

# *Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps*

Author:

Yorick Kuijs  
Cloud Solution Architect @ Microsoft  
[yorick.kuijs@microsoft.com](mailto:yorick.kuijs@microsoft.com)

Date:

23 November 2022

Version:

v2.0



## 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.

© 2022 Microsoft Corporation. All rights reserved.

## Changelog

Version	Date	Author	Changes
1.0	1 November 2020	Yordan Bechev Yorick Kuijs	First release
1.0.1	3 November 2020	Yorick Kuijs	Updated incorrect links
1.1	2 December 2020	Yorick Kuijs	Incorporated feedback from Zaki Semar Shahul Added Azure Conditional Access for the used service account
1.2	1 October 2021	Yorick Kuijs	Corrected issues Added Certificate authentication scenario
1.21	23 December 2021	Yorick Kuijs	Corrected download link to scripts after migration to new website
2.0	23 November 2022	Yorick Kuijs	Major update: Combining scenarios, demonstrating new flexible setup.  Reviewed by Brian Lalancette, Andi Krüger, Jeffrey Rosen, Andrew Piskai, Dean Sesko and Albert Boland.

## Table of Contents

1	Introduction.....	5
1.1	Microsoft 365 and DevOps.....	5
1.2	Setup.....	6
1.3	Assumptions.....	6
2	Solution Description.....	8
2.1	DevOps Configuration.....	8
2.1.1	Build Pipeline .....	8
2.1.2	Release Pipeline .....	8
2.2	DSC configuration.....	9
2.3	Customize the solution .....	9
2.4	App Registration Overview.....	10
3	Prerequisites.....	11
3.1	Virtual Machine .....	11
3.2	Azure DevOps.....	11
3.3	Azure .....	11
3.4	Microsoft 365.....	11
3.5	Licenses .....	12
4	Preparation .....	13
4.1	Preparing the Virtual Machine (Phase 1).....	13
4.1.1	Configure PowerShell requirements.....	13
4.1.2	Create Azure DevOps agent service account .....	13
4.1.3	Creating the Microsoft 365 authentication certificate.....	14
4.1.4	Configure the Local Configuration Manager.....	14
4.2	Preparing the Microsoft 365 tenant.....	17
4.2.1	Create an account for DSC in Microsoft 365 .....	17
4.2.2	Create an App Registration in Azure Active Directory.....	18
4.2.3	Add the App Registration to the Exchange Administrators role.....	20
4.3	Preparing the Azure DevOps environment .....	22
4.3.1	Create a new project in Azure DevOps.....	22
4.3.2	Create an Agent Pool in Azure DevOps.....	23
4.3.3	Create Personal Access Token.....	25
4.4	Preparing the Virtual Machine (Phase 2).....	27
4.4.1	Installing and configuring the Azure DevOps Agent on the virtual machine .....	27

4.5	Preparing the Azure Key Vault .....	32
4.5.1	Create an App Registration .....	32
4.5.2	Granting the App Registration Reader permissions to the Azure Subscription..	34
4.5.3	Create a new Azure Key Vault .....	37
4.5.4	Adding a Service Connection to Azure to the Azure DevOps project .....	40
5	Configuring Azure DevOps.....	45
5.1	Populate scripts.....	45
5.2	Add secrets to your Key Vault.....	51
5.3	Configure Azure DevOps project .....	52
5.3.1	Create Build pipeline.....	52
5.3.2	Create the Deployment Release pipeline.....	55
5.3.3	Validate that changes to the config are deployed successfully .....	63
5.3.4	Create a scheduled Compliancy Test Release pipeline.....	65
6	Troubleshooting.....	73
6.1	Error: Service connection could not be found.....	73
6.2	Error: Release pipeline throws an error about the PSGallery not found .....	73
7	Security Enhancements .....	74
7.1	Using Azure Conditional Access to secure service account.....	74
8	Script details .....	79
9	Learning materials.....	82
9.1	Desired State Configuration.....	82
9.2	Microsoft365DSC.....	83
9.3	Git.....	83
10	Acronyms.....	84

## 1 Introduction

Microsoft 365 is a very popular productivity cloud solution. Each customer has their own tenant which stores their data, applications and configuration. Using the Administration Portal (<https://admin.microsoft.com>) each customer can configure and manage their tenant.

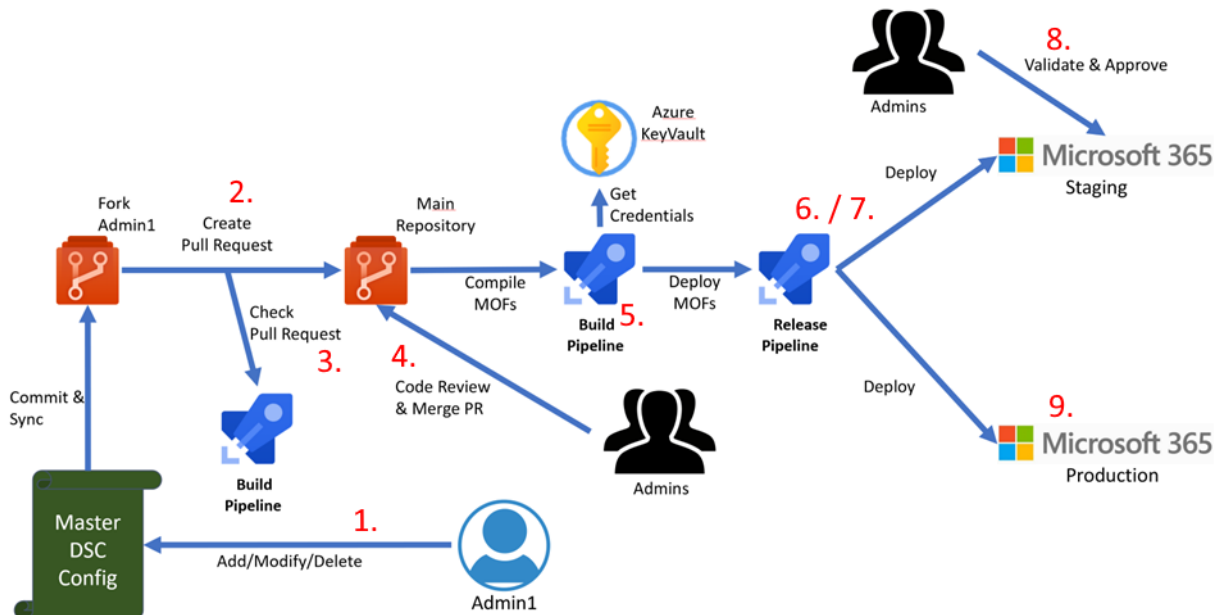
Many companies are adopting DevOps practices and are interested in applying them against Microsoft 365 as well. Infrastructure as Code and Continuous Deployment/Continuous Integration (CD/CI) are important concepts in DevOps.

Microsoft365DSC is a PowerShell Desired State Configuration (DSC) module which can configure and manage Microsoft 365 in a true DevOps style: "Configuration-as-Code".

### 1.1 Microsoft 365 and DevOps

When you perform management of your Microsoft 365 tenant manually, there is no way to consistently deploy changes and to monitor for changes. By using "Configuration-as-Code" principles, you document/define the configuration of your tenant in code. You can then deploy this configuration programmatically to your tenant and periodically check if the defined/intended configuration still matches the actual configuration. The tool that allows you to do this is Microsoft365DSC (<https://microsoft365dsc.com>).

By adding CD/CI capabilities, for example by using Azure DevOps, you can also add additional quality gates making sure changes to your configuration are deployed in a controlled and consistent way.



Steps:

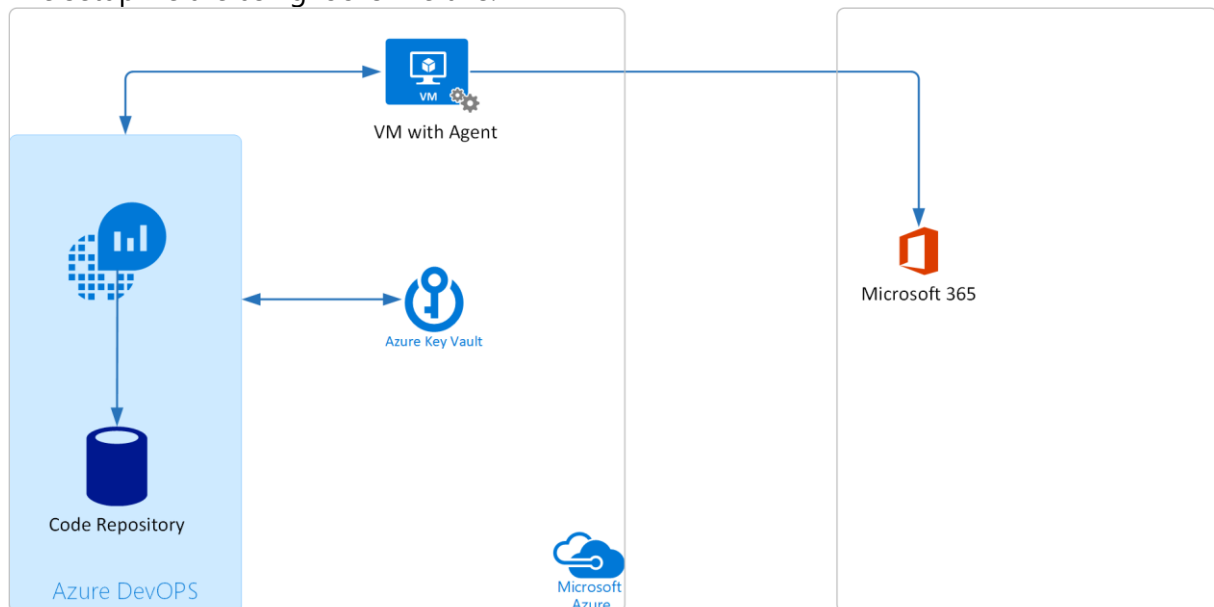
1. Admin1 updates the configuration in his personal copy (fork)
2. When done, Admin1 creates a Pull Request to have his changes merged into the Main repository

3. The quality assurance process starts:
  - a. An automated process runs certain quality checks against the Pull Request
  - b. Other admins validate the changes via a peer review process.
4. When both succeed, the Pull Request is merged.
5. The Merge initiates a Build pipeline, which retrieves credentials from Azure Key Vault and compiles what are known as *MOF files*
6. Once the Build pipeline completes successfully, a Release pipeline starts, which deploys the generated MOF file to the Staging environment
7. After a successful deployment, the Release pipeline sends a notification to the admins and waits for approval
8. The admins check if the change has been deployed successfully and if the desired result has been achieved. If that is the case, they approve the deployment.
9. After the approval is given, the Release pipeline performs the deployment to the Production environment
10. The change has now been automatically and consistently deployed to all environments

## 1.2 Setup

In this document we are going to describe the process and steps required to implement a basic Configuration-as-Code setup using Microsoft365DSC, Azure DevOps and Azure Key Vault. Changes to Microsoft 365 are made within a Git repository in Azure DevOps and then fully and automatically deployed to a Microsoft 365 tenant.

