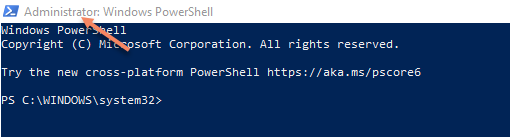
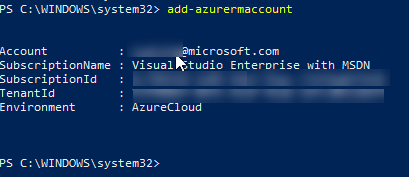
Lab 1 - The Basics

Lab 1: Part 1: Installing Azure PowerShell

**Installing Azure PowerShell: in this section you will learn how to install Azure PowerShell**

* + Start Windows PowerShell using the Run as Administrator option (Right click the PowerShell Icon and select Run as Administrator), Notice the window title containing the word "Administrator" in the title.
  + 
  + Type install-module azurerm -allowclobber
  + To login, type add-azurermaccount, a window prompt will request credentials
  + Provide your username and password
  + Once logged-in the PowerShell window will show your default subscription information
  + 

* + To get a list of available subscription to your account, type get-azurermsubscription, and capture name of subscription used for this training
  + To switch to between subscription use select-azurermsubscription '{subscriptionname'}

Lab 1: Part 2: Create VM using PowerShell via cloud shell

* + **Launch Azure Cloud Shell,**
  + **Create an Azure resource group** with New-AzResourceGroup. A resource group is a logical container into which Azure resources are deployed and managed.
    - type new-azurermresourcegroup -name 'az{youralias}automationrg' -location UK West
  + **Create virtual machine,** Create a VM with New-AzVM. Provide names for each of the resources and the New-AzVM cmdlet creates if they don't already exist. When prompted, provide a username and password to be used as the sign-in credentials for the VM:
    - Type:

New-AzVm `

-ResourceGroupName "az{youralias}automationrg" `

-Name "az{youralias}VM" `

-Location "UK West" `

-VirtualNetworkName "az{youralias}Vnet" `

-SubnetName "az{youralias}Subnet" `

-SecurityGroupName "az{youralias}NetworkSecurityGroup" `

-PublicIpAddressName "az{youralias}PublicIpAddress" `

-OpenPorts 80,3389

* + **To see the public IP address of the VM**, use the Get-AzPublicIpAddress cmdlet:
    - Get-AzPublicIpAddress -ResourceGroupName "az{youralias}automationrg" | Select "IpAddress"
  + Use the following command to create a remote desktop session from your local computer. Replace the IP address with the public IP address of your VM.mstsc /v:{publicIpAddress}
  + In the Windows Security window, select More choices, and then select Use a different account. Type the username as localhost\username, enter password you created for the virtual machine, and then click OK. You may receive a certificate warning during the sign-in process. Click Yes or Continue to create the connection
  + To see your VM in action, install the IIS web server. Open a PowerShell prompt on the VM and run the following command:
    - Install-WindowsFeature -name Web-Server -IncludeManagementTools
    - With IIS installed and port 80 now open on your VM from the Internet, use a web browser of your choice to view the default IIS welcome page. Use the public IP address of your VM obtained in a previous step.
  + When no longer needed, you can use the Remove-AzResourceGroup cmdlet to remove the resource group, VM, and all related resources:// Do not delete the VM or the RG as you will need it later in the Lab..
    - Remove-AzResourceGroup -Name az{youralias}automationrg

Lab1: Part 3: Create Azure Automation Account

**Creating Azure Automation Account:** In this section you will create an Azure Automation Account using PowerShell and the Azure portal

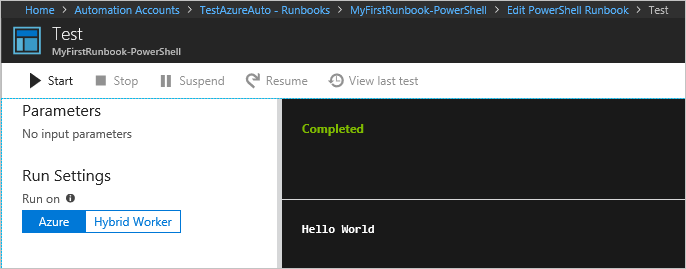
* + To create using PowerShell
    - Start windows powershell
    - Login to the subscription you are using for the purposes of this lab
    - Type, new-azurermautomationaccount -name 'az{youralias}autoaccount' -resourcegroup 'az{youralias}automationrg' -location ukwest
    - To verify type, get-azurermautomationaccount

