Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps

Authors:

Yordan Bechev
Premier Field Engineer at Microsoft
yordan.bechev@microsoft.com

Yorick Kuijs
Premier Field Engineer at Microsoft
yorick.kuijs@microsoft.com

Date: October 1st 2021
Version: v1.2

Disclaimer

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

Changelog

Version	Date	Author	Changes
1.0	November 1st 2020	Yordan Bechev Yorick Kuijs	First release
1.0.1	November 3rd 2020	Yorick Kuijs	Updated incorrect links
1.1	December 2nd 2020	Yorick Kuijs	Incorporated feedback from Zaki Semar Shahul Added Azure Conditional Access for the used service account
1.2	October 1st 2021	Yorick Kuijs	Corrected issues Added Certificate authentication scenario

Table of Contents

1 Introduction

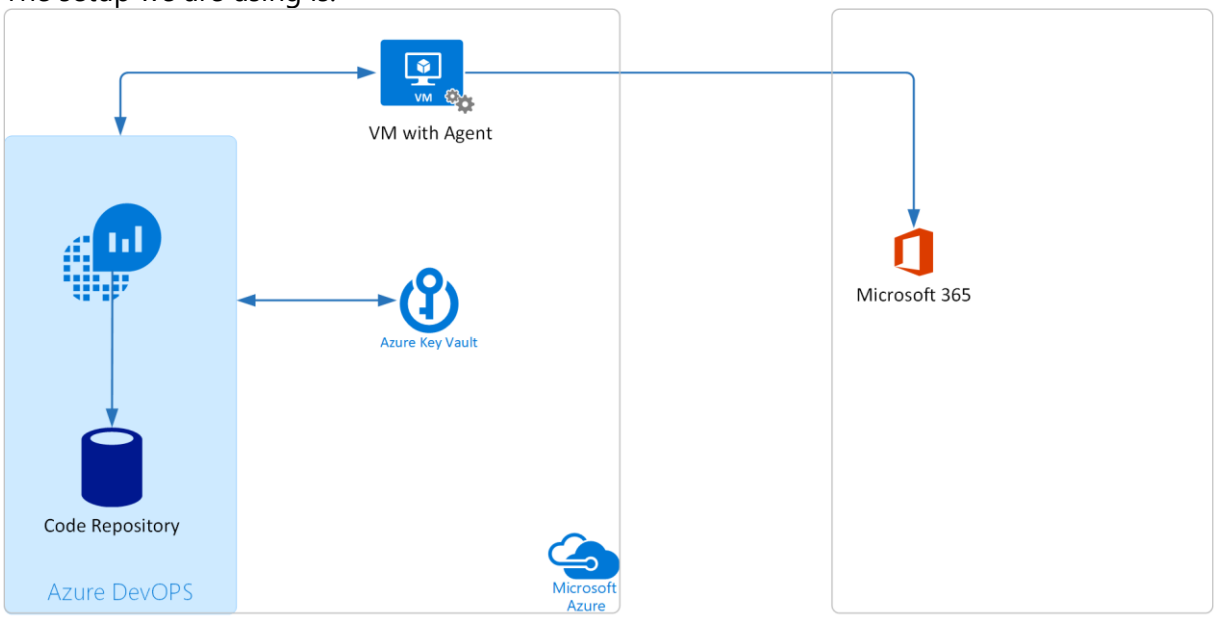
Microsoft 365 is the very popular productivity cloud solution of Microsoft. Each customer has its own tenant in which their data is stored. Using the Administration Portal (https://admin.microsoft.com) each customer can configure and manage their own tenant.

Many companies are adopting DevOps practices and are applying these practices against Microsoft 365 as well. Infrastructure as Code and Continuous Deployment/Continuous Integration are important concepts in DevOps.

Microsoft365Dsc is a PowerShell Desired State Configuration (DSC) module, which can configure and manage Microsoft 365 in a true DevOps style: Configuration as Code.

In this document we are going to describe the process and steps required to implement Configuration as Code using Microsoft365Dsc, Azure DevOps and Azure KeyVault. Changes to Microsoft 365 are done on a Git repository in Azure DevOps and then fully automatically deployed to a Microsoft 365 tenant.

The setup we are using is:



Chapter 5 “Security Enhancements” describe two alternatives that implement different scenarios to enhanced security.

2 Prerequisites

To deploy DSC configurations, we need a machine that will do the actual deployment to Microsoft 365. This can be a physical or virtual machine. In this guide we assume the use of a virtual machine. The requirements for this virtual machine are:

- Windows Server 2016 or above
- .Net Framework 4.7 or higher
    - https://dotnet.microsoft.com/download/dotnet-framework
- PowerShell v5.1
    - Installed by default on all current versions of Windows Server
- Up to date PowerShellGet:
  Install-PackageProvider Nuget –Force
  Install-Module –Name PowerShellGet –Force
  Note: If you run into issues downloading these updates, check out the following ` article: https://devblogs.microsoft.com/powershell/powershell-gallery-tls-support/
- A local account with administrative privileges, to deploy configurations from Azure DevOps

We are using Azure DevOps to store, compile and deploy the configurations. This means we need:

- An Azure DevOps tenant and permissions to configure this tenant
- A project in Azure DevOps

We also need a Microsoft 365 tenant, which is going to be managed using Microsoft365Dsc. In this tenant we need:

- An account with Global Administrator privileges, used to access the Admin Portal
- A service account with Global Administrative privileges, used to deploy setting using DSC
    - This account cannot be configured to use Multi-Factor Authentication
    - The actual required permissions depend on the used resources

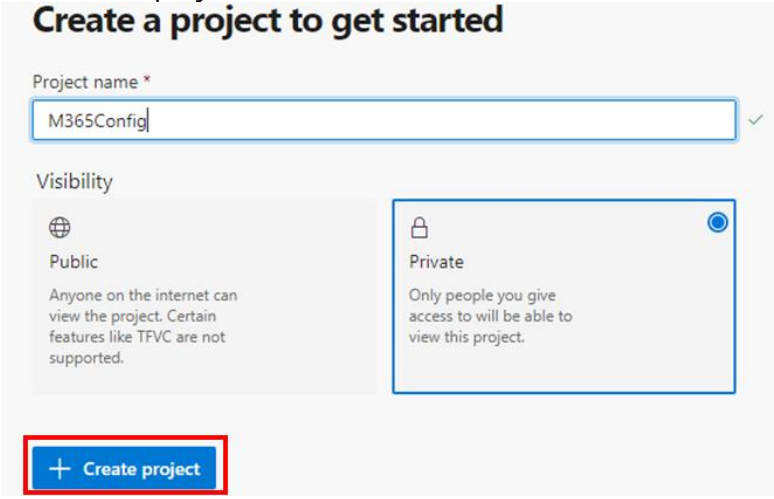
3 Preparation

3.1 Create a DSC account in Microsoft 365

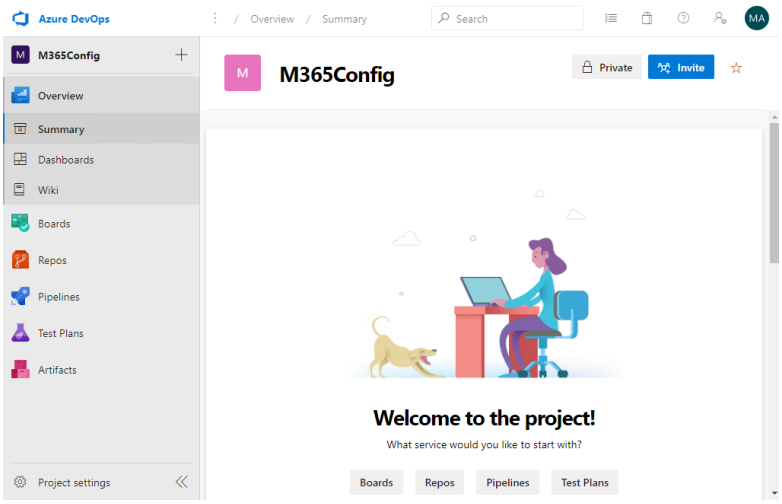
- Open an Internet browser
- Browse to the Microsoft 365 Admin Portal
- Create a new account
    - For example: DscConfigAdmin
    - Don't assign any license
    - Grant the user Global Admin permissions
        - More limited permissions possible depending on the resources in your configuration

3.2 Create a new project in Azure DevOps

- Log into the Azure DevOps portal
- Create new project



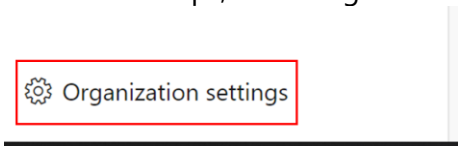
- When the project is created, the project is opened automatically



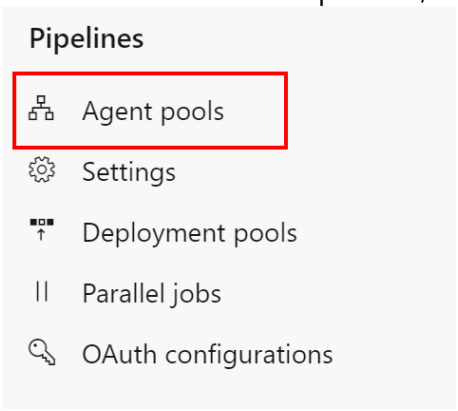
3.3 Create an Agent Pool in Azure DevOps

- Browse to the main Azure DevOps page
- Create a new Agent Pool

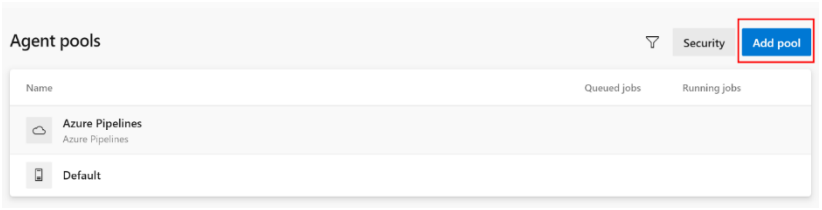
- In Azure DevOps, click "Organization Settings" in the lower left corner



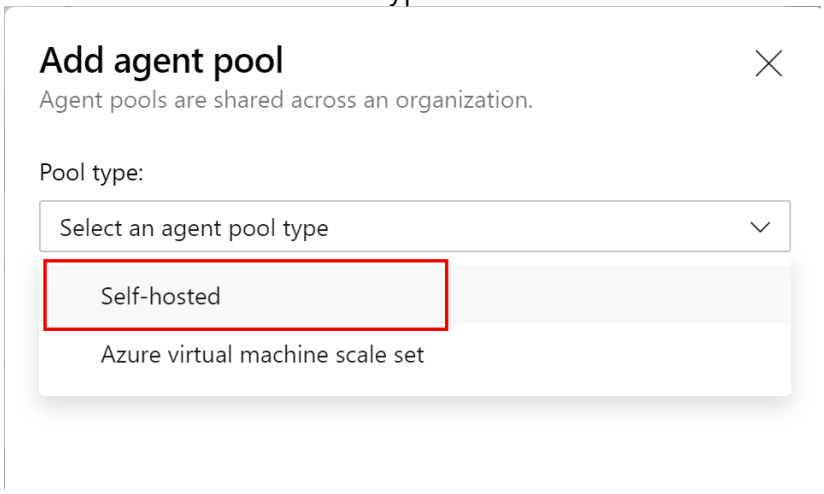
- Scroll down and under "Pipelines", click "Agent Pools"



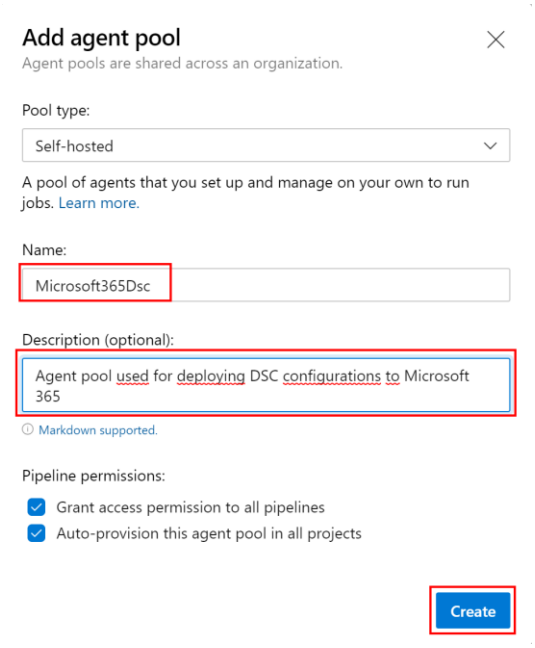
- Create a new Agent Pool by clicking the "Add pool" button in the upper right corner



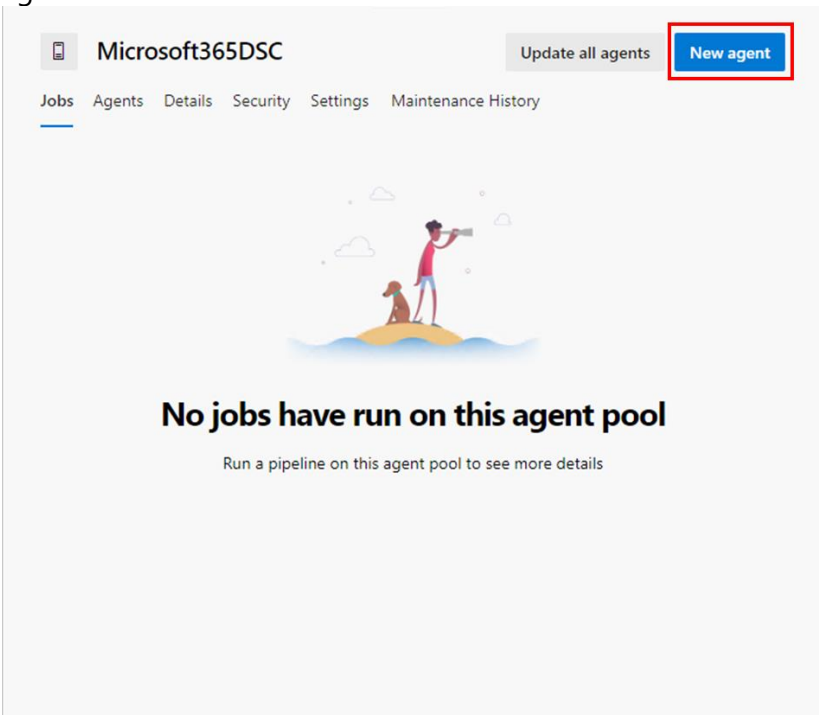
- Select “Self-hosted” as “Pool type”



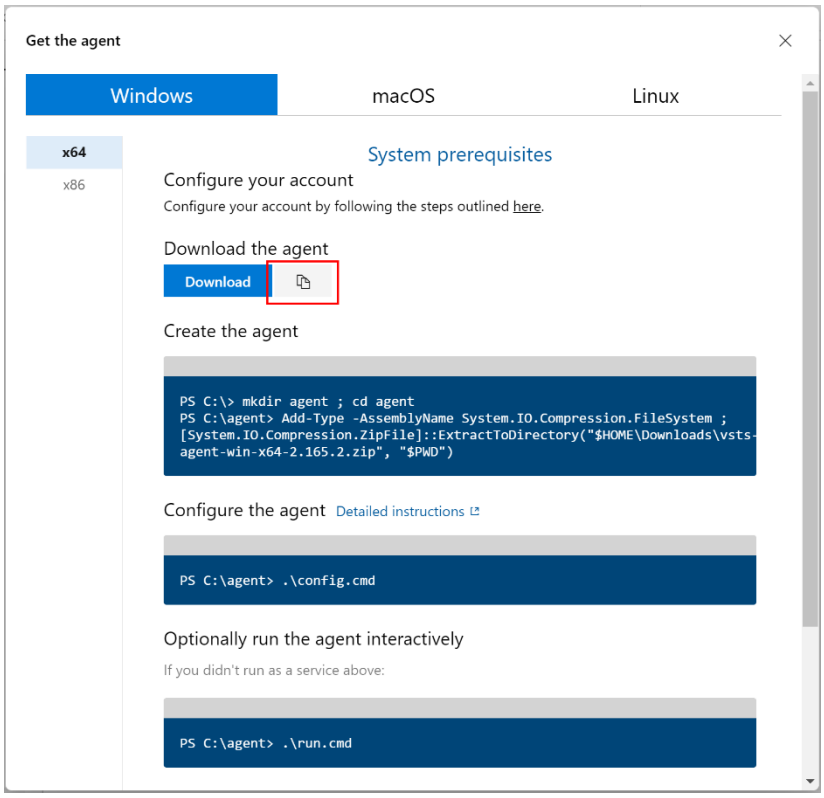
- Enter a Name (for example: Microsoft365Dsc) and Description for the new pool and click "Create"



- Click the newly created pool to open the pool
- Click the "New agent" button to open the required information to add a new agent



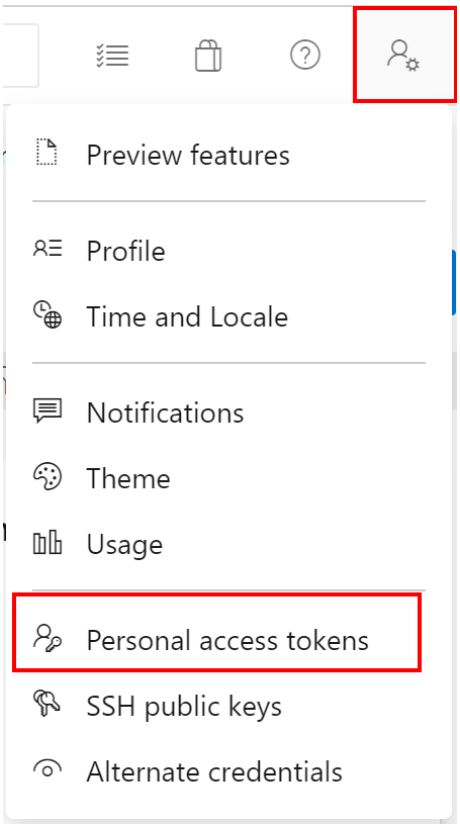
- Copy the download link and download the agent on the virtual machine



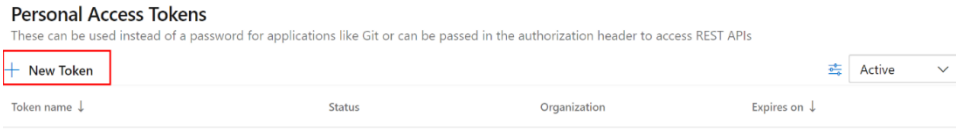
- Extract the downloaded zip file to the C:\Agent folder

3.4 Create Personal Access Token

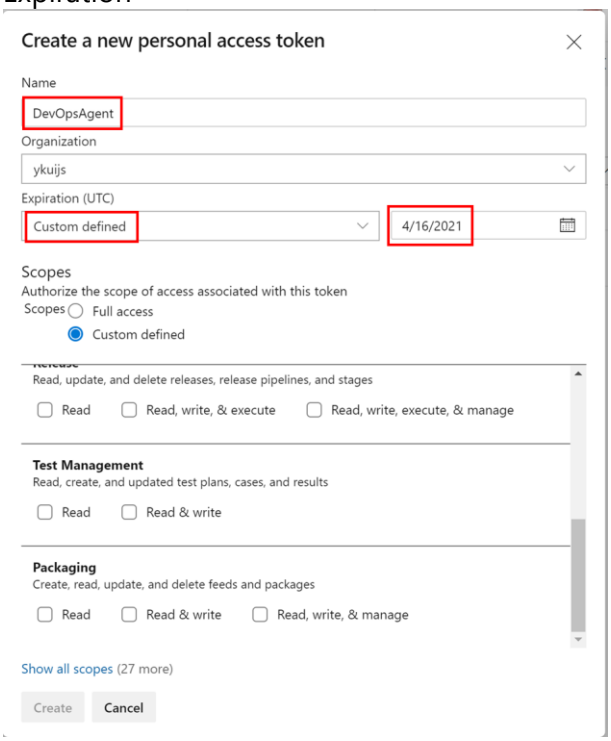
- Open Azure DevOps
- Click the user icon in the upper right corner and select the "Personal access tokens" menu item



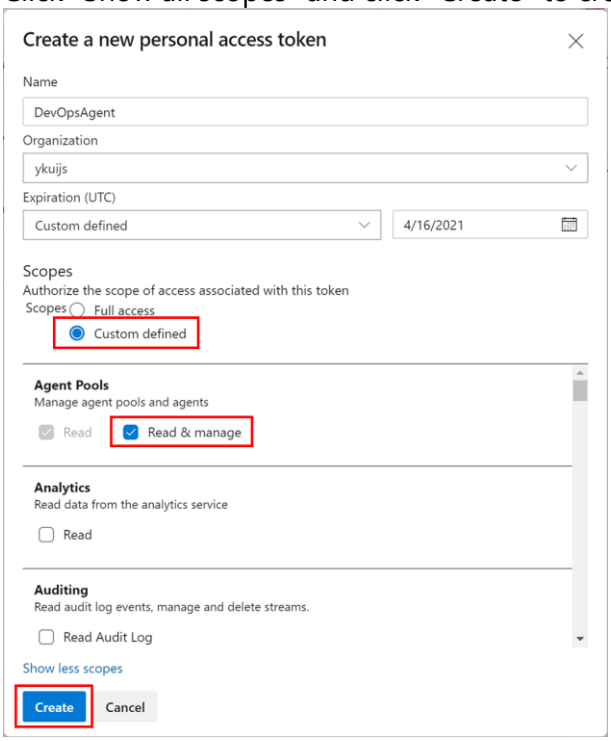
- Click "New Token" to create a new token



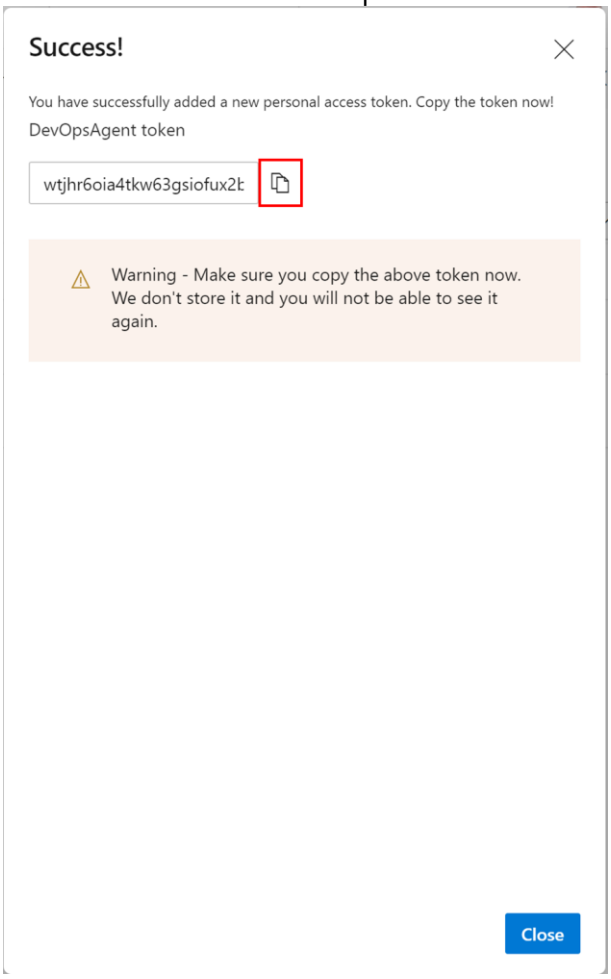
- Enter a Name and select next year (not possible to select more than a year) as Expiration



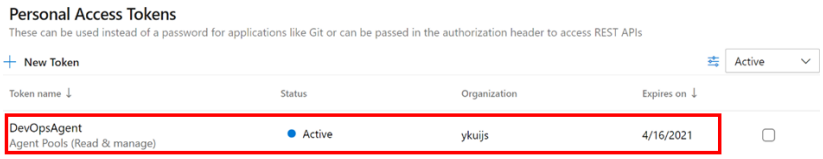
- Click "Show all scopes" and click "Create" to create the token



- IMPORTANT: Copy and store the generated token in a secure place. You cannot retrieve the token at a later point in time.