The setup we are using looks like this:



## 1.3 Assumptions

This document assumes you are familiar with creating and deploying PowerShell Desired State Configurations (DSC).

**IMPORTANT:** If you are new to PowerShell DSC, please first check out the first two links in paragraph 9.1. These are recordings of two very good PowerShell DSC training courses that will give you a good understanding of what PowerShell DSC is and how it works. A good foundation for the skills you need when working with PowerShell DSC.

## 2 Solution Description

This solution consists of multiple components. In this chapter, the solution is described in more detail.

### 2.1 DevOps Configuration

In Azure DevOps, you can use various components:

- Build and Release pipelines
- Microsoft Hosted Agents and Self-Hosted Agents

This solution uses all these components.

#### 2.1.1 Build Pipeline

The Build pipeline is responsible for running the Build script, which compiles the DSC configuration into a MOF file. It is using Microsoft Hosted agents to run the Build script, since the requirements for compiling DSC configurations are limited and can easily be covered by the Microsoft Hosted agents.

The Build script prepares the Microsoft Hosted agent, reads the data files, retrieves the required information from Azure Key Vault and uses this information to compile the MOF file. Each environment has its own data file (in the DataFiles folder) and will compile its own MOF file.

**NOTE:** This solution only uses one environment, but you can extend this if needed.

The result of the Build script is an Output folder in which all components are placed that are required for the deployment of the MOF files. These are:

- The MOF files themselves
- The deployment script
- The DSCResources.ps1 file, to determine which version of Microsoft365DSC is used and must be installed.

At the end of the pipeline, the entire Output folder is packaged as a Zip file and attached to the Build pipeline as an artifact.

#### 2.1.2 Release Pipeline

This solution creates two Release pipelines:

- For deploying the DSC configuration to the environment(s)
- For checking the compliance of the environment(s) with the latest configuration

The Release pipelines use Self Hosted agents (on a virtual machine) to run the corresponding scripts.

The deployment pipeline uses the Build artifacts to deploy the generated MOF file to the corresponding environment. The deploy script prepares the self-hosted agent by installing Microsoft 365 authentication certificate (if not present), installing Microsoft365DSC (including all required modules) and deploying the MOF file to the specified environment. Each environment will be its own stage in the Release pipeline.



**NOTE:** This solution only creates one Stage for one environment, but more stages can be created if needed.

The compliance pipeline uses the Build artifacts to check if all environments are still compliant with the desired state. It retrieves all MOF files in the artifact and runs a compliance check for each of them.

When done, it will create a summary report and either send this via an e-mail or via a Teams channel message. This is configurable in the script.

## 2.2 DSC configuration

The DSC configuration uses so-called Composite Resources, which are a way to structure DSC resources into separate configurations. So instead of creating:

One huge DSC configuration with all DSC resources for all workloads, which will become very hard to read and maintain.

You now have multiple smaller and dedicated composite resources and one main DSC configuration (M365Configuration.ps1) which is responsible for calling each of the composite resources.

For this purpose, a M365Config module is created (included in the scripts) which contains a composite resource for each workload. Each composite resource contains all DSC resources for that workload, which makes it much easier to read and maintain.

**Note:** See the link in "Composite Resources" paragraph 9.1 for more information on Composite resources

## 2.3 Customize the solution

Of course, you can update the setup described and/or the DSC configuration to fit your specific situation. Better yet, you should update the sample configuration with your own settings!

As mentioned earlier, the current solution only uses a single Microsoft 365 tenant / environment (named "Production"), but this can be extended to include multiple environments like Test and Acceptance tenants. To do this:

- Create an account and app registration in each tenant (paragraph 4.2)
- Add a data file to the DataFiles folder for each tenant and update this with the correct information for that tenant (paragraph 5.1)
- Populate the secrets for that environment in Azure Key Vault (paragraph 5.2)
- Add a new stage to the release pipeline for each new environment. Make sure you update the value of the "Environment" parameter with the name of the newly created data file (paragraph 5.3.2)

- In the configuration window for the Azure PowerShell task, configure the following settings:
  - Azure Subscription: "KeyVaultConnection" or whatever name you gave the service connection
  - Select "Script File Path" as "Type" and browse to the "deploy.ps1" file by clicking the "..." button
  - Enter "-Environment **Production**" as Script Arguments
  - Select "Stop" as ErrorActionPreference and check the "Fail on Standard Error" checkbox

The solution is targeted to a specific version of Microsoft365DSC which is **v1.22.1019.1** - the most recent version at the time of writing. If you want to use a different version of Microsoft365DSC, edit the DscResources.psd1 file in the repository and enter the desired version.

### 2.4 App Registration Overview

This solution is using various App Registrations in various places. This paragraph provides an overview of all user app registrations and their purpose. The rest of this document refers to the numbers in this overview:

Nr	Name	Description
1.	Microsoft 365 authentication	<p>This app registration is used by Microsoft365DSC to authenticate with the Microsoft 365 tenant using application credentials. Each tenant you are managing using this solution requires its own app registration.</p> <p>The process to create this app registration can be found in paragraph 4.2.2.</p>
2.	Azure authentication	<p>The Azure DevOps project is using this app registration to authenticate with Azure and retrieve credentials from the Azure Key Vault.</p> <p>The process to create this app registration can be found in paragraph 4.5.1.</p>
3.	Mail authentication	<p>If you choose to use e-mail to send status reports, you need an app registration to authenticate against Microsoft 365 so you can use that as SMTP server.</p> <p>The process to create this app registration can be found here: <a href="https://learn.microsoft.com/en-us/graph/auth-register-app-v2">https://learn.microsoft.com/en-us/graph/auth-register-app-v2</a> <a href="https://learn.microsoft.com/en-us/graph/auth-v2-service#2-configure-permissions-for-microsoft-graph">https://learn.microsoft.com/en-us/graph/auth-v2-service#2-configure-permissions-for-microsoft-graph</a></p>

## 3 Prerequisites

### 3.1 Virtual Machine

To deploy DSC configurations, we require a machine that will serve to perform the actual deployment to Microsoft 365. This can either be a physical or virtual machine; in this guide we assume the use of a virtual machine. The requirements for this virtual machine are:

1. Windows Server 2016 / Windows 10 or above
  - Recommended to have at least 2 CPUs and 8GB of memory.
  - x64 version is required.

**Note:** Using the ARM version of Windows is not supported

2. .Net Framework 4.7 or higher
  - <https://dotnet.microsoft.com/download/dotnet-framework>
3. PowerShell v5.1
  - Installed by default on all current versions of Windows Server

**Note:** Later PowerShell versions aren't supported at this time, because some modules used by Microsoft365DSC don't support those PowerShell versions yet.

### 3.2 Azure DevOps

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

1. An Azure DevOps tenant and permissions to configure this tenant
2. A project in Azure DevOps

### 3.3 Azure

To be able to connect to Microsoft 365, you need credentials and a way to store these securely. This solution uses Azure Key Vault to store the password, application secret or certificate securely. The pipelines use Azure Key Vault to retrieve the required information when this is needed.

### 3.4 Microsoft 365

We also need a Microsoft 365 tenant, which will be managed using Microsoft365DSC.

In this tenant we need:

1. An account with Global Administrator privileges, used to access the Admin Portal
2. A service account with Global Administrative privileges, used to deploy settings using DSC
  - This account does not support being configured to use Multi-Factor Authentication
  - The actual required permissions will depend on the resources used and workloads configured (e.g. Exchange Online, Teams)
3. An App Registration with the appropriate permissions to Microsoft 365
  - Steps to create this app registration are described in paragraph 4.2.2 of this whitepaper

### 3.5 Licenses

You can either use a fully licensed or a trial version of the above-mentioned products.

Microsoft365DSC is open-source and available under a MIT license (<https://github.com/microsoft/Microsoft365DSC/blob/master/LICENSE>), which means that you do not need to purchase any license and can use it for free.

## 4 Preparation

### 4.1 Preparing the Virtual Machine (Phase 1)

#### 4.1.1 Configure PowerShell requirements

This solution needs a few components to be installed for it to work. In this step we are going to install these components:

- Log on to the virtual machine with Administrative credentials
- Open an elevated Windows PowerShell window
- Update PowerShellGet by executing the following commands:

```
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/>

**Note 2:** It is possible that the PowerShell Gallery isn't registered correctly in your installation. In that case "Get-PSRepository" will not return any results. If so, run the following command:

```
Register-PSRepository -Default
```

- Install the Az.KeyVault module by executing the following command:

```
Install-Module Az.KeyVault -Force
```

This command should install the Az.KeyVault and Az.Accounts modules to "C:\Program Files\WindowsPowerShell\Modules" folder.

- (Windows client versions only) Enable Windows Remote Management by executing the following command:

```
Enable-PSRemoting -Force
```

#### 4.1.2 Create Azure DevOps agent service account

The Azure DevOps agent needs a service account with the correct permissions. In this step we are going to create this account and assign local Administrator permissions:

- Log onto the virtual machine
- Open "Computer Management"
- Create a local service account, for example: "DevOpsAgent"

This account will be used to run the Azure DevOps agent with, which is used by Azure DevOps to deploy configurations to Microsoft 365.

**Note:** Make sure you use a long and complex password.

- Add this account to the local Administrators group

### 4.1.3 Creating the Microsoft 365 authentication certificate

To authenticate against Microsoft 365, we need a certificate. In this step we are going to create a certificate:

- Log onto the virtual machine with administrative credentials
- Open an elevated Windows PowerShell window
- Create and export a new authentication certificate by running the following PowerShell commands:

- **NOTE:** Update the [PASSWORD] parameter to your own password

```
$clientCert = New-SelfSignedCertificate -Subject  
"CN=Microsoft365DSC" -CertStoreLocation "Cert:\LocalMachine\My"  
-KeyExportPolicy Exportable -KeySpec Signature  
  
$password = ConvertTo-SecureString -String "[PASSWORD]" -  
AsPlainText -Force  
  
Export-PfxCertificate -Cert $clientCert -FilePath  
C:\M365ClientCert.pfx -Password $password  
  
Export-Certificate -Cert $clientCert -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>

- Copy the created file "C:\M365ClientCert.cer" and store it for later use
- Run the following command, copy the displayed Thumbprint and document it for later use

```
$clientCert.Thumbprint
```

**Note:** Repeat these steps for each environment you are going to manage (only one environment described in this whitepaper). More information about implementing more environments, see paragraph 2.3.