* + To create using the Azure Portal
    - Sign in to Azure [https://portal.azure.com](https://portal.azure.com/)
    - Click the Create a resource button found on the upper left-hand corner of Azure.
    - Mi&osoft Azure (Preview) 
      Azure services 
      Create a 
      resource 
      All resources 
      Storage 
      accounts 
      Virtual 
      machines 
      Recent resources 
      Name 
      Web Apps 
      Cogniti' 
      Storage 
      B2C Ter 
      Resourc 
      Networ 
    - Select IT & Management Tools, and then select Automation.
    - Machine generated alternative text:
      Home > 
      New 
      New 
      p Search the Marketplace 
      Azure Marketplace 
      See all 
      Get started 
      Recently created 
      Recently created 
      Al + Machine Learning 
      Analytics 
      Blockchain 
      Compute 
      Containers 
      Databases 
      Developer Tools 
      DevOps 
      Identity 
      Integration 
      Internet of Things 
      Media 
      Mixed Reality 
      IT & Management Tools 
      Networking 
      Software as a Service (SaaS) 
      Featured 
      See all 
      Azure Security Center 
      Learn more 
      Azure Migrate 
      Quickstart tutorial 
      Application Insights 
      Quickstart tutorial 
      Log Analytics 
      Quickstart tutorial 
      Automati 
      Learn more 
      Backup and Site Recovery 
      Quickstart tutorial 
      Splunk Enterprise (preview) 
      Learn more 
      PREVIEW 
    - Enter the account information. For Create Azure Run As account, choose Yes so that the artifacts to simplify authentication to Azure are enabled automatically. It is important to note, that when creating an Automation Account, the name cannot be changed after it is chosen. Automation Account names are unique per region and resource group. Names for Automation Accounts that were deleted may not be immediately available. One Automation Account can manage resources across all regions and subscriptions for a given tenant. When complete, click Create, to start the Automation account deployment.
    - Microsoft Azure (Preview) 
      p Search resources, services, a 
      Home > New > Marketplace > Automation Add Automation Account 
      Add Automation Account 
      Name 
      Enter the account name... 
      Subscription * 
      Microsoft Azure Internal Consumptio 
      Resource group 
      Select existing... 
      Create new 
      Location * 
      Japan East 
      Create Azu 
      n As account * 
      No 
      This will create Azure Run As 
      account in the Automation 
      account which are useful for 
      authenticating with Azure to 
      manage Azure resources from 
      Automation runbooks. Note 
      that the creation of Azure Run 
      As account may affect the 
      security of the 
      subscription.Learn more 
    - When the deployment has completed, click All Services, select Automation Accounts and select the Automation Account you created.
  + Run one of the tutorial runbooks.
    - Click Runbooks under PROCESS AUTOMATION. The list of runbooks is displayed. By default several tutorial runbooks are enabled in the account.
    - Select the AzureAutomationTutorialScript runbook. This action opens the runbook overview page.
    - Click Start, and on the Start Runbook page, click OK to start the runbook.
    - After the Job status becomes Running, click Output or All Logs to view the runbook job output. For this tutorial runbook, the output is a list of your Azure resources.

Lab 2: Automate Runbooks

Lab 2: Part 1: Create a PowerShell Runbook

* + **Create a Runbook:** in this section you will practice creating a Runbook and also you will check the runbook gallery, You start with a simple runbook that you test and publish while you learn how to track the status of the runbook job. Then you modify the runbook to actually manage Azure resources, in this case starting an Azure virtual machine. Lastly, you make the runbook more robust by adding runbook parameters. You start by creating a simple runbook that outputs the text Hello World.

