Azure Dev & Test



Vision Scope

Prepared for

Microsoft

20-Nov-15

Version 1.0 Draft

Prepared by

**Stanislav Zhelyazkov**

Principal Consultant

stanislav.zhelyazkov@lumagate.com

Contributors

**Add Contributors to Doc Properties**

Table of Contents

[1 Before you begin 3](#_Toc435542871)

[2 Lab 1: Deploy Automation account via ARM template and delegate access to Tenant 3](#_Toc435542872)

[3 Lab 2: Access Automation account as Tenant 7](#_Toc435542873)

[4 Lab 3: Setup Source Control Integration 8](#_Toc435542874)

[5 Lab 4: Manage Azure Backup with Process Automation 16](#_Toc435542875)

[6 Lab 5: Create Webhook 21](#_Toc435542876)

[7 Lab 6: Execute Runbook via Webhook 22](#_Toc435542877)

[8 Lab 7: Register Azure VM Autoamtion DSC 24](#_Toc435542878)

[9 Lab 8: Apply DSC Configuration 25](#_Toc435542879)

1. Before you begin

The objective of this Lab is to learn how to use Azure Automation trough examples.

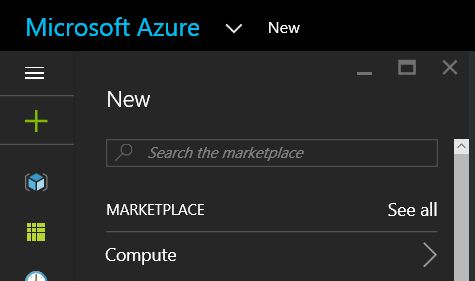
In order to complete these labs, you need to ensure you have the following access and tools properly configured:

* Admin access to an Azure Subscription (minimum Trial Subscription)
* Microsoft Azure PowerShell (1.0.2 or higher)
* PowerShell ISE or Console

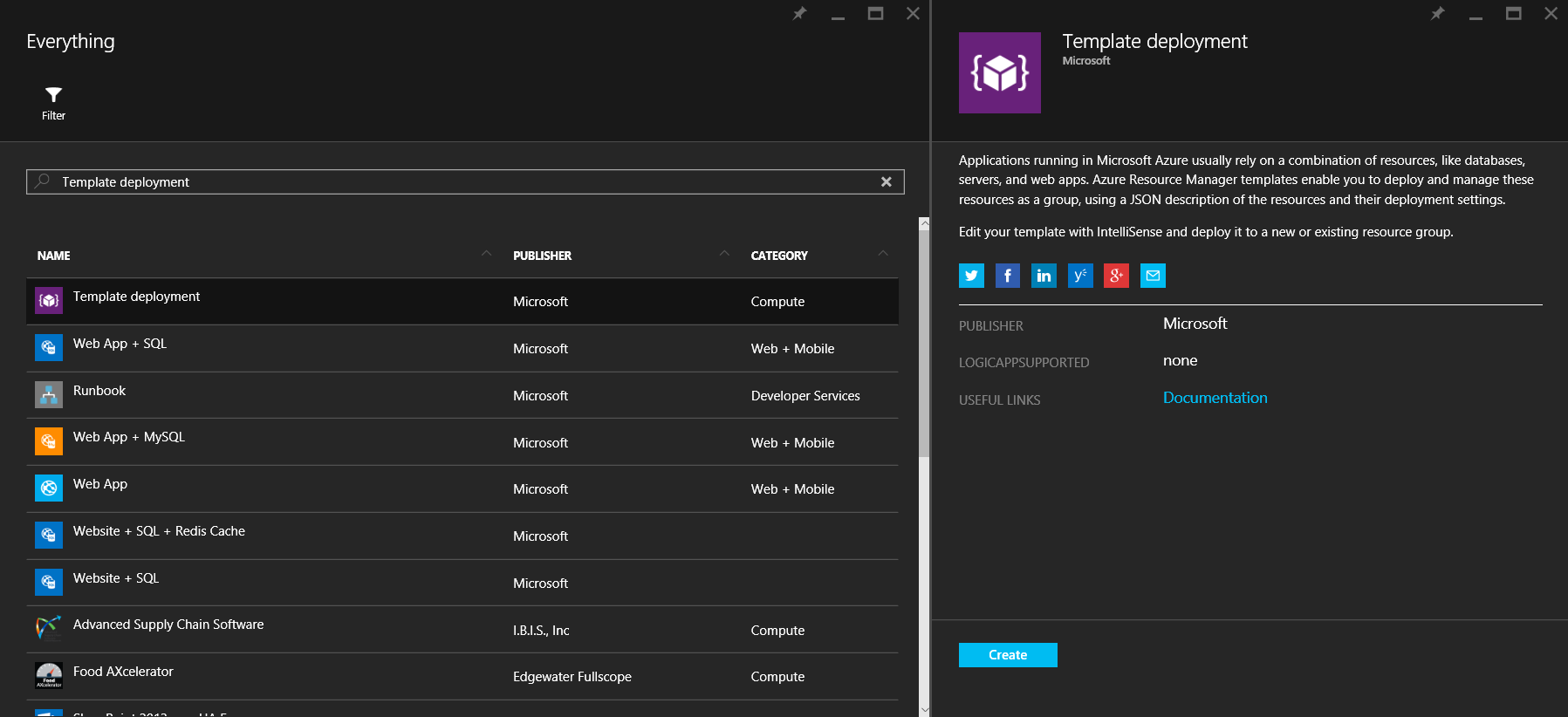
1. Lab 1: Deploy Automation account via ARM template and delegate access to Tenant

In this lab, you will become familiar with the deployment of Azure Auotmation account via Azure Resource Manager Template.

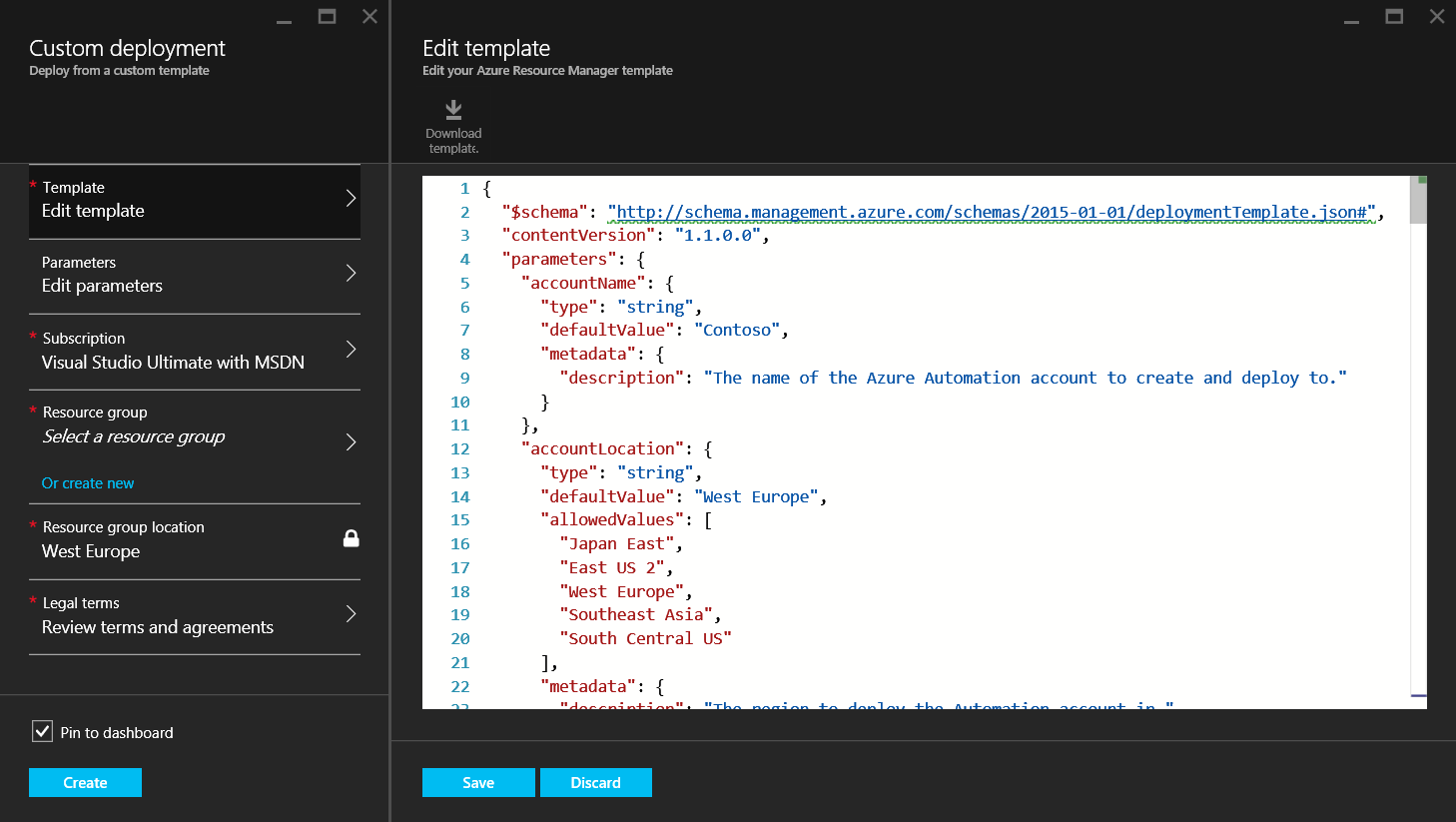
1. From your computer, open your preferred web browser and navigate to <https://portal.azure.com>
2. Sign in with your credentials that has access to an Azure subscription (you need Admin access)
3. Once you are logged in, click on the green “+” sign on the left menu. Click on the “See all” link.



1. In search bar type “template deplpoyment” (without quoates) and press enter key.
2. Select Template Deployment and click create.



1. From Custom Deployment blade click on Edit template.
2. Copy and paste the code from deploy\_AutomationAccount.json file into the Edit template blade by replacing the existing code. Click Save.



1. Click on Edit parameters and enter the information below and click OK:

AccountName: <Name of the Automation account to be created>

AccountLocation: <Choose region for the Automation account>

PricingTier: <Pricing Tier for the Automation account>

AzureSubscriptionIDValue: <The ID of your Azure Subscription. Can be found in Subscriptions blade>

BackupVautlNameValue: <The name of the Azure Backup Vault that you will create later in this lab. Leave default value>

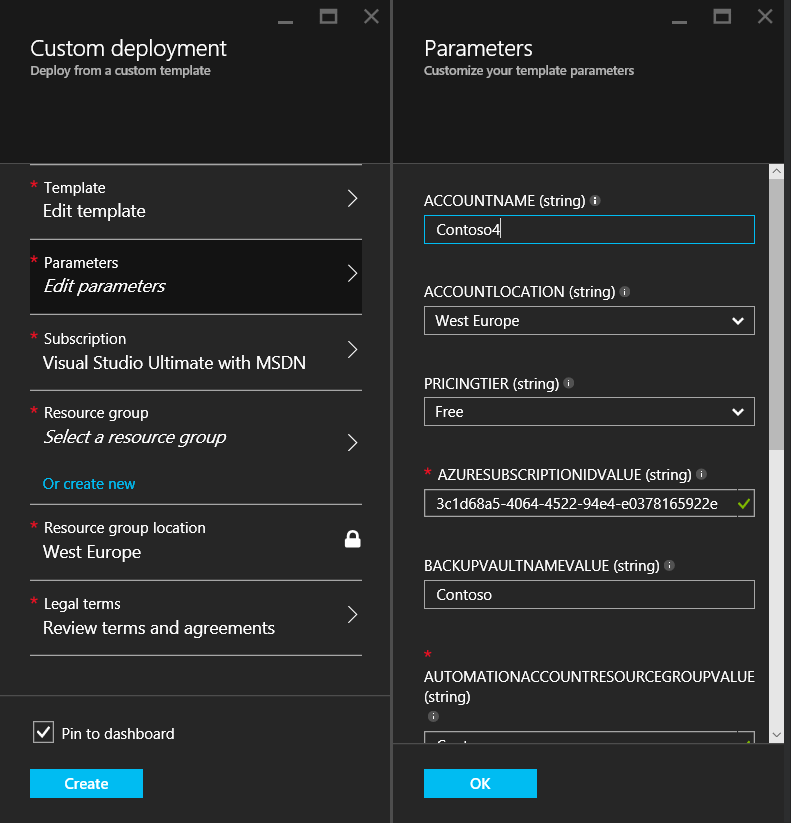
AutomationAccountResourceGroupValue: <The name of the Azure Resource Group where you will place this Automation account>

AzureServiceAccountUserName: <The username of the service account from Azure AD that you will use in Azure Automation to manage Azure resources. Value should be in the following format [username@domain.com](mailto:username@domain.com)>

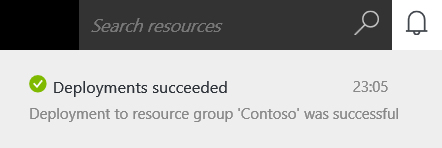
AzureServiceAccountPassword: <The password of the service account from Azure AD that you will use in Azure Automation to manage Azure resources>

DomainServiceAccountUserName: <The username of the service account from Active Directory that you will use to manage domain resources with Azure Automation. Value should be in the following format domain\username>