### 4.1.4 Configure the Local Configuration Manager

We need an encryption certificate to encrypt the credentials used in the DSC configuration. In this step we are creating this certificate:

- Log onto your virtual machine with administrative credentials
- Open an elevated PowerShell ISE and run the following command

```
$certForDSC = 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 document the value:

```
$certForDSC.Thumbprint
```

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

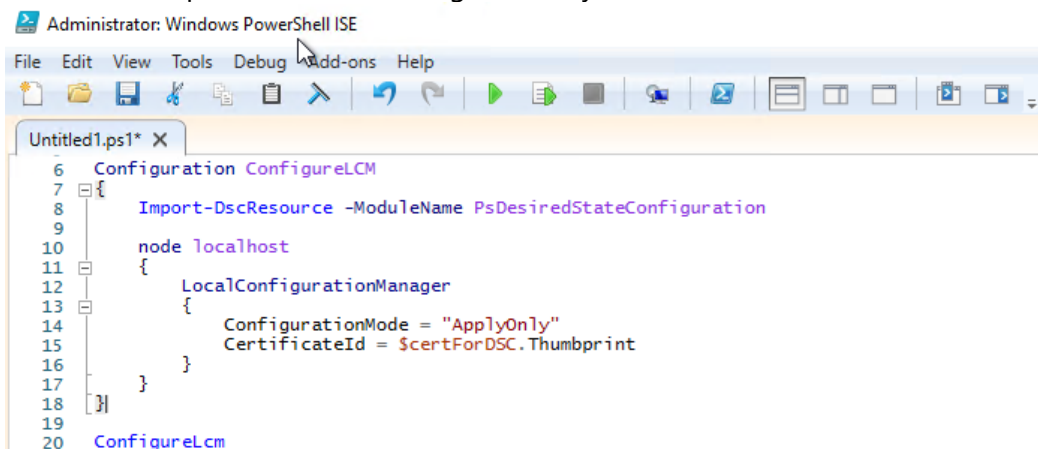
```
Export-Certificate -Cert $certForDSC -FilePath  
C:\DSCCertificate.cer
```

- In the PowerShell window, browse to the folder "C:\M365Dsc"
  - Create this folder if it does not yet exist

- Paste the following code in the white script pane:

```
Configuration ConfigureLCM  
{  
    Import-DscResource -ModuleName  
    PsDesiredStateConfiguration  
  
    node localhost  
    {  
        LocalConfigurationManager  
        {  
            ConfigurationMode = "ApplyOnly"  
            CertificateId = $certForDSC.Thumbprint  
        }  
    }  
}  
  
ConfigureLcm
```

- Run the code (press F5 or click the green "Play" icon)



- A prompt will be shown indicating that the localhost.meta.mof has been created. Note the output path and replace the string <OutputDirectory> with it, below.

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

```
Set-DscLocalConfigurationManager -Path <OutputDirectory>
-Verbose
```

The output should look like this:

```
PS C:\M365DSC> Set-DscLocalConfigurationManager -path C:\M365DSC\Config\Lcm -Verbose
VERBOSE: Performing the operation "Start-DscConfiguration: SendMetaConfigurationApply" on target "MSFT_DSCLocalConfigurationManager".
VERBOSE: Perform operation 'Invoke CimMethod' with following parameters, ''methodName' = SendMetaConfigurationApply, 'className' = MSFT_DSCLocalConfigurationManager, 'namespaceName' = root/Microsoft/Windows/DesiredStateConfiguration'.
VERBOSE: An LCM method call arrived from computer M365AUTOMATIONV with user sid S-1-5-21-773870280-229285111-1404544293-500.
VERBOSE: [M365AUTOMATIONV]: LCM: [ Start Set ]
VERBOSE: [M365AUTOMATIONV]: LCM: [ Start Resource ] [MSFT_DSCLocalConfigurationManager]
VERBOSE: [M365AUTOMATIONV]: LCM: [ Start Set ] [MSFT_DSCLocalConfigurationManager]
VERBOSE: [M365AUTOMATIONV]: LCM: [ End Set ] [MSFT_DSCLocalConfigurationManager] in 0.0210 seconds.
VERBOSE: [M365AUTOMATIONV]: LCM: [ End Resource ] [MSFT_DSCLocalConfigurationManager]
VERBOSE: [M365AUTOMATIONV]: LCM: [ End Set ]
VERBOSE: [M365AUTOMATIONV]: LCM: [ End Set ] in 0.0610 seconds.
VERBOSE: Operation 'Invoke CimMethod' complete.
VERBOSE: Set-DscLocalConfigurationManager finished in 0.118 seconds.
```

- To validate a successful configuration of the thumbprint, run Get-DscLocalConfigurationManager. The "CertificateID" parameter should now show the Certificate Thumbprint of your certificate and the "ConfigurationMode" should show "ApplyOnly".

```
PS C:\M365DSC> Get-DscLocalConfigurationManager

ActionAfterReboot      : ContinueConfiguration
AgentId                : 0C476C8E-2937-11ED-83CB-000D3AA93695
AllowModuleOverwrite   : False
CertificateID          : 6D094530ACB5F5E67EDFF394843941A43193EEEB
ConfigurationDownloadManagers : {}
ConfigurationID        : 
ConfigurationMode      : ApplyOnly
ConfigurationModeFrequencyMins : 15
Credential             : 
DebugMode              : {NONE}
DownloadManagerCustomData : 
DownloadManagerName    : 
LCMCompatibleVersions  : {1.0, 2.0}
LCMState               : Idle
LCMStateDetail         : 
LCMVersion             : 2.0
StatusRetentionTimeInDays : 10
SignatureValidationPolicy : NONE
SignatureValidations    : {}
MaximumDownloadSizeMB  : 500
PartialConfigurations  : 
RebootNodeIfNeeded     : False
RefreshFrequencyMins   : 30
RefreshMode            : PUSH
ReportManagers         : {}
ResourceModuleManagers : {}
PSComputerName         :
```

**Note:** We configure the ApplyOnly setting because we will use a pipeline to implement the monitoring functionality, later in this document.

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



## 4.2 Preparing the Microsoft 365 tenant

### 4.2.1 Create an account for DSC in Microsoft 365

The solution needs an account with administrative privileges that can be used to manage the Microsoft 365 settings. In this step we are creating such an account and assigning it the Global Administrator permissions:

- Open an Internet browser
- Browse to the Microsoft 365 Admin Portal (<https://admin.microsoft.com>)
- Create a new account
  - For example: "DscAdmin"
  - Do not assign any license
  - Grant the user Global Admin permissions

**Note:** More limited permissions may suffice, depending on the resources used in your configuration

- Make sure this account does **not** have Multi-Factor Authentication (MFA) enabled!

### 4.2.2 Create an App Registration in Azure Active Directory

Some of the Microsoft 365 workloads also support authentication using application credentials. To use this, an app registration must be created in Azure Active Directory<sup>1</sup>, which is granted the correct permissions.

Microsoft365DSC has a cmdlet that can create and manage an app registration for you, including permissions. In this we will create a new app registration using this cmdlet:

- Log onto your virtual machine
- Open an elevated PowerShell window
- Install the most recent versions of Microsoft365DSC and Az.Resources modules by running the following command:

```
Install-Module Microsoft365DSC, Az.Resources
```

- Run the following cmdlet. Replace "<APPNAME>" with the name you want to use for the app registration, for example "Microsoft365DSC":