* + In the Azure portal, open your Automation account.
  + Click Runbooks under Process Automation to open the list of runbooks.
  + Create a new runbook by clicking the + Add a runbook button and then Create a new runbook.
  + Give the runbook the name az{youralias}MyFirstRunbook-PowerShell.
  + In this case, you're going to create a PowerShell runbook so select Powershell for Runbook type.
  + Click Create to create the runbook and open the textual editor.
  + type Write-Output "Hello World." in the body of the script.
  + Save the runbook by clicking Save
  + Test the runbook, Click Test pane to open the Test pane.
  + Click Start to start the test. This should be the only enabled option, a runbook job is created and its status displayed. The job status starts as Queued indicating that it's waiting for a runbook worker in the cloud to come available. It moves to Starting when a worker claims the job, and then Running when the runbook actually starts running. When the runbook job completes, its output is displayed. ou should see Hello World.
  + 
  + Close the Test pane to return to the canvas.
  + The runbook that you created is still in Draft mode. It must be published before you can run it in production. When you publish a runbook, you overwrite the existing Published version with the Draft version. In your case, you don't have a Published version yet because you just created the runbook.
    - Click Publish to publish the runbook and then Yes when prompted.

* + If you scroll left to view the runbook in the Runbooks pane now, it shows an Authoring Status of Published.

* + Scroll back to the right to view the pane for MyFirstRunbook-PowerShell. The options across the top allows to start the runbook, view the runbook, schedule it to start at some time in the future, or create a webhook so it can be started through an HTTP call.

* + You want to start the runbook, so click Start and then click Ok when the Start Runbook page opens.

* + A job page is opened for the runbook job that you created. You can close this pane, but in this case you leave it open so you can watch the job's progress.

* + The job status is shown in Job Summary and matches the statuses that you saw when you tested the runbook.
  + Once the runbook status shows Completed, under Overview click Output. The Output pane is opened, and you can see your Hello World.

* + Machine generated alternative text:
    Overview 
    Input 
    Errors 
    Output 
    Ou tpu t 
    Warnings 
    All Logs 
  + Close the Output page.

* + Click All Logs to open the Streams pane for the runbook job. You should only see Hello World in the output stream, but this output can show other streams for a runbook job such as Verbose and Error if the runbook writes to them.
  + Machine generated alternative text:
    Overview 
    Input 
    Errors 
    Output 
    Ou tpu t 
    Warnings 
    All Logs 
  + Close the Streams page and the Job page to return to the MyFirstRunbook-PowerShell page.

* + Under Details, click Jobs to open the Jobs pane for this runbook. This page lists all of the jobs created by this runbook. You should only see one job listed since you only ran the job once.
  + You can click this job to open the same Job pane that you viewed when you started the runbook. This action allows you to go back in time and view the details of any job that was created for a particular runbook.
  + You've tested and published your runbook, but so far it doesn't do anything useful. You want to have it manage Azure resources. It is not able to do that though unless You have it authenticate using a Run As connection that is automatically created when you create your automation account. You use the Run As connection with the Connect-AzureRmAccount cmdlet. If you are managing resources across multiple subscriptions, you need to use the -AzureRmContext parameter along with Get-AzureRmContext.
    - Open the textual editor by clicking Edit on the MyFirstRunbook-PowerShell page.

* + You don't need the Write-Output line anymore, so go ahead and delete it.

* + Type or copy and paste the following code that handles the authentication with your Automation Run As account:

# Ensures you do not inherit an AzureRMContext in your runbook

Disable-AzureRmContextAutosave –Scope Process

$connection = Get-AutomationConnection -Name AzureRunAsConnection

while(!($connectionResult) -And ($logonAttempt -le 10))

{

$LogonAttempt++

# Logging in to Azure...

$connectionResult = Connect-AzureRmAccount `

-ServicePrincipal `

-Tenant $connection.TenantID `

-ApplicationID $connection.ApplicationID `

-CertificateThumbprint $connection.CertificateThumbprint

Start-Sleep -Seconds 30

}

* + Click Test pane so that you can test the runbook.

* + Click Start to start the test. Once it completes, you should receive output similar to the following, displaying basic information from your account. This output confirms that the Run As Account is valid.
  + Machine generated alternative text:
    Completed 
    Envi ronments 
    {[AzureC10ud, AzureC10ud] , 
    [AzureChinaC10ud, AzureChinaC10ud] , 
    Context 
    [AzureUSGovernment, AzureUSCovernment]} Microsoft. Azur... 
  + Now that your runbook is authenticating to your Azure subscription, you can manage resources. You add a command to start a virtual machine. You can pick any virtual machine in your Azure subscription, and for now you hardcode that name in the runbook.