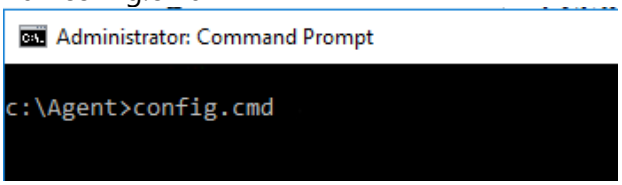


- Click "Close" to close the wizard. Your token is now created.

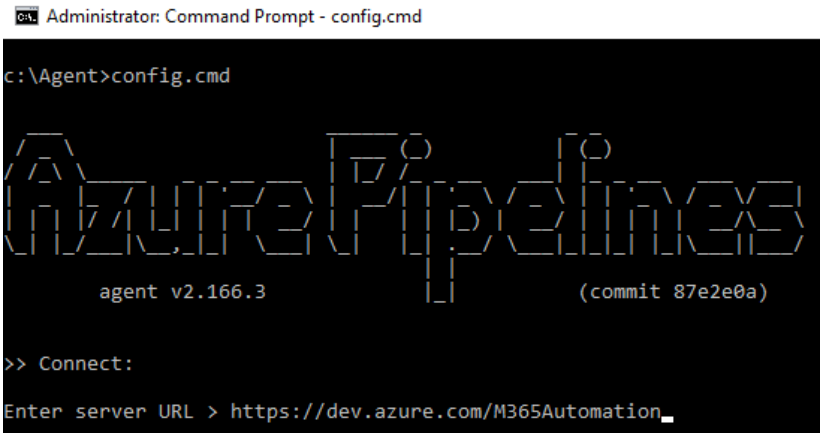


3.5 Configure Azure DevOps Agent on the virtual machine

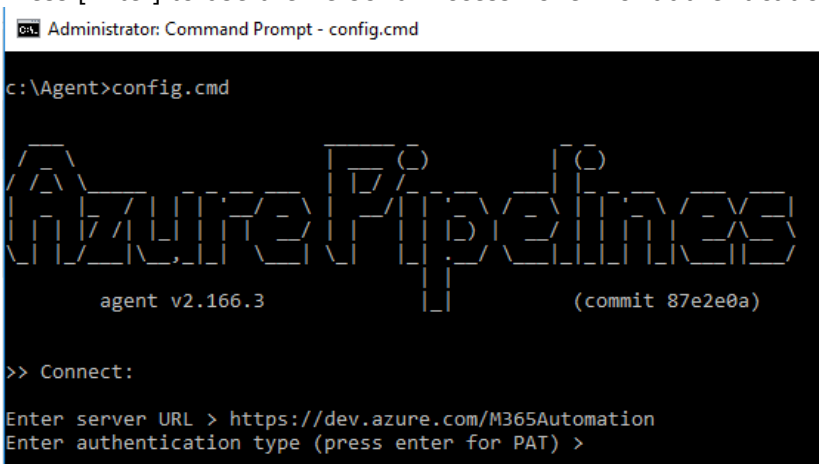
- Connect to your virtual machine
- Create a service account, either local or domain, for the Azure DevOps agent.
  - NOTE: The account needs local Administrator permissions to be able to push configurations to the Local Configuration Manager.
- Open an elevated Command Prompt
- Browse to the C:\Agent folder
- Run config.cmd



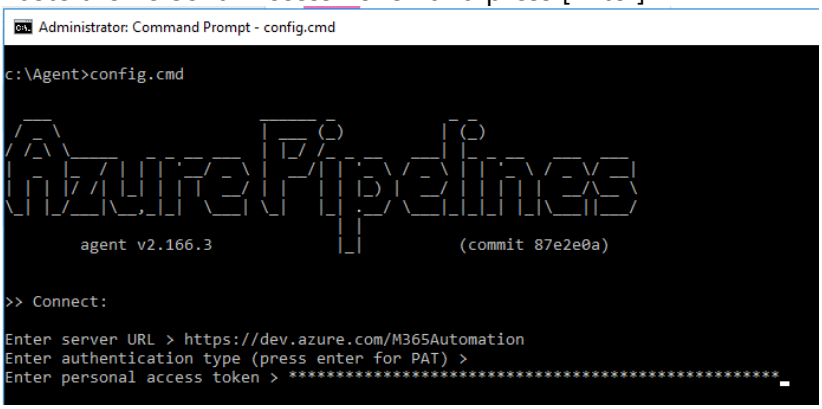
- Enter the Server URL as https://dev.azure.com/<org_name> and press [Enter]



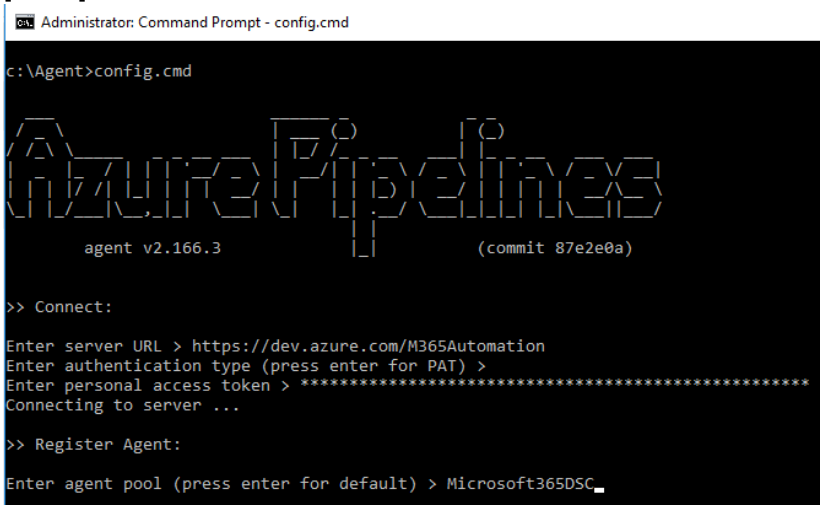
- Press [Enter] to use the Personal Access Token for authentication



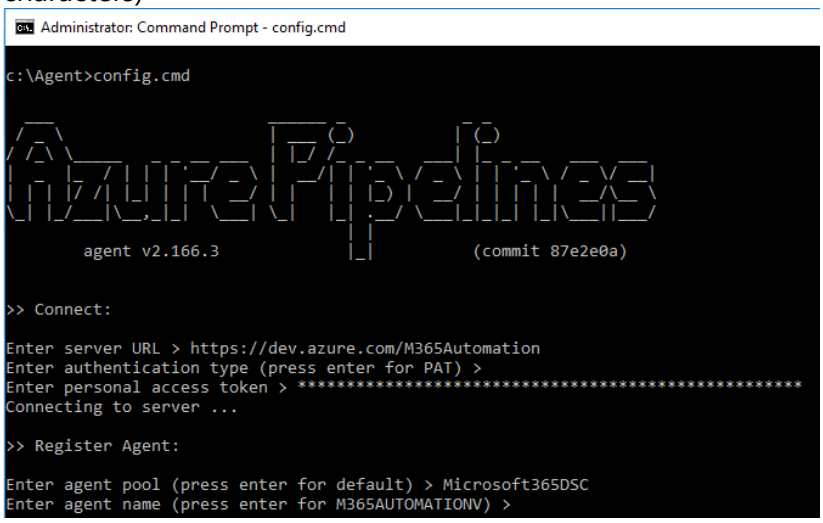
- Paste the Personal Access Token and press [Enter]



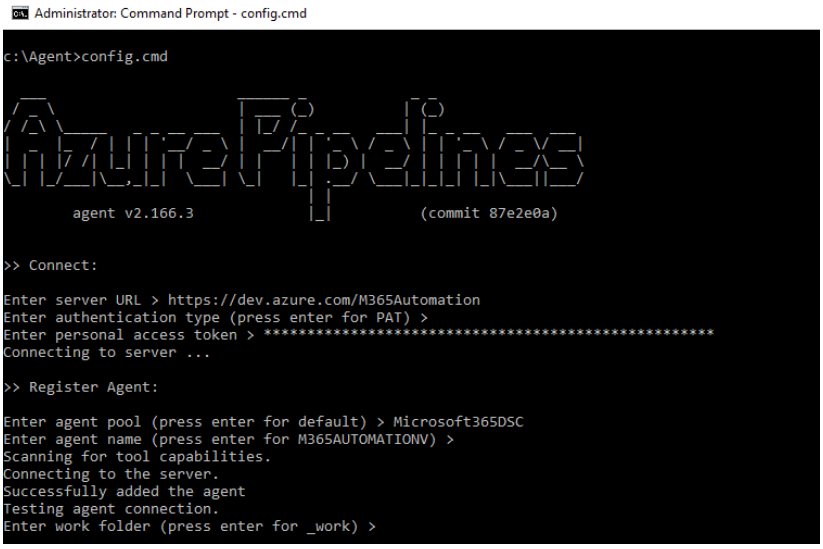
- Enter "Microsoft365Dsc" (use the name specified earlier) as the Agent Pool and press [Enter]



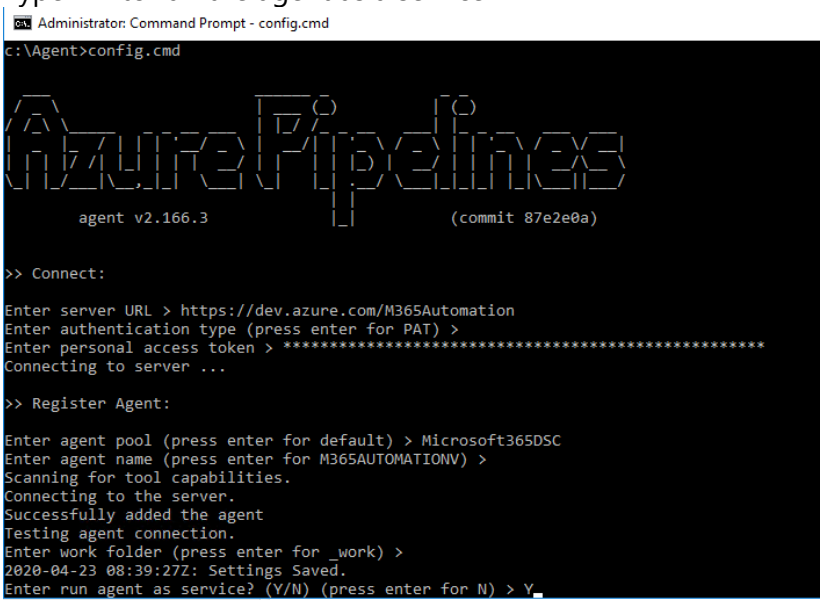
- Enter a custom Agent name or press [Enter] to use the server name (max 15 characters)



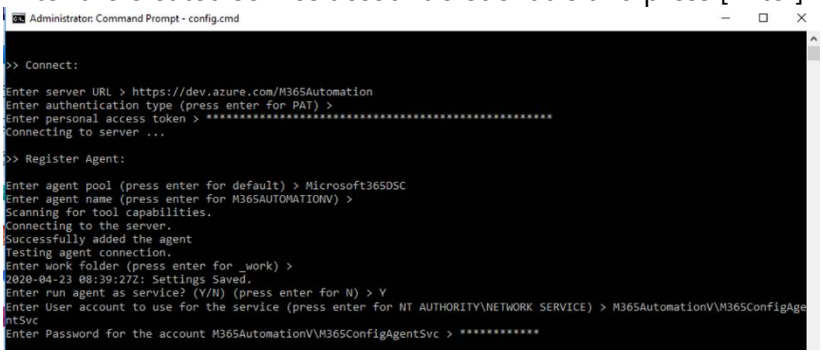
- The Agent checks some prerequisites. Press [Enter] to use the default work folder



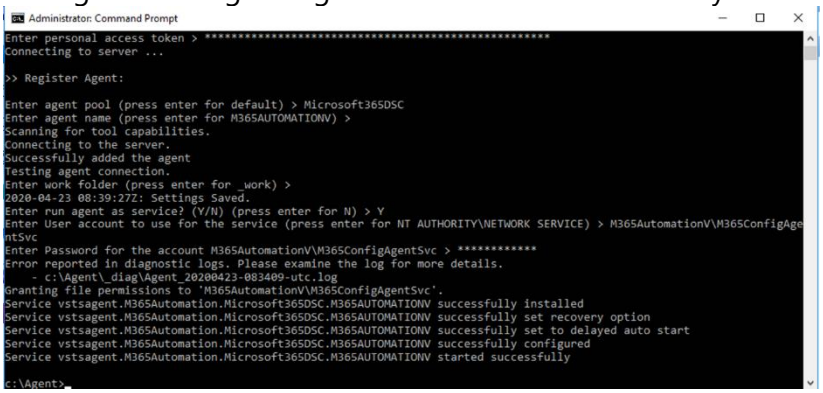
- If prompted: Press Enter to acknowledge “N” for “Perform an unzip for each step”
- Type "Y" to run the agent as a service

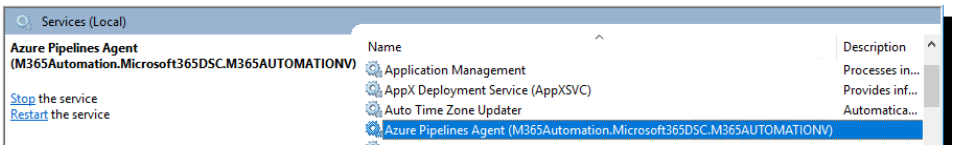


- Enter the created service account credentials and press [Enter]

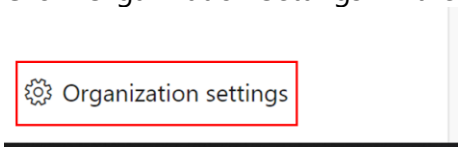


- The agent is being configured and started automatically.

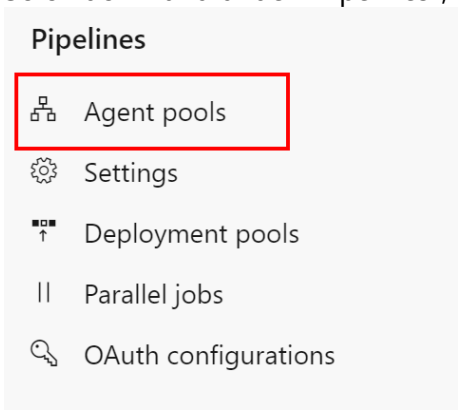




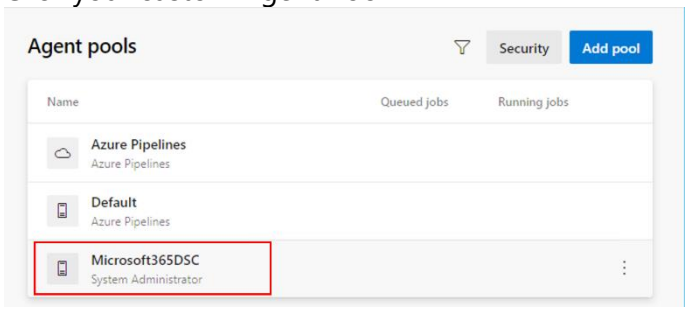
- Verify agent is successfully registered in Azure DevOps
  - Open the Azure DevOps portal
  - Click "Organization Settings" in the lower left corner



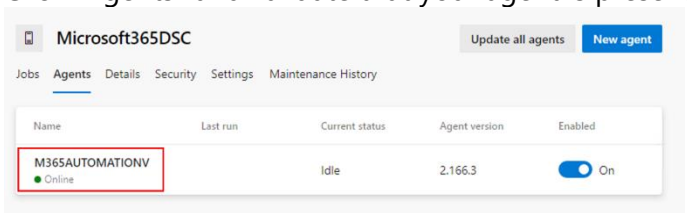
  - Scroll down and under "Pipelines", click "Agent Pools"



  - Click your custom Agent Pool



  - Click "Agents" and validate that your agent is present and Online

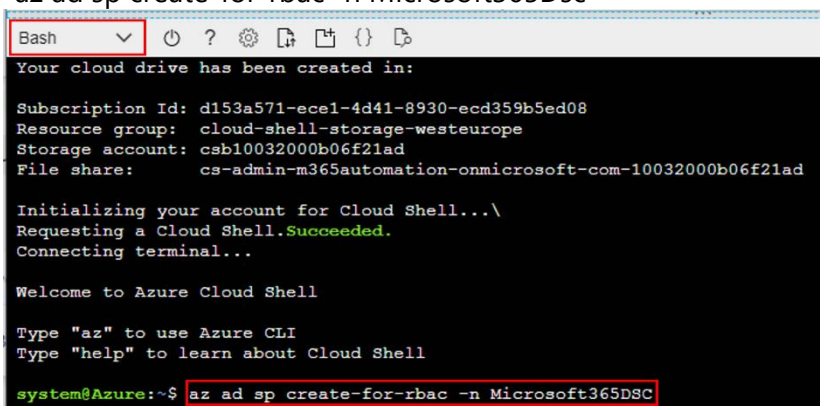


3.6 Configure Azure Key Vault

3.6.1 Create Service Principle Name

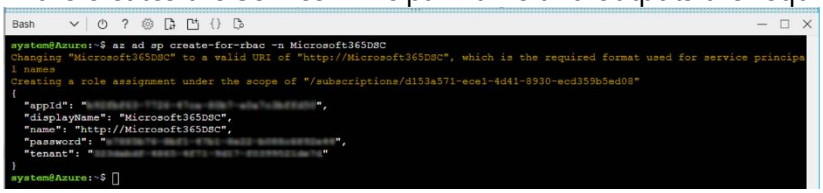
- Log into the Azure Portal
- Open Cloud Shell

- Make sure you select the Bash shell and enter:
  "az ad sp create-for-rbac -n Microsoft365Dsc"



Note: You can change "Microsoft365Dsc" for a custom string.

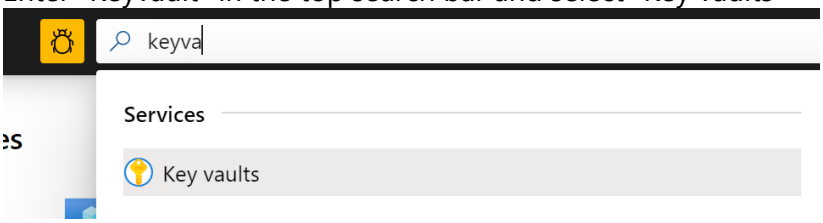
- Azure creates the Service Principal Name and outputs the required information.



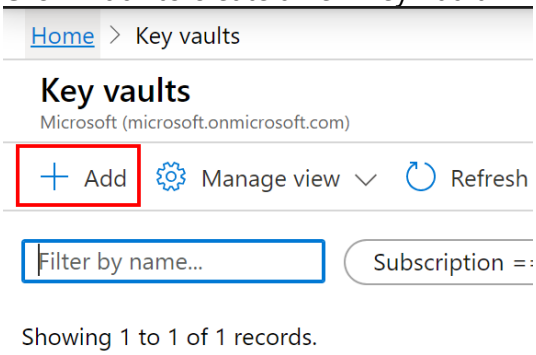
- Copy this information and store it in a secure place

3.6.2 Create Azure KeyVault

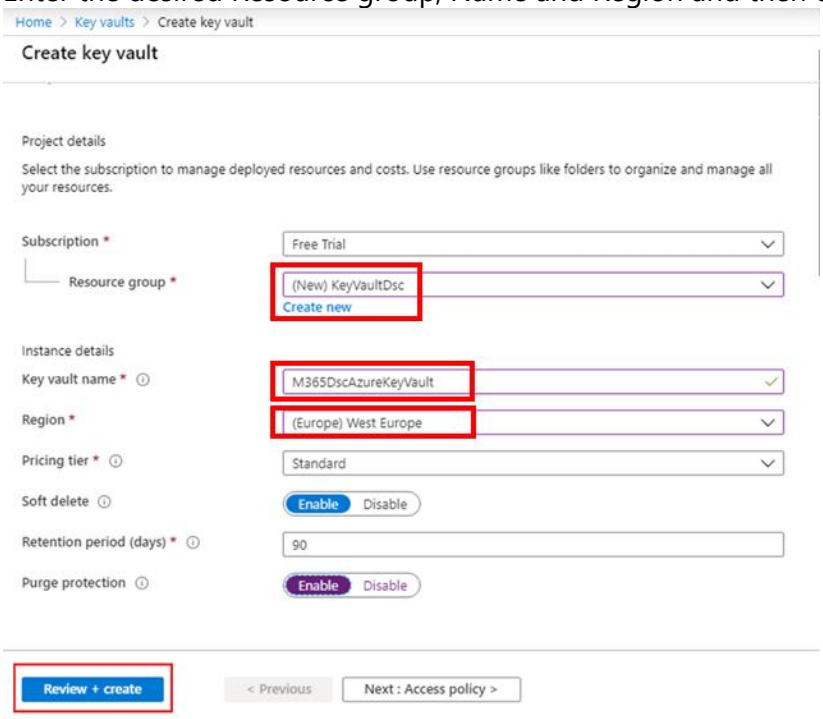
- Log into the Azure Portal
- Enter "Keyvault" in the top search bar and select "Key vaults"



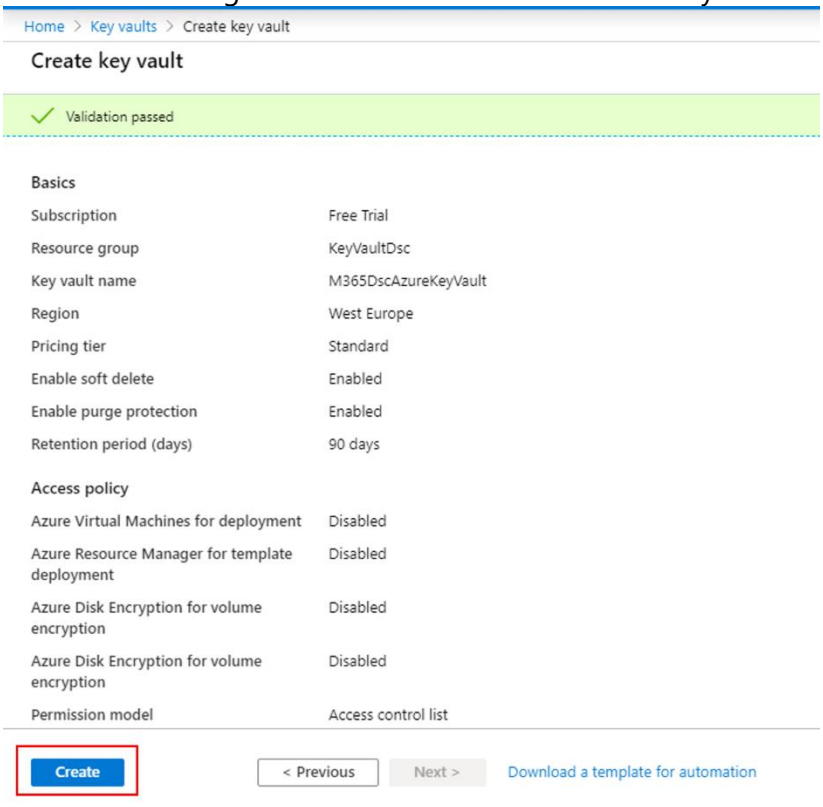
- Click "Add" to create a new Key Vault



- Enter the desired Resource group, Name and Region and then click "Review + create"