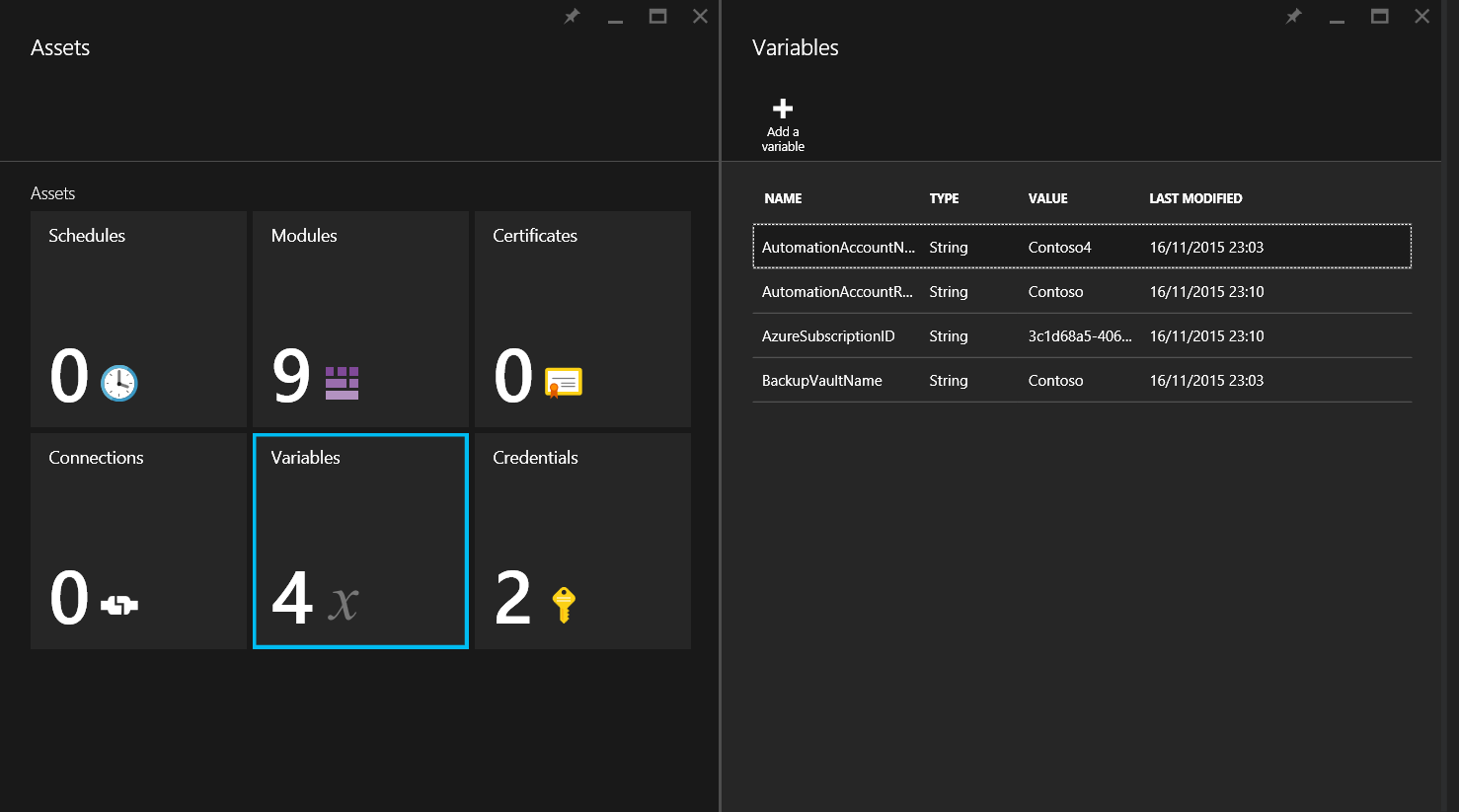
DomainServiceAccountPassword: <The password of the service account from Active Directory that you will use to manage domain resources with Azure Automation >



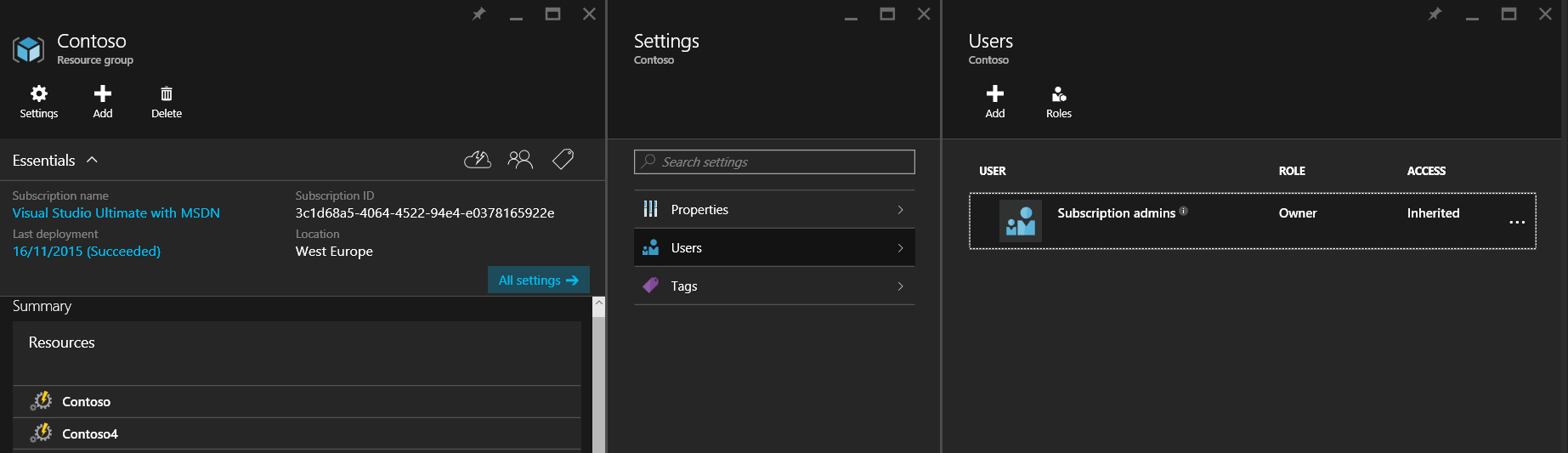
1. Click on Select a resource group and choose existing resource group to deploy the Automation account to.
2. Click on Review terms and agrrements and click Purchase.
3. In Custom deployment blade click Create.
4. A notification for the deployment will appear in the upper right corner of the Azure Portal.
5. Wait until the deployment is finished successfully.



1. Navigate to Automation accounts blade. Select the newly create automation account.
2. Select Assets tile and take a look what assets have been created during deployment in Modules, Variables and Credentials.



1. Open Resource group blade.
2. Select the resource group where Automation account was deployed. Click on All settings and then Users.



1. In Users blade click on Add.
2. In Select a role blade choose Automation Operator.
3. In Add Users blade select Tenant user.
4. In Add access blade click OK.

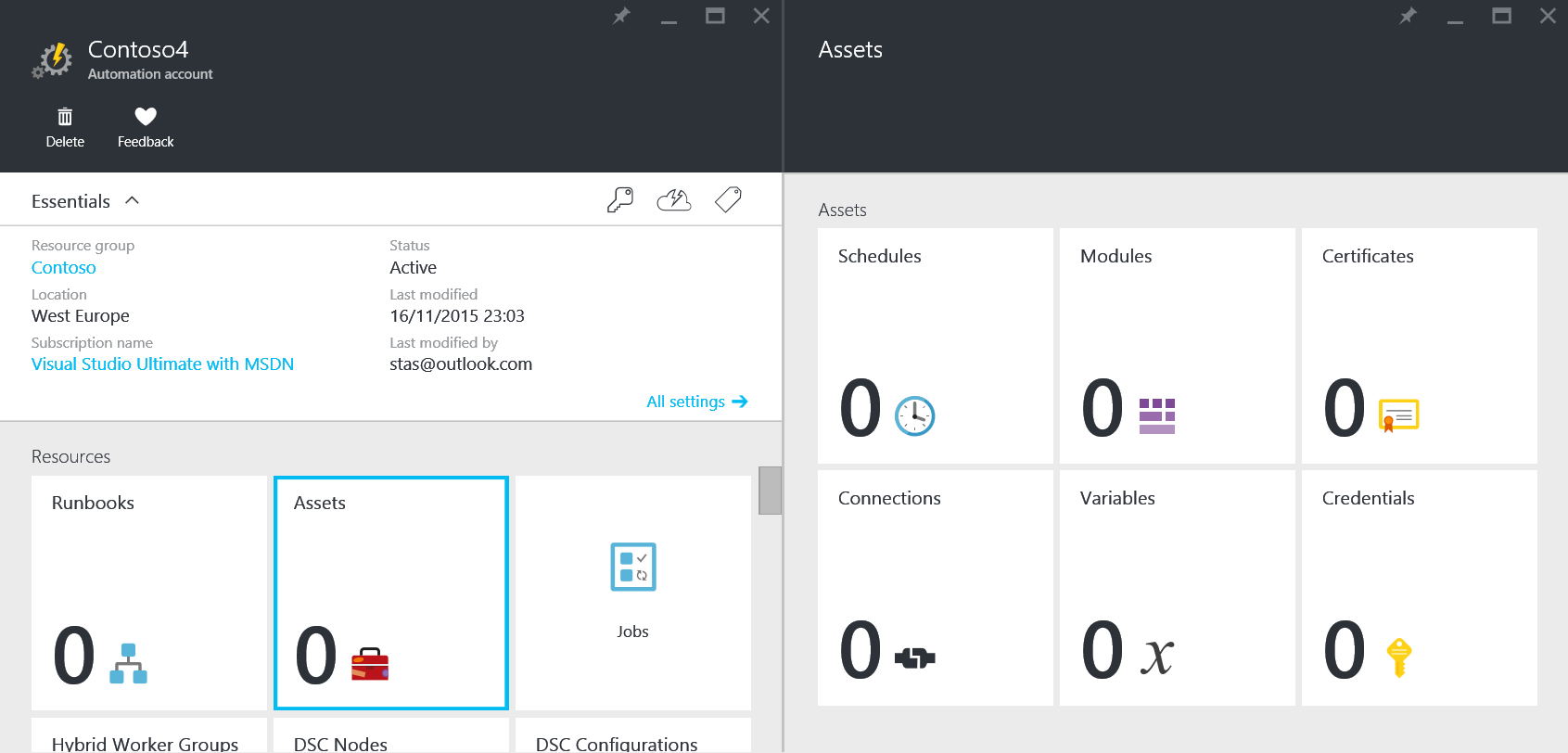


**Summary**

You have now successfully completed Lab 1, where you have done the following:

* Deployed Automation account along with variables, credentials and modules with ARM Template
* Explored Automation account assets with Administrator account
* Assigned Automation Operator role to Tenant

1. Lab 2: Access Automation account as Tenant
2. From your computer, open your preferred web browser and navigate to <https://portal.azure.com> but login with the Tenant users to which was assigned access.
3. Navigate to Automation accounts blade.
4. Click on the newly create Automation account.
5. This user account will be able to start, stop, suspend and resume jobs but will not be able to create or see assets or perform other advanced tasks.



**Summary**

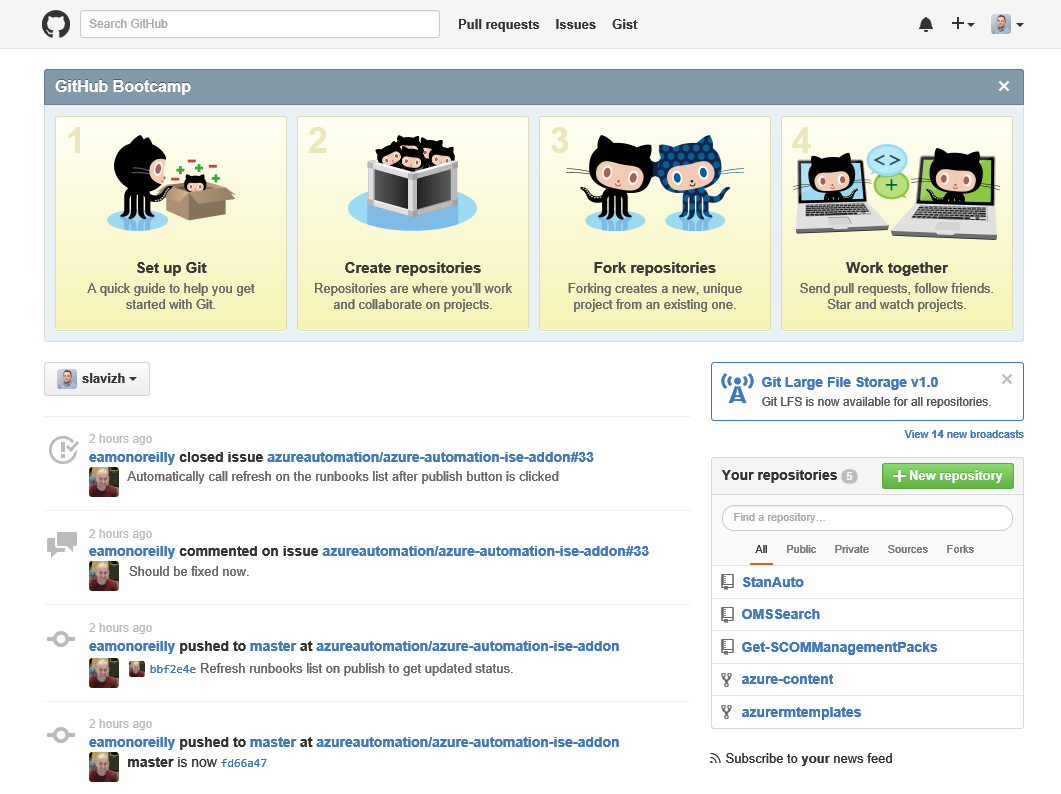
You have now successfully completed Lab 2, where you have done the following:

* Logged in as Tenant to Automation account

1. Lab 3: Setup Source Control Integration

In this lab, you will setup source control in Azure Automation to GitHub repository. In the repository you will place runbooks that will be synced to the Automation account.

1. Browse to <https://github.com>. Create account or login with existing if you have one.
2. On GitHub homepage click on New Repository.