* + After Connect-AzureRmAccount, type Start-AzureRmVM -Name 'az{youralias}VM' -ResourceGroupName 'az{youralias}automationrg' providing the name and Resource Group name of the virtual machine to start.
    - # Ensures you do not inherit an AzureRMContext in your runbook

Disable-AzureRmContextAutosave –Scope Process

$connection = Get-AutomationConnection -Name AzureRunAsConnection

while(!($connectionResult) -And ($logonAttempt -le 10))

{

$LogonAttempt++

# Logging in to Azure...

$connectionResult = Connect-AzureRmAccount `

-ServicePrincipal `

-Tenant $connection.TenantID `

-ApplicationID $connection.ApplicationID `

-CertificateThumbprint $connection.CertificateThumbprint

Start-Sleep -Seconds 30

}

Start-AzureRmVM -Name 'az{youralias}VM' -ResourceGroupName 'az{youralias}automationrg'

* + Save the runbook and then click Test pane so that you can test it.

* + Click Start to start the test. Once it completes, check that the virtual machine was started.
  + Your runbook currently starts the virtual machine that you hardcoded in the runbook, but it would be more useful if you specify the virtual machine when the runbook is started. You add input parameters to the runbook to provide that functionality.
    - Add parameters for VMName and ResourceGroupName to the runbook and use these variables with the Start-AzureRmVM cmdlet as in the following example.
      * Param(

[string]$VMName,

[string]$ResourceGroupName

)

# Ensures you do not inherit an AzureRMContext in your runbook

Disable-AzureRmContextAutosave –Scope Process

$connection = Get-AutomationConnection -Name AzureRunAsConnection

while(!($connectionResult) -And ($logonAttempt -le 10))

{

$LogonAttempt++

# Logging in to Azure...

$connectionResult = Connect-AzureRmAccount `

-ServicePrincipal `

-Tenant $connection.TenantID `

-ApplicationID $connection.ApplicationID `

-CertificateThumbprint $connection.CertificateThumbprint

Start-Sleep -Seconds 30

}

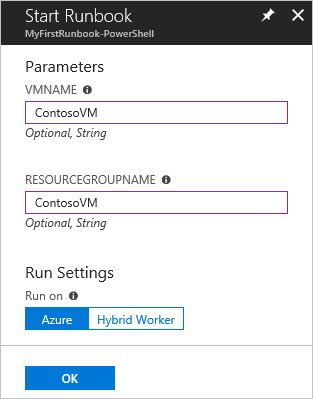
Start-AzureRmVM -Name $VMName -ResourceGroupName $ResourceGroupName

* + Save the runbook and open the Test pane. You can now provide values for the two input variables that are used in the test.

* + Close the Test pane.

* + Click Publish to publish the new version of the runbook.

* + Stop the virtual machine that you started in the previous step.

* + Click OK to start the runbook. Type in the VMName and ResourceGroupName for the virtual machine that you're going to start.
  + 
  + When the runbook completes, check that the virtual machine was started.

Lab 2: Part 2: Create Graphical Runbook

* + **Create Graphical Runbook**, In the Azure portal, open your Automation account.

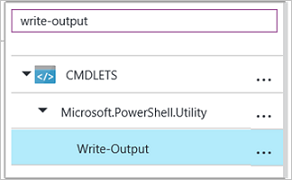
* + The Automation account page gives you a quick view of the resources in this account. You should already have some Assets. Most of those assets are the modules that are automatically included in a new Automation account. You should also have the Credential asset that's mentioned in the prerequisites.

* + Select Runbooks under PROCESS MANAGEMENT to open the list of runbooks.

* + Create a new runbook by selecting + Add a runbook, then click Create a new runbook.

* + Give the runbook the name az{youralias}MyFirstRunbook-Graphical.

* + In this case, you're going to create a graphical runbook so select Graphical for Runbook type.
  + New runbook

* + Click Create to create the runbook and open the graphical editor.
  + **Add activities**, in the Library control, click in the search textbox and type Write-Output. The search results are shown in the following image:
  + 
  + Scroll down to the bottom of the list. You can either right-click Write-Output and select Add to canvas, or click the ellipses next to the cmdlet and then select Add to canvas.

* + Click the Write-Output activity on the canvas. This action opens the Configuration control page, which allows you to configure the activity.

* + The Label defaults to the name of the cmdlet, but you can change it to something more friendly. Change it to Write Hello World to output.

* + Click Parameters to provide values for the cmdlet's parameters.