```
Update-M365DSCAzureAdApplication -ApplicationName '<APPNAME>'
-Permissions @(@{ Api =
"SharePoint";PermissionName="Sites.FullControl.All"},@{ Api =
"Exchange";PermissionName="Exchange.ManageAsApp"}) -
AdminConsent -Type Certificate -CertificatePath
C:\M365ClientCert.cer
```

**Note:** This command configures all permissions required in this solution. If you add more resources, you might need to add more permissions as described on the resources pages on <https://microsoft365dsc.com>.

---

<sup>1</sup> Use App Registration nr 1. See paragraph 2.3 for more information.

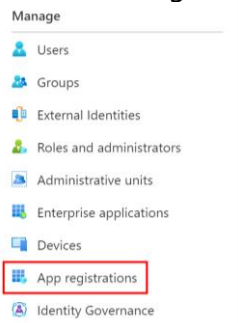
- The script outputs the ApplicationID and TenantID. Copy this information and document it for later use.

**Important:** Only perform the next steps if the script shows an error that consent could not be provided successfully:

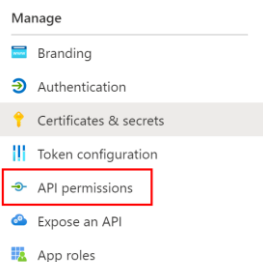
- Open the Azure Portal (<https://portal.azure.com>)
- Log on using an account from the same domain as your Microsoft 365 tenant
- Go to Azure Active Directory



- Under “Manage”, click “App registrations”



- Click on the app that was created using the “Update-M365DSCAzureAdApplication” cmdlet
- Click the option “API permissions”



- Click “Grant admin consent for <org name>” to grant these permissions

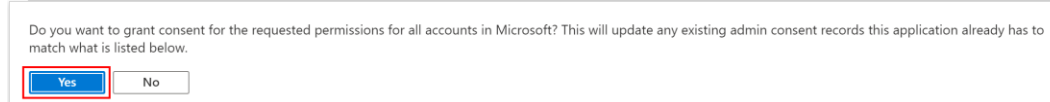
Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

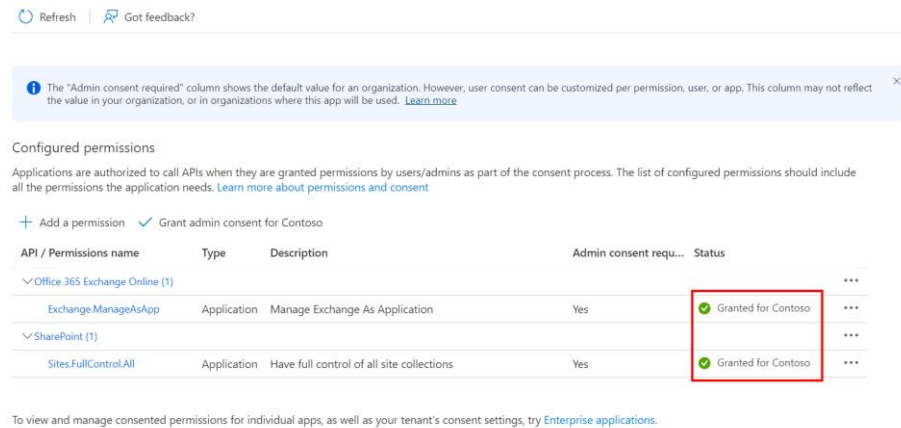
+ Add a permission ☒ Grant admin consent for Microsoft

API / Permissions name	Type	Description	Admin consent req...	Status
▼ SharePoint (1)				
Sites.FullControl.All	Application	Have full control of all site collections	Yes	⚠ Not granted for Microsoft

- 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>"



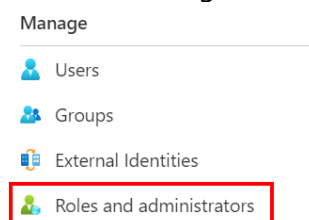
### 4.2.3 Add the App Registration to the Exchange Administrators role

When you are using app credentials to manage Exchange (as is done in this solution), you need to add the app registration to the Exchange Administrators role group. That way the app registration has the correct permissions to manage Exchange. In this step, you will add the created app registration to the "Exchange Administrator" role group:

- Open the Azure Portal (<https://portal.azure.com>)
- Log on using an account from the same domain as your Microsoft 365 tenant
- Go to Azure Active Directory

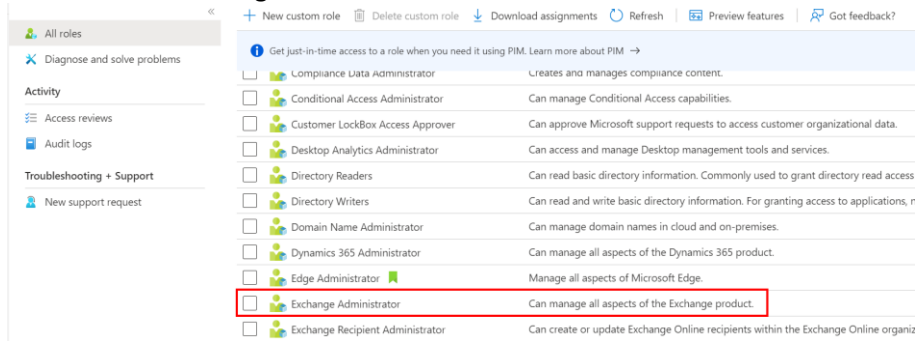


- Under "Manage", click "Roles and administrators"

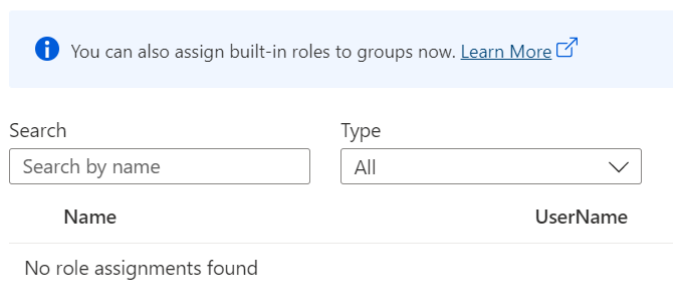
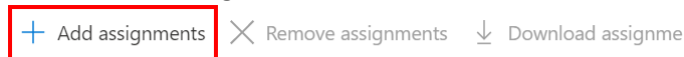


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

- Look for the “Exchange Administrator” role

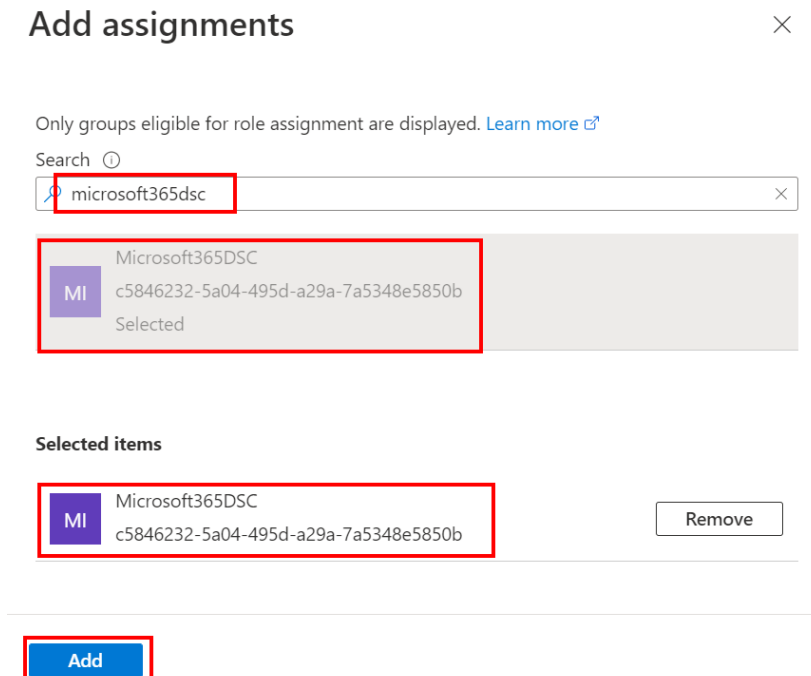


- Click the “Add assignments” button



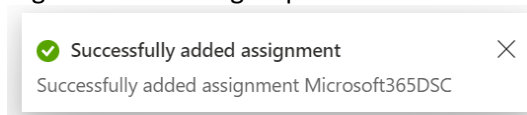
- Search for the App Registration that was created in the previous paragraph by entering “Microsoft365DSC”. Then select the app that appears, make sure it appears under “Selected items” and click “Add”.

### Add assignments

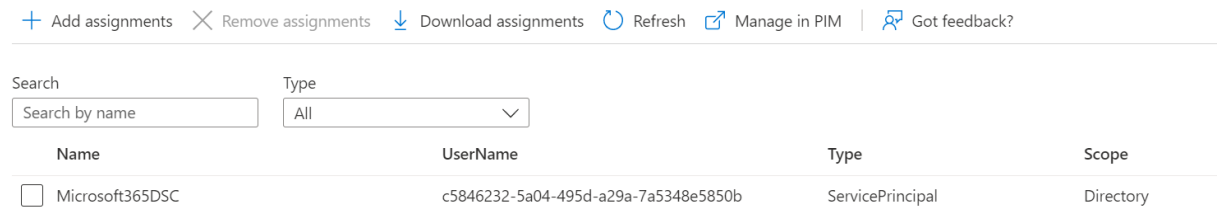


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

- In the upper-right corner, a message appears confirming the successful addition of the app registration to the group



- The app registration should be present in the group assignments

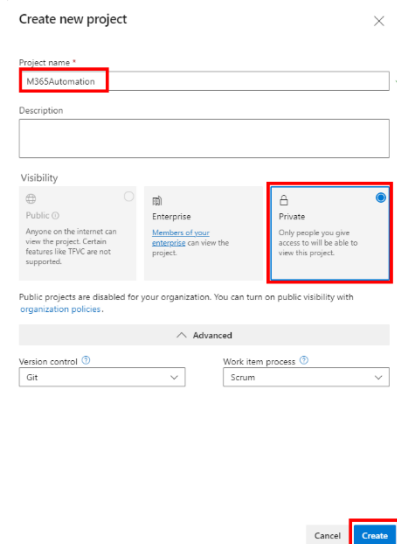


### 4.3 Preparing the Azure DevOps environment

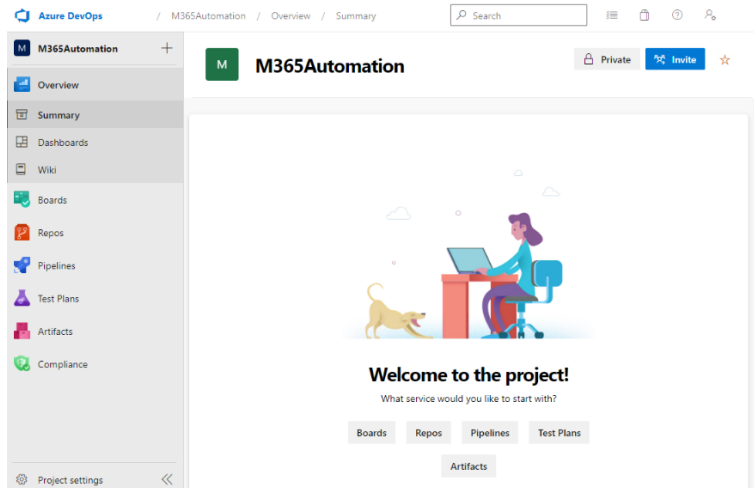
#### 4.3.1 Create a new project in Azure DevOps

We need a new project in Azure DevOps in which the DSC configurations will be stored and from where the deployments will be executed. In this step we will create a new project:

- Log into the Azure DevOps portal
- Click the "New project" button in the upper-right corner
- Enter "M365Automation" as project name (or use your own name) and select "Private". Leave all other settings as default and click "Create"



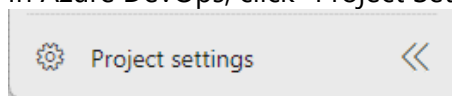
- Once the project is created, it is opened automatically



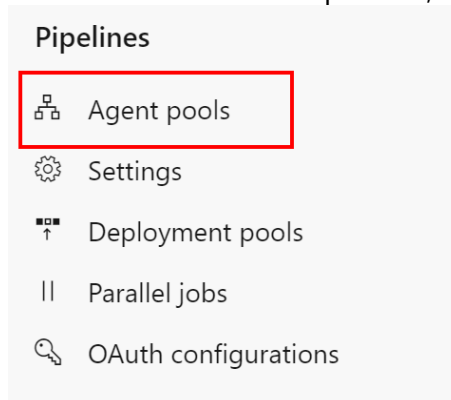
### 4.3.2 Create an Agent Pool in Azure DevOps

The Azure DevOps agents will perform the actual deployment. Each self-hosted agent needs to be placed in its own Agent Pool. In this step, we will create a dedicated Agent Pool for this solution:

- Browse to the main Azure DevOps page
- Create a new Agent Pool
  - In Azure DevOps, click "Project Settings" in the lower left corner



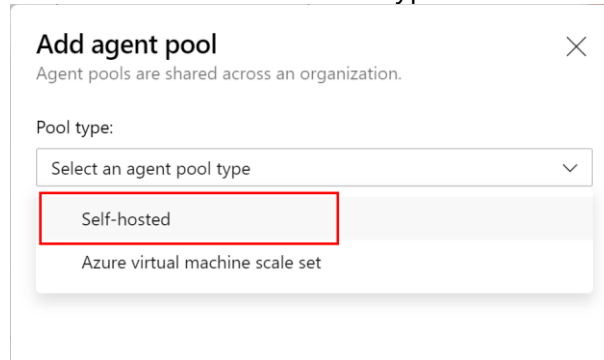
- Scroll down and under "Pipelines", click "Agent Pools"



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



- Select "Self-hosted" as "Pool type"



**Add agent pool** ×

Agent pools are shared across an organization.

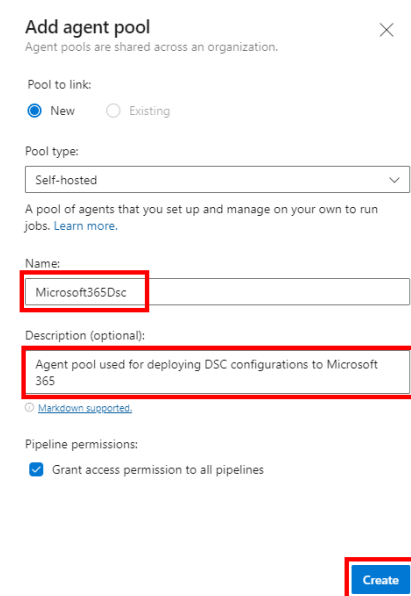
Pool type:

Select an agent pool type

Self-hosted

Azure virtual machine scale set

- Enter a Name (for example: Microsoft365DSC) and Description for the new pool and click "Create"



**Add agent pool** ×

Agent pools are shared across an organization.

Pool to link:

☒ New ☐ Existing

Pool type:

Self-hosted

A pool of agents that you set up and manage on your own to run jobs. [Learn more.](#)

Name:

Microsoft365Dsc

Description (optional):

Agent pool used for deploying DSC configurations to Microsoft 365

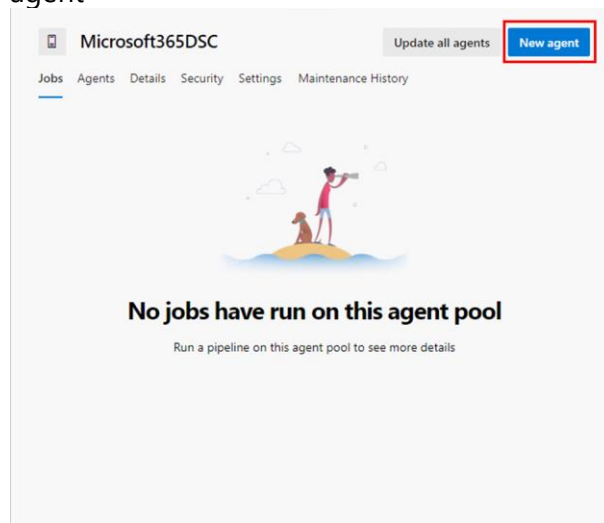
[Markdown supported.](#)

Pipeline permissions:

☒ Grant access permission to all pipelines


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



**Microsoft365DSC** Update all agents New agent

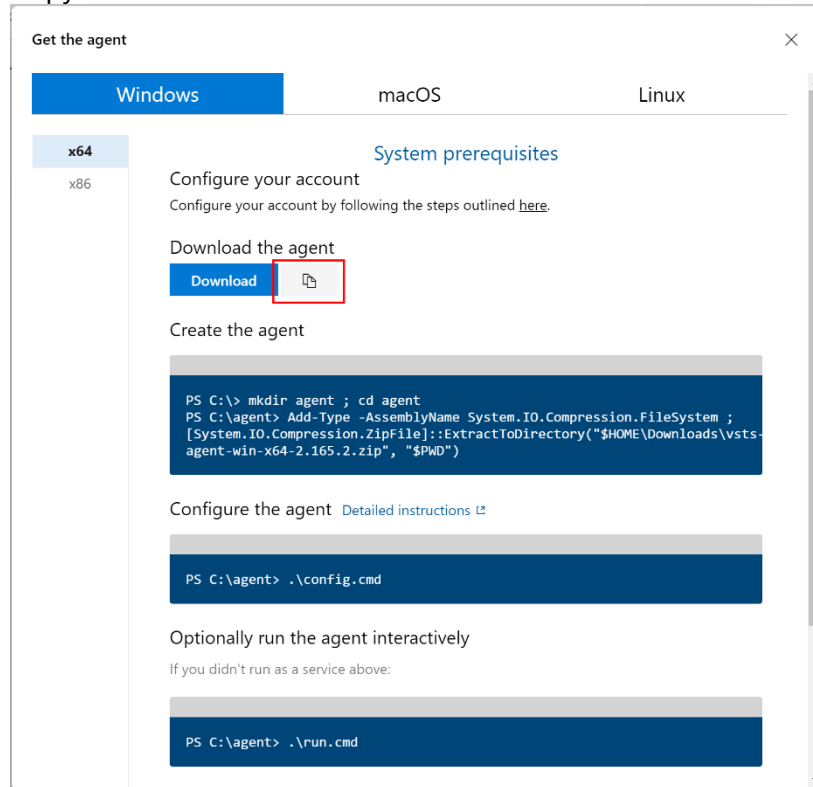
[Jobs](#) [Agents](#) [Details](#) [Security](#) [Settings](#) [Maintenance History](#)



**No jobs have run on this agent pool**

Run a pipeline on this agent pool to see more details

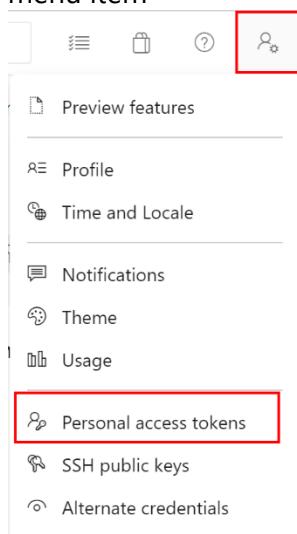
- Copy the download link to be used later in this document



### 4.3.3 Create Personal Access Token

The Azure DevOps agent needs to be able to connect to Azure DevOps with the correct credentials. It is using a Personal Access Token (PAT) to do this. In this step we will create a new PAT to be used by the Azure DevOps agent:

- 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

**Personal Access Tokens**  
These can be used instead of a password for applications like Git or can be passed in the authorization header to access REST APIs

[+ New Token](#) Active

Token name	Status	Organization	Expires on
------------	--------	--------------	------------

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

**Create a new personal access token**

Name

Organization

Expiration (UTC)

Scopes  
Authorize the scope of access associated with this token  
Scopes ☐ Full access ☒ Custom defined

**Release**  
Read, update, and delete releases, release pipelines, and stages  
☐ Read ☐ Read, write, & execute ☐ Read, write, execute, & manage

**Test Management**  
Read, create, and updated test plans, cases, and results  
☐ Read ☐ Read & write

**Packaging**  
Create, read, update, and delete feeds and packages  
☐ Read ☐ Read & write ☐ Read, write, & manage

[Show all scopes \(27 more\)](#)

- Click "Show all scopes", select "Read & manage" under Agent Pools, and click "Create" to create the token

**Create a new personal access token**

Name

Organization

Expiration (UTC)

Scopes  
Authorize the scope of access associated with this token  
Scopes ☐ Full access ☒ Custom defined

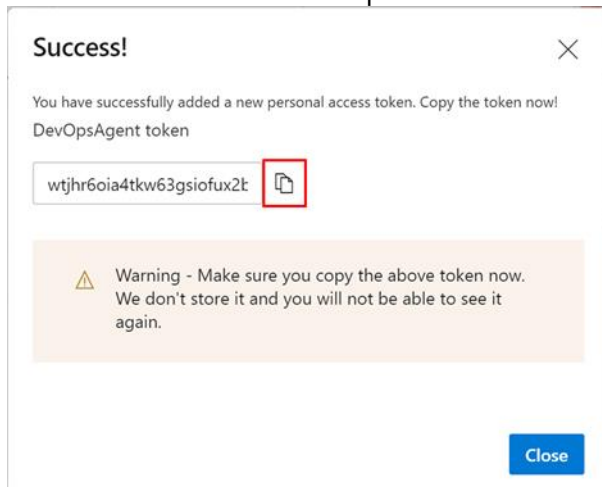
**Agent Pools**  
Manage agent pools and agents  
☒ Read ☒ Read & manage

**Analytics**  
Read data from the analytics service  
☐ Read

**Auditing**  
Read audit log events, manage and delete streams.  
☐ Read Audit Log

[Show less scopes](#)

- **IMPORTANT:** Copy and record 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.

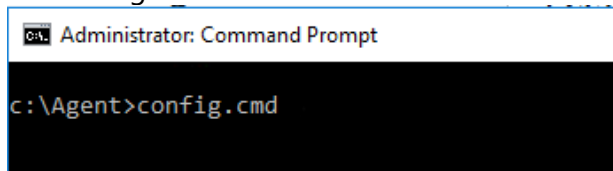


## 4.4 Preparing the Virtual Machine (Phase 2)

### 4.4.1 Installing and configuring the Azure DevOps Agent on the virtual machine

All Azure DevOps agent prerequisites have now been configured. In this step we will install the agent on the virtual machine:

- Connect to your virtual machine with administrative credentials
- Download the Azure DevOps agent using the download link found in the last step of paragraph 4.3.2.
- Create a new folder e.g. `C:\Agent` and extract the downloaded zip to that folder
- 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>](https://dev.azure.com/<org_name>)" and press [Enter]