1. In new repository page, input the following settings and click Create Repository.

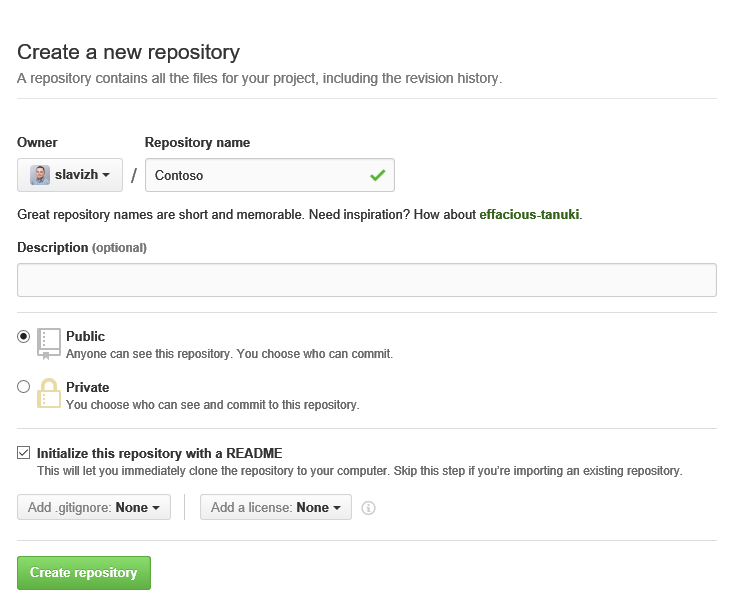
Repository Name: Contoso

Type: Public

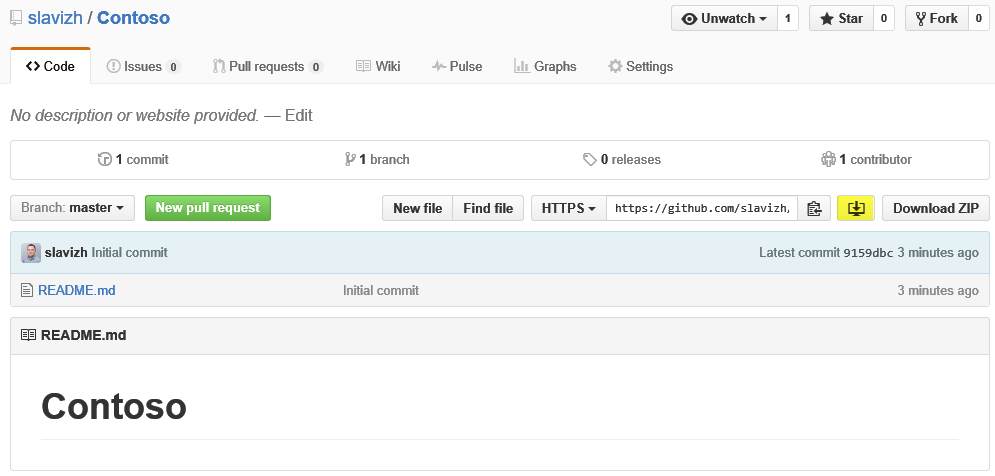
Initialize this repository with a Readme: Check

Add .gitignore: None

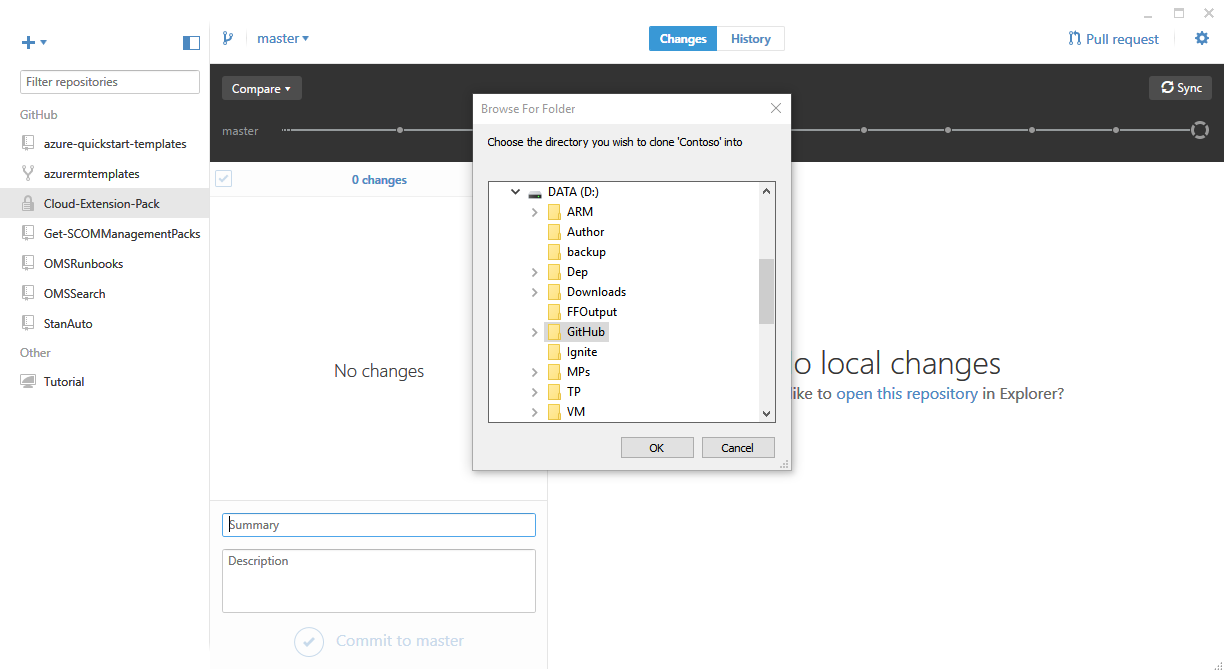
Add a license: None



1. Download GitHub Desktop from <https://desktop.github.com> and install it on your local computer. Start GitHub Desktop and logon with your GitHub account.
2. From Contoso repository page click on Clone in Desktop.



1. GitHub Desktop will be opened with window to choose location on your local computer for Contoso repository and click OK.



1. Browse the chosen location for Contoso repository on your local computer and create folder named “SyncedRunbooks” in it. Inside the folder copy and paste the following runbooks:

New-AzureBackupVault.ps1

New-AzureBackupVMProtectionPolicy.ps1

Register-AzureClassicVMToAzureBackup.ps1

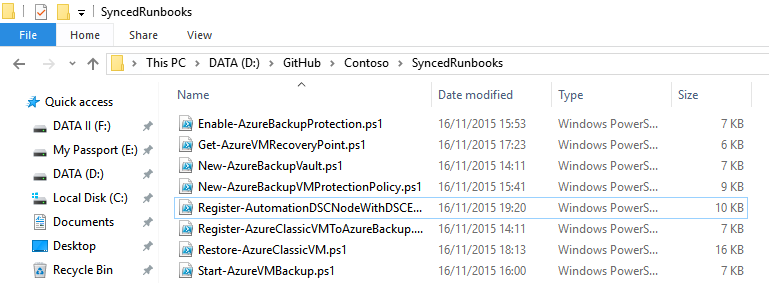
Enable-AzureBackupProtection.ps1

Start-AzureVMBackup.ps1

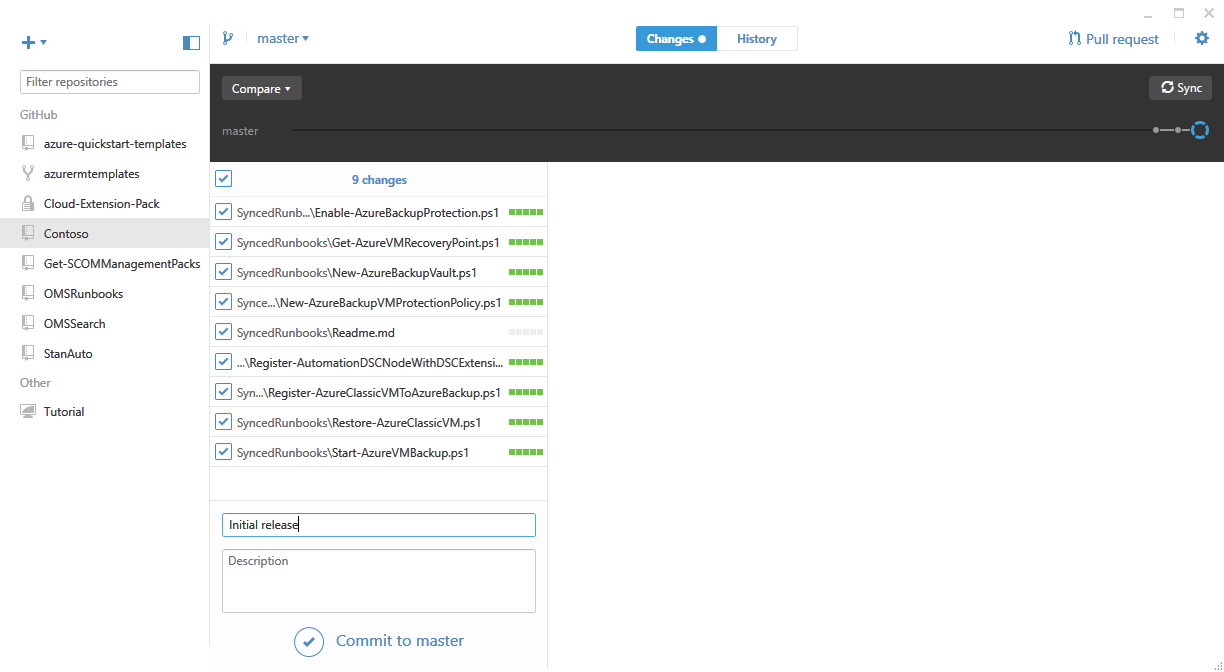
Get-AzureVMRecoveryPoint.ps1

Restore-AzureClassicVM.ps1

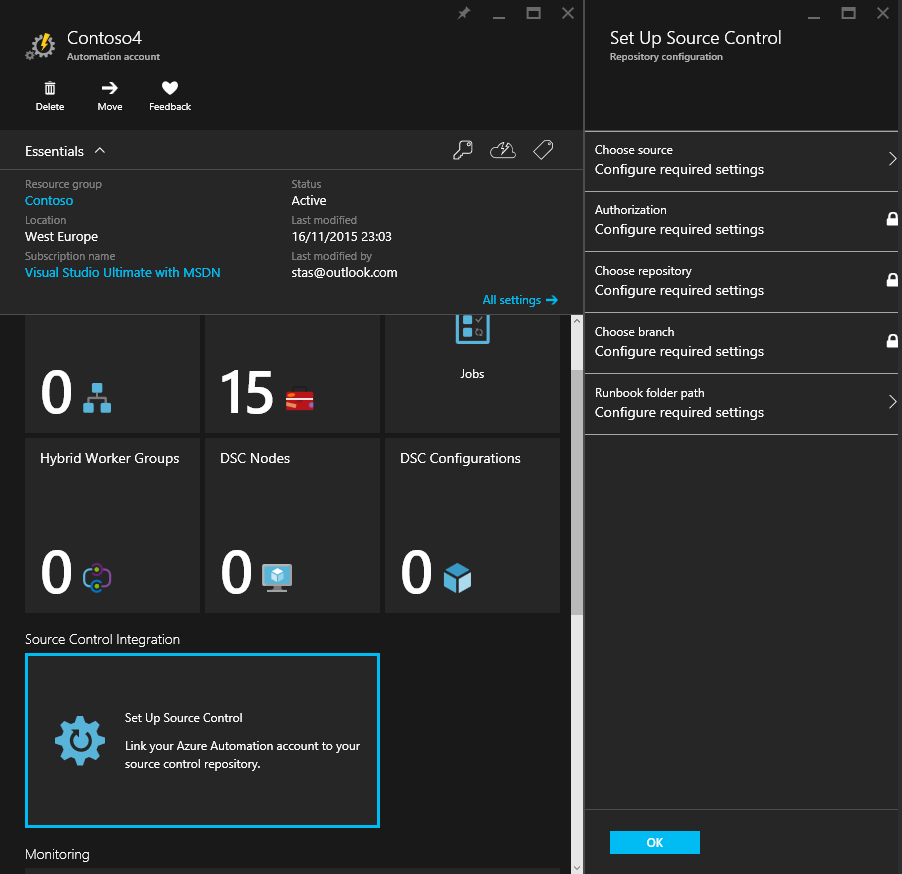
Register-AutomationDSCNodeWithDSCExtension.ps1



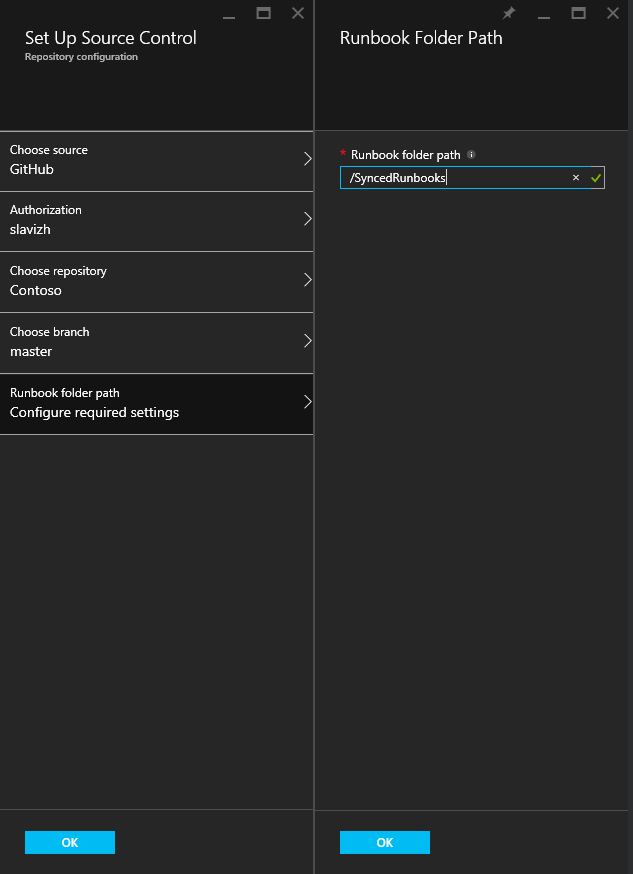
1. Switch to GitHub Desktop application. There will be one uncommitted change for Contoso repository. Enter a name for the change and click Commit to master.