* + Some cmdlets have multiple parameter sets, and you need to select which you one to use. In this case, Write-Output has only one parameter set, so you don't need to select one.

* + Select the InputObject parameter. This is the parameter where you specify the text to send to the output stream.

* + In the Data source dropdown, select PowerShell expression. The Data source dropdown provides different sources that you use to populate a parameter value.

* + You can use output from such sources such as another activity, an Automation asset, or a PowerShell expression. In this case, the output is just Hello World. You can use a PowerShell expression and specify a string.

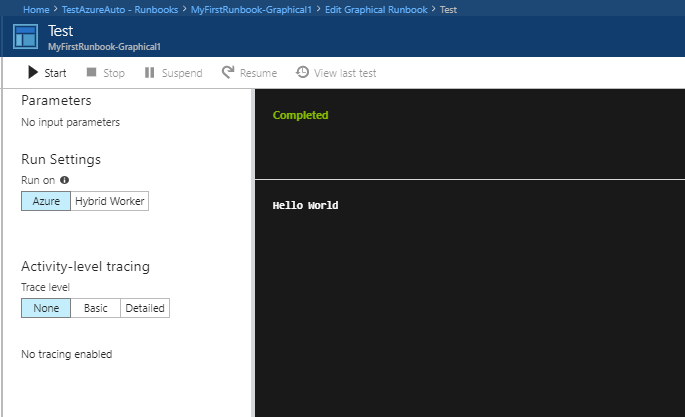
* + In the Expression box, type "Hello World" and then click OK twice to return to the canvas.

* + Save the runbook by clicking Save.
  + **Test runbook,**select Test pane to open the Test page.

* + Click Start to start the test. This should be the only enabled option.

* + A runbook job is created and its status displayed in the pane.

* + The job status starts as Queued indicating that it is waiting for a runbook worker in the cloud to become available. It then moves to Starting when a worker claims the job, and then Running when the runbook actually starts running.

* + When the runbook job completes, its output is displayed. In this case, you see Hello World.
  + 
  + Close the Test page to return to the canvas.
  + **Publish and start the run book,** Select Publish to publish the runbook and then Yes when prompted.

* + If you scroll left to view the runbook in the Runbooks page, it shows an Authoring Status of Published.

* + Scroll back to the right to view the page for az{youralias}MyFirstRunbook-Graphical.

* + The options across the top allow us to start the runbook, schedule it to start at some time in the future, or create a webhook so it can be started through an HTTP call.

* + Select Start and then Yes when prompted to start the runbook.

* + A job page is opened for the runbook job that was created. Verify that the Job status shows Completed.

* + Once the runbook status shows Completed, click Output. The Output page is opened, and you can see the Hello World in the pane.

* + Close the Output page.

* + Click All Logs to open the Streams page for the runbook job. You should only see Hello World in the output stream, but this can show other streams for a runbook job such as Verbose and Error if the runbook writes to them.

* + Close the All Logs page and the Job page to return to the az{youralias}MyFirstRunbook-Graphical page.

* + To view all the jobs for the runbook close the Job page and select Jobs under RESOURCES. This lists all the jobs created by this runbook. You should only see one job listed since you only ran the job once.

* + You can click this job to open the same Job pane that you viewed when you started the runbook. This allows you to go back in time and view the details of any job that was created for a particular runbook.
  + **Create variable assets,** You've tested and published your runbook, but so far it doesn't do anything useful. You want to have it manage Azure resources. Before you configure the runbook to authenticate, you create a variable to hold the subscription ID and reference it after you set up the activity to authenticate in step 6 below. Including a reference to the subscription context allows you to easily work between multiple subscriptions. Before proceeding, copy your subscription ID from the Subscriptions option off the Navigation pane.

* + In the Automation Accounts page, select Variables under SHARED RESOURCES.
  + Select Add a variable.
  + In the New variable page, in the Name box, enter AzureSubscriptionId and in the Value box enter your Subscription ID. Keep string for the Type and the default value for Encryption.
  + Click Create to create the variable.
  + **Add authentication,** Now that you have a variable to hold the subscription ID, you can configure the runbook to authenticate with the Run As credentials You do that by adding the Azure Run As connection Asset and Connect-AzureRmAccount cmdlet to the canvas.
    - Navigate back to your runbook and select Edit on the az{youralias}MyFirstRunbook-Graphical page.