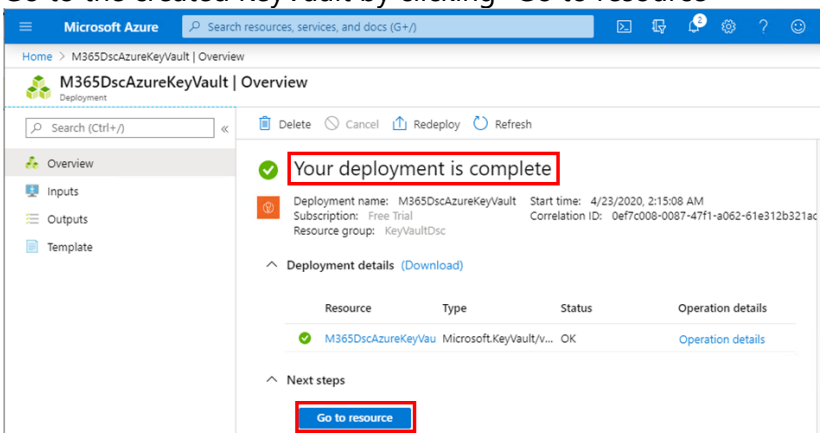


- Review the settings and click "Create" to create the Key Vault

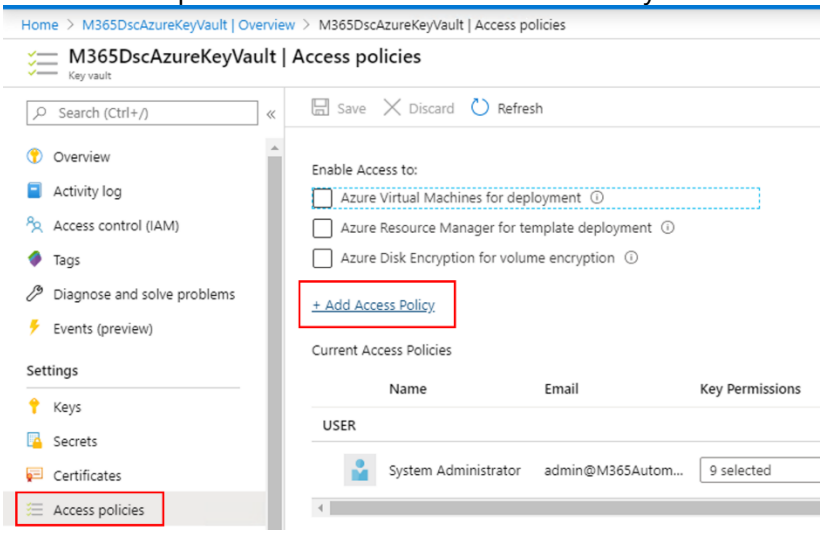


- The KeyVault will be created

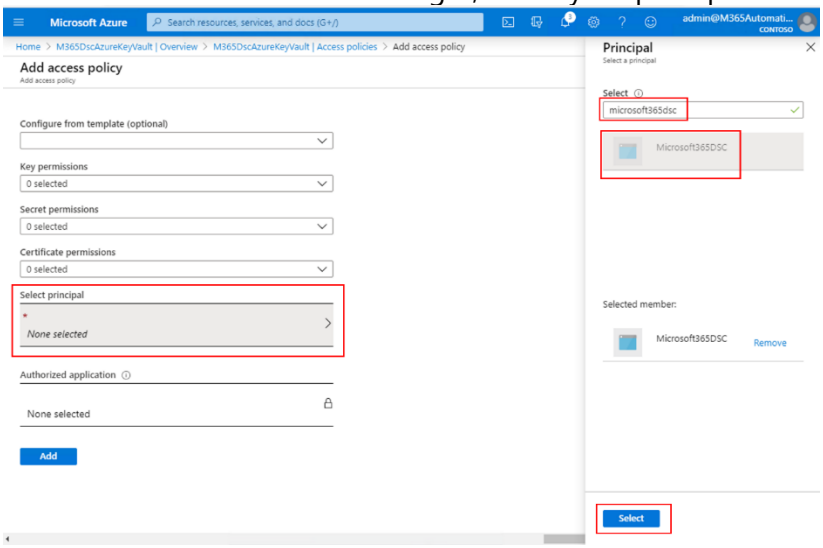
- Go to the created KeyVault by clicking "Go to resource"



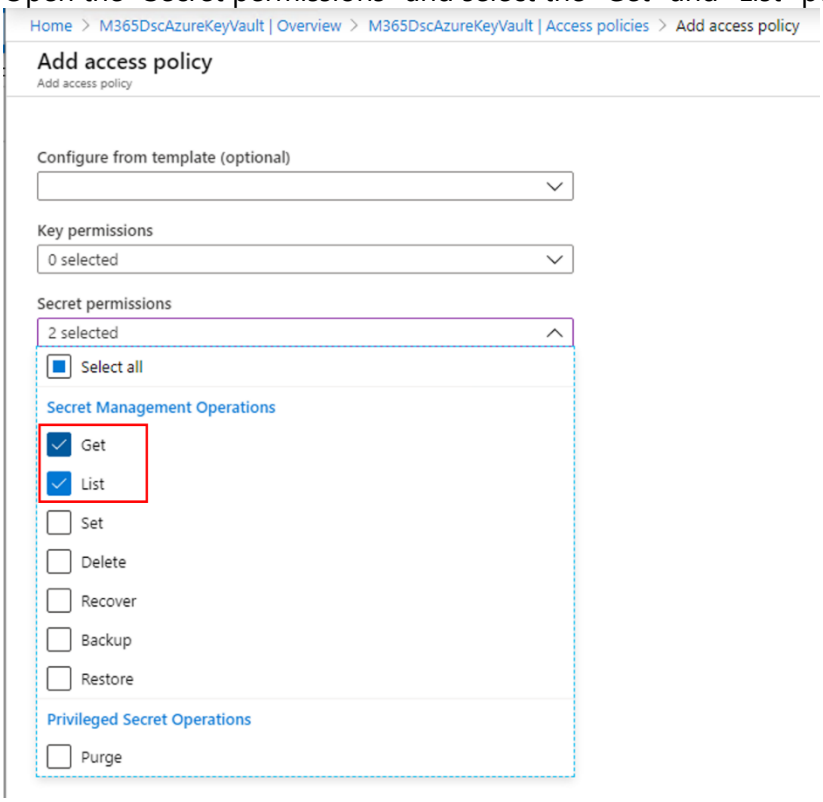
- Click "Access policies" and click "Add Access Policy"



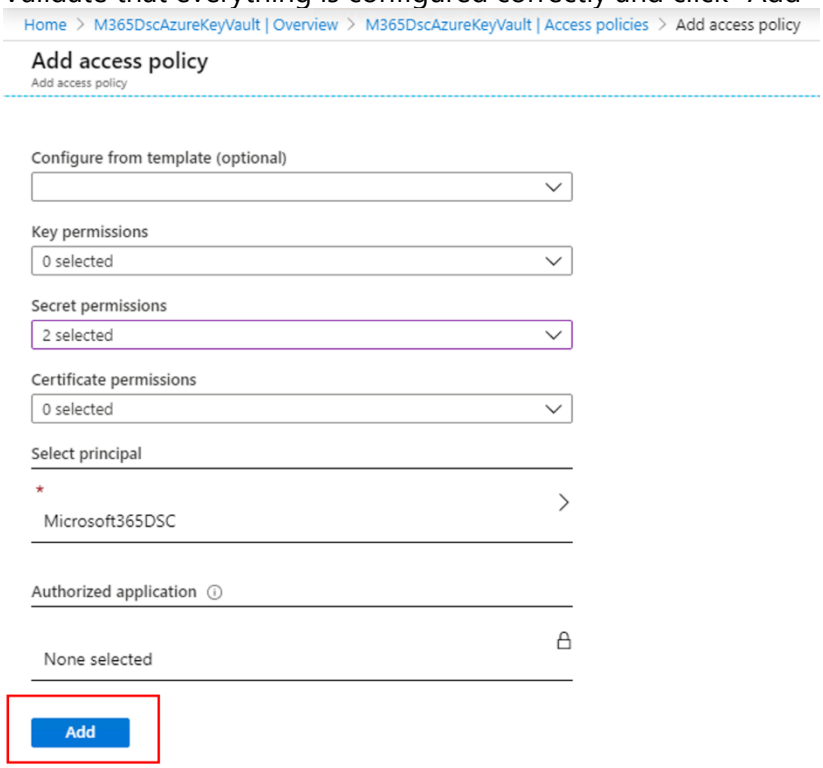
- Select the "Select principal" option, enter the Service Principal Name you created earlier in the search box on the right, select your principal and click "Select".



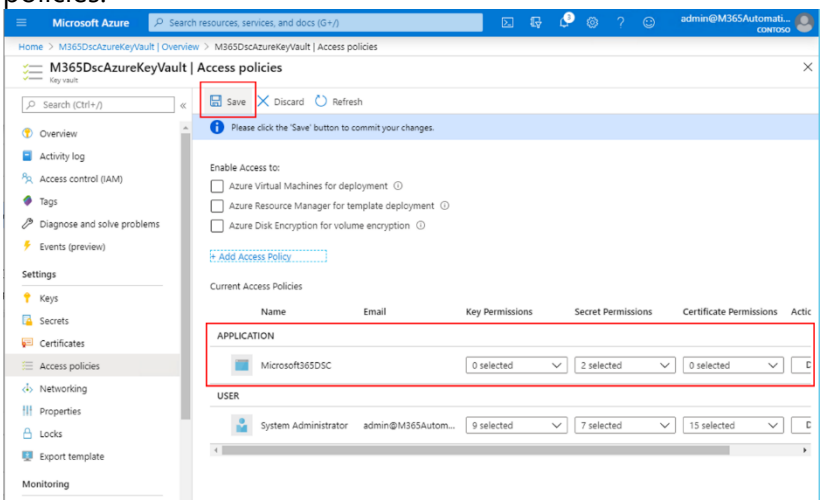
- Open the "Secret permissions" and select the "Get" and "List" permissions



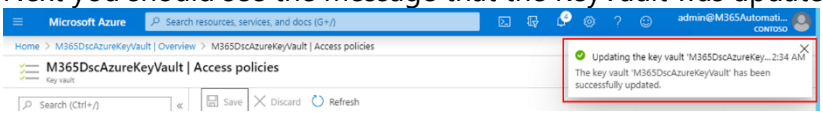
- Validate that everything is configured correctly and click "Add"



- Validate that the access policy has been added and click "Save" to store the new policies.

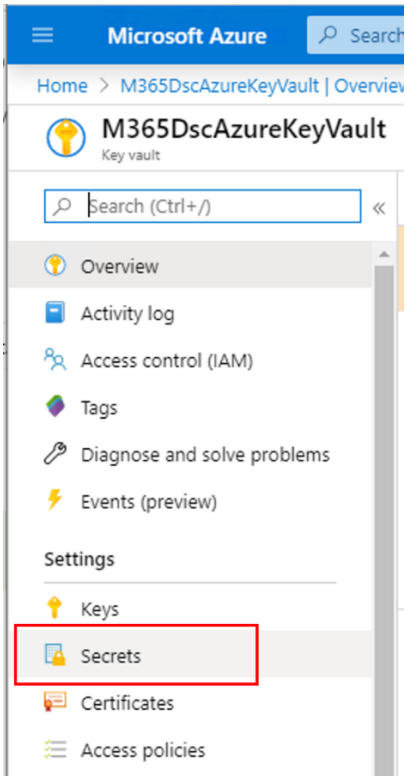


- Next you should see the message that the KeyVault was updated successfully

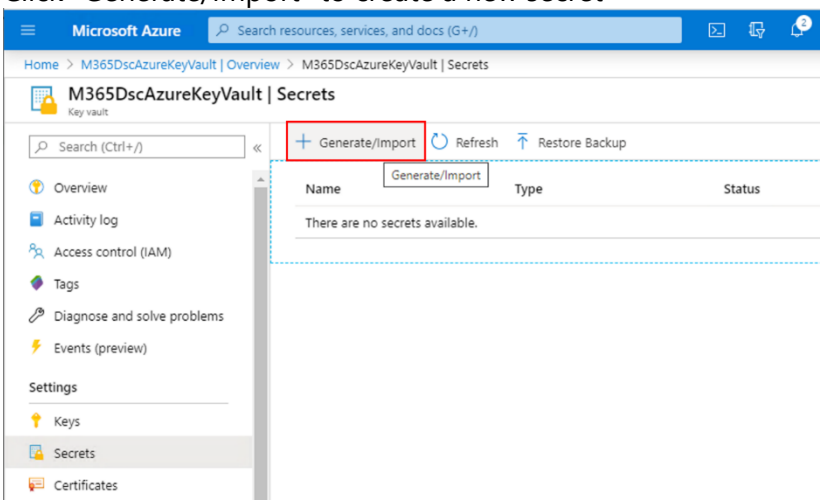


3.6.3 Add secrets to your Vault

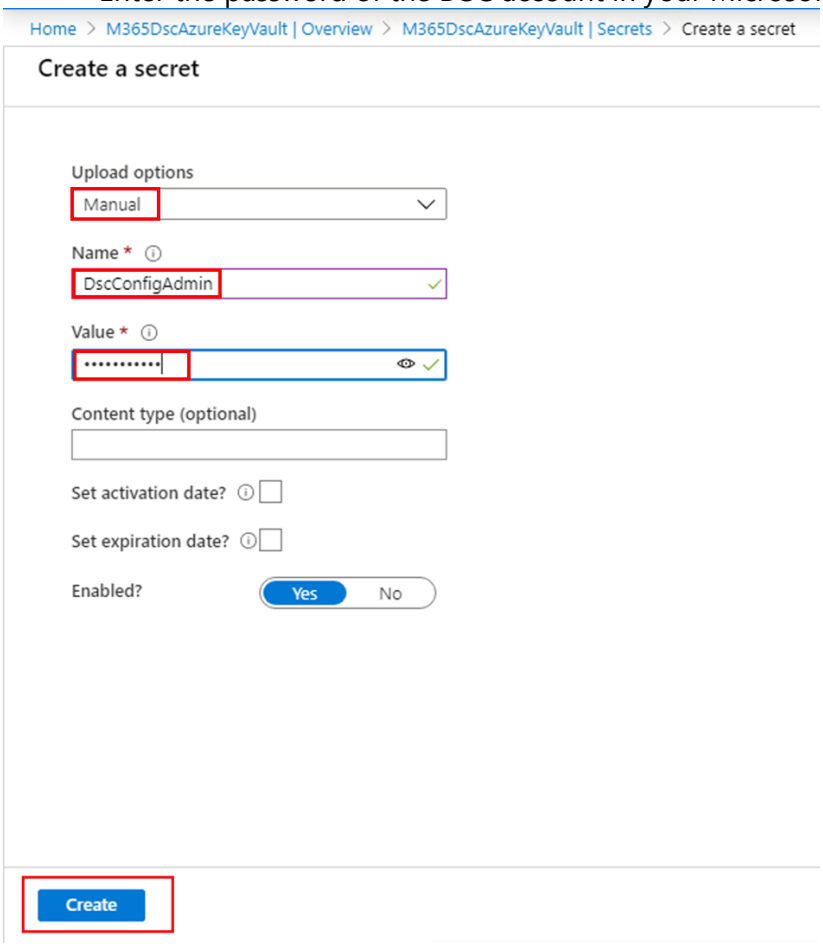
- Click "Secrets" in the left menu



- Click "Generate/Import" to create a new secret



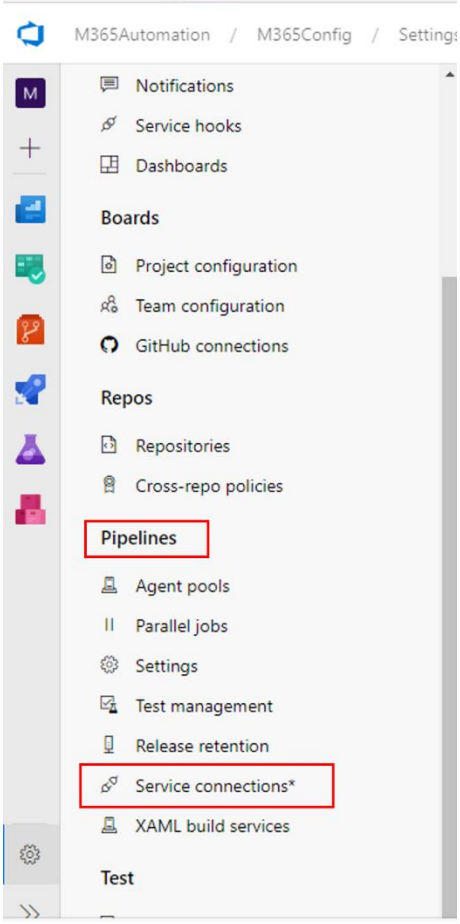
- Enter the correct information and click "Create"
  - Select "Manual" under "Upload options"
  - Use "DscConfigAdmin" as the Name
  - Enter the password of the DSC account in your Microsoft 365 tenant



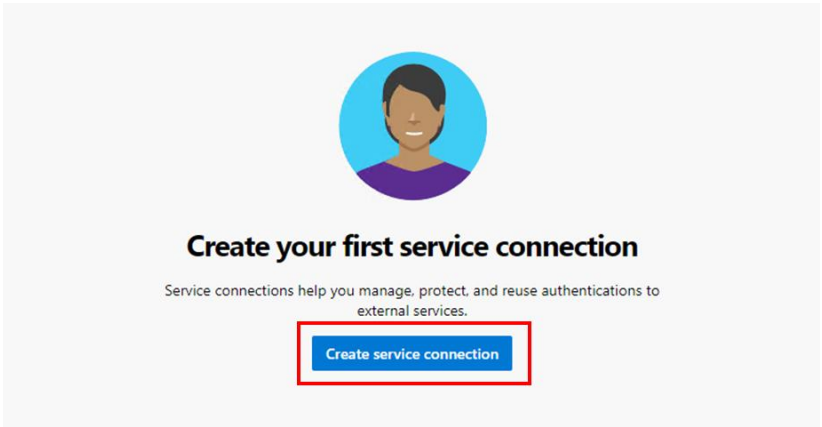
3.6.4 Adding Service Connection to the Azure DevOps project

- Open the Azure DevOps Portal
- Browse to your project

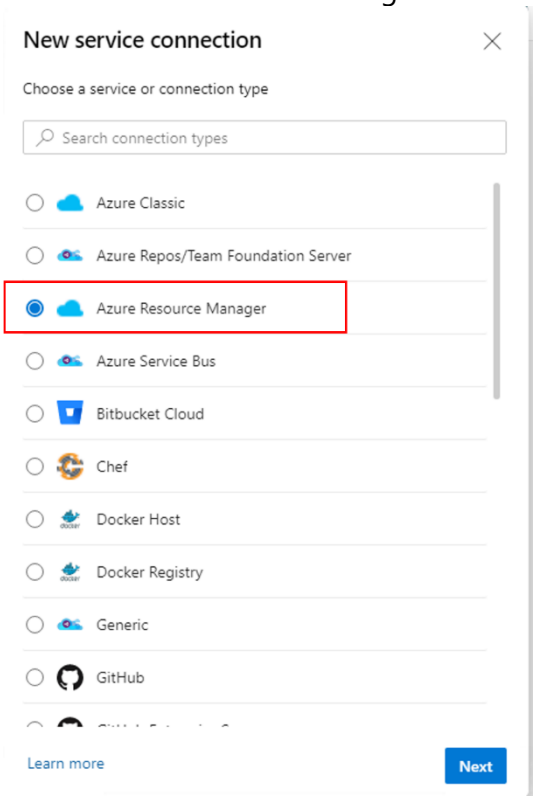
- Click "Project Settings" in the lower left corner
- Scroll to the "Pipelines" section and select "Service connections*"



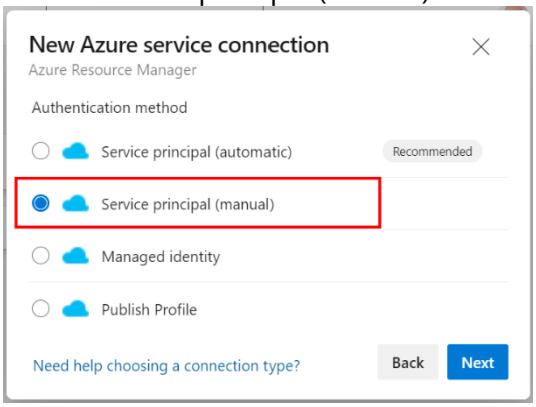
- Click "Create service connection"



- Select "Azure Resource Manager" and click “Next”

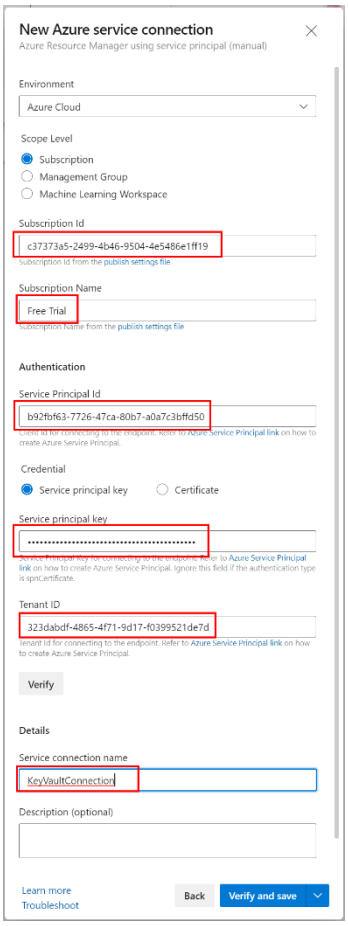


- Select “Service principal (manual)” and click “Next”



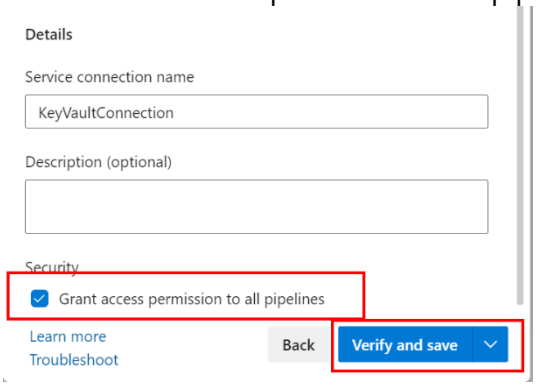
NOTE: We are using the already created Service Principal Name

- Enter the information you stored when creating the Service Principal Name
    a. Enter the GUID of the subscription in which the KeyVault was created as the “Subscription Id”
    b. Enter the name of the subscription in which the KeyVault was created as the “Subscription Name”
    c. Enter the "appId" as the "Service principal client ID"
    d. Enter the "password" as the "Service principal key"
    e. Enter the "tenant" as the "Tenant ID" (potentially already populated)
    f. Enter a "Service connection name"

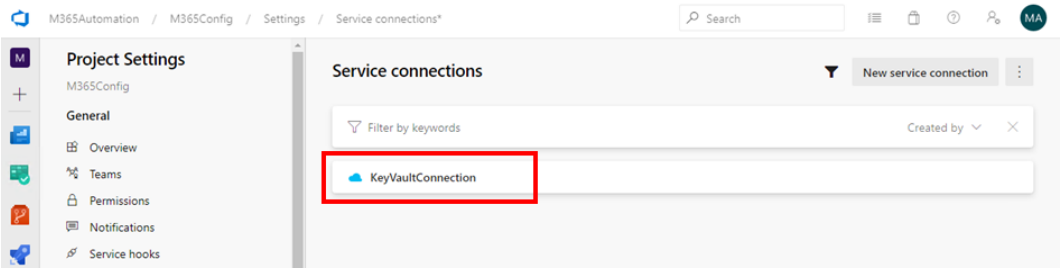


- Click "Verify" to validate the entered information. The status "Verified" (in green) should appear behind the Verify button.