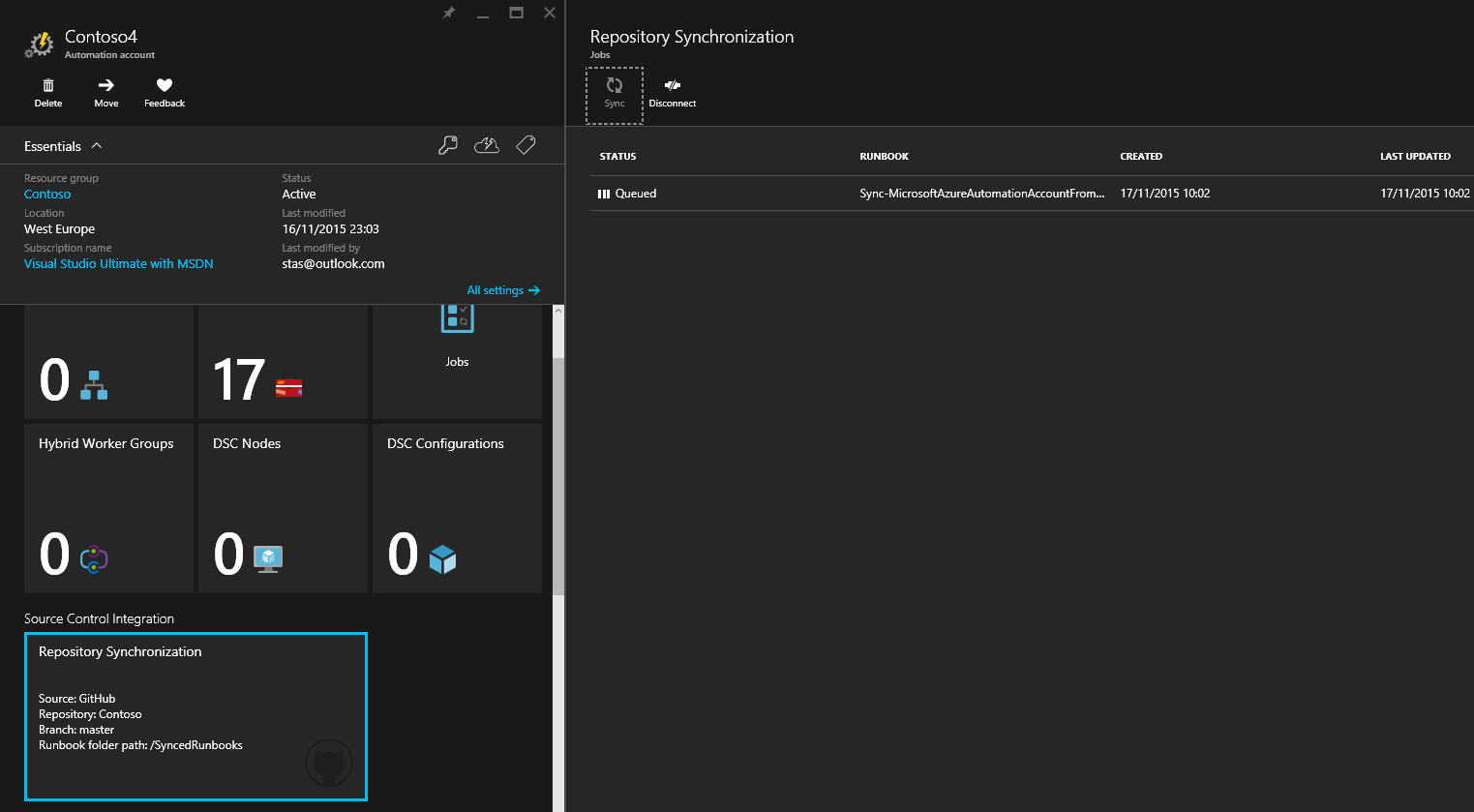
1. Next click on Sync button
2. After successful sync on repository web page you should see SyncedRunbooks folder appear along with the runbooks.
3. From your computer, open your preferred web browser and navigate to <https://portal.azure.com>
4. Sign in with your credentials that has access to an Azure subscription as Admin.
5. Open Automation accounts blade.
6. Select the automation account created in previous lab.
7. Click on Source Control tile.



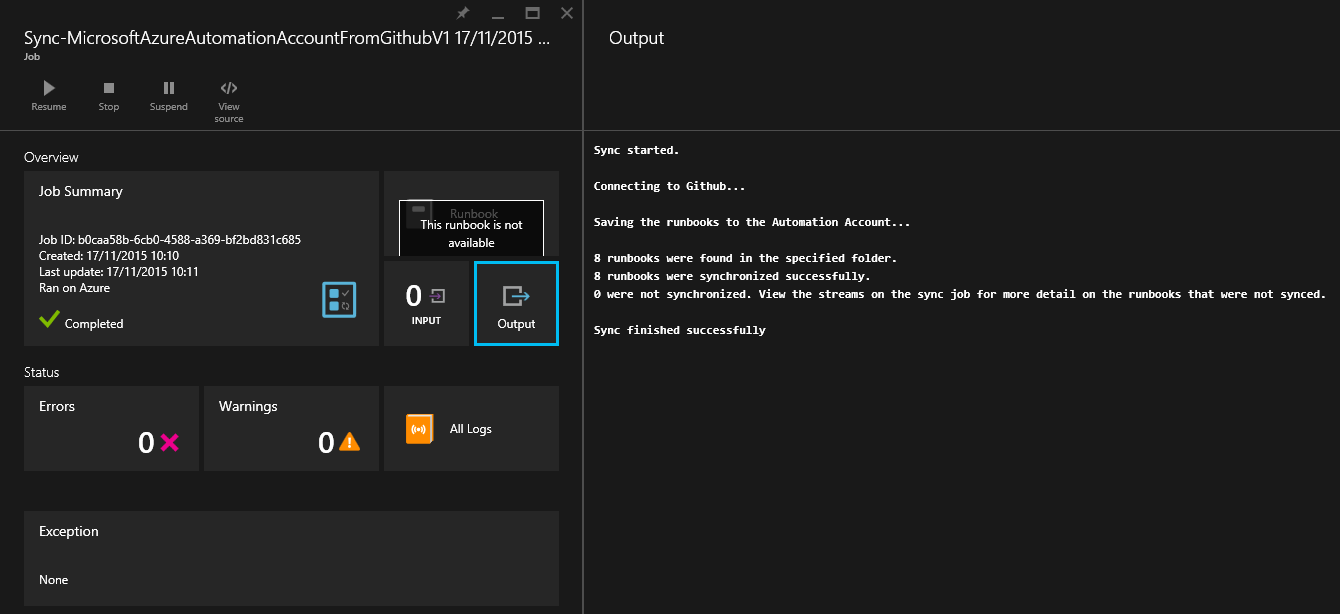
1. In Set Up Source Control blade click on Choose source and select GitHub.
2. In Set Up Source Control blade click on Authorization and then Authorize. When asked for credentials provide your GitHub account credentials. Click OK once the authorization is complete.
3. In Set Up Source Control blade click on Choose Repository. From list of repositories choose Contoso repository.
4. In Set Up Source Control blade click on Choose branch. Select master.
5. In Set Up Source Control blade click on Runbook folder path. In Runbook folder path field enter /SyncedRunbooks and click OK.



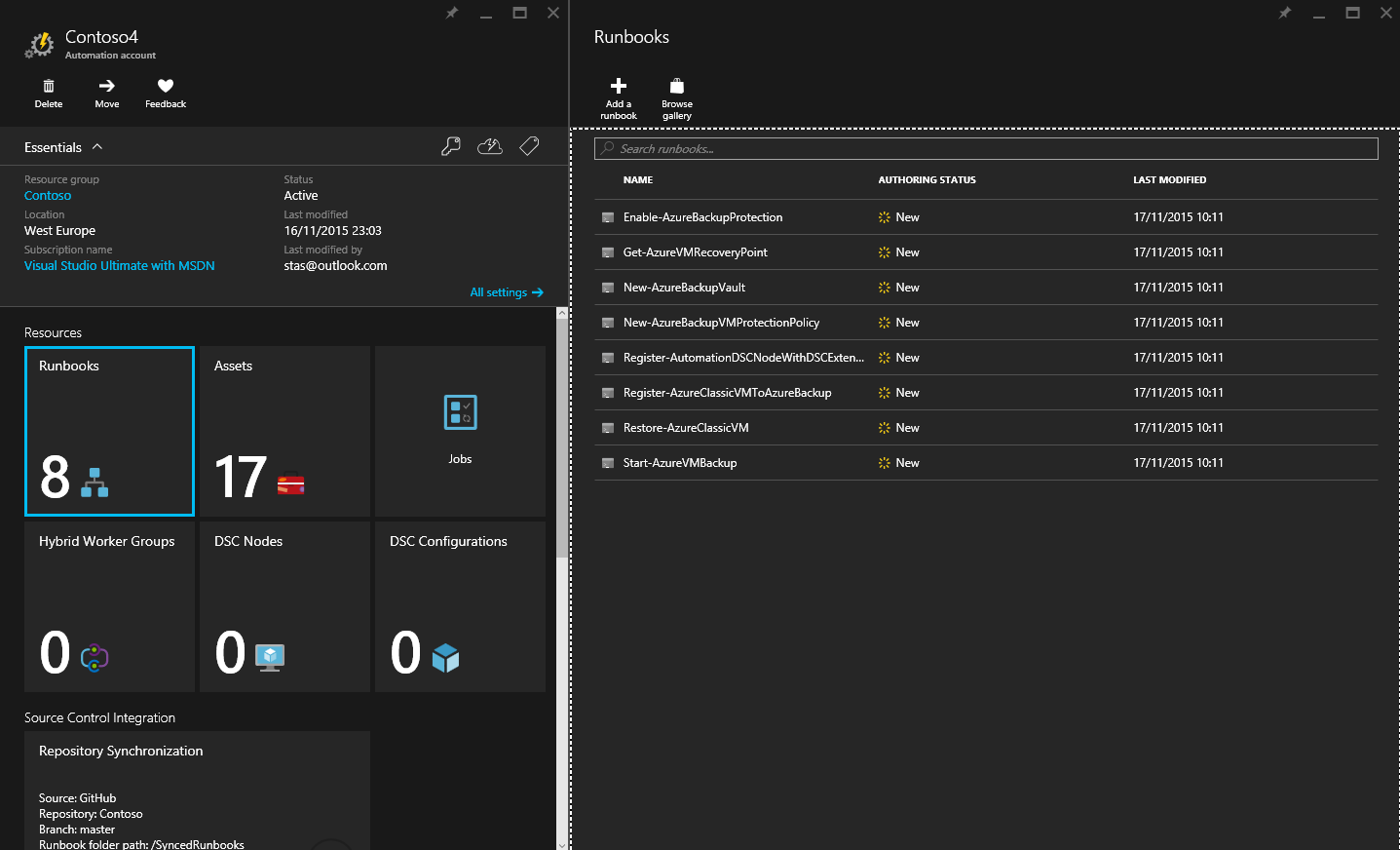
1. On Set Up Source Control blade click OK.
2. Open Repository Synchronization blade and click on Sync button. A job will appear in the list.



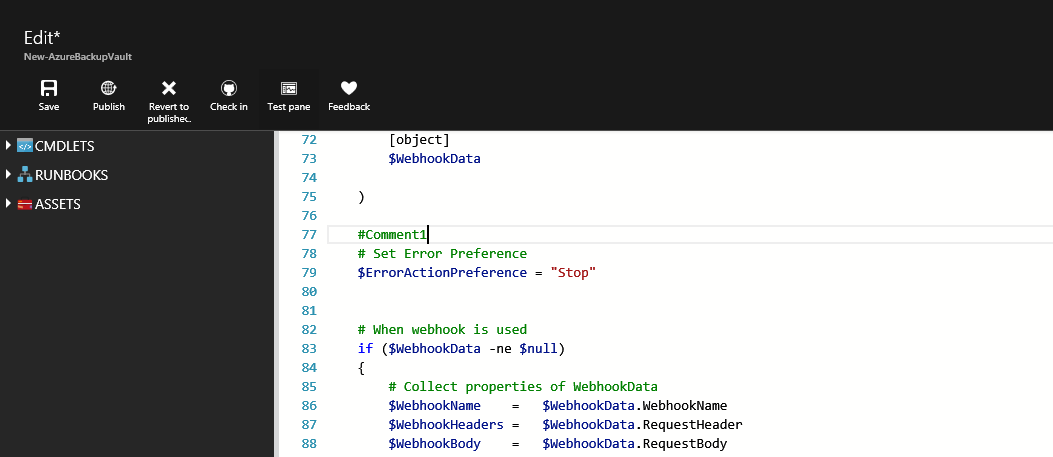
1. Click on the job and then on Output tile. When the job completes status information will appear.