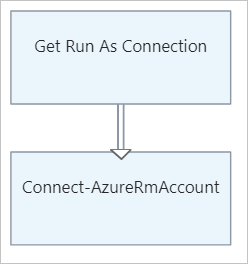
* + You don't need the Write Hello World to output anymore, so click the ellipses (...) and select Delete.

* + In the Library control, expand ASSETS, Connections, and add AzureRunAsConnection to the canvas by selecting Add to canvas.

* + In the Library control, type Connect-AzureRmAccount in the search textbox.

Add Connect-AzureRmAccount to the canvas.

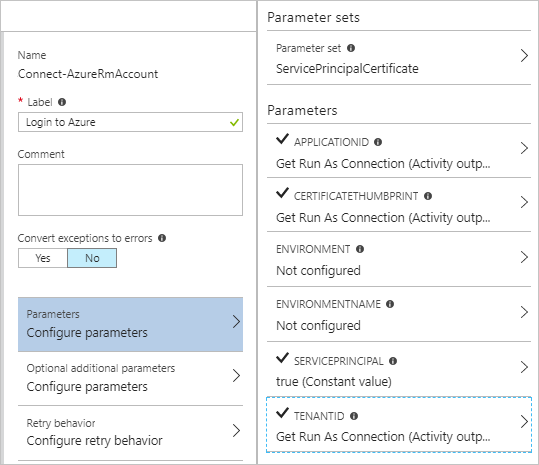
* + Hover over Get Run As Connection until a circle appears on the bottom of the shape. Click the circle and drag the arrow to Connect-AzureRmAccount. The arrow that you created is a link. The runbook starts with Get Run As Connection and then run Connect-AzureRmAccount.



* + On the canvas, select Connect-AzureRmAccount and in the Configuration control pane type Login to Azure in the Label textbox.

* + Click Parameters and the Activity Parameter Configuration page appears.

* + Connect-AzureRmAccount has multiple parameter sets, so you need to select one before you can provide parameter values. Click Parameter Set and then select the ServicePrincipalCertificate parameter set.

* + Once you select the parameter set, the parameters are displayed in the Activity Parameter Configuration page. Click APPLICATIONID.
  + 
  + In the Parameter Value page, select Activity output for the Data source and select Get Run As Connection from the list, in the Field path textbox type ApplicationId, and then click OK. You are specifying the name of the property for the Field path because the activity outputs an object with multiple properties.

* + Click CERTIFICATETHUMBPRINT, and in the Parameter Value page, select Activity output for the Data source. Select Get Run As Connection from the list, in the Field path textbox type CertificateThumbprint, and then click OK.

* + Click SERVICEPRINCIPAL, and in the Parameter Value page, select ConstantValue for the Data source, click the option True, and then click OK.

* + Click TENANTID, and in the Parameter Value page, select Activity output for the Data source. Select Get Run As Connection from the list, in the Field path textbox type TenantId, and then click OK twice.

* + In the Library control, type Set-AzureRmContext in the search textbox.

* + Add Set-AzureRmContext to the canvas.

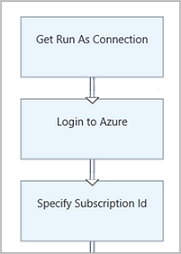
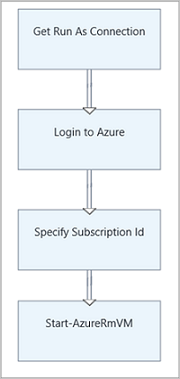
* + On the canvas, select Set-AzureRmContext and in the Configuration control pane type Specify Subscription Id in the Label textbox.

* + Click Parameters and the Activity Parameter Configuration page appears.

* + Set-AzureRmContext has multiple parameter sets, so you need to select one before you can provide parameter values. Click Parameter Set and then select the SubscriptionId parameter set.

* + Once you select the parameter set, the parameters are displayed in the Activity Parameter Configuration page. Click SubscriptionID

* + In the Parameter Value page, select Variable Asset for the Data source and select AzureSubscriptionId from the list and then click OK twice.