- Select "Grant access permission to all pipelines" and click "Verify and save"



- The "Service connection" is now created and displaying



3.7 Configure the Local Configuration Manager

- Create a Desired State Configuration signing certificate
  - Log onto your virtual machine
  - Open an elevated PowerShell session and run the following command

```
$cert = New-SelfSignedCertificate -Type
DocumentEncryptionCertLegacyCsp -DnsName 'DSCNode Document
Encryption' -HashAlgorithm SHA256 -NotAfter (Get-
Date).AddYears(10)
```

NOTE: This will create a self-signed signing certificate for the Local Configuration Manager to use. You can also use a certificate created via a Certificate Authority.

  - Run the following command and store the value:

```
$cert.Thumbprint
```

- Export public certificate (required for MOF compilation)
  - Export the public certificate to a CER file by running the following command:

```
Export-Certificate -Cert $cert -FilePath
C:\DSCCertificate.cer
```

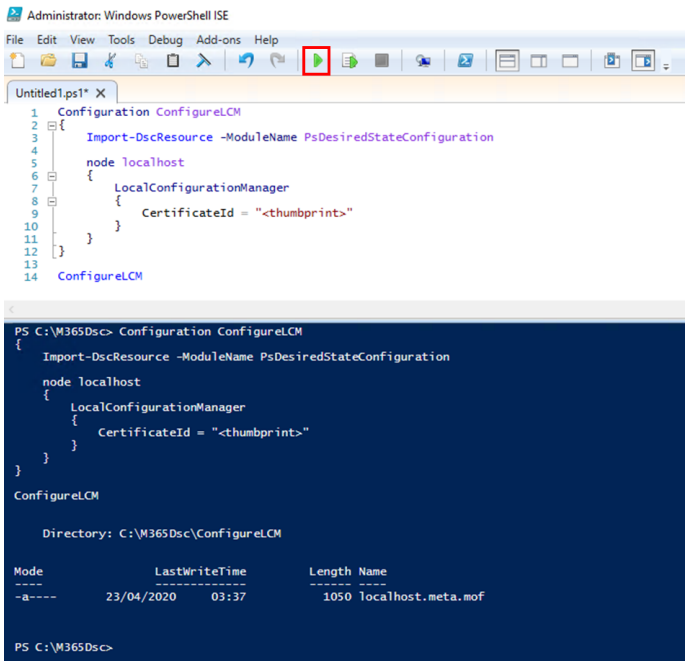
- Configure the certificate Thumbprint (and the mode eventually and/or other settings) in Local Configuration Manager
  - Log onto your virtual machine
  - Open an elevated PowerShell ISE
  - Browse to a folder M365Dsc (create if it not yet exists)
  - Paste the following code:

```
Configuration ConfigureLCM
{
    Import-DscResource -ModuleName
PsDesiredStateConfiguration

    node localhost
    {
        LocalConfigurationManager
        {
            CertificateId = "<thumbprint>"
        }
    }
}

ConfigureLcm
```

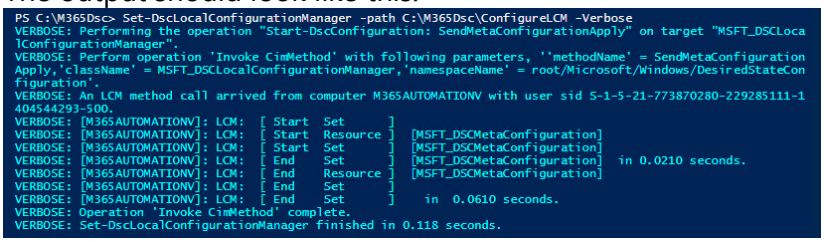
  - Update the "<thumbprint>" with your own certificate thumbprint and run the code (F5)



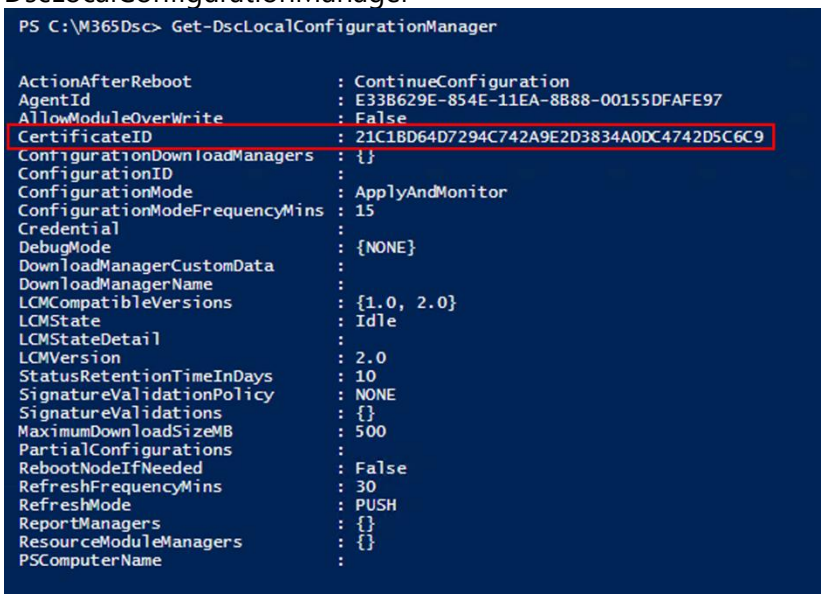
- Run the following command to deploy the Local Configuration Manager config:

```
Set-DscLocalConfigurationManager -Path
C:\M365Dsc\ConfigureLcm -Verbose
```

The output should look like this:



- To validate a successful configuration of the thumbprint, run Get-DscLocalConfigurationManager



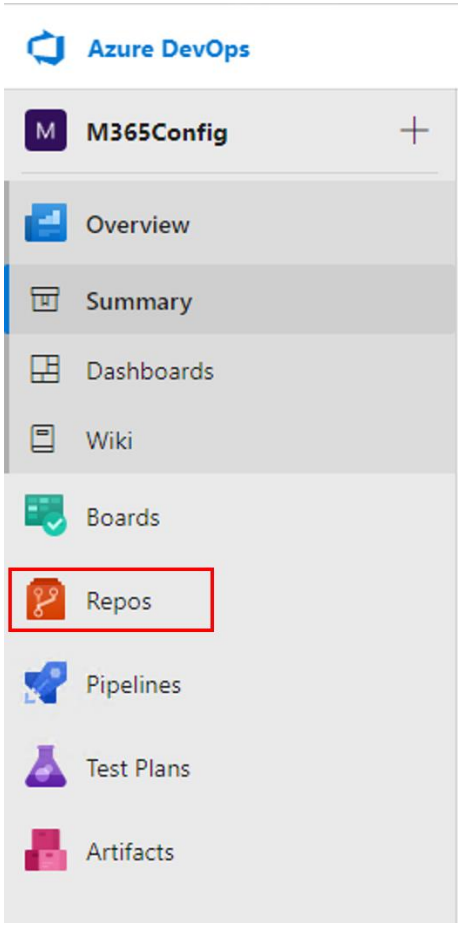
- Optional: Secure your certificate
  - Export the certificate to PFX format
  - Delete the certificate from the certificate store
  - Reimport the certificate from the PFX file and do not select the option to make the private key exportable
  - Import the PFX file into Azure KeyVault for secure backup

4 Configuring Azure DevOps

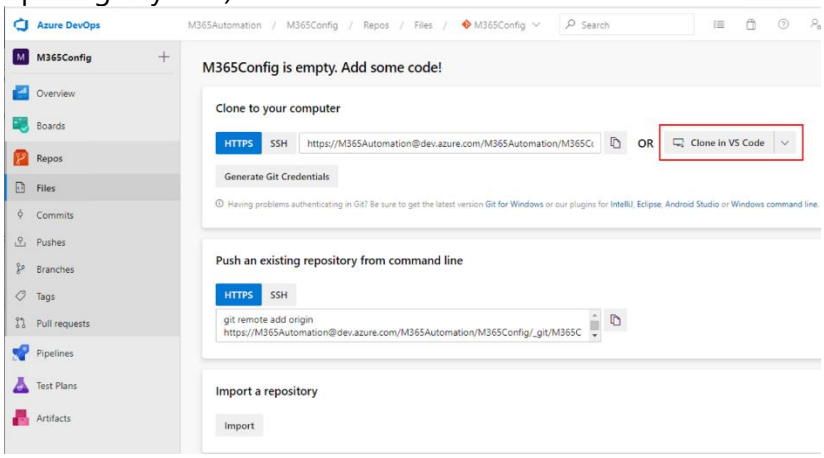
4.1 Populate scripts

- Download and install Visual Studio Code from https://code.visualstudio.com
- Download and install Git from https://git-scm.com
    - Download the most recent version of Git by clicking the "Download" button

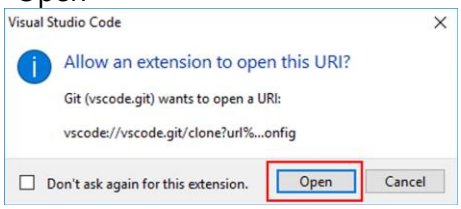
    - Run the downloaded installer and use the default settings
- Download the DSC scripts from https://microsoft365dsc.com/Pages/Resources/Whitepapers/M365Automation.zip
    - This package contains several scripts, check chapter 6 “Script details” for more details.
- Upload the scripts to the DevOps repository
    - Open Azure DevOps Portal and browse to your project
    - Click the "Repos" icon in the left menu


Azure DevOps
M365Config
Overview
Summary
Dashboards
Wiki
Boards
Repos
Pipelines
Test Plans
Artifacts

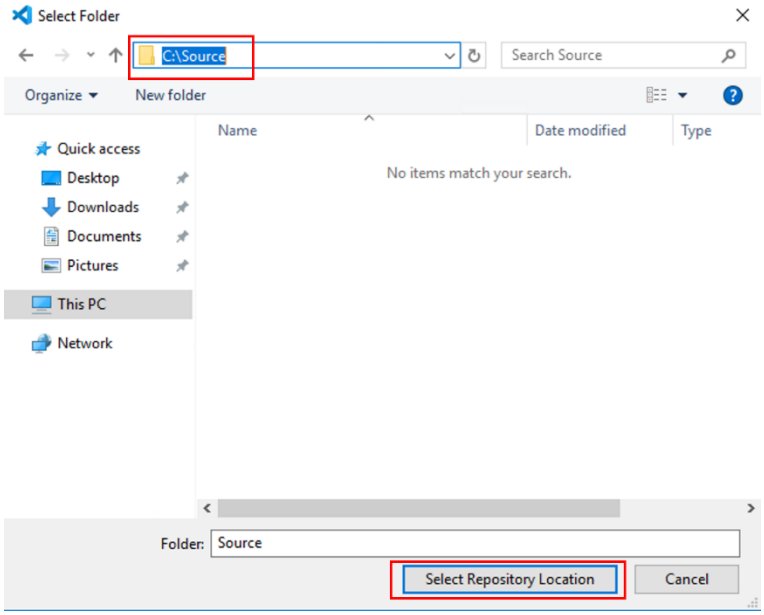
- Click on "Clone in VS Code" (acknowledge any browser notifications for opening any files)



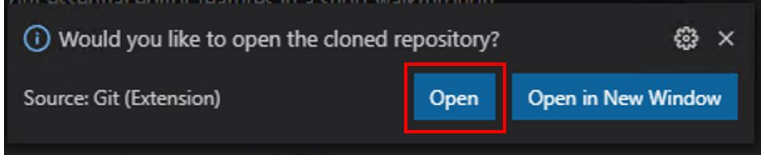
- Acknowledge that Visual Studio Code can open the external URL by clicking "Open"



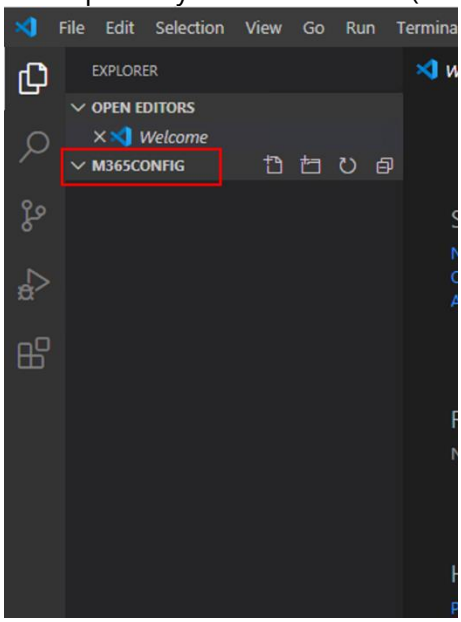
- Select "C:\Source" as the source folder (create if does not exist) and select "Select Repository Location"



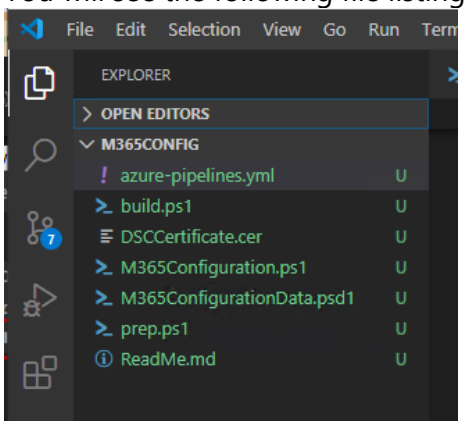
- Login with your Microsoft 365 admin account
- Click "Open" in the bottom right corner to open the cloned folder



- The repository is now available (but still empty) in Visual Studio Code



- Open Explorer and browse to your C:\Source\<project> folder
- Copy the script files from the script download package to this folder
- Copy the DSCCertificate.cer file which you created in paragraph 3.7 to the folder
- You will see the following file listing:



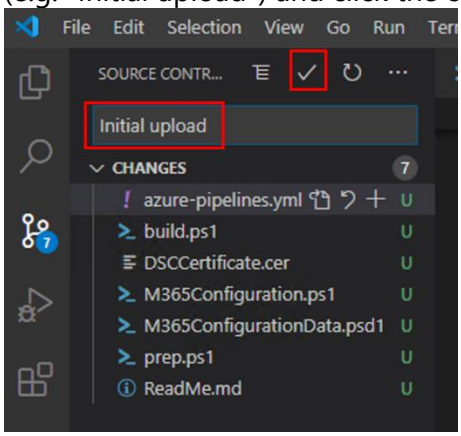
- Open the file "M365ConfigurationData.psd1" and update the domain names used in your environment.

```
NonNodeData = @{
    HRSiteCol = @{
        Url   = 'https://M365Automation.sharepoint.com/sites/hr'
        Owner = 'admin@M365Automation.onmicrosoft.com'
    }
}
```

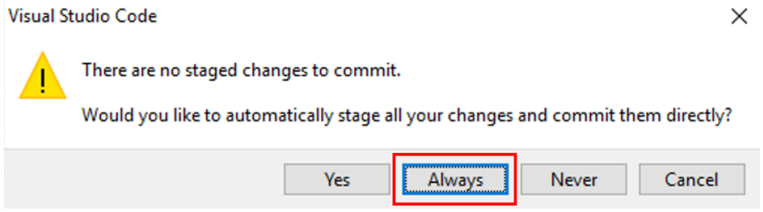
- Open the file "build.ps1" and update account name to the account used in your environment.

```
$password = ConvertTo-SecureString -String $env:SAPW -AsPlainText -Force
$username = "admin@M365Automation.onmicrosoft.com"
$cred = New-Object System.Management.Automation.PSCredential($username, $password)
```

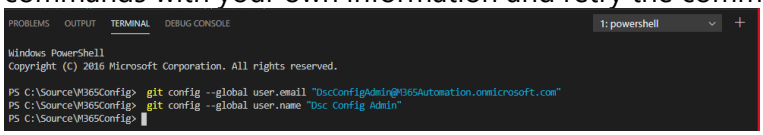
- Click on the Git Source Control icon in the left menu, type a commit message (e.g. "Initial upload") and click the checkmark icon



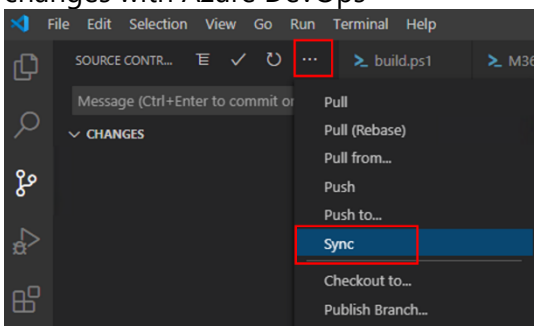
- Click "Always" if you get the message that there are no staged changes to commit.



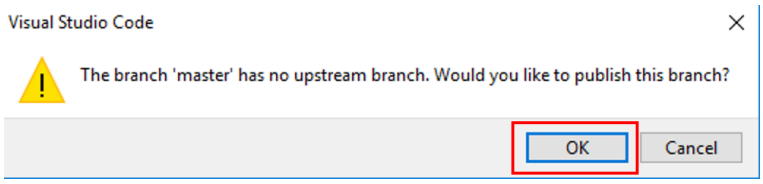
- If you get an error about an unknown e-mail address, run the following commands with your own information and retry the commit:



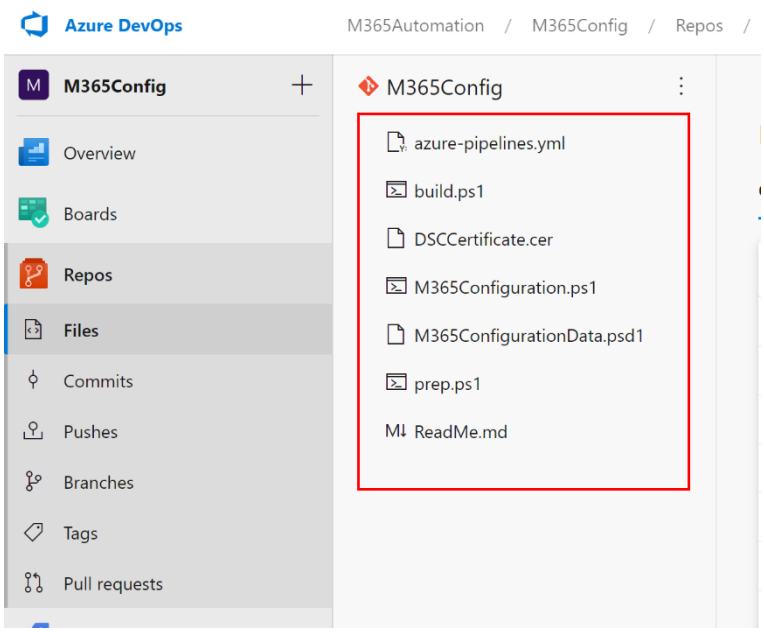
- Click the three dots icon and select "Push, Pull > Sync" to sync your local changes with Azure DevOps



- You might get the below message when running the Sync. Click "OK" to publish the branch to DevOps



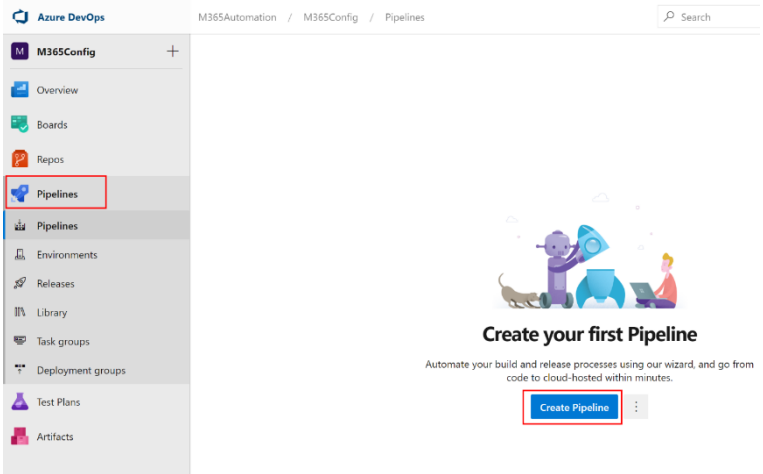
- Validate a successful sync by opening the Azure DevOps Portal, browsing to Repos and validating that all files have been uploaded



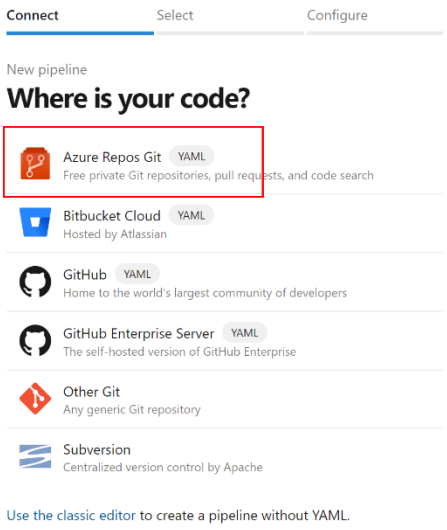
4.2 Configure Azure DevOps project

4.2.1 Create Build pipeline

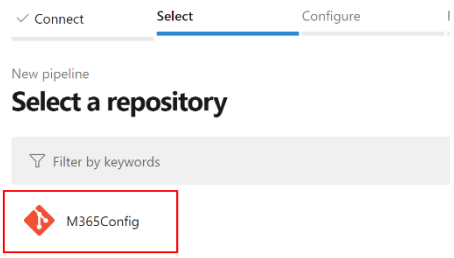
- Go to the Azure DevOps Portal
- Click "Pipeline" and click "Create Pipeline"