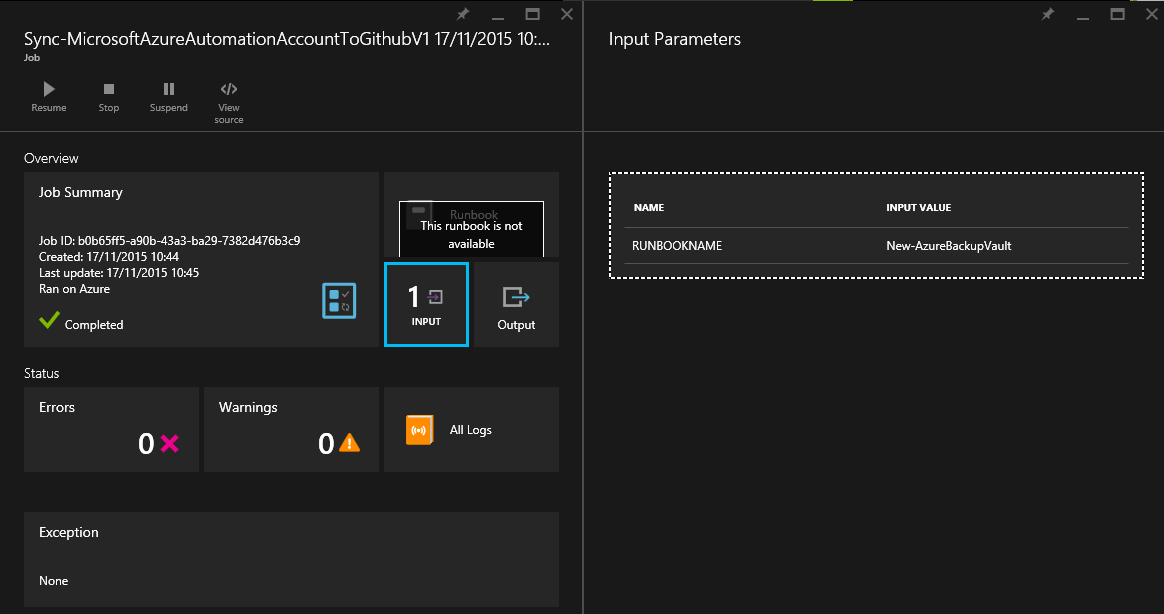
1. Click on Runbooks tile. All synced runbooks will be with status New.



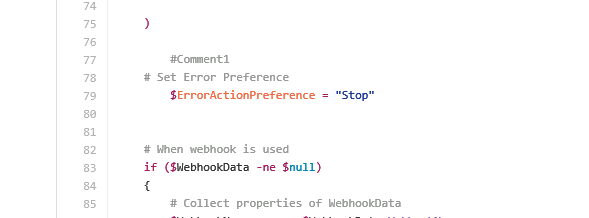
1. Click on the first runbook. Click Edit. Take a look at the runbook code. Click Publish.
2. Repeat previous step for all other runbooks. In order runbook to be started with latest version you will need to Publish it first.
3. Select on New-AzureBackupVault runbook and click Edit.
4. In Edit pane add small comment like #Comment1.



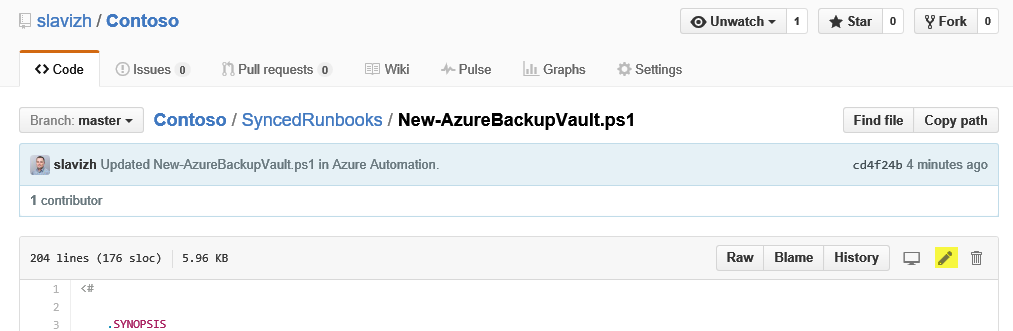
1. Click on Check in button. Confirm check in by clicking Yes. This will create a job that will push changes to GitHub repository.
2. Click Publish.
3. Click on Repository Synchronization tile.
4. Select the first job select Input.



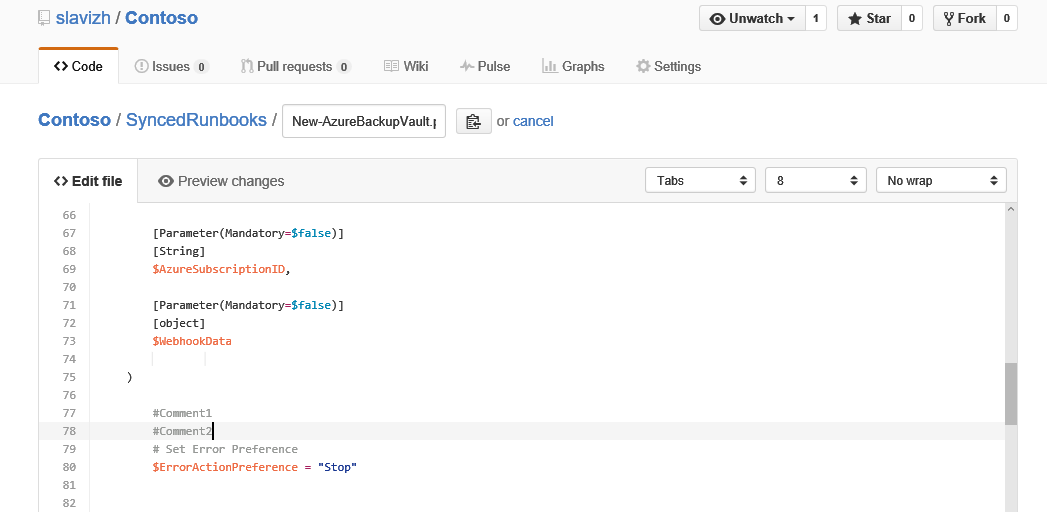
1. Browse to <https://github.com>. Open Contoso repository. Open SyncedRnbooks folder. Open New-AzureBackupVault.ps1 runbook. Comment will be synced.



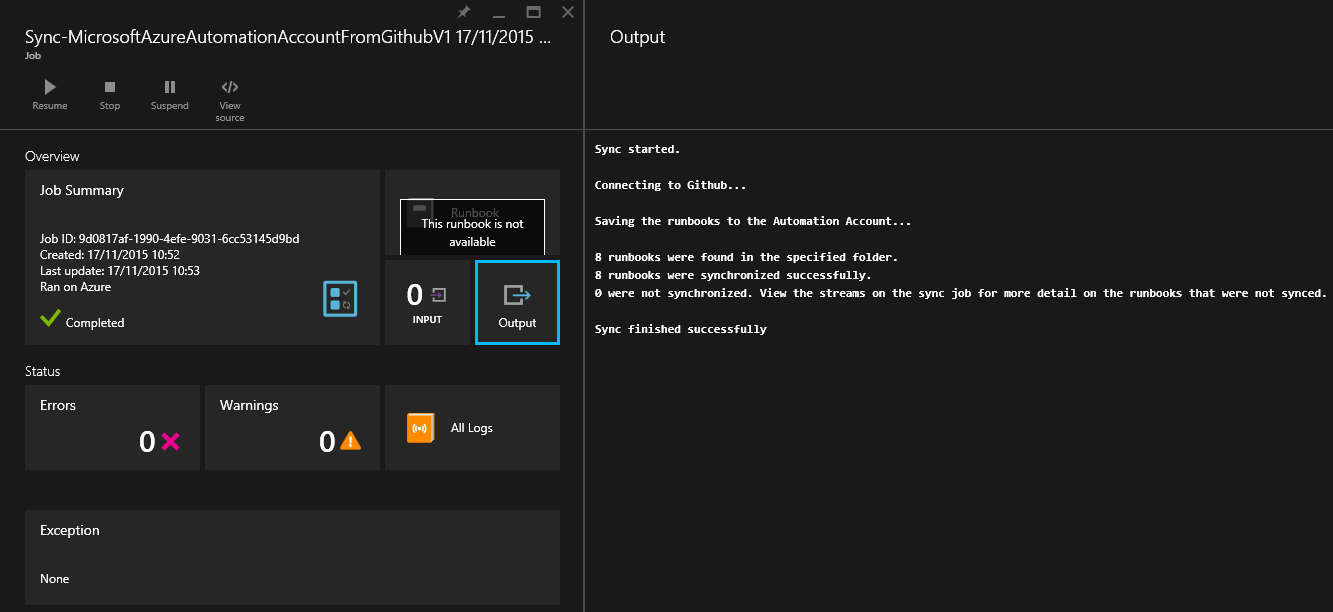
1. From the top of the page click on Edit file icon.



1. Add another comment to the file.



1. In Commit Changes at the bottom of the page enter comment and click Commit Changes.
2. From your computer, open your preferred web browser and navigate to <https://portal.azure.com>
3. Sign in with your credentials that has access to an Azure subscription as Admin.
4. Open Automation accounts blade.
5. Select the automation account created in previous lab.
6. Click on Repository Synchronization tile.
7. Click on Sync button.
8. A new job will be created to sync the latest changes. Select Output of the job.



1. Return to Runbooks tile. All runbooks will be In edit status.
2. Select New-AzureBackupVault runbook. Click Edit. #Comment 2 will be synced.
3. Publish all runbooks that are in status in Edit.

**Summary**

You have now successfully completed Lab 3, where you have done the following:

* Set up GitHub repository
* Configured Source Control with GitHub
* Synced runbooks from GitHub Repository
* Checked changes made in Azure Portal to GitHub repository

1. Lab 4: Manage Azure Backup with Process Automation

In this lab you will use Automation runbooks as Tenant to manage Azure Backup.

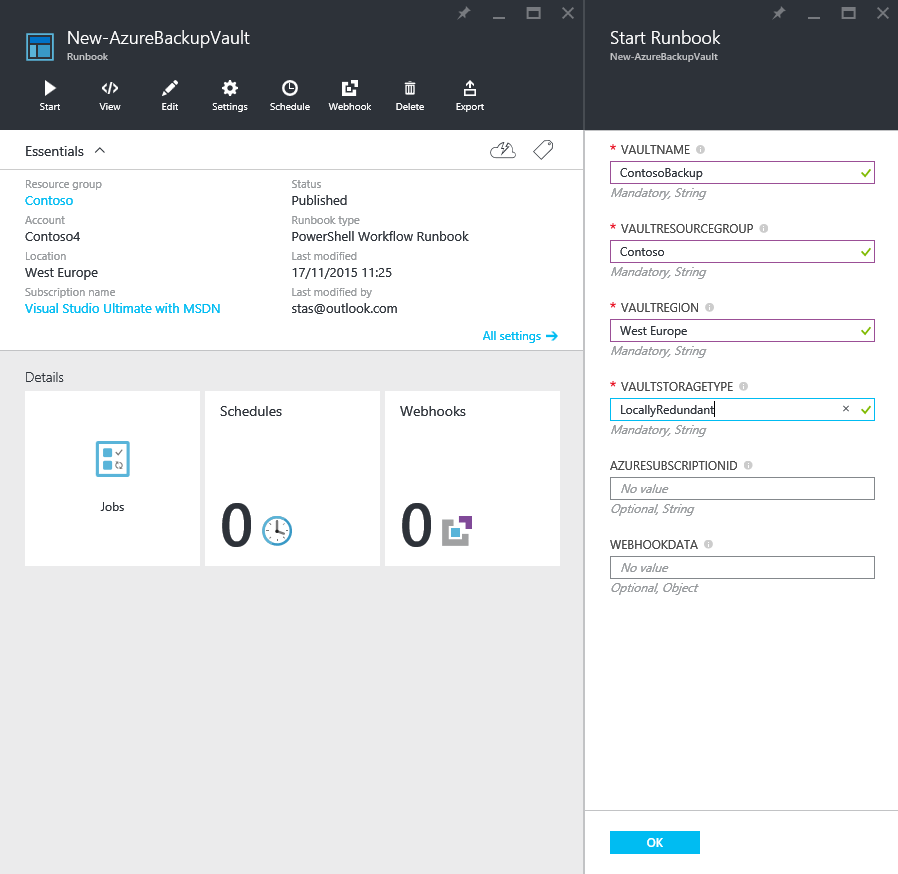
1. From your computer, open your preferred web browser and navigate to <https://portal.azure.com>
2. Sign in with your credentials that has access to an Azure subscription as Tenant
3. Open Automation accounts blade.
4. Select the automation account created in previous lab.
5. Click on Runbooks tile.
6. Click on New-AzureBackupVault runbook.
7. Click on Start.
8. Enter the following parameters and click OK.

VaultName: <Name of the Azure Backup Vault that will be created>

VaultResourceGroup: <Name of the resource group where ther Azure Backup Vault will be placed>

VaultRegion: <Azure region where the Azure Backup Vault will be created>

VaultStorageType: <Type of storage for the Azure Backup Vault. Possible values are GeoRedundant or LocallyRedundant.



1. A job will be started to create Azure Backup Vault. Once the Backup Vault is created the name of the Backup Vault will be saved as variable in Asset store.
2. Once the job is completed Azure Backup Vault is cerated.
3. Select New-AzureBackupVMProtectionPolicy runbook. Click Start.
4. Enter the following parameters and click OK.