* + Hover over Login to Azure until a circle appears on the bottom of the shape. Click the circle and drag the arrow to Specify Subscription Id. Your runbook should look like the following at this point:
  + 
  + **Add activity to start a VM**, In the Library control, type Start-AzureRm in the search textbox.
  + Add Start-AzureRmVM to the canvas and then click and drag it underneath Specify Subscription Id.
  + Hover over Specify Subscription Id until a circle appears on the bottom of the shape. Click the circle and drag the arrow to Start-AzureRmVM.
  + Select Start-AzureRmVM. Click Parameters and then Parameter Set to view the sets for Start-AzureRmVM. Select the ResourceGroupNameParameterSetName parameter set. ResourceGroupName and Name have exclamation points next them. This indicates that they are required parameters. Also note both expect string values.
  + Select Name. Select PowerShell expression for the Data source and type in the name of the virtual machine surrounded with double quotes that you start with this runbook. Click OK.
  + Select ResourceGroupName. Use PowerShell expression for the Data source and type in the name of the resource group surrounded with double quotes. Click OK.
  + Click Test pane so that you can test the runbook.
  + Click Start to start the test. Once it completes, check that the virtual machine was started.
  + 
  + **Add additional input parameters**, Open the graphical editor by clicking Edit on the MyFirstRunbook-Graphical pane.
  + Select Input and output and then Add input to open the Runbook Input Parameter pane.
  + Specify VMName for the Name. Keep string for the Type, but change Mandatory to Yes. Click OK.
  + Create a second mandatory input parameter called ResourceGroupName and then click OK to close the Input and Output pane.
  + Runbook Input Parameters
  + Select the Start-AzureRmVM activity and then click Parameters.
  + Change the Data source for Name to Runbook input and then select VMName.
  + Change the Data source for ResourceGroupName to Runbook input and then select ResourceGroupName.
  + Start-AzureVM Parameters
  + Save the runbook and open the Test pane. You can now provide values for the two input variables that you use in the test.
  + Close the Test pane.
  + Click Publish to publish the new version of the runbook.
  + Stop the virtual machine that you started in the previous step.
  + Click Start to start the runbook. Type in the VMName and ResourceGroupName for the virtual machine that you're going to start.
  + When the runbook completes, check that the virtual machine was started.
  + **Create a conditional link,** You now modify the runbook so that it only attempts to start the virtual machine if it is not already started. You do this by adding a Get-AzureRmVM cmdlet to the runbook that gets the instance level status of the virtual machine. Then you add a PowerShell Workflow code module called Get Status with a snippet of PowerShell code to determine if the virtual machine state is running or stopped. A conditional link from the Get Status module only runs Start-AzureRmVM if the current running state is stopped. Finally, You output a message to inform you if the VM was successfully started or not using the PowerShell Write-Output cmdlet.

* + Open MyFirstRunbook-Graphical in the graphical editor.

* + Remove the link between Specify Subscription Id and Start-AzureRmVM by clicking on it and then pressing the Delete key.

* + In the Library control, type Get-AzureRm in the search textbox.

* + Add Get-AzureRmVM to the canvas.

* + Select Get-AzureRmVM and then Parameter Set to view the sets for Get-AzureRmVM. Select the GetVirtualMachineInResourceGroupNameParamSet parameter set. ResourceGroupName and Name have exclamation points next them. This indicates that they are required parameters. Also note both expect string values.

* + Under Data source for Name, select Runbook input and then select VMName. Click OK.

* + Under Data source for ResourceGroupName, select Runbook input and then select ResourceGroupName. Click OK.

* + Under Data source for Status, select Constant value and then click True. Click OK.

* + Create a link from Specify Subscription Id to Get-AzureRmVM.

* + In the library control, expand Runbook Control and add Code to the canvas.

* + Create a link from Get-AzureRmVM to Code.

* + Click Code and in the Configuration pane, change label to Get Status.

* + Select Code parameter, and the Code Editor page appears.

* + In the code editor, paste the following snippet of code:

$StatusesJson = $ActivityOutput['Get-AzureRmVM'].StatusesText

$Statuses = ConvertFrom-Json $StatusesJson

$StatusOut =""

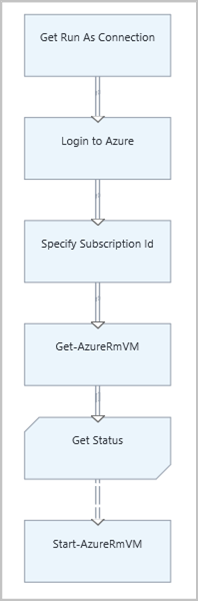
foreach ($Status in $Statuses){

if($Status.Code -eq "Powerstate/running"){$StatusOut = "running"}

elseif ($Status.Code -eq "Powerstate/deallocated") {$StatusOut = "stopped"}

}

$StatusOut

* + Create a link from Get Status to Start-AzureRmVM.
  + 
  + Runbook with Code Module

* + Select the link and in the Configuration pane, change Apply condition to Yes. Note the link turns to a dashed line indicating that the target activity only runs if the condition resolves to true.