```
Administrator: Command Prompt - config.cmd

c:\Agent>config.cmd

Azure Pipelines
agent v2.166.3 (commit 87e2e0a)

>> Connect:
Enter server URL > https://dev.azure.com/M365Automation_
```

**NOTE:** The agent will be unable to register if you specify the organization name including the project name ([https://dev.azure.com/<org\\_name>/<project>](https://dev.azure.com/<org_name>/<project>)).

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

```
Administrator: Command Prompt - config.cmd

c:\Agent>config.cmd

Azure Pipelines
agent v2.166.3 (commit 87e2e0a)

>> Connect:
Enter server URL > https://dev.azure.com/M365Automation
Enter authentication type (press enter for PAT) >
```

- Paste the Personal Access Token and press [Enter]

```
Administrator: Command Prompt - config.cmd

c:\Agent>config.cmd

Azure Pipelines
agent v2.166.3 (commit 87e2e0a)

>> Connect:
Enter server URL > https://dev.azure.com/M365Automation
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
```

- Enter "Microsoft365DSC" (or use the name specified earlier) as the Agent Pool and press [Enter]

```
Administrator: Command Prompt - config.cmd

c:\Agent>config.cmd

Azure Pipelines
agent v2.166.3 (commit 87e2e0a)

>> Connect:

Enter server URL > https://dev.azure.com/M365Automation
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) > Microsoft365DSC_
```

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

```
Administrator: Command Prompt - config.cmd

c:\Agent>config.cmd

Azure Pipelines
agent v2.166.3 (commit 87e2e0a)

>> Connect:

Enter server URL > https://dev.azure.com/M365Automation
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) > Microsoft365DSC
Enter agent name (press enter for M365AUTOMATIONV) >
```

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

```
Administrator: Command Prompt - config.cmd

c:\Agent>config.cmd

Azure Pipelines
agent v2.166.3 (commit 87e2e0a)

>> Connect:

Enter server URL > https://dev.azure.com/M365Automation
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) > Microsoft365DSC
Enter agent name (press enter for M365AUTOMATIONV) >
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
```

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

```
Administrator: Command Prompt - config.cmd
c:\Agent>config.cmd

(Azure Pipelines)
agent v2.166.3 (commit 87e2e0a)

>> Connect:

Enter server URL > https://dev.azure.com/M365Automation
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) > Microsoft365DSC
Enter agent name (press enter for M365AUTOMATIONV) >
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2020-04-23 08:39:27Z: Settings Saved.
Enter run agent as service? (Y/N) (press enter for N) > Y
```

- Press Enter to accept the default value for the "SERVICE\_SID\_TYPE\_UNRESTRICTED" setting
- Enter the created service account credentials in paragraph 3.1.2 (use the format ComputerName\AccountName) and press [Enter]

```
Administrator: Command Prompt - config.cmd

>> Connect:

Enter server URL > https://dev.azure.com/M365Automation
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) > Microsoft365DSC
Enter agent name (press enter for M365AUTOMATIONV) >
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2020-04-23 08:39:27Z: Settings Saved.
Enter run agent as service? (Y/N) (press enter for N) > Y
Enter User account to use for the service (press enter for NT AUTHORITY\NETWORK SERVICE) > M365AutomationVM365ConfigAgentSvc
Enter Password for the account M365AutomationVM365ConfigAgentSvc > *****
```

- The agent is being configured. Press Enter to start the service automatically.