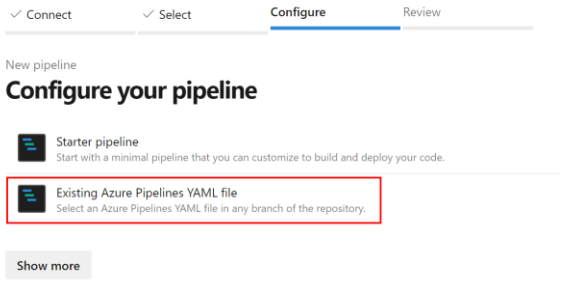
- Select "Azure Repos Git"



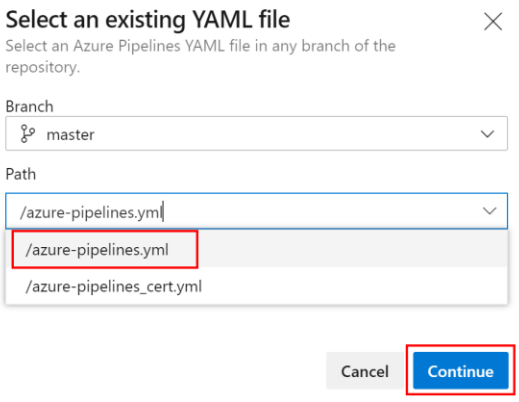
- Click the name of your Project



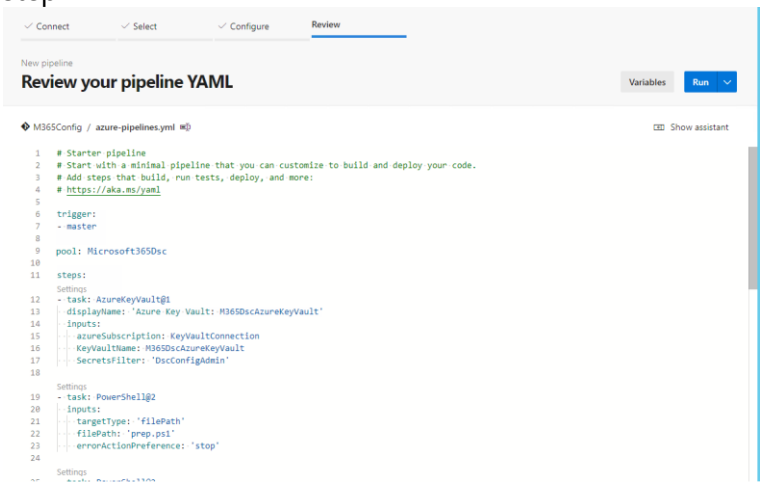
- Select the "Existing Azure Pipelines YAML file"



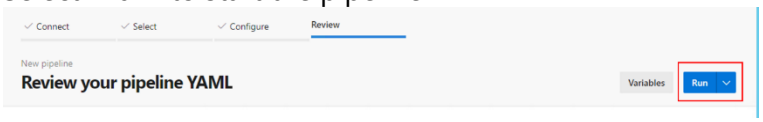
- Select the file "azure-pipelines.yml" and click "Continue"



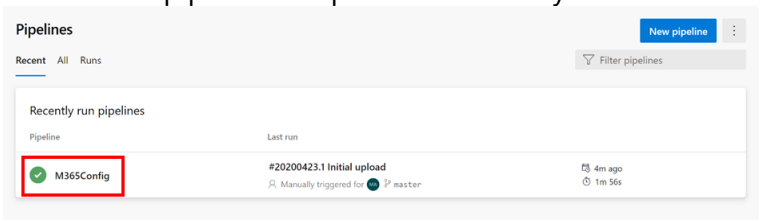
- The pipeline then shows you the azure-pipelines.yml file you uploaded in a previous step



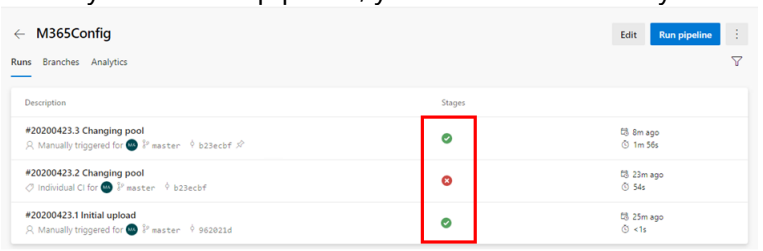
- Select "Run" to start the pipeline



- Check if the pipeline completes successfully



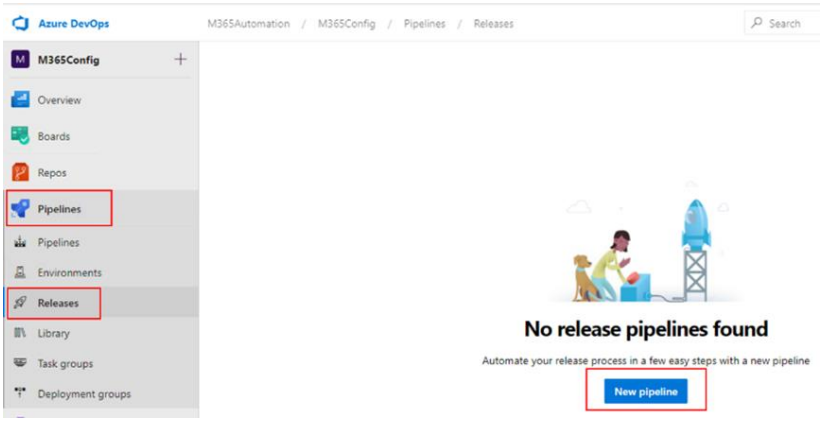
- When you click the pipeline, you can see the history of all runs



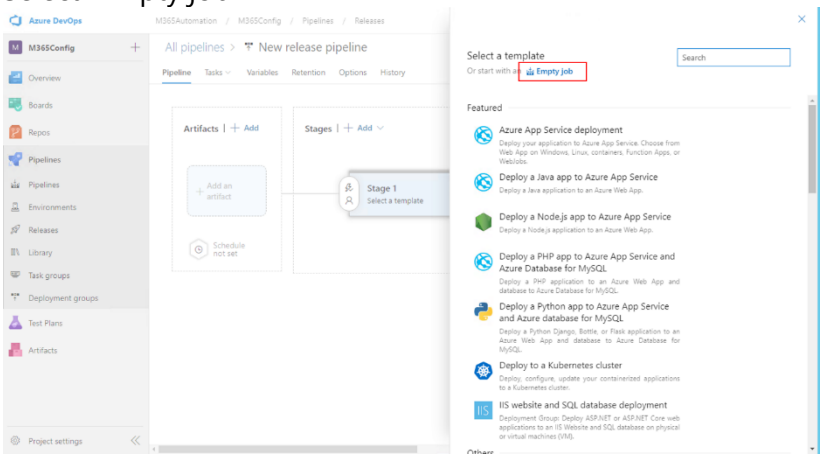
- When you click a run, you can see the logging and other details.

4.2.2 Create Release pipeline

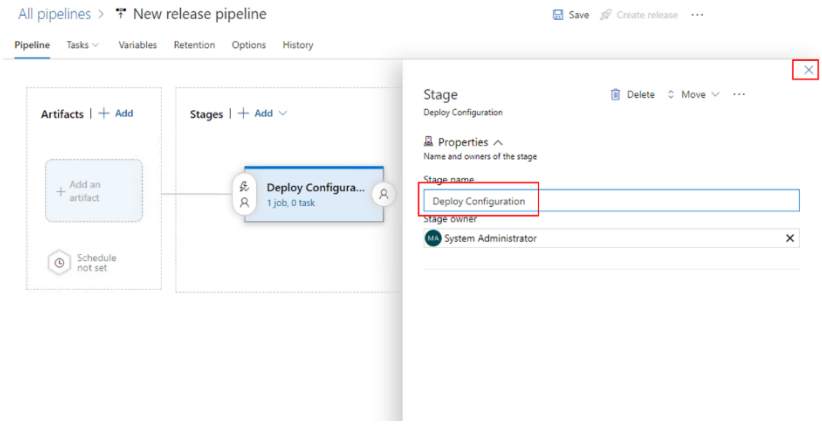
- Go to the Azure DevOps Portal
- Click "Pipelines", click "Releases" and then click "New pipeline"



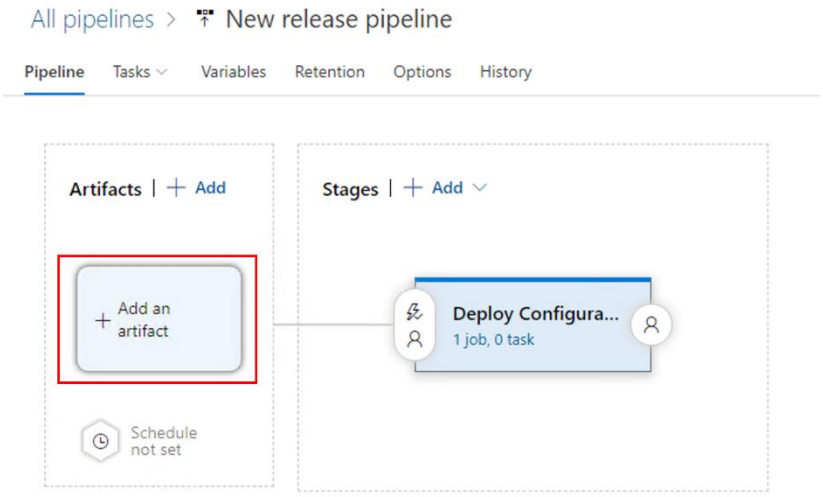
- Select "Empty job"



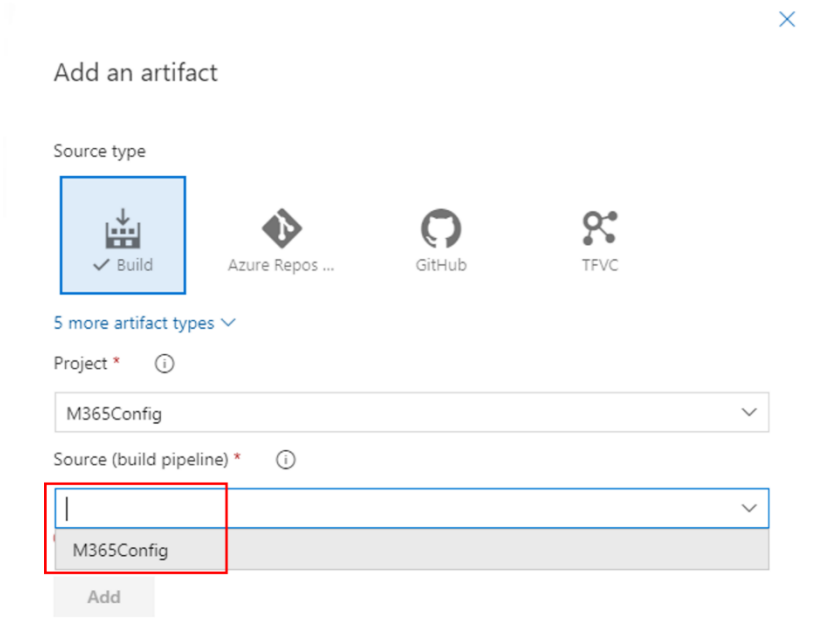
- Give the Stage a name and close the pane



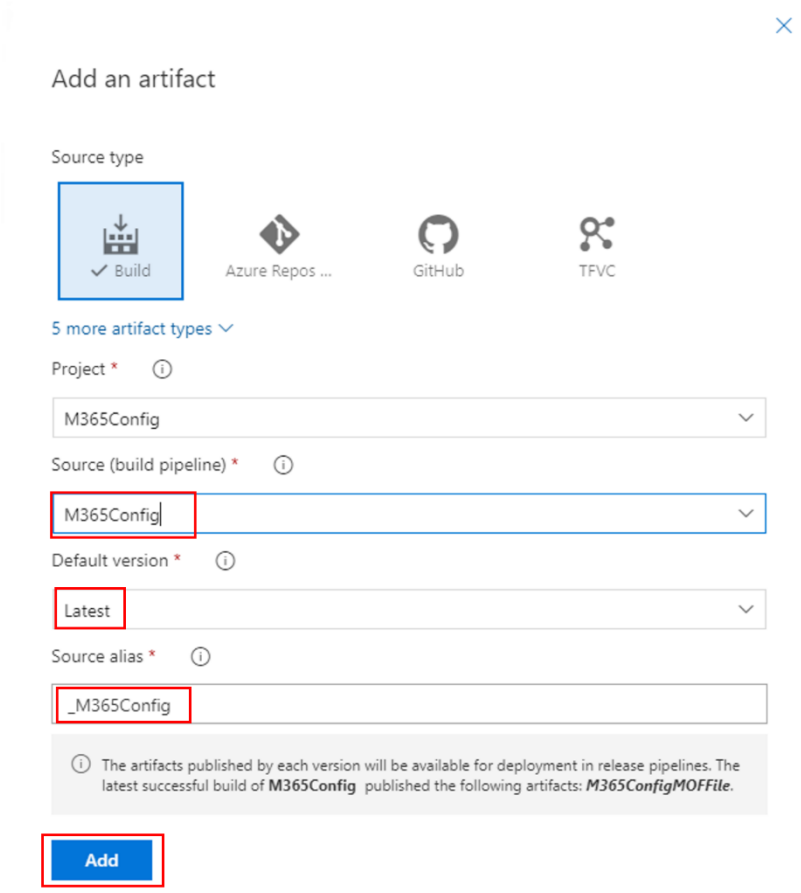
- Click "Add an artifact"



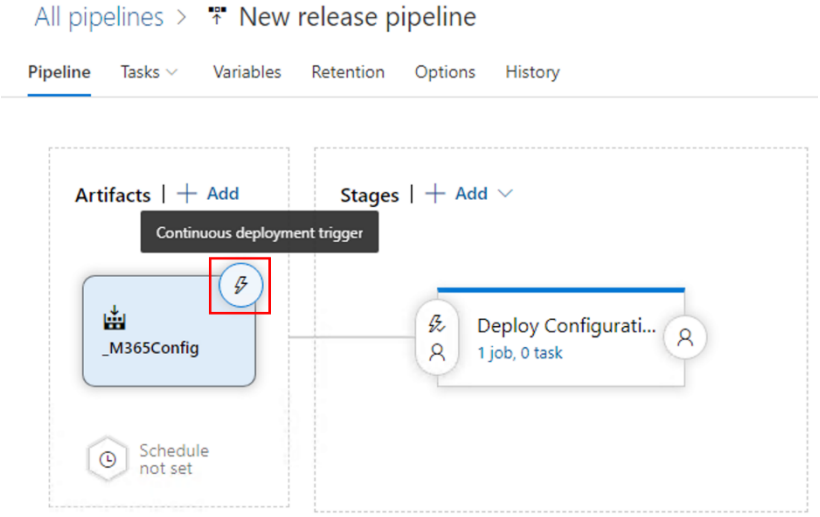
- Under "Source" select the build pipeline



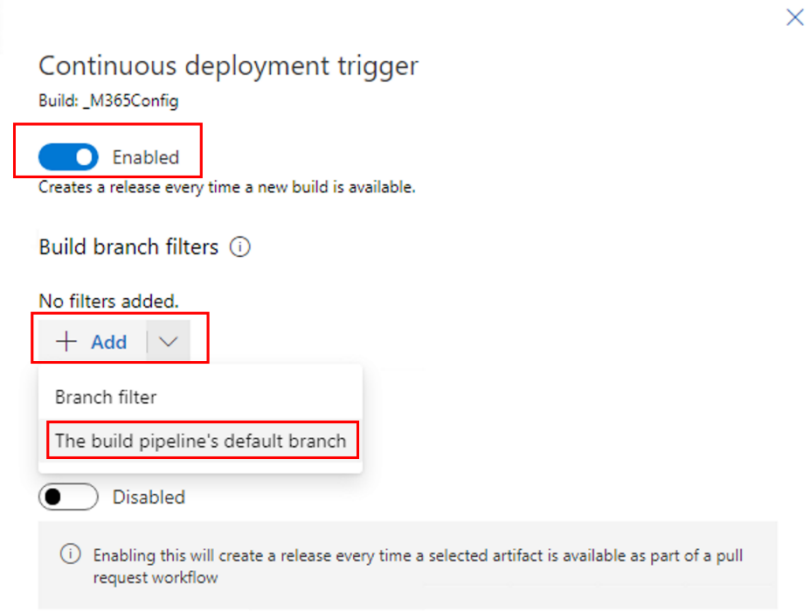
- After selecting the Source, more options will appear. Leave them default and click "Add".
  NOTE: Notice the "Source alias" value. We need this value in a next step.



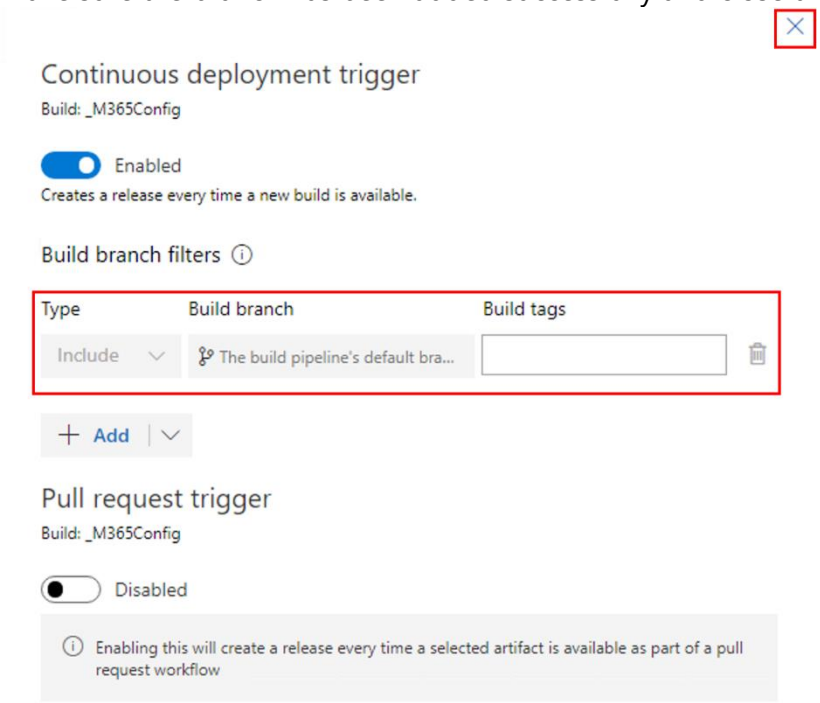
- Configure the Release pipeline triggers by clicking the Lightning icon next to Artifacts



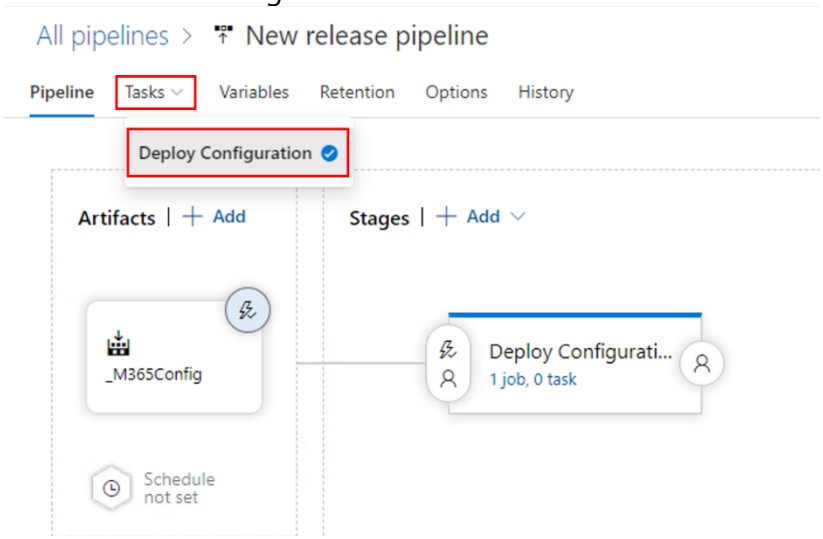
- Enable the "Continuous deployment trigger", under "Build branch filters" click "Add" and select "The build pipeline's default branch"



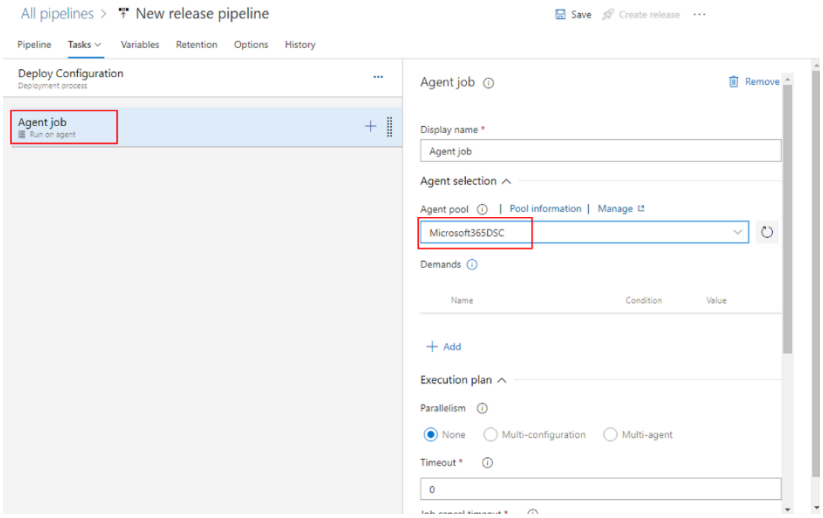
- Make sure the branch has been added successfully and close the pane



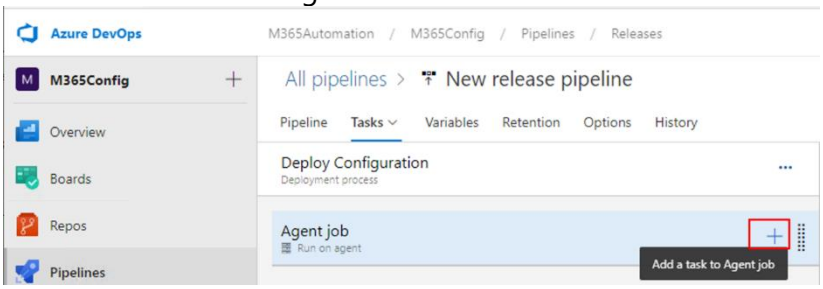
- Select "Tasks > <Stage name>"



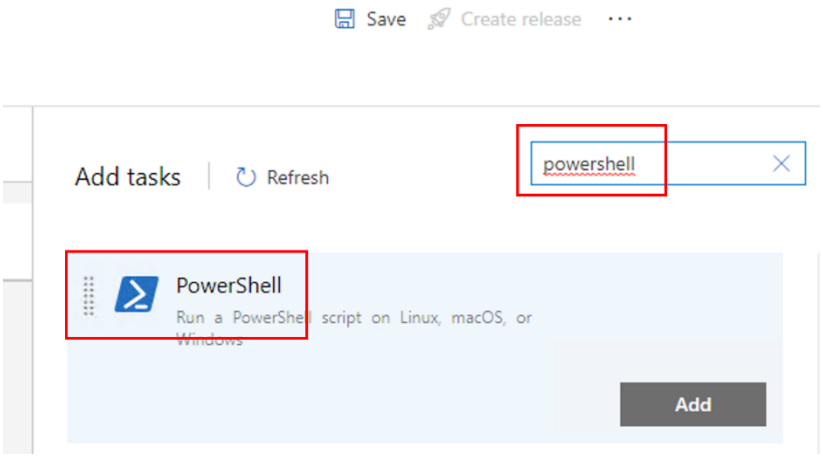
- Select the task "Agent job" in the left part of the pane and change the "Agent pool" to "Microsoft365Dsc". Leave the rest default.