* + For the Condition expression, type $ActivityOutput['Get Status'] -eq "Stopped". Start-AzureRmVM now only runs if the virtual machine is stopped.

* + In the Library control, expand Cmdlets and then Microsoft.PowerShell.Utility.

* + Add Write-Output to the canvas twice.

* + On the first Write-Output control, click Parameters and change the Label value to Notify VM Started.

* + For InputObject, change Data source to PowerShell expression and type in the expression "$VMName successfully started.".

* + On the second Write-Output control, click Parameters and change the Label value to Notify VM Start Failed

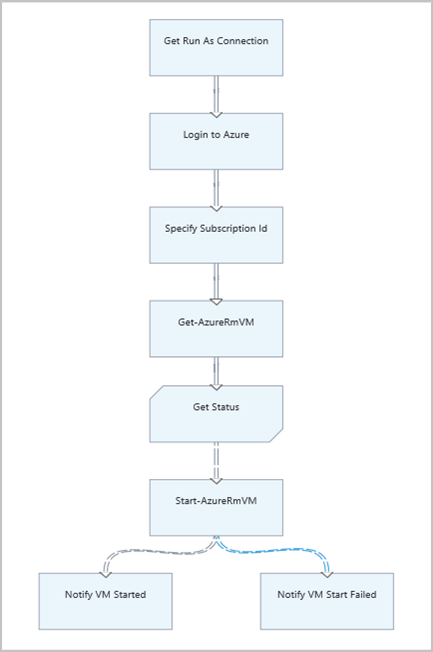
* + For InputObject, change Data source to PowerShell expression and type in the expression "$VMName could not start.".

* + Create a link from Start-AzureRmVM to Notify VM Started and Notify VM Start Failed.

* + Select the link to Notify VM Started and change Apply condition to True.

* + For the Condition expression, type $ActivityOutput['Start-AzureRmVM'].IsSuccessStatusCode -eq $true. This Write-Output control now only runs if the virtual machine is successfully started.

* + Select the link to Notify VM Start Failed and change Apply condition to True.

* + For the Condition expression, type $ActivityOutput['Start-AzureRmVM'].IsSuccessStatusCode -ne $true. This Write-Output control now only runs if the virtual machine is not successfully started. Your runbook should look like the following image:
  + 
  + Runbook with Write-Output

* + Save the runbook and open the Test pane.

* + Start the runbook with the virtual machine stopped, and it should start.

Lab 3: Azure CLI

Lab 3: Part 1: Installing Azure CLI and login to Azure

* + Download using this link
  + Open CLI window
  + Type az Login
  + From a browser window that is already logged in to Azure, Navigate to <https://aka.ms/devicelogin> and input the displayed code
  + Another way of doing this you can use the command az login -u username -p password
  + A third option is login using a service principle using a certificate
  + In CLI window type az ad sp create-for-rbac --name az{yourid}app --create-cert
  + Capture the AppID, .pem file path, and tenant ID
  + Navigate to Azure AD
  + Navigate to App Registrations
  + Find your app az{yourid}app and capture App ID
  + Now to login using certificate
    - Az login --service-principal -u {AppID} -p {PemFilePath} -tenant {tenantid}

Lab 3: Part 2: Create VM and Stop Running VM's

* + To list your subscriptions in table format type az account list --output table
  + To select a subscription type az account set --subscription "{subscriptionname}"
  + To find out what commands you can do with a VM, type az vm
  + To list your VM's type az vm list --output table
  + To create another VM type, az vm create --resource-group {resourcegroupname} --name az{youralias}vm2 --image win2016datacenter
  + To query for specific VM's for example get a list of running VM's, type az vm list -d --query "[powerState=='VM Running']" --output table
  + To turn off VM, type az vm stop --ids $(az vm list -d --query "[powerState=='VM Running'].id" --output tsv)