```
Administrator: Command Prompt

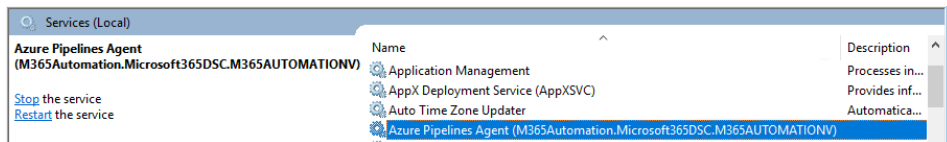
Enter personal access token > *****
Connecting to server ...

>> Register Agent:

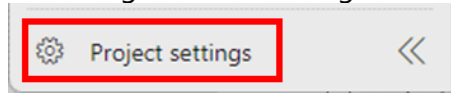
Enter agent pool (press enter for default) > Microsoft365DSC
Enter agent name (press enter for M365AUTOMATIONV) >
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2020-04-23 08:39:27Z: Settings Saved.
Enter run agent as service? (Y/N) (press enter for N) > Y
Enter User account to use for the service (press enter for NT AUTHORITY\NETWORK SERVICE) > M365AutomationVM365ConfigAgentSvc
Enter Password for the account M365AutomationVM365ConfigAgentSvc > *****
Error reported in diagnostic logs. Please examine the log for more details.
- c:\Agent\diag\Agent_20200423-083409-utc.log
Granting file permissions to 'M365AutomationVM365ConfigAgentSvc'.
Service vstsagent.M365Automation.Microsoft365DSC.M365AUTOMATIONV successfully installed
Service vstsagent.M365Automation.Microsoft365DSC.M365AUTOMATIONV successfully set recovery option
Service vstsagent.M365Automation.Microsoft365DSC.M365AUTOMATIONV successfully set to delayed auto start
Service vstsagent.M365Automation.Microsoft365DSC.M365AUTOMATIONV successfully configured
Service vstsagent.M365Automation.Microsoft365DSC.M365AUTOMATIONV started successfully