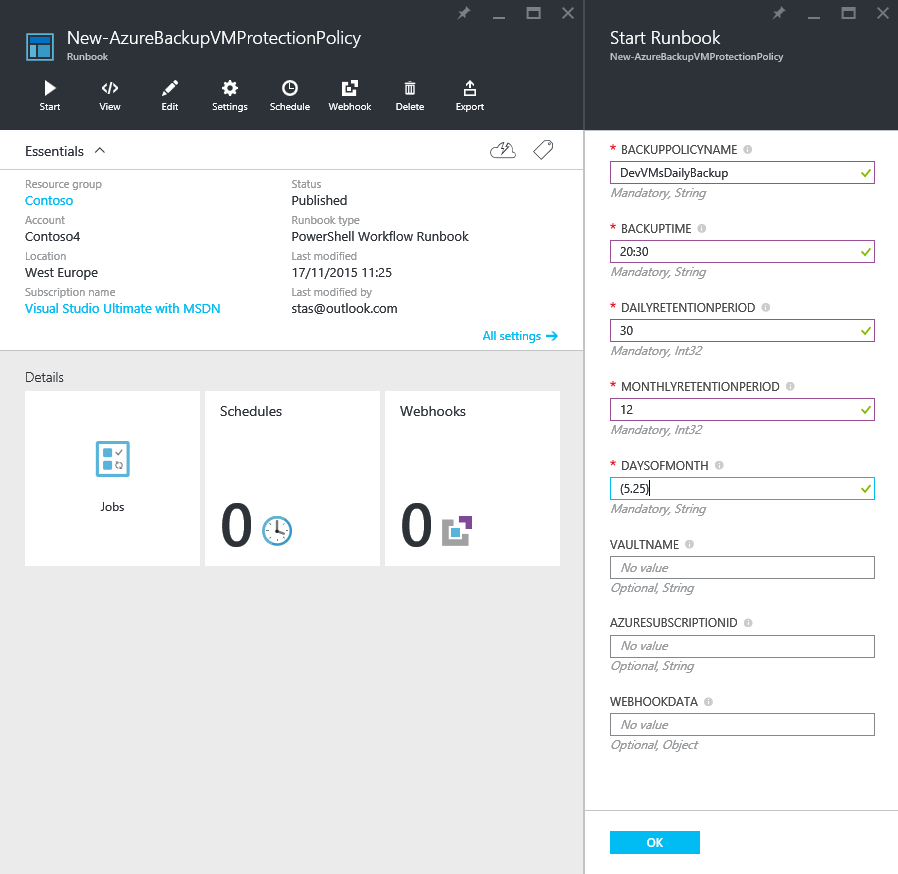
BackupPolicy: DevVMsDailyBackup

BackupTime: 20:30

DailyRetentionPeriod: 30

MonthlyRetentionPeriod: 12

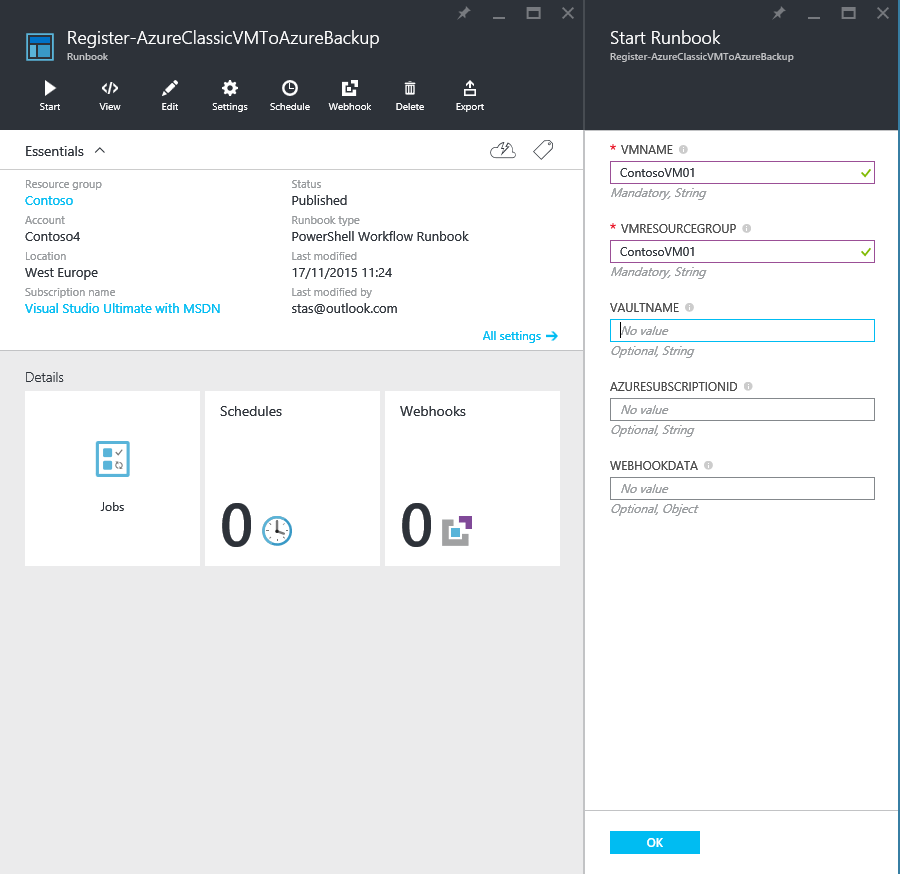
DaysOfMonth: (5,25)



1. A job will be started to create Backup Policy.
2. Once the job is completed the backup policy is created.
3. Select New-AzureClassicVMToAzureBackup runbook. This runbook will register Azure Classic VM to Backup Vault. Click Start.
4. Enter the following parameters and click OK.

VMName: <The Name of the Azure Classic VM>

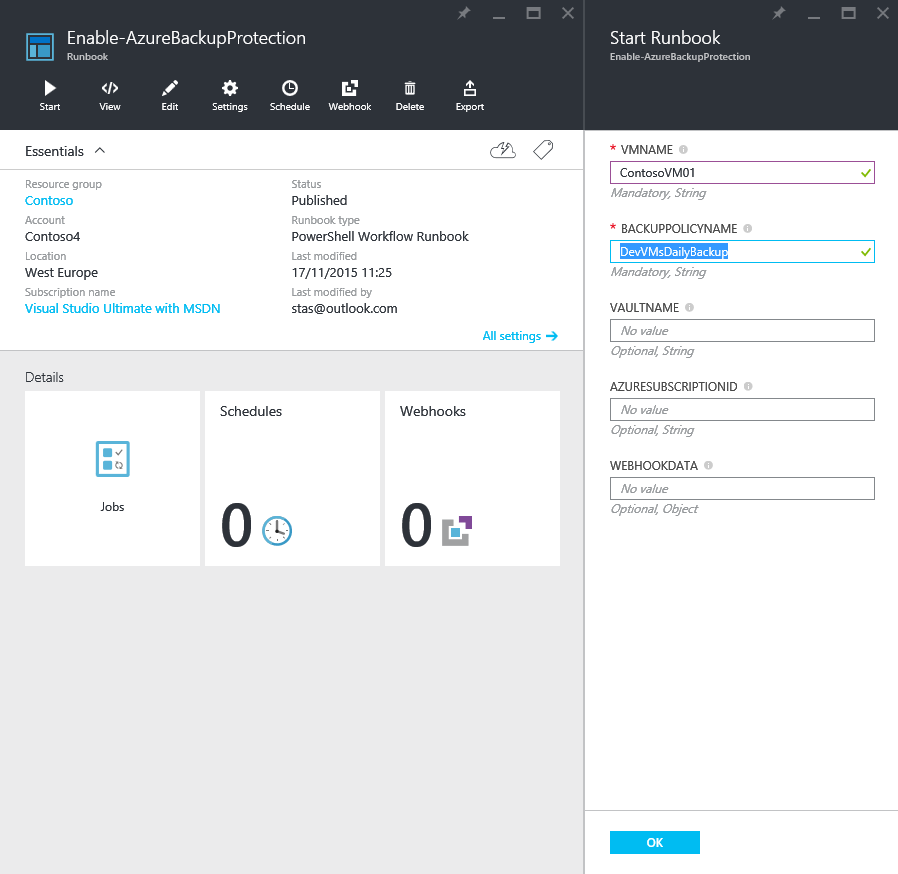
VMResourceGroup: <The Name of the Resource Group where the Azure Classic VM is located>



1. A job will be started to register Azure Classic VM to Backup Vault.
2. Once the job is completed the Azure Classic VM will be register to Backup Vault.
3. Select Enable-AzureBackupProtection runbook. This runbook will assign backup policy to the registered VM so it can start protecting it. Click Start.
4. Enter the following parameters and click OK.

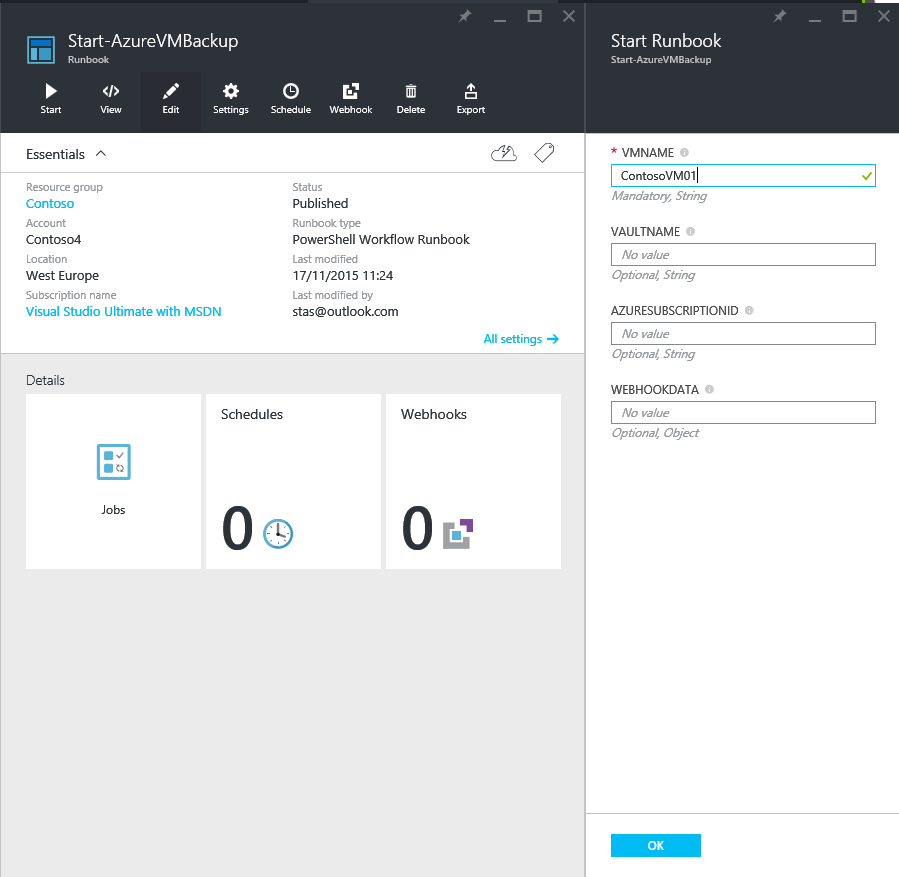
VMName: <The Name of the Azure Classic VM>

BackupPolicyName: DevVMsDailyBackup



1. A job will be started to add the VM to backup policy.
2. Once the job is completed the VM is added to backup policy.
3. Select Start-AzureVMBackup runbook. The backup of the Azure VM will start at the specified time in the backup policy but this runbook we can initiate the backup now. Click Start.
4. Enter the following parameters and click OK.

VMName: <The Name of the Azure Classic VM>



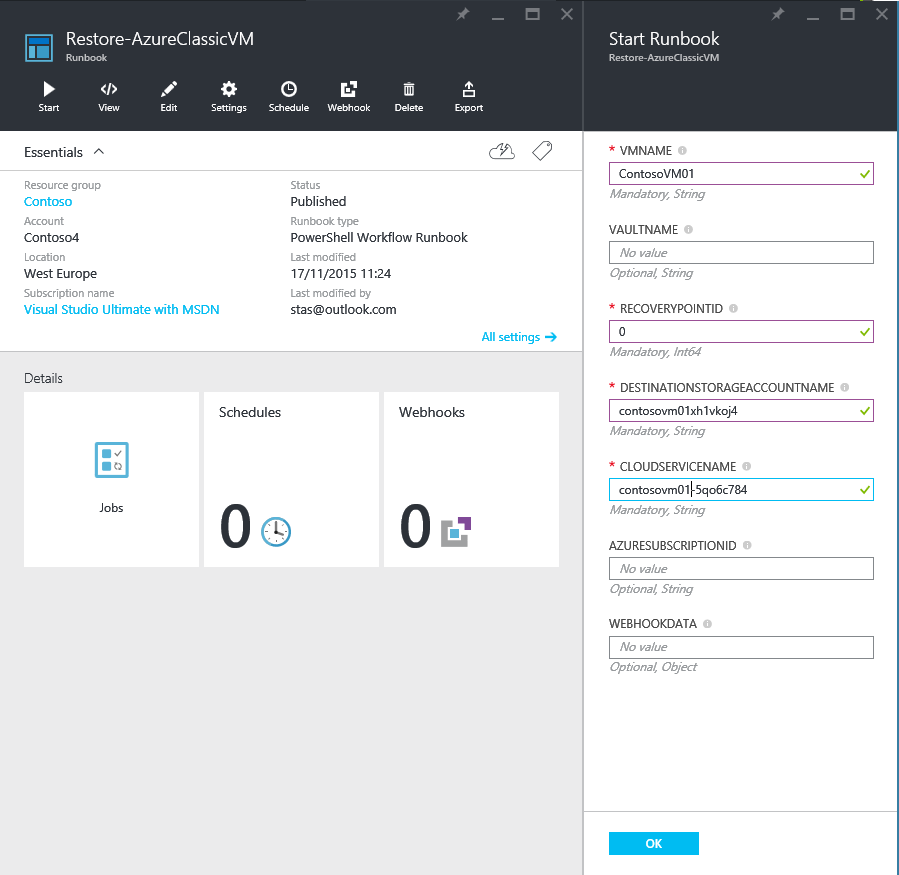
1. A job will be started to add the VM to backup policy. The backup job can take some time until it is completed. This job needs to be completed in order to use the next runbook.
2. Once the job is completed a recovery point will be created for the VM.
3. Select Restore-AzureClassicVM runbook. This runbook will restore Azure Classic VM from Backup. The existing Classic VM needs to be deleted in advance, because it will be restored in the same Cloud Service. If needed login as Admin to Azure Portal and delete the VM only with its disks. Click Start.
4. Enter the following parameters and click OK.

