Job Summary**, and after a few seconds, the same output as the testing you have done earlier will be shown under **output**.
6. Go back to your runbook. Under the **Jobs** tile, you can seen a list of job created will filter above.
7. Click a job, you can see the **summary** and **history** of the job.

## Step 5 - Add authentication to manage Azure resources

We’ve tested and published our runbook, but so far it doesn’t do anything useful. We want to have it manage Azure resources. It won’t be able to do that though unless we have it authenticate using the credentials that are referred to in the [prerequisites](#prerequisites). We do that with the **Add-AzureAccount** cmdlet.

1. Open the textual editor by clicking **Author** tile > **Draft** tag > **Edit runbook**.
2. We don’t need the **Write-Output** line anymore, so go ahead and delete it.
3. Position the cursor on a blank line between the braces.
4. Click **Insert** > **Setting** > **Get Windows PowerShell Credential**, choose the credential you want.
5. If you don’t have a credential, you can add one by clicking **Manage** > **Add Credential** to create one. For more information, see [Azure Active Directory user and Automation Credential asset](/documentation/articles/automation-configuring).
6. In front of **Get-AutomationPSCredential**, type *$Credential =* to assign the credential to a variable.
7. On the next line, type *Add-AzureAccount -Credential $Credential –Environment AzureChinaCloud*.

* workflow test  
  {  
   $Credential = Get-AutomationPSCredential -Name "<your credential>"  
   Add-AzureAccount –Credential $Credential –Environment AzureChinaCloud  
  }

1. Click **Test** and then **Yes** when prompted.
2. Once it completes, you should receive output similar to the following that returns the information for the user in the credential. This confirms that the credential is valid. 

## Step 6 - Add code to start a virtual machine

Now that our runbook is authenticating to our Azure subscription, we can manage resources. We’ll add a command to start a virtual machine. You can pick any virtual machine in your Azure subscription, and for now we’ll be hardcoding that name into the cmdlet.

1. After *Add-AzureAccount*, type *Start-AzureVM -Name ‘VMName’ -ServiceName ‘VMServiceName’* providing the name and service name of the virtual machine to start.

* workflow test  
  {  
   $Credential = Get-AutomationPSCredential -Name "<your credential>"  
   Add-AzureAccount –Credential $Credential –Environment AzureChinaCloud  
   Start-AzureVM -Name "<your vm>" -ServiceName "<your vm service>"  
  }

1. Save the runbook and then click **Test** so that we can test it.

## Step 7 - Add an input parameter to the runbook

Our runbook currently starts the virtual machine that we hardcoded in the runbook, but it would be more useful if we could specify the virtual machine when the runbook is started. We will now add input parameters to the runbook to provide that functionality.

1. Add parameters for *VMName* and *VMServiceName* to the runbook and use these variables with the **Start-AzureVM** cmdlet as in the following image. workflow test { Param ( [string]VMServiceName )

* $Credential = Get-AutomationPSCredential -Name "<your credential>"  
   Add-AzureAccount –Credential $Credential –Environment AzureChinaCloud  
   Start-AzureVM -Name $VMName -ServiceName $VMServiceName  
  }

1. Save the runbook and open the Test pane. Note that you can now provide values for the two input variables that will be used in the test.
2. Close the Test pane.
3. Click **Publish** to publish the new version of the runbook.
4. Stop the virtual machine that you started in the previous step.
5. Click **Start** to start the runbook. Type in the **VMName** and **VMServiceName** for the virtual machine that you’re going to start.
6. When the runbook completes, check that the virtual machine was started.