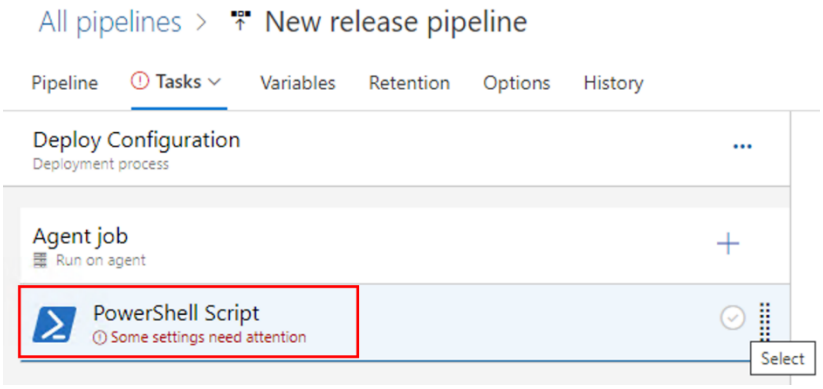
- Click the "+" behind "Agent Job"



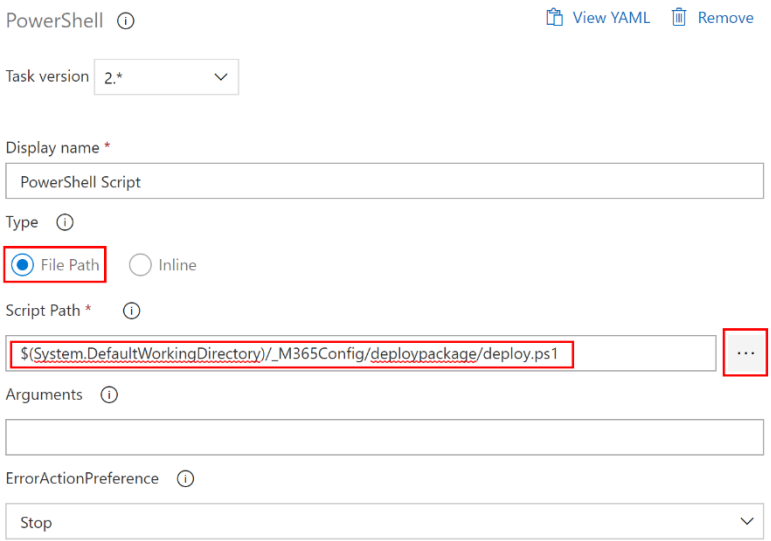
- Search for "PowerShell" and select "PowerShell".
  Note: Do not select the "PowerShell on Target Machines" task



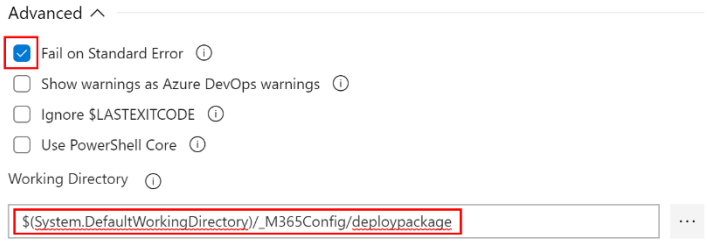
- Select the "PowerShell" task



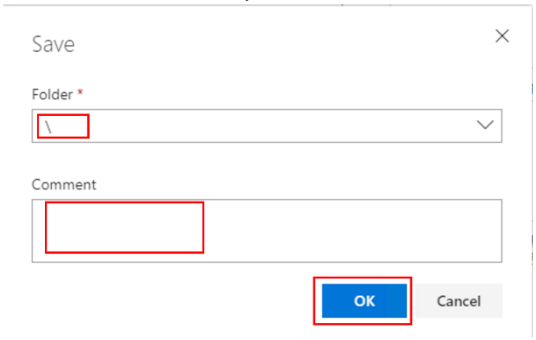
- Select "File Path" as "Type" and browse to the deploy.ps1 file by clicking the "..." button



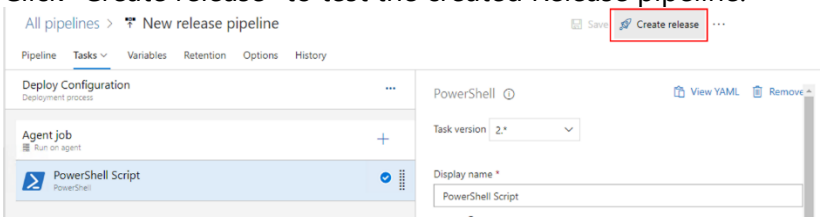
- Open the “Advanced” section and select “Fail on Standard Error”



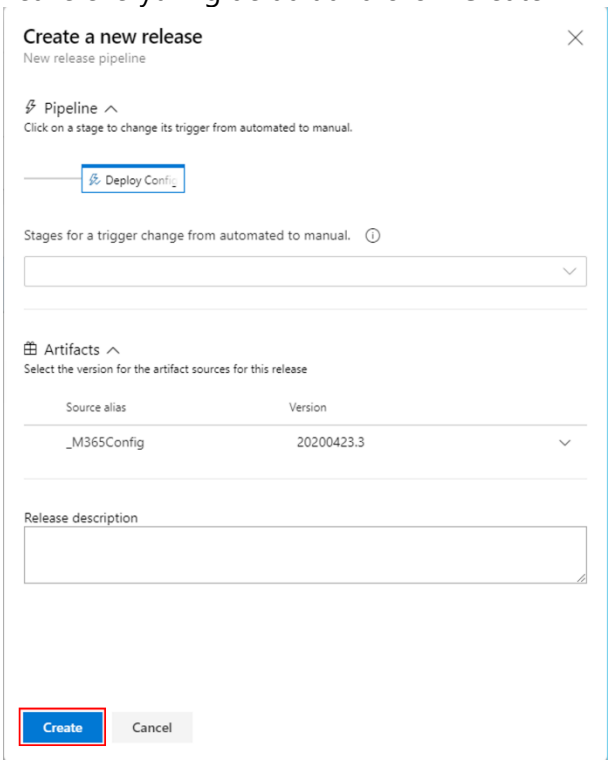
- Click "Save". Use "\" as the folder and add a comment if you prefer. Click "OK"



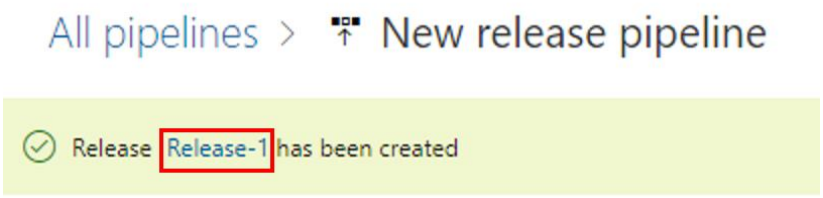
- Click "Create release" to test the created Release pipeline.



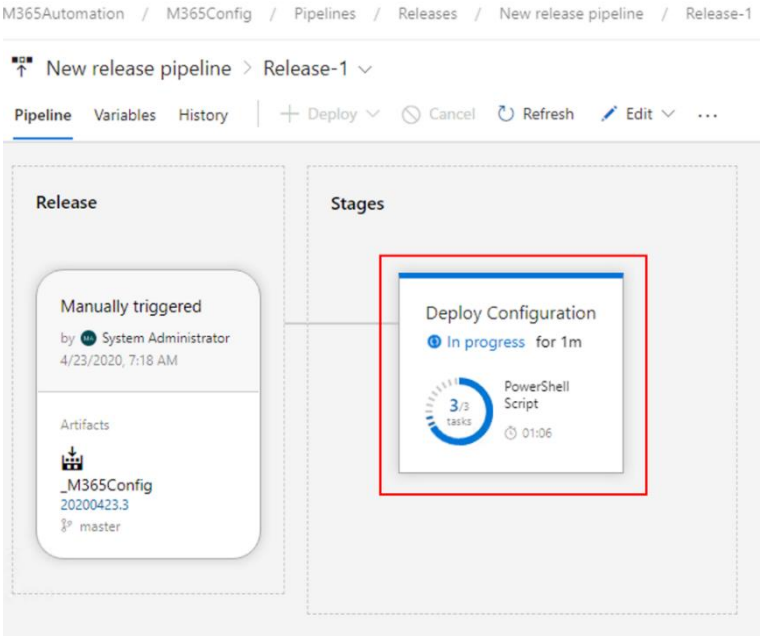
- Leave everything default and click "Create"



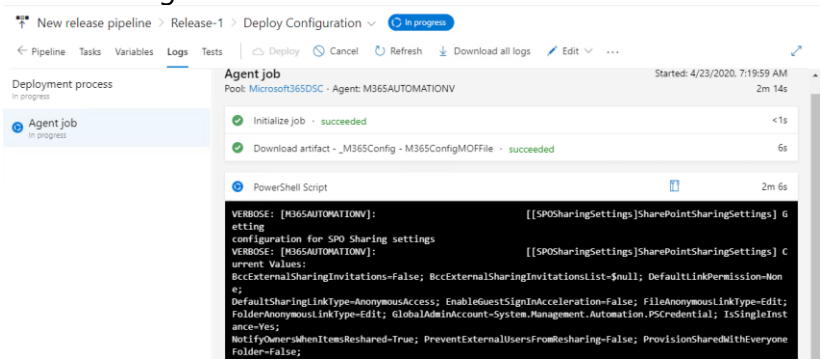
- Click "Release-<nr>" in the top bar to open the release and review its progress



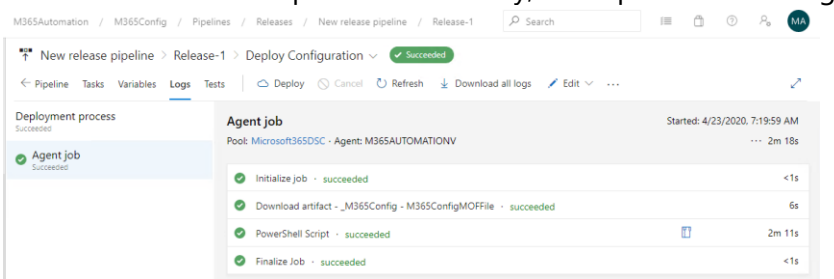
- Review the progress



- Click the stage for more details

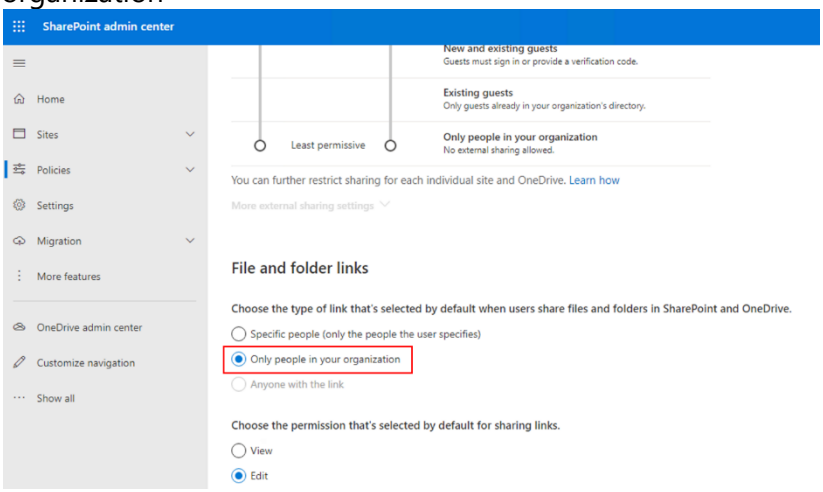


- When the release completes successfully, all steps should have green check marks.

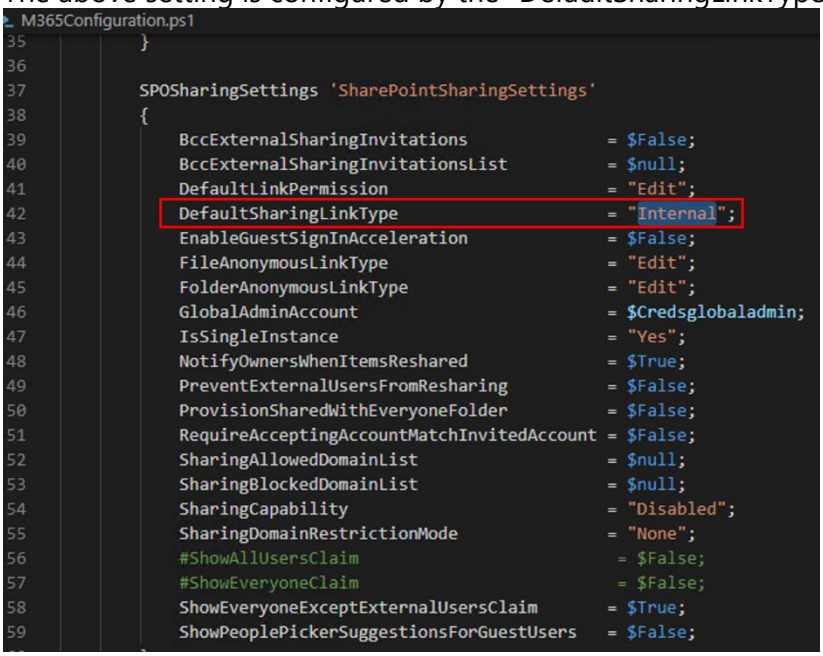


4.2.3 Validate that changes to the config are deployed successfully

- Make sure the following setting is configured:
  SharePoint Admin Center > Policies > File and folder links > Only people in your organization



- The above setting is configured by the "DefaultSharingLinkType" DSC setting:

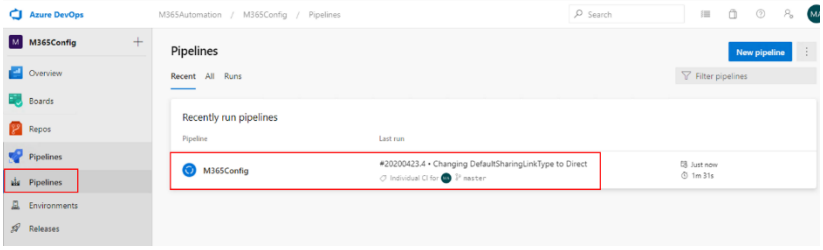


- Change this setting from "Internal" to "Direct"

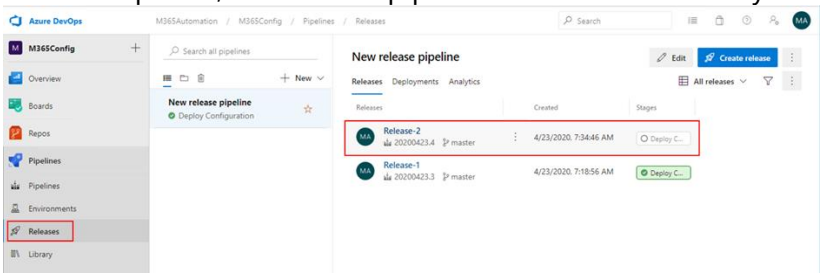
```
M365Configuration.ps1
35      }
36
37      SPOSharingSettings 'SharePointSharingSettings'
38      {
39          BccExternalSharingInvitations             = $False;
40          BccExternalSharingInvitationsList         = $null;
41          DefaultLinkPermission                     = "Edit";
42          DefaultSharingLinkType                    = "Direct";
43          EnableGuestSignInAcceleration             = $False;
44          FileAnonymousLinkType                     = "Edit";
45          FolderAnonymousLinkType                   = "Edit";
46          GlobalAdminAccount                        = $Credsglobaladmin;
47          IsSingleInstance                          = "Yes";
48          NotifyOwnersWhenItemsReshared             = $True;
49          PreventExternalUsersFromResharing         = $False;
50          ProvisionSharedWithEveryoneFolder         = $False;
51          RequireAcceptingAccountMatchInvitedAccount = $False;
52          SharingAllowedDomainList                  = $null;
53          SharingBlockedDomainList                  = $null;
54          SharingCapability                         = "Disabled";
55          SharingDomainRestrictionMode              = "None";
56          #ShowAllUsersClaim                         = $False;
57          #ShowEveryoneClaim                         = $False;
58          ShowEveryoneExceptExternalUsersClaim      = $True;
59          ShowPeoplePickerSuggestionsForGuestUsers  = $False;
60      }
```

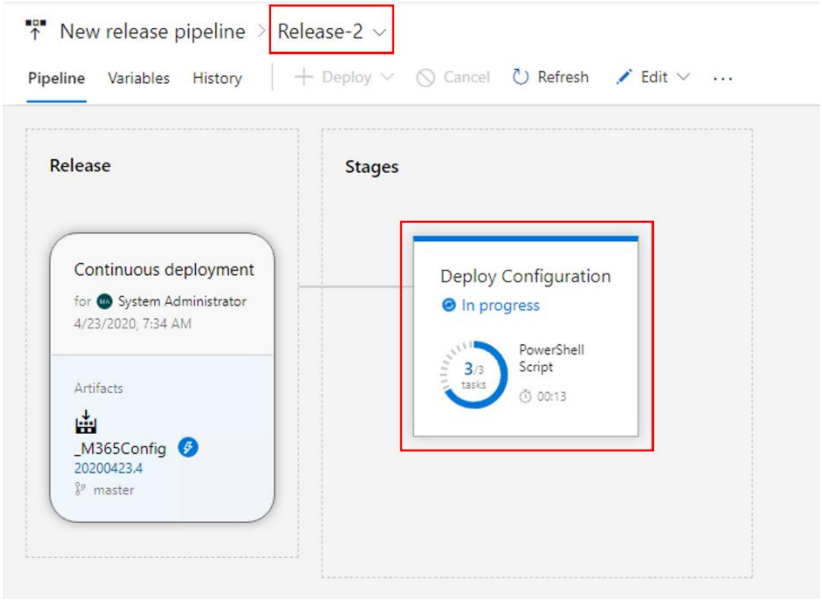
- Save the file, go to the Git Source Control section, enter a commit description, commit the change and sync the repository with Azure DevOps

- Open the Build Pipeline, which should have started

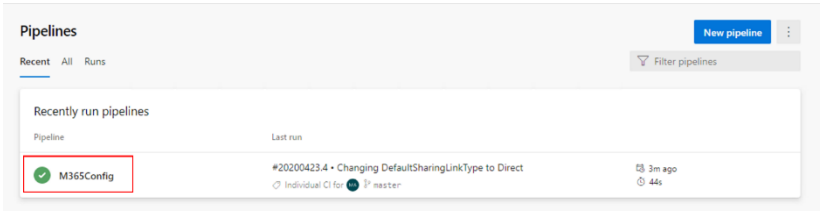


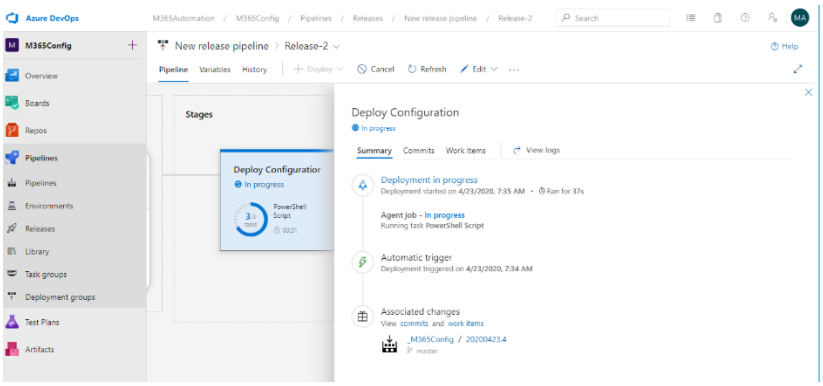
- Once completed, the Release pipeline should automatically start

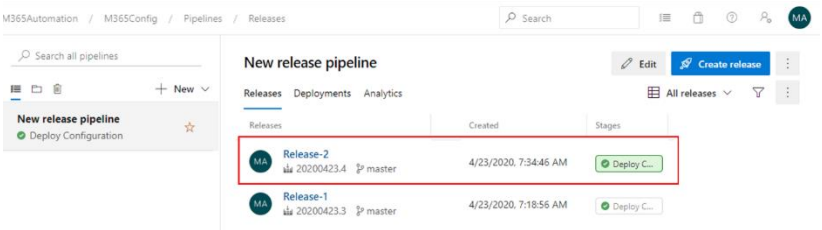


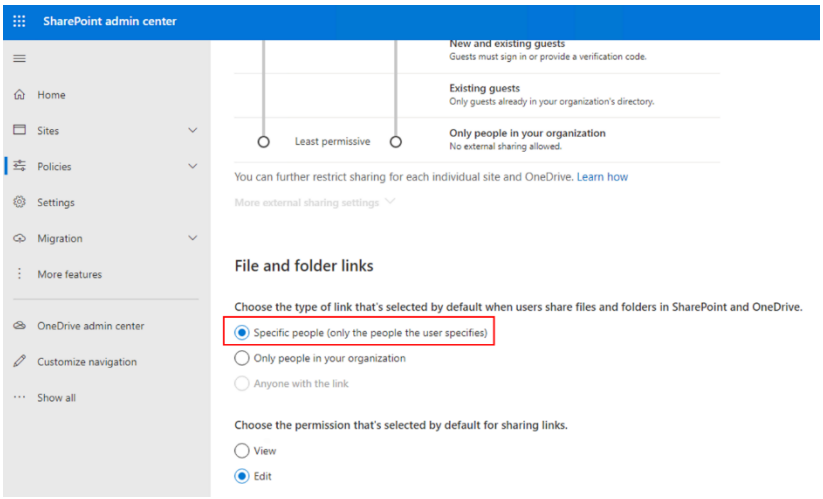


- When the Release pipeline completes, the setting should have been changed in the SharePoint Admin Portal








SharePoint admin center
Home
Sites
Policies
Settings
Migration
More features
OneDrive admin center
Customize navigation
Show all
New and existing guests
Guests must sign in or provide a verification code.
Existing guests
Only guests already in your organization's directory.
Least permissive
Only people in your organization
No external sharing allowed.
You can further restrict sharing for each individual site and OneDrive. Learn how
More external sharing settings
File and folder links
Choose the type of link that's selected by default when users share files and folders in SharePoint and OneDrive.
Specific people (only the people the user specifies)
Only people in your organization
Anyone with the link
Choose the permission that's selected by default for sharing links.
View
Edit

5 Security Enhancements

5.1 Using Azure Conditional Access to secure service account

Azure Conditional Access[1] can be used to prevent the created service account login into Microsoft 365, except when coming from a specified location / IP address. This section describes the steps to implement this restriction.