VMName: <The Name of the Azure Classic VM>

RecoveryPointID <Enter 0 to restore from latest recovery point>

DestinationStorageAccountName: <The same storage account name where previous VM (before restore) was located>

CloudServiceName: <The cloud service where the previous VM was located>



1. A job will be started to add the VM to backup policy. The VM will be restored with previous size and disks.
2. Once the job is completed the VM will be in booting state. Additional network settings like endpoints may need to be configured after the restore.

**Summary**

You have now successfully completed Lab 4, where you have done the following:

* As Tenant you used several Automation runbooks to manage Azure Backup created by Service Admin

1. Lab 5: Create Webhook

This lab will focus on creating a webhook for a runbook that will be given to Tenant who will use it to execute the runbook with simple HTTP request.

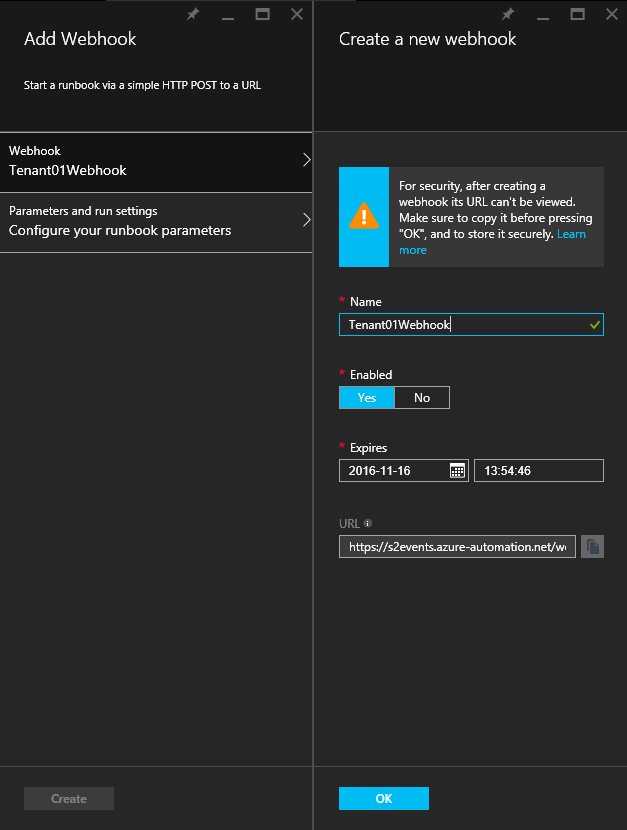
1. From your computer, open your preferred web browser and navigate to <https://portal.azure.com>
2. Sign in with your credentials that has access to an Azure subscription as Admin.
3. Open Automation accoutns blade.
4. Select the Automation account createad in previous lab.
5. Click on Runbooks tile.
6. Select Start-AzureVMBackup runbook.
7. Click on Webhook.
8. From Add Webhook blade click on Create new webhook.
9. Enter the following parameters and click OK.

Name: Tenant01Webhook

Enabled: Yes

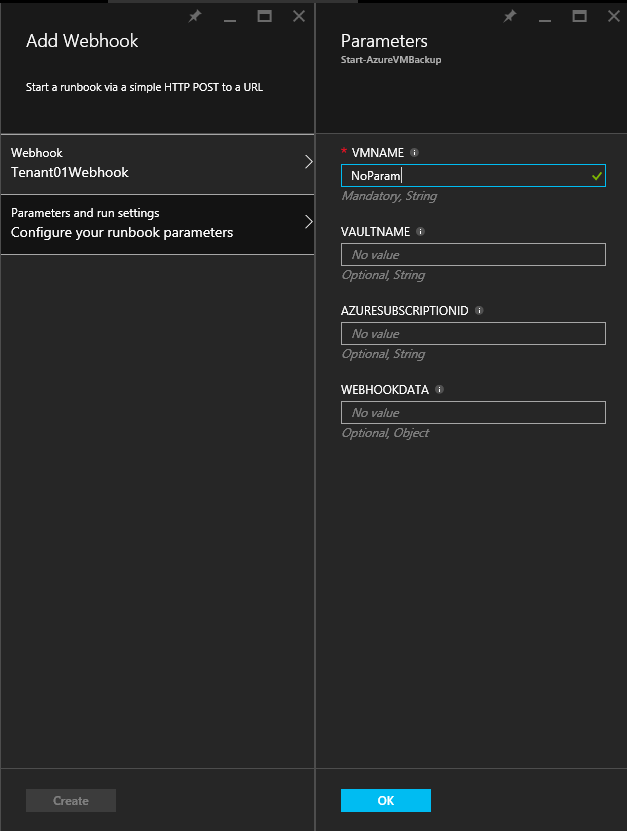
Expires: <leave default>

URL: <leave default cannot be changed. Copy the URL>



1. From Add Webhook blade click on Configure your runbook parameters.
2. Enter the following parameters and click OK.

VMname: NoParam



1. From Add Webhook blade click on Create.
2. Once the webhook is create it will be visible by clicking on the webhook tile for that specific runbook. URL needs to be saved on secure place as once created it cannot be retrieved.

**Summary**

You have now successfully completed Lab 5, where you have done the following:

* Created webhook for a runbook for which the URL will be passed to Tenant

1. Lab 6: Execute Runbook via Webhook

This lab will focus on executing runbook via webhook with simple HTTP request generated with PowerShell.

1. From your computer open PowerShell ISE.
2. Copy and paste the code below in PowerShell ISE.

# Variables to Set

$WebhookURI = "https://s2events.azure-automation.net/webhooks?token=CU3sajg3Sq%2f3QVrzvONlACbK5iFmJJq6pYUKAWY4Bzw%3d"

$VMName = "ContosoVM01"

# Static Variables

$headers = @{"AuthorizationValue"="Contoso"}

$WebhookBody = @([pscustomobject]@{VMName=$VMName})

# Convert to JSON format

$body = ConvertTo-Json `

-InputObject $WebhookBody

# Execute HTTP request

$response = Invoke-WebRequest `

-Method Post `

-Uri $WebhookURI `

-Headers $headers `

-Body $body

# Show response

$response

#Get Job ID

$jobid = (ConvertFrom-Json ($response.Content)).jobids[0]

# Show Job ID information

$jobid

1. Change the first two variables according to your environment.
2. Execute the code.
3. The response below will be returned with status code 202 which means the job has been started.

PS F:\OneDrive\CSP> # Show response

$response

StatusCode : 202

StatusDescription : Accepted

Content : {"JobIds":["742ef588-8ee6-43e9-82da-29f0efff7522"]}

RawContent : HTTP/1.1 202 Accepted

Pragma: no-cache

Strict-Transport-Security: max-age=31536000; includeSubDomains

Content-Length: 51

Cache-Control: no-cache

Content-Type: application/json; charset=utf-8

Dat...

Forms : {}

Headers : {[Pragma, no-cache], [Strict-Transport-Security, max-age=31536000; includeSubDomains], [Content-Length, 51], [Cache-Control, no-cache]...}

Images : {}

InputFields : {}

Links : {}

ParsedHtml : mshtml.HTMLDocumentClass

RawContentLength : 51

PS F:\OneDrive\CSP> #Get Job ID

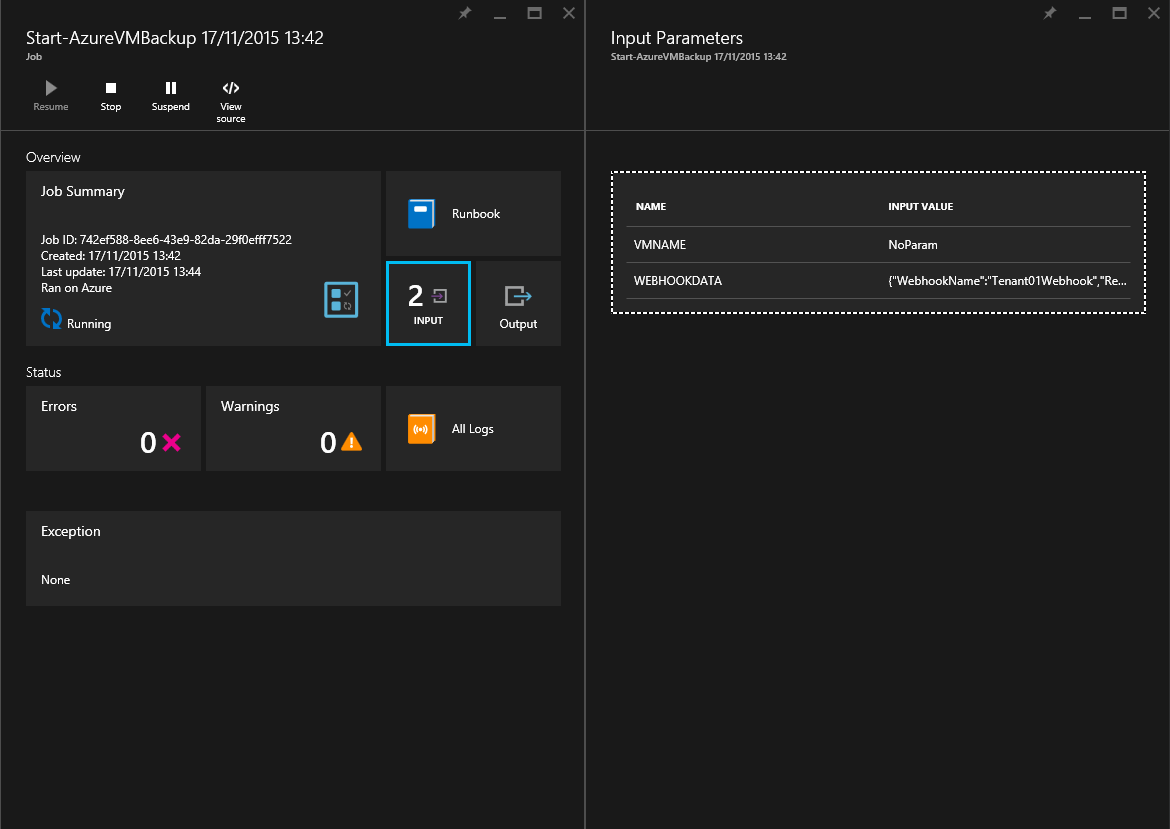
$jobid = (ConvertFrom-Json ($response.Content)).jobids[0]

PS F:\OneDrive\Customers\CSP> # Show Job ID information

$jobid

742ef588-8ee6-43e9-82da-29f0efff7522

1. From your computer, open your preferred web browser and navigate to <https://portal.azure.com>
2. Sign in with your credentials that has access to an Azure subscription as Admin.
3. Open Automation accoutns blade.
4. Select the Automation account createad in previous lab.
5. Click on Jobs tile.
6. The last job should be in Starting or Running state.
7. Select the last job.
8. Click on Input tile to see what data has been passed.



**Summary**

You have now successfully completed Lab 6, where you have done the following:

* Executed runbook via webhook with HTTP request

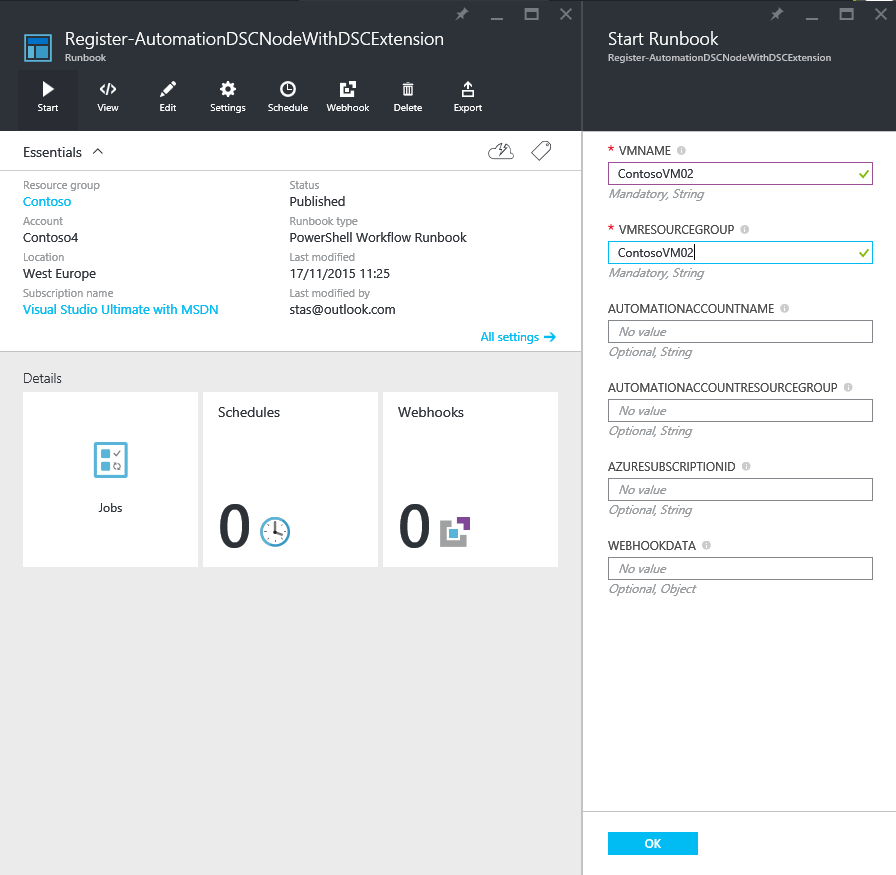
1. Lab 7: Register Azure VM Autoamtion DSC

This lab will focus on registering Azure v2 VM in Automation DSC as Tenant.