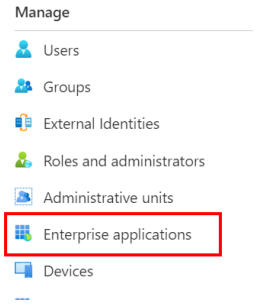
Requirements:

- All VMs have a fixed IP address configured
- List of the IP address of all the VMs
- Name of DSC service account created in paragraph 3.1, e.g., "DscConfigAdmin"

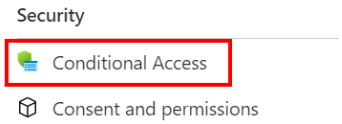
Steps

- Open the Azure Portal (https://portal.azure.com)
- Go to Azure Active Directory

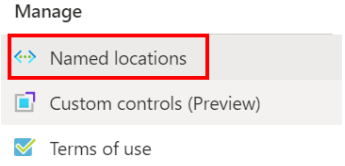
- Under "Manage", click "Enterprise applications"



- Under "Security", click "Conditional Access"

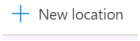


- First, we are going to create a Named Location
- Under "Manage", click "Named locations"

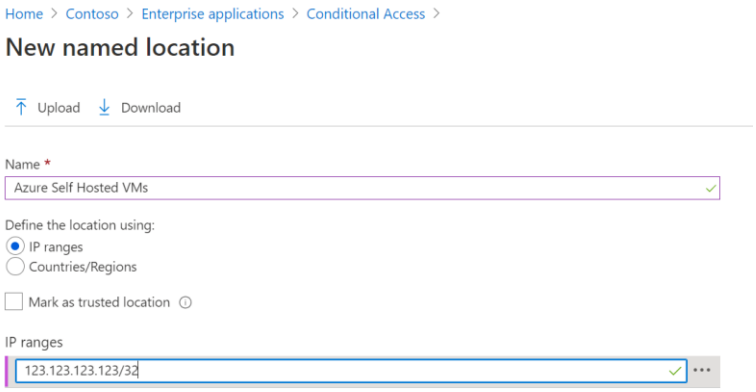


[1] Azure AD Premium P1 license required

- Click “New location”

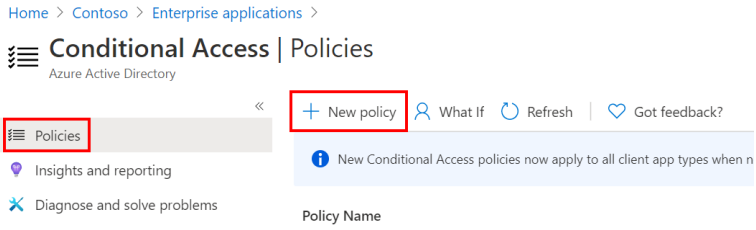


- Enter the required information:
  - Name: “Azure Self Hosted VMs” (or any other name you want to use)
  - Define the location using: “IP Ranges”
  - IP ranges: The public IP address of the VM in the “123.123.123.123/32” format

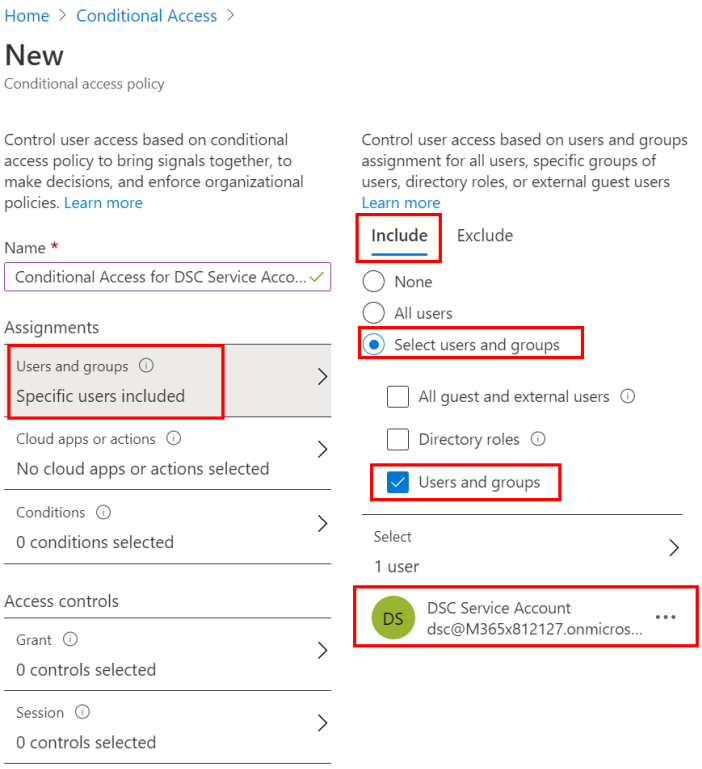


- Click “Create” to create the Named location

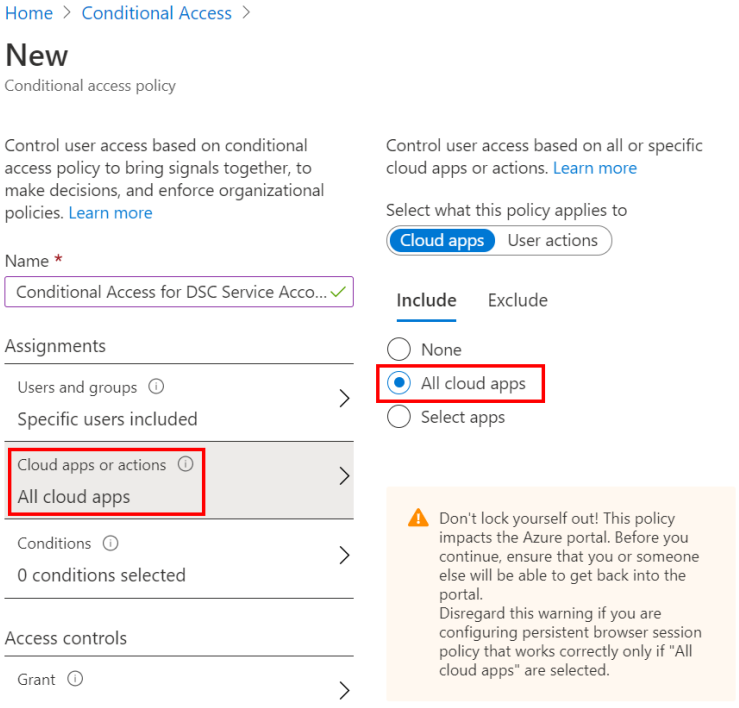
- Next, select “Policies” and click “New policy”



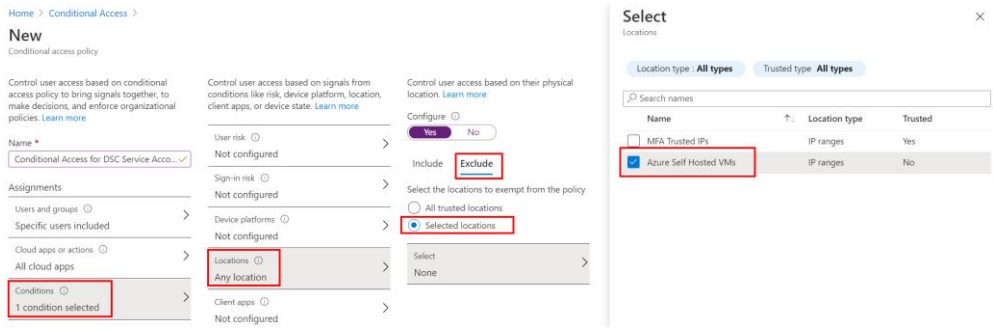
- Create a new policy, using the following settings:
  - Name: “Conditional Access for DSC Service Account” (or the name you would like to use)
  - Users and groups > Include
    - Select “Selected users and groups”
    - Check “Users and groups”
    - Search and select the DSC Service Account



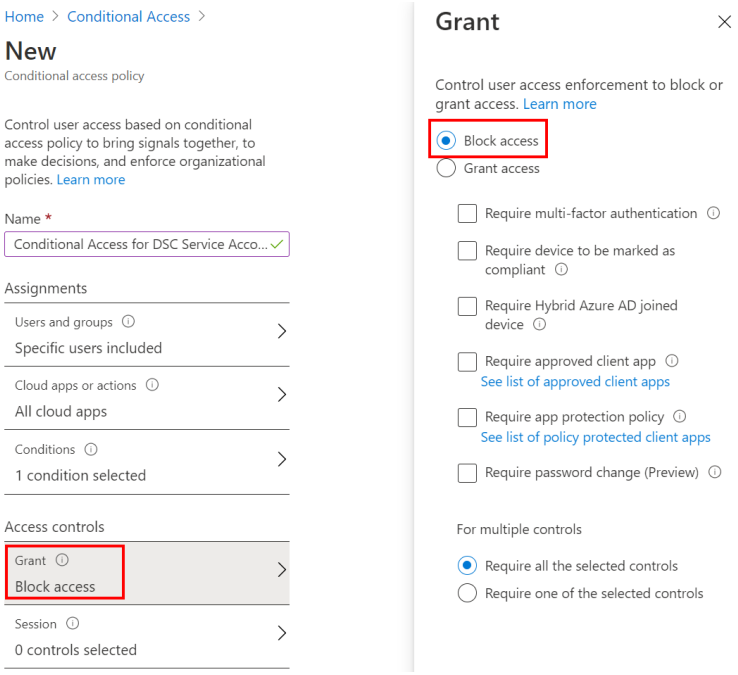
- Cloud apps or actions: Select “All cloud apps”



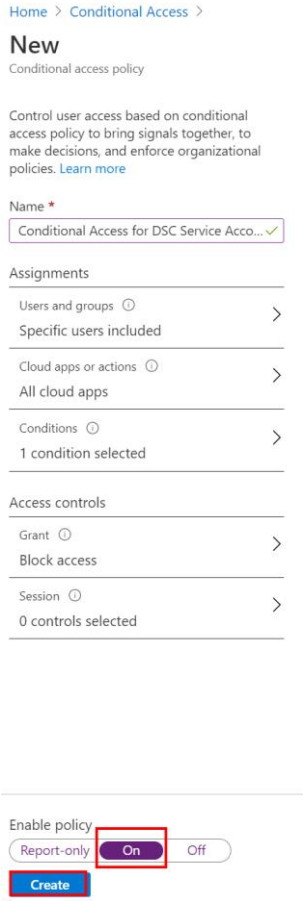
- Conditions > Locations
  - Include: “Any location”
  - Exclude: Select “Selected locations” and select the newly created Named location “Azure Self Hosted VMs”



- Access controls > Grant
  - Select “Block access”



- Under "Enable policy", select "On" to activate the policy and click "Create"



- The DSC service account can now only be used to authenticate from the Azure DevOps Self Hosted VMs

5.2 Using Certificates instead of Username/Password for authentication

The Microsoft 365 PowerShell modules, used by Microsoft365DSC, are supporting several authentication methods:

- Username / Password
- Certificate stored in the computers certificate store / Thumbprint of certificate
- Certificate stored in PFX file / Password of the file
- Application secret
    - Only supported by a few modules and therefore not implemented in Microsoft365DSC (yet).

The following table shows the supported authentication method for each Microsoft365DSC workload:

Workload	PowerShell Module	Credential	Certificate Thumbprint	Certificate Path	Application Secret*
AzureAD	AzureADPreview (Connect-AzureAD)	✓	✓	✗	✗
Exchange Online	ExchangeOnlineManagement (Connect-ExchangeOnline)	✓	✓	✓	✗
Intune	Microsoft.Graph.Intune (Connect-MSGraph)	✓	✗	✗	✗
Office 365	AzureADPreview (Connect-AzureAD)	✓	✓	✗	✗
OneDrive	SharePointPnPPowerShellOnline (Connect-PnPOnline)	✓	✓	✓	✗
PowerApps	Microsoft.PowerApps.Administration.PowerShell	✓	✓	✗	✗
Planner	Microsoft.Graph.Authentication (Connect-Graph)	✗	✓	✗	✗
Security & Compliance Center	ExchangeOnlineManagement (Connect-IPPSSession)	✓	✗	✗	✗
Skype for Business Online	MicrosoftTeams (New-CSOnlineSession)	✓	✗	✗	✗
SharePoint Online	SharePointPnPPowerShellOnline (Connect-PnPOnline)	✓	✓	✓	✗
Teams	MicrosoftTeams (New-CSOnlineSession)	✓	✓	✗	✗
	* Application secret is possible with the OneDrive, PowerApps and SharePoint cmdlets, but has not been implemented for Microsoft365DSC yet.				

This section describes the required steps to use certificate with thumbprint authentication instead of a username / password.

To automate the deployment of the certificate to the Azure DevOps agent, we are storing the certificate and the password in Azure KeyVault. In the Release pipeline we are checking if the certificate is present and import it if not.

To implement this scenario, follow the regular process as described in chapter 3 "Preparation" and 4 "Configuring Azure DevOps", but replace paragraphs 4.2.1 and 4.2.2 with the below steps.

5.2.1 Creating the authentication certificate

- Log onto the virtual machine
- Open an elevated PowerShell window
- Create and export a new certificate by running the following PowerShell commands:
  - **NOTE:** Update the [PASSWORD] parameter to your own password

```
$cert = New-SelfSignedCertificate -Subject "CN=Microsoft365DSC"
-CertStoreLocation "Cert:\LocalMachine\My" -KeyExportPolicy
Exportable -KeySpec Signature

$password = ConvertTo-SecureString -String "[PASSWORD]" -
AsPlainText -Force

Export-PfxCertificate -Cert $cert -FilePath
C:\M365ClientCert.pfx -Password $password

Export-Certificate -Cert $cert -FilePath C:\M365ClientCert.cer
```

For more information on creating a certificate for application authentication, see: https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-self-signed-certificate

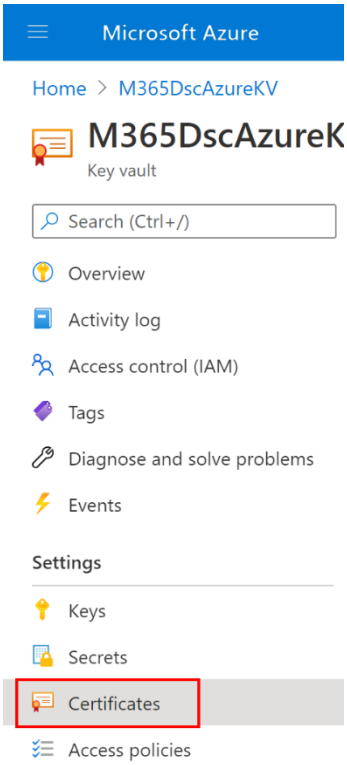
- Run the following command, copy the outputted Thumbprint and store it for later use

```
$cert.Thumbprint
```

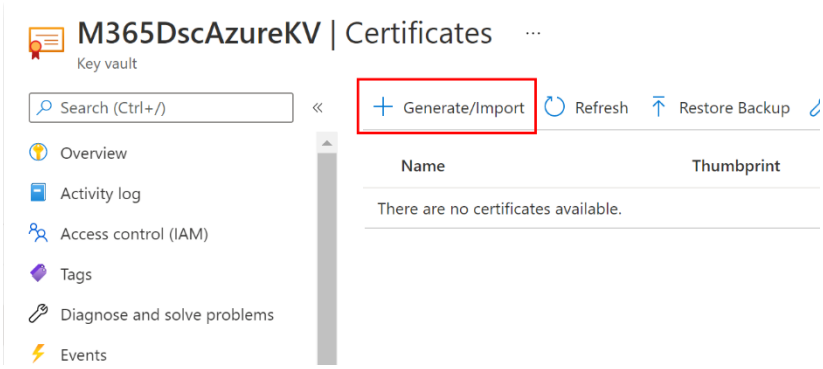
5.2.2 Adding certificate to Azure KeyVault

Instead of paragraph 3.6.3 "Add secrets to your Vault", take the following steps to add the certificate to the KeyVault:

- Click "Certificates" in the left menu



- Click "Generate/Import" to create a new secret



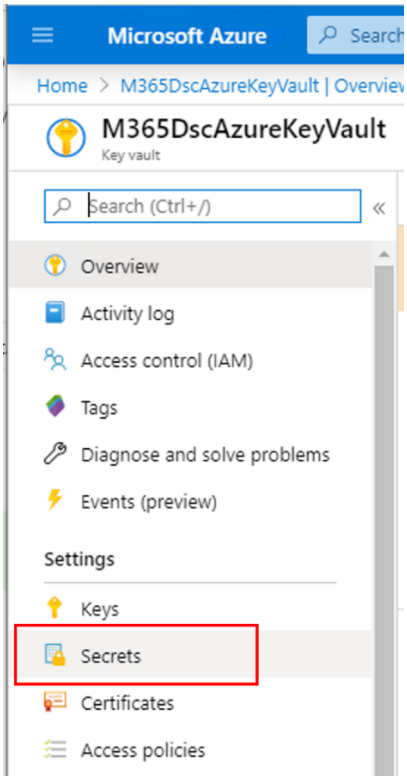
- Enter the correct information and click "Create"
  - Select "Import" under "Method of Certificate Creation"

- Use "M365ClientCert" as the “Certificate Name”
- At “Upload Certificate File” click the file browse icon at the right and select the “C:\M365ClientCert.pfx” file
- Enter the password of the certificate you specified while creating the certificate

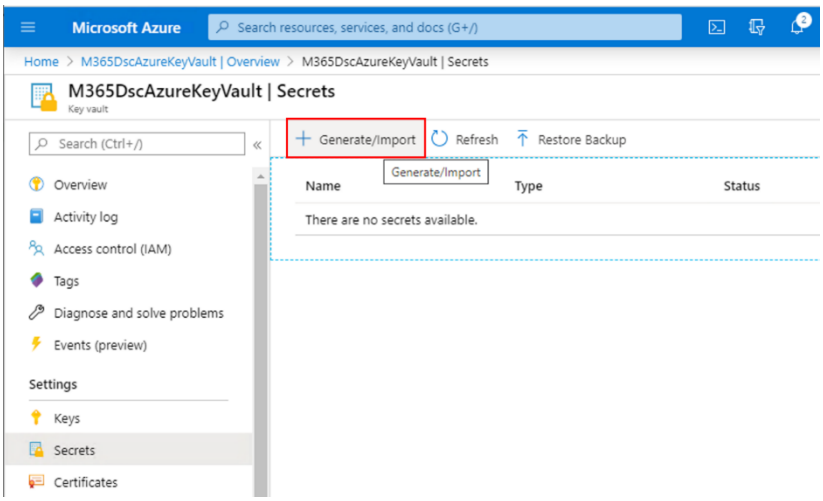
5.2.3 Adding the certificate password to Azure KeyVault

The Release pipeline needs the password of the certificate to import the certificate into the local certificate store. We are storing this password in KeyVault as well.

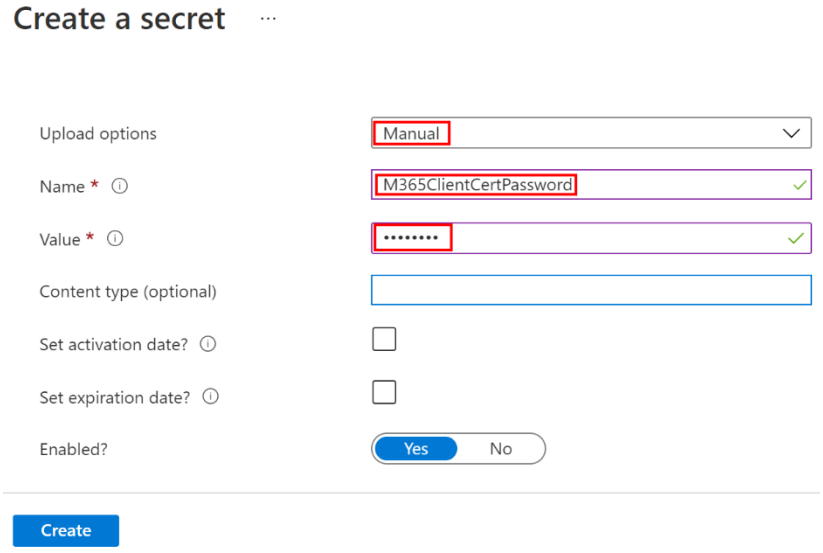
- Click "Secrets" in the left menu



- Click "Generate/Import" to create a new secret



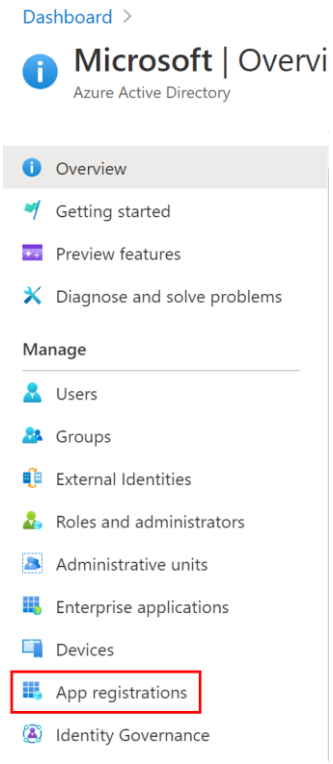
- Enter the correct information and click "Create"
  - Select "Manual" under "Upload options"
  - Use "M365ClientCertPassword" as the Name
  - Enter the password of the certificate you specified while creating the certificate



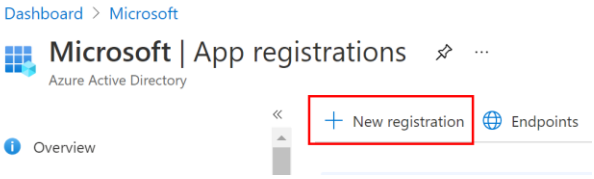
5.2.4 Create an App Registration in Azure Active Directory

- Open the Azure Portal (https://portal.azure.com)
- Go to Azure Active Directory

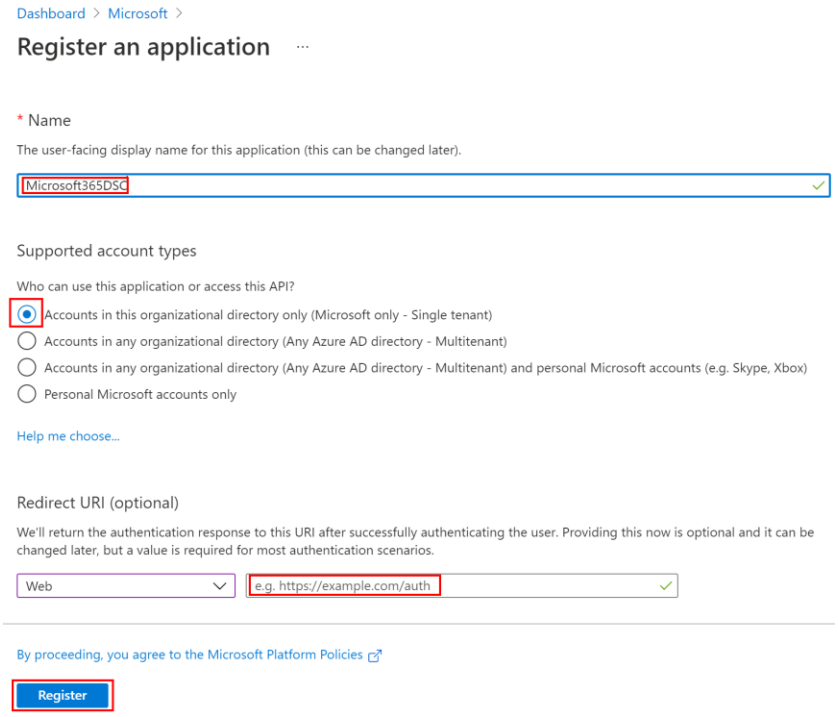
- Under “Manage”, click “App registrations”



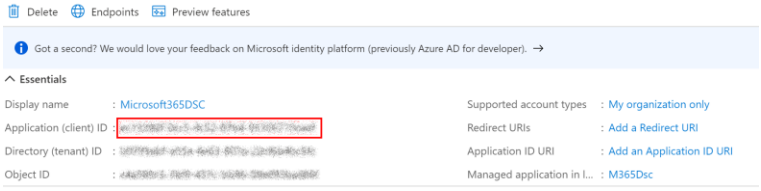
- Click “New registration” to create a new registration



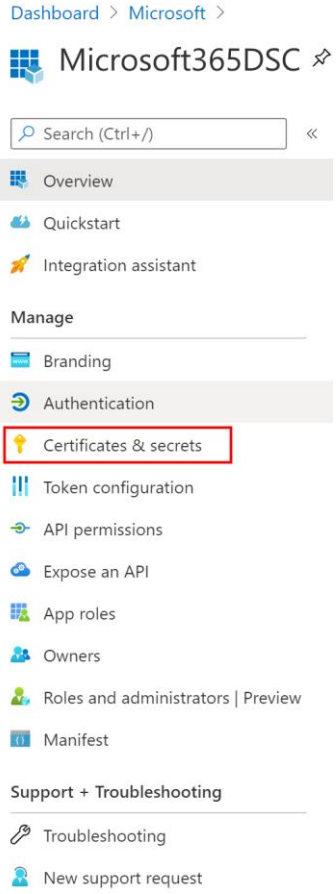
- Enter the following information and click “Register”:
  - Name: Microsoft365DSC
  - Supported account types: Accounts in this organizational directory only
  - Redirect URI: Select Web and leave the URI empty



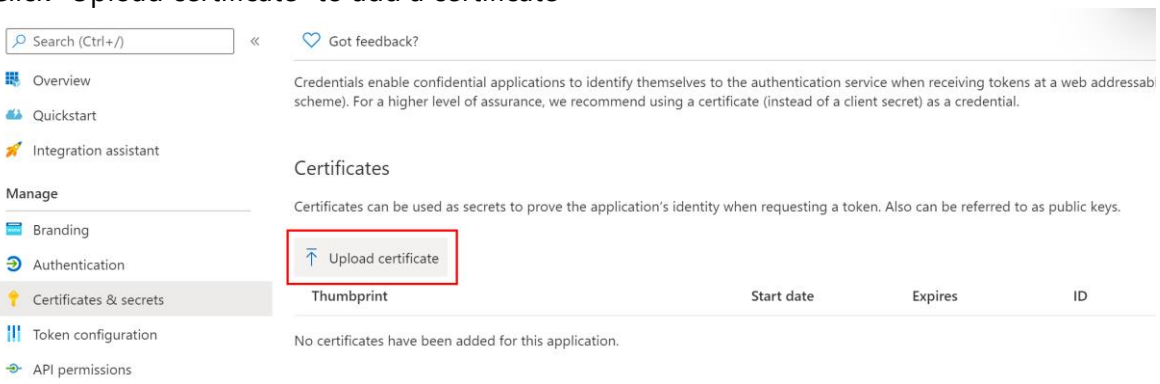
- Copy the "Application (client) ID" and store it for later use



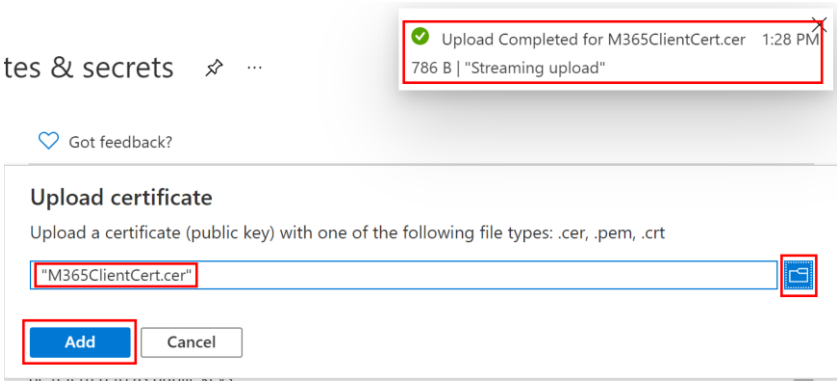
- Click the option “Certificates & secrets”



- Click “Upload certificate” to add a certificate

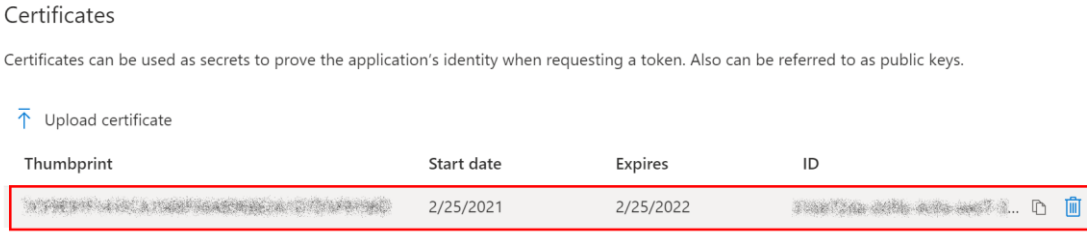


- In the popup, click the browse icon, select the "C:\M365ClientCert.cer" file and click "Add"

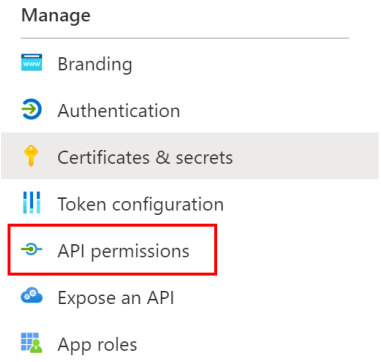


NOTE: After selecting the file, an upload message appears.

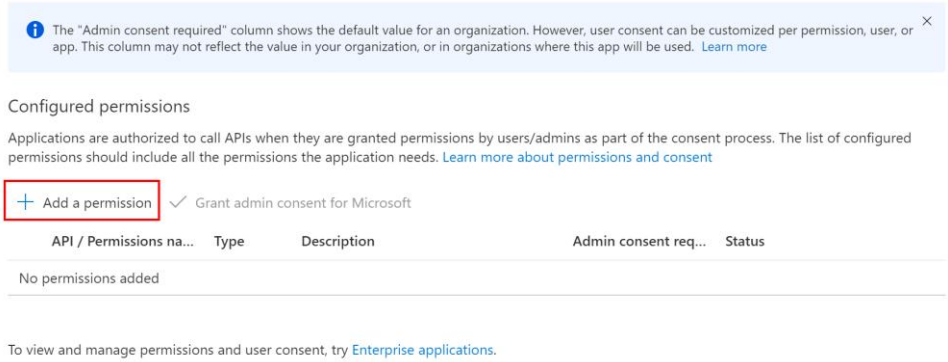
- The certificate has been added:



- Click the option "API permissions"

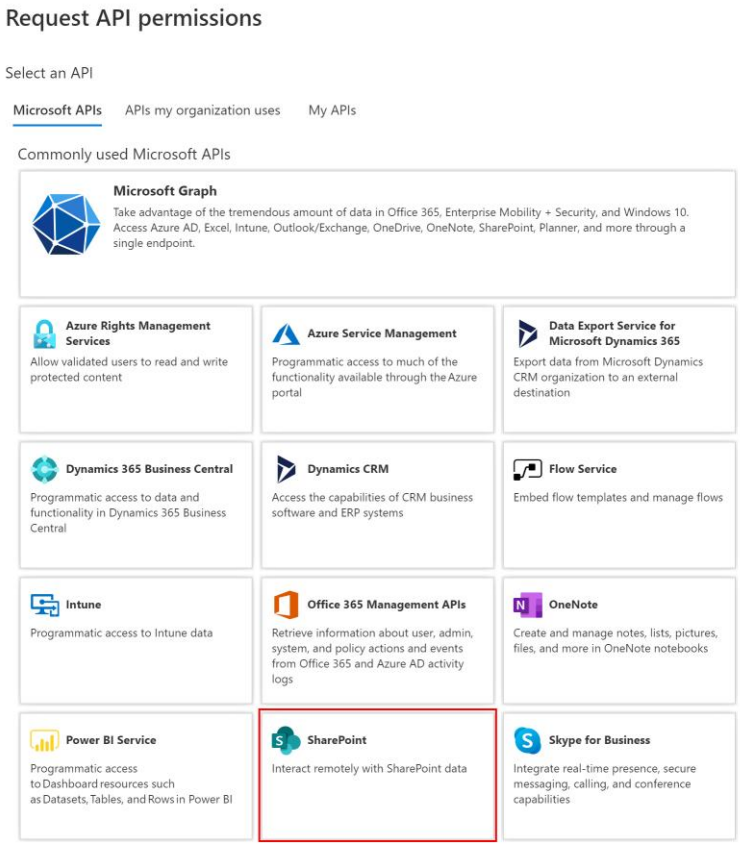


- Click "Add a permission"

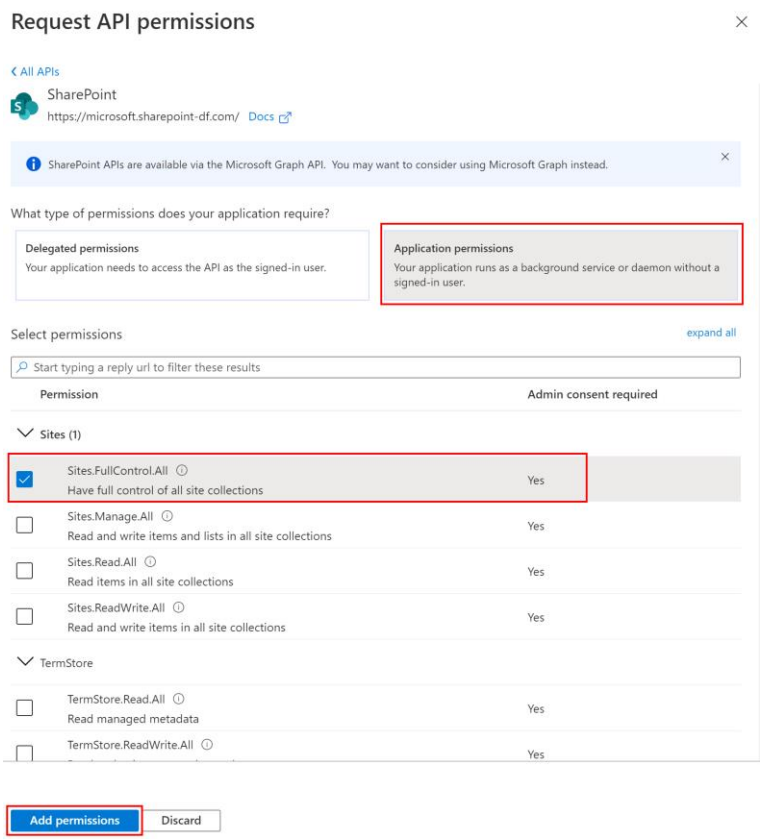


NOTE: Each resource/workload requires different permissions. This example only requires the currently listed permissions.

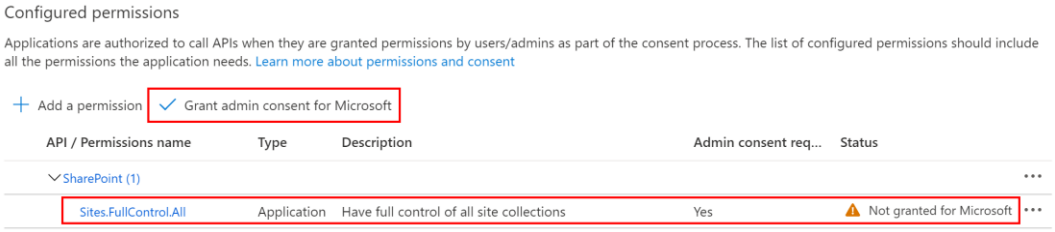
- Select “SharePoint”



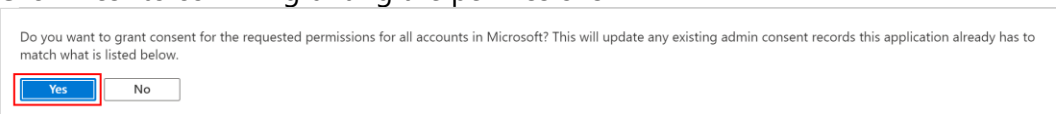
- Click “Application permissions”, select “Sites.FullControl.All” and click “Add permissions”



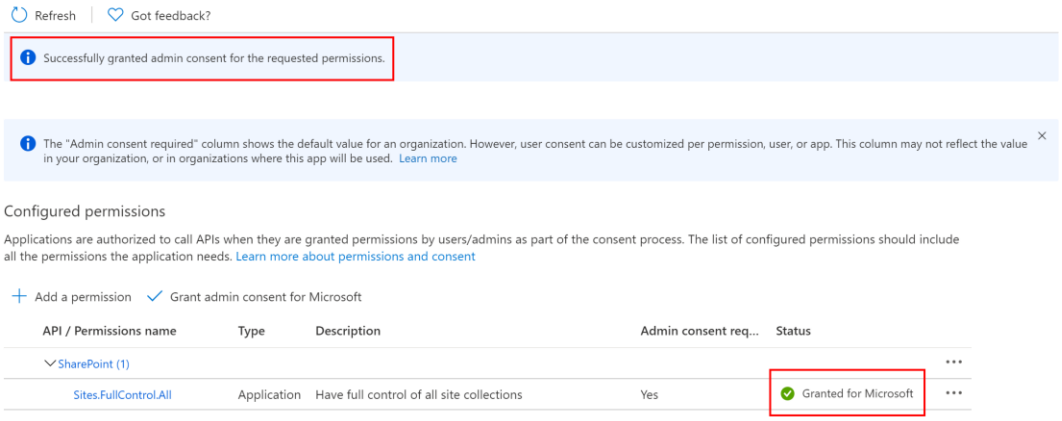
- The permission has been added, but permission has not yet been granted. Click "Grant admin consent for <org name>" to grant these permissions



- Click "Yes" to confirm granting the permissions



- You should receive the message that the permissions have been granted and see that the status is "Granted for <org name>"



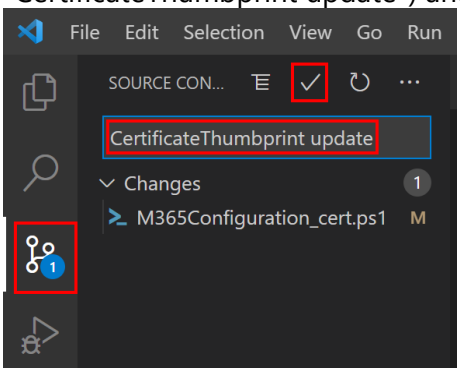
5.2.5 Updating the DSC configuration with the certificate thumbprint

- Open Visual Studio Code

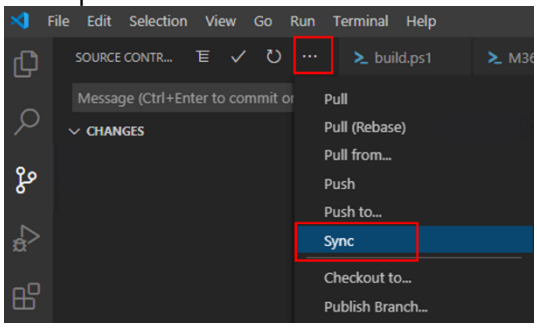
- Open the file "M365Configuration_cert.ps1"

- For each resource, update the following parameters:
  - ApplicationId: Application ID stored in paragraph 5.2.4 "Create an App Registration in Azure Active Directory".
  - TenantId: Change <tenant> with your tenant name.
  - CertificateThumbprint: Certificate thumbprint stored in the last step of paragraph 5.2.1 "Creating the authentication certificate".

- Click on the Git Source Control icon in the left menu, type a commit message (e.g. "CertificateThumbprint update") and click the checkmark icon



- Click the three dots icon and select "Sync" to sync your local changes with Azure DevOps

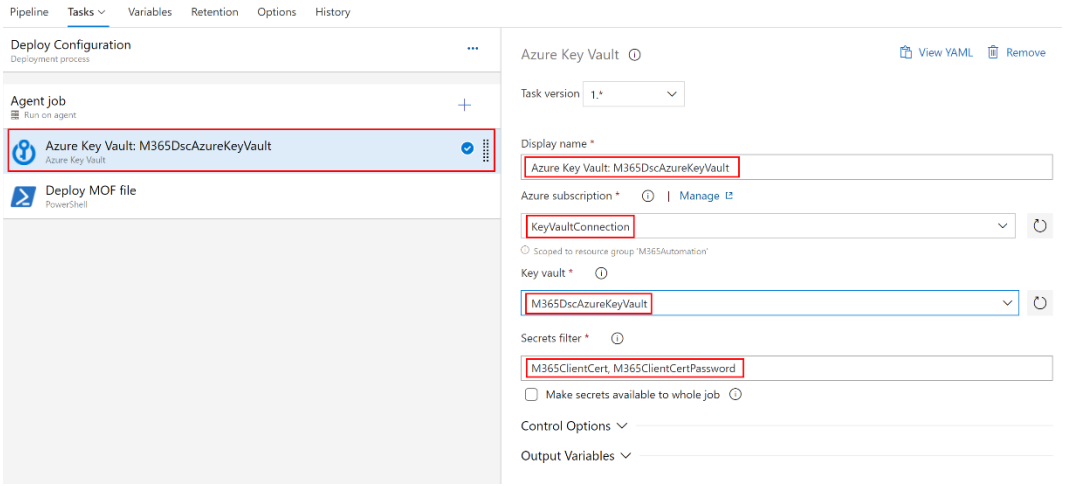


5.2.6 Creating the Build and Release pipelines

To create the Build and Release pipelines using certificate, you can now follow the paragraphs 4.2.1 “Create Build pipeline” and 4.2.2 “Create Release pipeline”, with two exceptions

1.) Change the following files:
- azure-pipelines.yml -> azure-pipelines_cert.yml
- build.ps1 -> build_cert.ps1
- deploy.ps1 -> deploy_cert.ps1
- M365Configuration.ps1 -> M365Configuration_cert.ps1

2.) Add a “Azure Key Vault” task to the Release pipeline (before the PowerShell task) with the following details:
- Display name : Azure KeyVault: M365DscAzureKeyVault
- Azure subscription : KeyVaultConnection
- Key vault : M365DscAzureKeyVault
- Secrets filter : M365ClientCert, M365ClientCertPassword



6 Script details

This whitepaper used some pre-created scripts. You can use these scripts as-is or tailor them to your own situation. This section describes what each script is for.

You can download the script package at:
https://microsoft365dsc.com/Pages/Resources/Whitepapers/M365Automation.zip

The package contains these files:

File name	Description
.gitignore	File used by Git, which specifies all files and folders Git has to ignore.
azure-pipelines.yml	The configuration file for the Azure DevOps Build Pipeline that is using the username/password. This file defines which steps are required to build the DSC MOF file.
azure-pipelines_cert.yml	The configuration file for the Azure DevOps Build Pipeline that is using Certificates for authentication (see paragraph 5.2). This file defines which steps are required to build the DSC MOF file.
build.ps1	The script that is responsible for retrieving the service account password from Azure KeyVault and building the DSC MOF file.
build_cert.ps1	Te script that is responsible for building the DSC MOF file and using a certificate for authentication.
deploy.ps1	Te script that is responsible for deploying the DSC MOF file to the LCM of the virtual machine.
deploy_cert.ps1	Te script that is responsible for importing the authentication certificate and deploying the DSC MOF file with certificate authentication to the LCM of the virtual machine.
DscResources.psd1	Data file that specifies the version of Microsoft365Dsc to be used.
M365Configuration.ps1	The Microsoft 365 configuration file which defines the target state using username/password.
M365Configuration_cert.ps1	The Microsoft 365 configuration file which defines the target state using certificate authentication.
M365ConfigurationData.psd1	The parameter file used to make sure the configuration can be used across multiple environments.
ReadMe.md	A project description file in Markdown format. This will be displayed when opening the repository in Azure DevOps.

7 Learning materials

7.1 Desired State Configuration

- Channel9: "Getting Started with PowerShell Desired State Configuration"
  - https://channel9.msdn.com/Series/Getting-Started-with-PowerShell-Desired-State-Configuration-DSC
- Channel9: "Advanced PowerShell Desired State Configuration"
  - https://channel9.msdn.com/Series/Advanced-PowerShell-Desired-State-Configuration-DSC-and-Custom-Resources
- Desired State Configuration Overview for Engineers
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/overview/dscforengineers
- Creating configurations
  - Configurations: https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/configurations
  - DependsOn: https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/resource-depends-on
  - DSC Resources: https://docs.microsoft.com/en-us/powershell/scripting/dsc/resources/resources
- Using configuration data in DSC
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/configdata
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/separatingenvdata
- Secure the MOF file
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/pull-server/securemof
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/configdatacredentials
- Local Configuration Manager
  - Configuring: https://docs.microsoft.com/en-us/powershell/scripting/dsc/managing-nodes/metaconfig
  - Push/Pull model: https://docs.microsoft.com/en-us/powershell/scripting/dsc/pull-server/enactingconfigurations
- Apply, Get, and Test Configurations on a Node
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/managing-nodes/apply-get-test
- Debugging DSC
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/troubleshooting/debugresource

7.2 Microsoft365Dsc

- Microsoft365dsc.com
  - https://microsoft365dsc.com/
- Microsoft365Dsc promotion video
  - https://aka.ms/m365dscpromo
- GitHub repository
  - https://github.com/microsoft/Microsoft365DSC
- What is Configuration-as-Code?
  - http://nikcharlebois.com/what-is-configuration-as-code
- Microsoft365Dsc YouTube channel
  - https://www.youtube.com/channel/UCveScabVT6pxzqYgGRu17iw
- PluralSight: “SharePoint Conference '19: IT Pros, Get Relevant! Upskilling for Today's Cloud” (subscription required)
  - https://www.pluralsight.com/courses/sharepoint-conference-2019-session-19

7.3 Git

- Git manual
  - https://git-scm.com/book/en/v2
- PluralSight: "How Git Works” (subscription required)
  - https://app.pluralsight.com/library/courses/how-git-works/table-of-contents
- PluralSight: "Mastering Git" (subscription required)
  - https://app.pluralsight.com/library/courses/mastering-git/table-of-contents

8 Acronyms

Acronym	Meaning
DSC	Desired State Configuration
LCM	Local Configuration Manager
VM	Virtual Machine