1. From your computer, open your preferred web browser and navigate to <https://portal.azure.com>
2. Sign in with your credentials that has access to an Azure subscription as Tenant.
3. Open Automation accoutns blade.
4. Select the Automation account createad in previous lab.
5. Click on Runbooks tile.
6. Select Register-AutomationDSCNodeWithDSCExtension runbook. This runbook will use DSC VM extension to register Azure v2 VM to Automation DSC as node. Click Start.
7. Enter the following parameters and click OK.

VMName: <Azure v2 Windows VM in a workgroup. This VM will be later joined into domain via Automation DSC.>

VMResourceGroup: <The resource group where this Azure v2 VM is located.>



1. A job will be started. The VM will be restarted during adding to Automation DSC for installing WMF 5.0
2. Once the job is completed the VM will be added as node to Automation DSC.

**Summary**

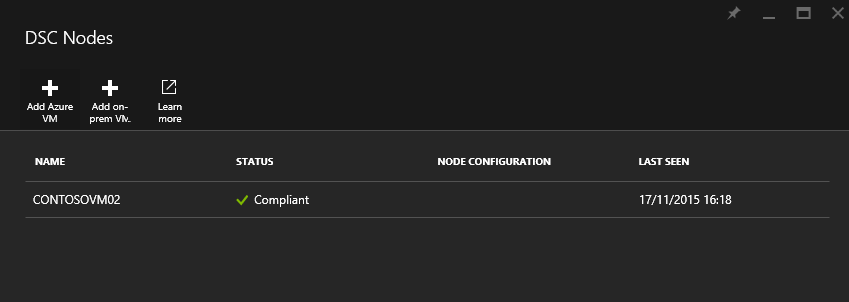
You have now successfully completed Lab 7, where you have done the following:

* Executed runbook that adds Azure v2 VM to Automation DSC.

1. Lab 8: Apply DSC Configuration

This lab will focus on applying DSC Configuration to a computer. The computer will be in a workgroup and configuration will be applied to join it to domain.

1. From your computer, open your preferred web browser and navigate to <https://portal.azure.com>
2. Sign in with your credentials that has access to an Azure subscription as Admin.
3. Open Automation accoutns blade.
4. Select the Automation account createad in previous lab.
5. Click on DSC Nodes tile. You will see one DSC node available.



1. Open PowerShell ISE on your local computer.
2. Execute the code below to import DSC Configuration, compile DSC Configuration and apply the Node Conifuration to DSC Node. Change the variables according to your environment.

# Variables to set

$AzureSubscriptionID = '3c1d65g1-4064-4522-94e4-e0378165922e'

$AutomationAccountResourceGroupName = 'Contoso'

$AutomationAccountName = 'Contoso4'

$DomainJoinConfigPath = 'C:\CSP\WorkgroupToDomain.ps1'

$VM = 'ContosoVM02'

$DomainName = 'contoso.com'

# Static Variables

$DomainJoinConfigName = 'WorkgroupToDomain'

# Login to Azure

Add-AzureRmAccount `

-Credential (Get-Credential) `

-SubscriptionId $AzureSubscriptionID `

# Import DSC Configuration

Import-AzureRmAutomationDscConfiguration `

-SourcePath $DomainJoinConfigPath `

-Description 'Configuration for Domain Join' `

-ResourceGroupName $AutomationAccountResourceGroupName `

-AutomationAccountName $AutomationAccountName `

-Published `

-force

$ConfigData = @{

AllNodes = @(

@{

NodeName = '\*'

PSDscAllowPlainTextPassword = $True

},

@{

NodeName = $VM

DomainName = $DomainName

}

)

}

$CompilationJob = Start-AzureRmAutomationDscCompilationJob `

-ResourceGroupName $AutomationAccountResourceGroupName `

-AutomationAccountName $AutomationAccountName `

# Login to Azure

Add-AzureRmAccount `

-Credential (Get-Credential) `

-SubscriptionId $AzureSubscriptionID `

# Import DSC Configuration

Import-AzureRmAutomationDscConfiguration `

-SourcePath $DomainJoinConfigPath `

-Description 'Configuration for Domain Join' `

-ResourceGroupName $AutomationAccountResourceGroupName `

-AutomationAccountName $AutomationAccountName `

-Published `

-force

# Configuration Data

$ConfigData = @{

AllNodes = @(

@{

NodeName = '\*'

PSDscAllowPlainTextPassword = $True

},

@{

NodeName = $VM

DomainName = $DomainName

}

)

}

# Compile DSC Configuration to Node Configuration

$CompilationJob = Start-AzureRmAutomationDscCompilationJob `

-ResourceGroupName $AutomationAccountResourceGroupName `

-AutomationAccountName $AutomationAccountName `

-ConfigurationName $DomainJoinConfigName `

-ConfigurationData $ConfigData

# Wait for Complication to finish

while($CompilationJob.EndTime –eq $null -and $CompilationJob.Exception –eq $null)

{$CompilationJob = $CompilationJob | Get-AzureRmAutomationDscCompilationJob

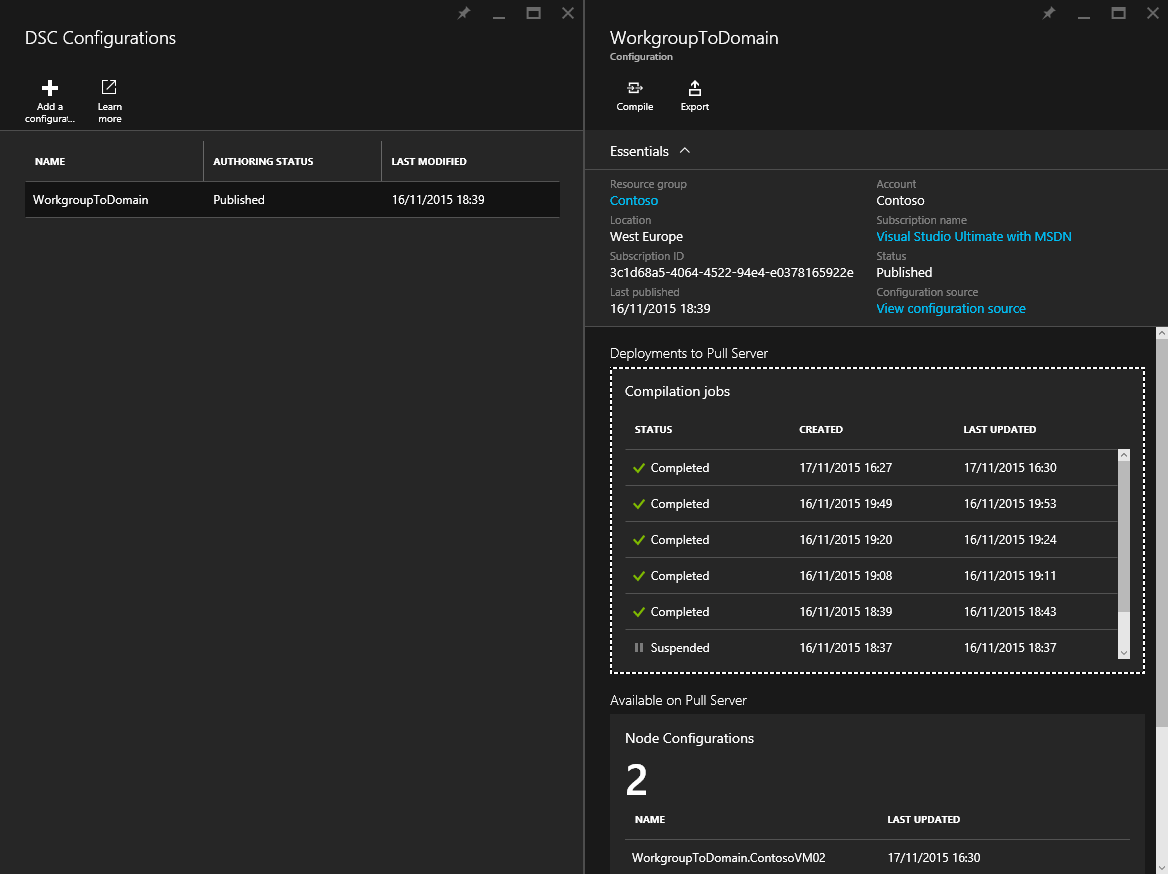
Start-Sleep -Seconds 3

}

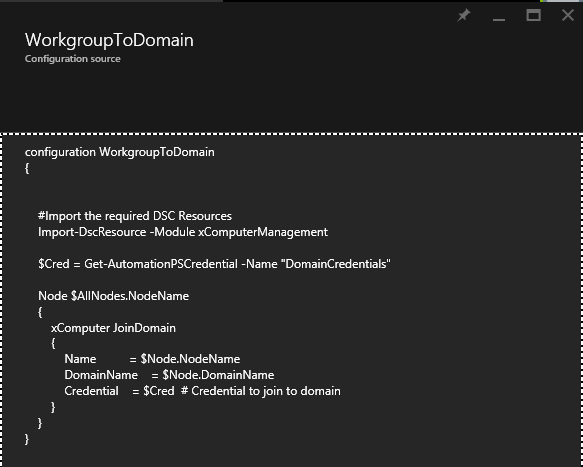
# Compliation job output

$CompilationJob | Get-AzureRmAutomationDscCompilationJobOutput –Stream Any

1. After the job compilation is completed navigate to Azure Portal -> Automation accounts blade.
2. Select the automation account created in previous lab. Click on DSC Configurations tile.
3. Click on WorkgroupToDomain configuration.



1. You can see the last compilation job as completed.
2. Click on View Configuration source to see the actual DSC Confiugration imported.



1. Execute the rest of the code below in PowerShell ISE to apply the Node Configuration to DSC Node.

# Get Azure Automation DSC Node

$NodeObj = Get-AzureRmAutomationDscNode `

-Name $VM `

-ResourceGroupName $AutomationAccountResourceGroupName `

-AutomationAccountName $AutomationAccountName

# Get Azure Automation Node Configuration

$ConfigurationObj = Get-AzureRmAutomationDscNodeConfiguration `

-ConfigurationName $DomainJoinConfigName `

-ResourceGroupName $AutomationAccountResourceGroupName `

-AutomationAccountName $AutomationAccountName

# Apply DSC node Configuration to DSC Node

Set-AzureRmAutomationDscNode `

-NodeConfigurationName $ConfigurationObj.Name `

-Id $NodeObj.Id `

-ResourceGroupName $AutomationAccountResourceGroupName `

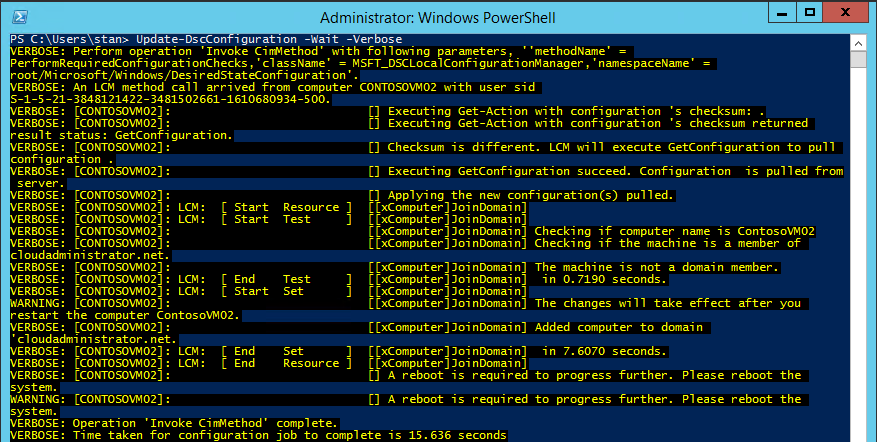
-AutomationAccountName $AutomationAccountName `

-Force

1. Logon via RDP to the computer to which you apply this configuration and in PowerShell console executed the command below.

# Execute locally on the machine

Update-DscConfiguration -Wait -Verbose



1. After the configuration is applied the computer will be joined to the Active Direcotry domain and restarted automatically.
2. The compliance of the DSC node can be checked with the code below.

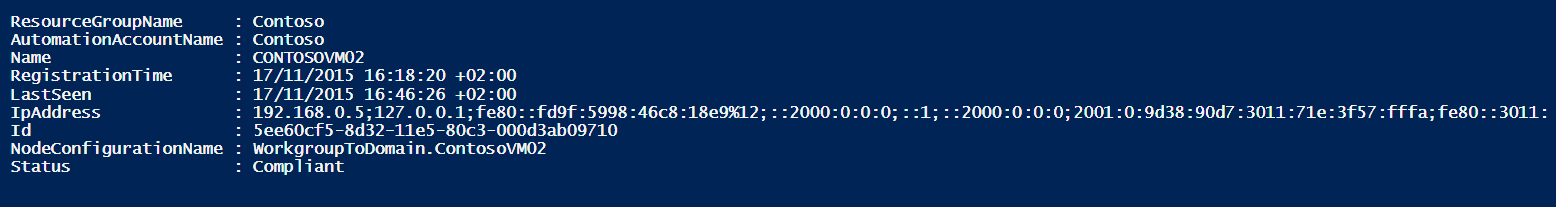
# Verify Compliance

Get-AzureRmAutomationDscNode `

-Id $NodeObj.Id `

-ResourceGroupName $AutomationAccountResourceGroupName `

-AutomationAccountName $AutomationAccountName



**Summary**

You have now successfully completed Lab 8, where you have done the following:

* Create DSC Configuration
* Compile DSC configuration
* Apply DSC Node Configuration
* Check DSC Node compliance

If the customer does not have a properly configured VPN or Azure Subscription in place, please have them consider the Azure IaaS Foundations IP at http://aka.ms/MCS\_EPG\_Azure\_Iaas-Foundation.

These are just examples; update the requirements adapting to the specific customer situation if necessary. Points to consider might include: tool usage (MAP versus customer provided), server locations, availability of test lab, and so on.