c:\Agent>
```

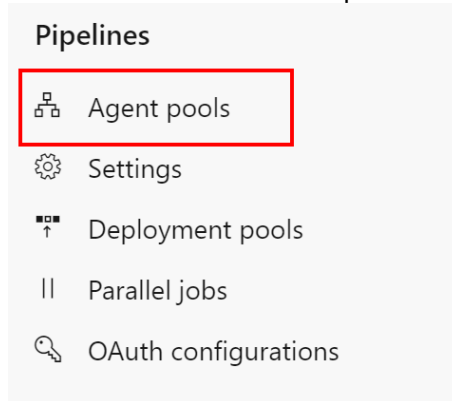
## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps



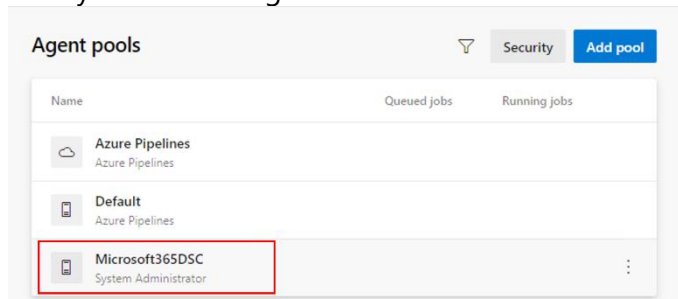
- Verify agent is successfully registered in Azure DevOps
  - Open the Azure DevOps portal
  - Click "Organization Settings" in the lower left corner



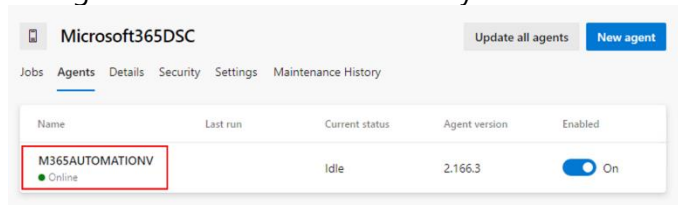
- Scroll down and under "Pipelines", click "Agent Pools"



- Click your custom Agent Pool



- Click "Agents" and validate that your agent is present and Online. The name of the agent will be the host name of your virtual machine:

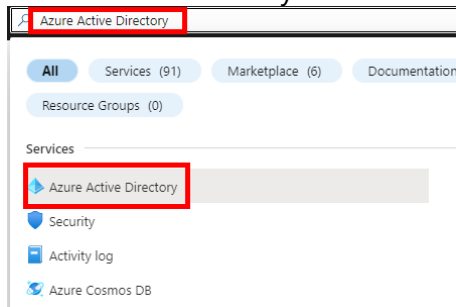


## 4.5 Preparing the Azure Key Vault

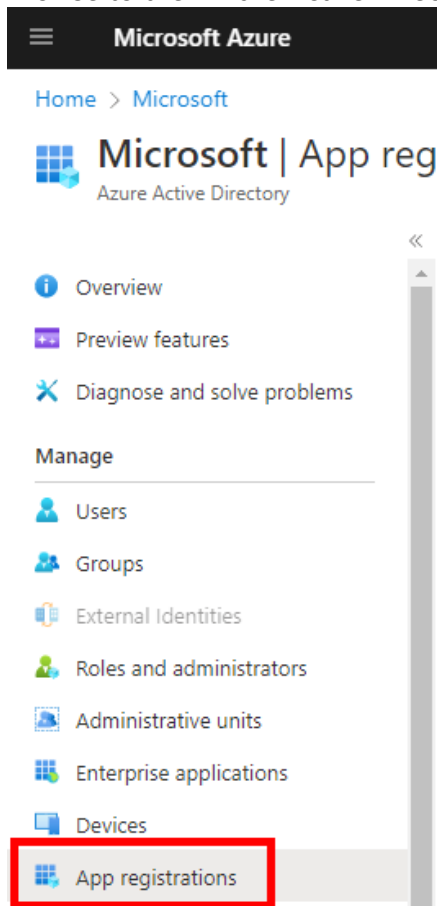
### 4.5.1 Create an App Registration

To allow Azure DevOps to retrieve secrets from Azure Key Vault, an app registration<sup>2</sup> is needed. In this step we are going to create this app registration:

- Log into the Azure Portal
- In the search box at the top of the page, type "Azure Active Directory" and click the Azure Active Directory icon that is found



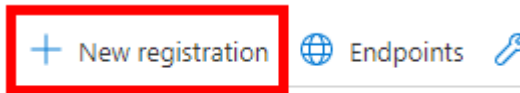
- Browse to the "Azure Active Directory" and select the "App Registrations" section



<sup>2</sup> Use App Registration nr 2. See paragraph 2.3 for more information.

## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

- To create a new App Registration, click "New registration"



- Enter "Microsoft365DSC\_DevOpsPipeline" as the Name (or use your own name) and click "Register"

Dashboard > Contoso | App registrations >

### Register an application

**Name**  
The user-facing display name for this application (this can be changed later).

Microsoft365DSC\_DevOpsPipeline ✓

**Supported account types**  
Who can use this application or access this API?

☒ Accounts in this organizational directory only (Contoso only - Single tenant)  
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)  
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)  
☐ Personal Microsoft accounts only

[Help me choose...](#)

**Redirect URI (optional)**  
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform:

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applications](#).

By proceeding, you agree to the [Microsoft Platform Policies](#)

**Register**

- Once the App Registration has been created, make sure you save the "Application (client) ID" and "Directory (tenant) ID" on the page that appears.

Delete Endpoints Preview features

#### Essentials

Display name	: <a href="#">Microsoft365DSC_DevOpsPipeline</a>	Client credentials	: <a href="#">0 certificate, 1 secret</a>
Application (client) ID	: <a href="#">42e94ab0-f18d-43cc-83a4-2459c77ac958</a>	Redirect URIs	: <a href="#">Add a Redirect URI</a>
Object ID	: 7c0f7341-1534-4627-882e-1272bdca8898	Application ID URI	: <a href="#">Add an Application ID URI</a>
Directory (tenant) ID	: <a href="#">16b3c013-d300-468d-ac64-7eda0820b6d3</a>	Managed application in L...	: <a href="#">Microsoft365DSC_DevOpsPipeline</a>
Supported account types	: <a href="#">My organization only</a>		

- Click the "Add a certificate or secret" link in the right column

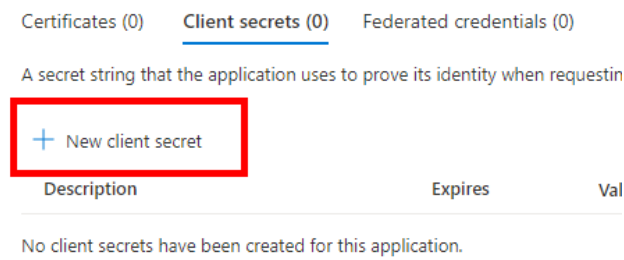
Delete Endpoints Preview features

#### Essentials

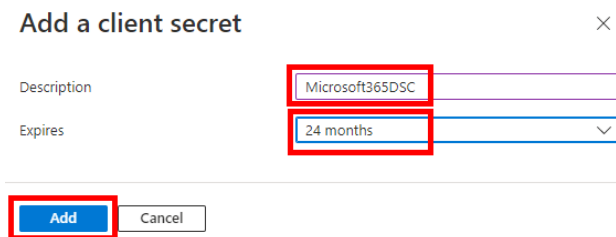
Display name	: <a href="#">Microsoft365DSC_DevOpsPipeline</a>	Client credentials	: <a href="#">Add a certificate or secret</a>
Application (client) ID	: 42e94ab0-f18d-43cc-83a4-2459c77ac958	Redirect URIs	: <a href="#">Add a Redirect URI</a>
Object ID	: 7c0f7341-1534-4627-882e-1272bdca8898	Application ID URI	: <a href="#">Add an Application ID URI</a>
Directory (tenant) ID	: 16b3c013-d300-468d-ac64-7eda0820b6d3	Managed application in L...	: <a href="#">Microsoft365DSC_DevOpsPipeline</a>
Supported account types	: <a href="#">My organization only</a>		



- Click the “New client secret” link to create a new secret



- Enter “Microsoft365DSC” as the Description (or use your own value), select “24 months” as Expires and click “Add”



- Copy and document the value of the created secret in a secure place.

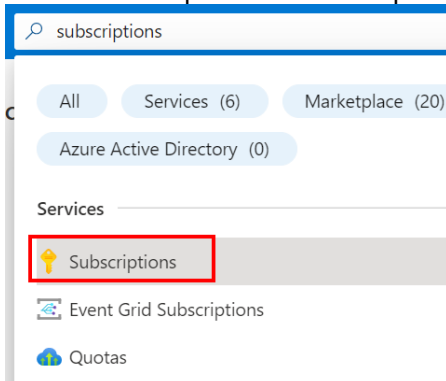
**Note:** This value will only be shown once!

Description	Expires	Value	Secret ID
Microsoft365DSC	8/10/2024		2f636980-bbf5-4312-a43f-bc7438e3ea1d

4.5.2 Granting the App Registration Reader permissions to the Azure Subscription

The new app registration<sup>3</sup> needs to connect to Azure Key Vault. However, it also needs Reader permissions to the Azure subscription. In this step we will configure these permissions:

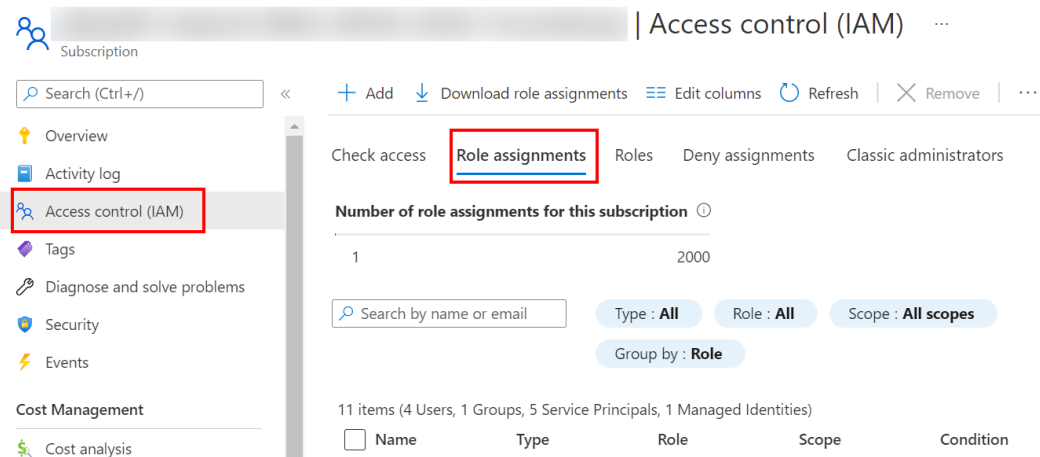
- Go back to the Azure Portal home page
- Enter “Subscriptions” in the top search bar and select “Subscriptions”



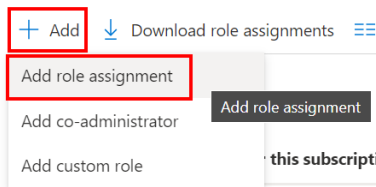
- Select the subscription in which the Key Vault will be created
- In the “Subscriptions” view, click “Access Control (IAM)”

<sup>3</sup> Use App Registration nr 2. See paragraph 2.3 for more information.

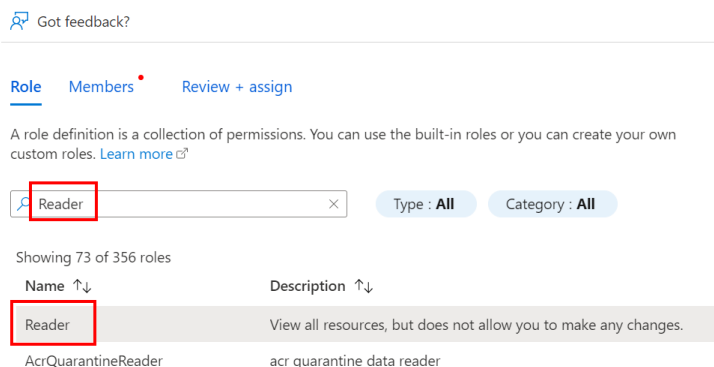
- Then select the "Role assignments" tab



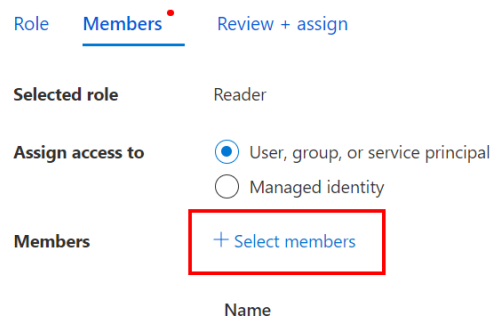
- Click the "Add" button and then click "Add role assignment"



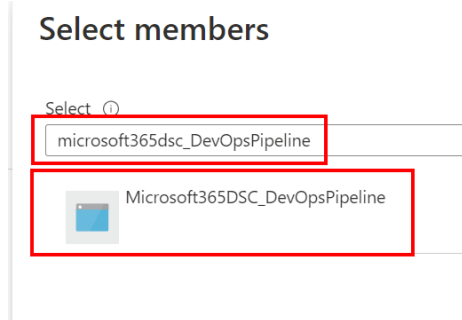
- Type "Reader" in the search bar and select the "Reader" role



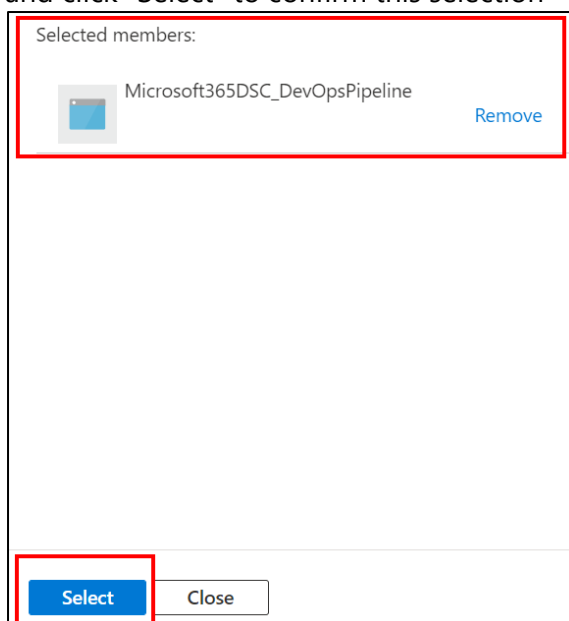
- Click "Next" to proceed to selecting the Members
- Click the "Select members" button



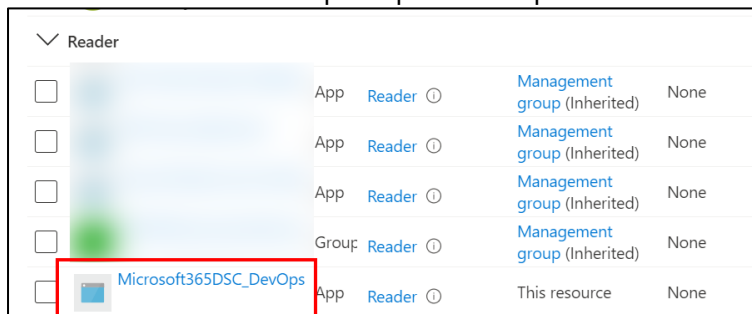
- Enter "Microsoft365DSC\_DevOpsPipeline" in the Select bar and click the found member



- Validate that the app registration has been moved to the "Selected members" section and click "Select" to confirm this selection



- Click the "Review + assign" button on the bottom of the screen to review the selections.
- Click the "Review + assign" button again to assign the permissions.
- Validate that the service principal is now present in the Readers group



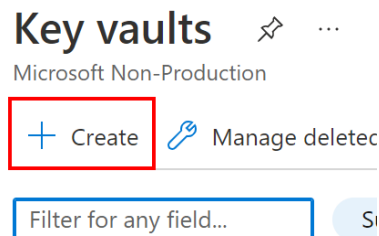
### 4.5.3 Create a new Azure Key Vault

The solution needs an Azure Key Vault to which the app registration<sup>4</sup> is granted permissions. In this step we will create a new Key Vault and grant the app registration the required permissions:

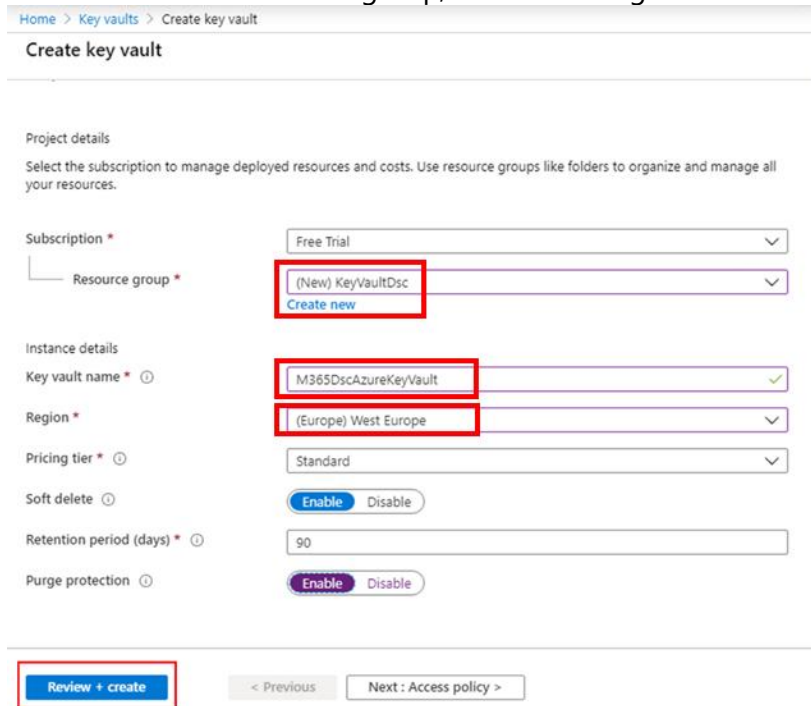
- Log into the Azure Portal
- Enter "Key vault" in the top search bar and select "Key vaults"



- Click "Create" to create a new Key Vault

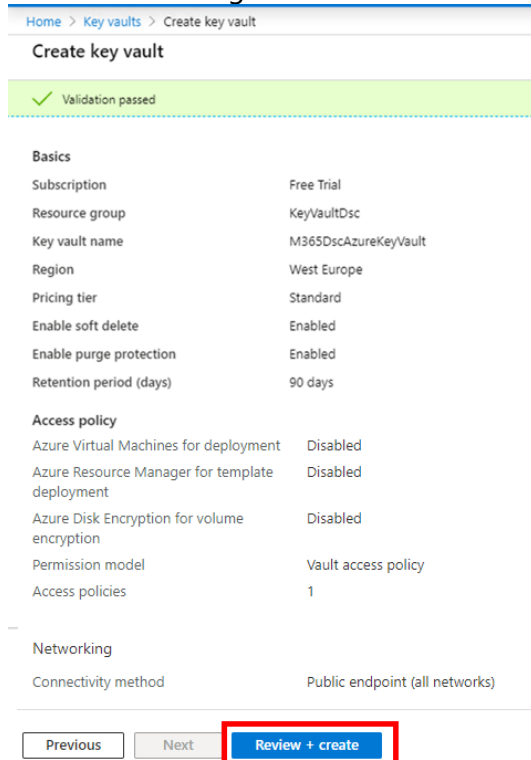


- Enter the desired Resource group, Name and Region and then click "Review + create"

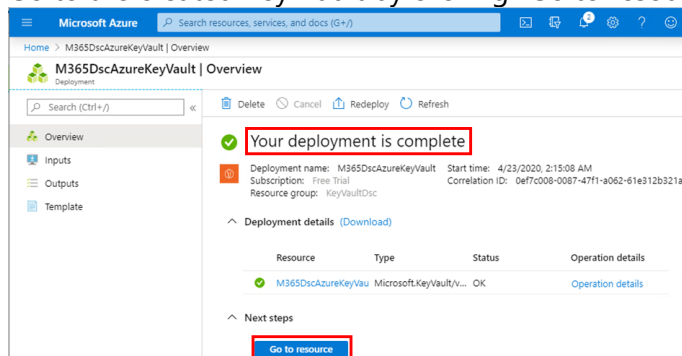
A screenshot of the 'Create key vault' form in the Azure Portal. The form is titled 'Create key vault' and has a breadcrumb trail: 'Home > Key vaults > Create key vault'. The form is divided into sections: 'Project details', 'Instance details', and 'Soft delete'. In the 'Project details' section, there are dropdown menus for 'Subscription' (set to 'Free Trial') and 'Resource group' (set to '(New) KeyVaultDsc' with a 'Create new' link below it). In the 'Instance details' section, there are dropdown menus for 'Key vault name' (set to 'M365DscAzureKeyVault'), 'Region' (set to '(Europe) West Europe'), and 'Pricing tier' (set to 'Standard'). There are also checkboxes for 'Soft delete' (set to 'Enable') and 'Purge protection' (set to 'Enable'). At the bottom of the form, there is a 'Review + create' button highlighted with a red box, and two other buttons: '< Previous' and 'Next: Access policy >'. The 'Review + create' button is a blue button with white text.

<sup>4</sup> Use App Registration nr 2. See paragraph 2.3 for more information.

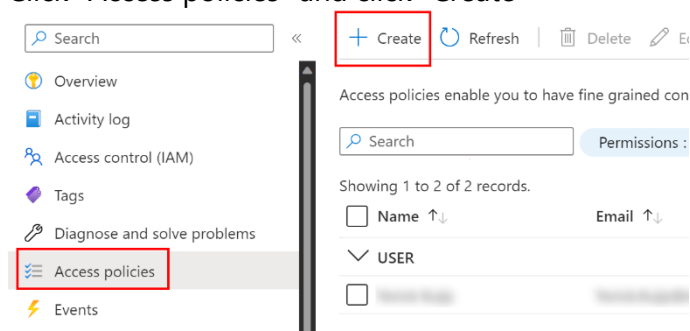
- Review the settings and click "Review + create" to create the Key Vault



- The Key Vault will be created
- Go to the created Key Vault by clicking "Go to resource"



- Click "Access policies" and click "Create"



- Select the "Get" and "List" permissions from "Secret permissions" and "Certificate permissions":

1 Permissions 2 Principal 3 Application (optional) 4 Review + create

Configure from a template  
Select a template

Key permissions	Secret permissions	Certificate permissions
Key Management Operations	Secret Management Operations	Certificate Management Operations
<input type="checkbox"/> Select all	<input type="checkbox"/> Select all	<input type="checkbox"/> Select all
<input type="checkbox"/> Get	<input checked="" type="checkbox"/> Get	<input checked="" type="checkbox"/> Get
<input type="checkbox"/> List	<input checked="" type="checkbox"/> List	<input checked="" type="checkbox"/> List
<input type="checkbox"/> Update	<input type="checkbox"/> Set	<input type="checkbox"/> Update
<input type="checkbox"/> Create	<input type="checkbox"/> Delete	<input type="checkbox"/> Create
<input type="checkbox"/> Import	<input type="checkbox"/> Recover	<input type="checkbox"/> Import
<input type="checkbox"/> Delete	<input type="checkbox"/> Backup	<input type="checkbox"/> Delete
<input type="checkbox"/> Recover	<input type="checkbox"/> Restore	<input type="checkbox"/> Recover

- Click Next, then select the "Select principal" option, enter the Service Principal Name you created in the previous paragraph (default "Microsoft365DSC\_DevOpsPipeline") in the search box on the right, select your principal and click "Select".

✔ Permissions ✔ Principal 3 Application (optional) 4 Review + create

Only 1 principal can be assigned per access policy.  
Use the new embedded experience to select a principal. The previous popup experience can be accessed here. [Select a principal](#)

Microsoft365DSC\_

Microsoft365DSC\_DevOpsPipeline  
8961e023-c915-43a9-a625-eb94e127d70c

- Click Next to skip through the Application (optional) screen, validate that everything is configured correctly and click "Create"

✓ Permissions   ✓ Principal   ✓ Application (optional)   4 Review + create

---

**Key Permissions**

Key Management Operations	None selected
Cryptographic Operations	None selected
Privileged Key Operations	None selected
Rotation Policy Operations	None selected

**Secret Permissions**

Secret Management Operations	None selected
Privileged Secret Operations	None selected

**Certificate Permissions**

Certificate Management Operations	None selected
Privileged Certificate Operations	None selected

**Principal**

Principal name	Microsoft365DSC_DevelopmentPipeline
Object ID	fa1360ae-a384-441f-88d8-53bd5307a393

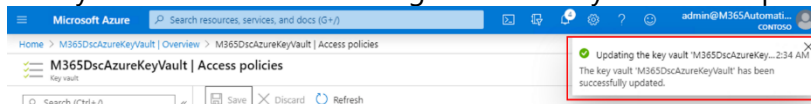
**Application**

Authorized application ⓘ	None selected
Object ID	None selected

---

Previous **Create**

- Next you should see the message that the Key Vault was updated successfully

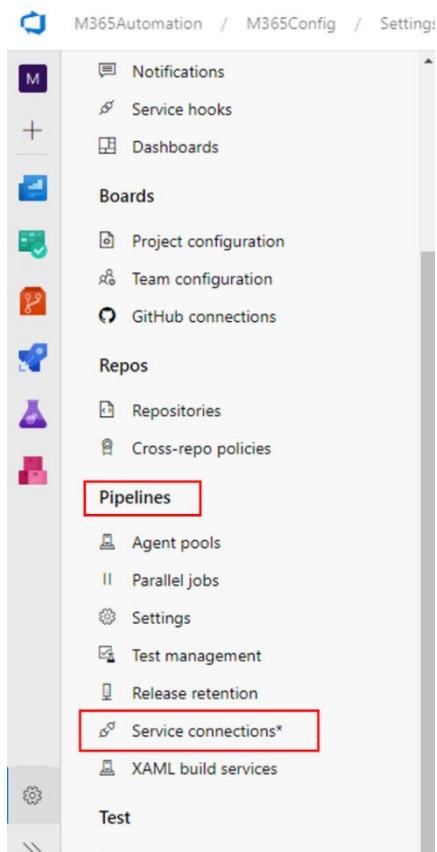


#### 4.5.4 Adding a Service Connection to Azure to the Azure DevOps project

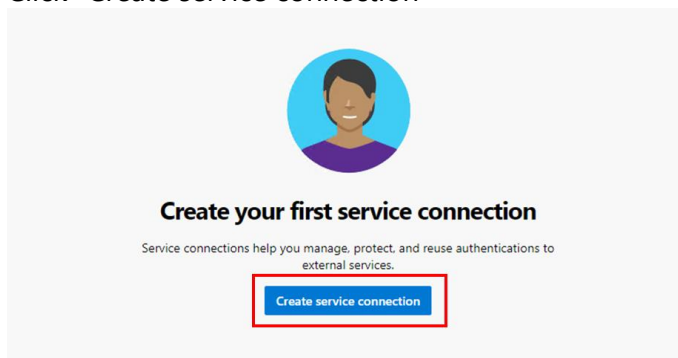
Now that the Key Vault has been created, a service connection can be created in Azure DevOps. DevOps will use this service connection to connect to Azure Key Vault. In this step we will create a new service connection in Azure DevOps:

- 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\*"

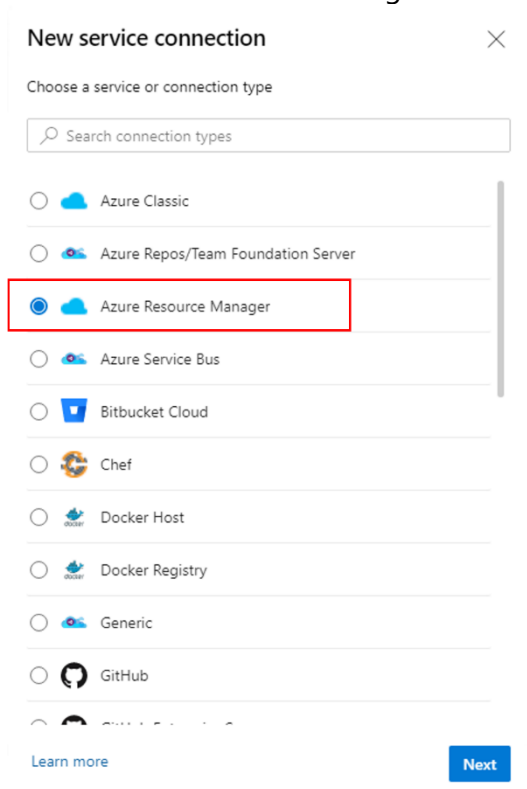


- Click "Create service connection"

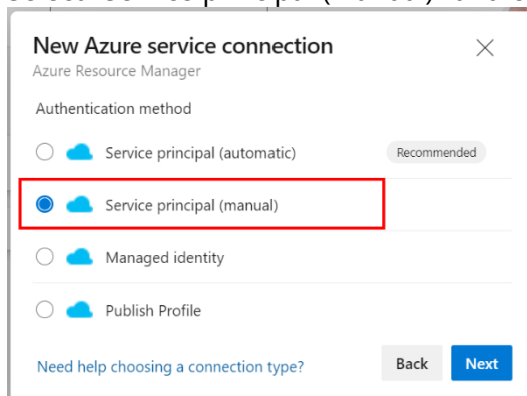




- Select "Azure Resource Manager" and click "Next"



- Select "Service principal (manual)" and click "Next"



NOTE: We are using the already created Service Principal Name

- Enter the information you documented when creating the App Registration<sup>5</sup> (paragraph 4.5.1):
  - a. Enter the GUID of the subscription in which the Key Vault was created as the "Subscription Id"
  - b. Enter the name of the subscription in which the Key Vault was created as the "Subscription Name"
  - c. Enter the documented value "Application (client) ID" as the "Service principal client ID"
  - d. Enter the documented value "Secret" as the "Service principal key"
  - e. Enter the documented value "Directory (tenant) ID" as the "Tenant ID" (potentially already populated)
  - f. Enter a "Service connection name", for example "KeyVaultConnection"

New Azure service connection  
Azure Resource Manager using service principal (manual)

Environment  
Azure Cloud

Scope Level  
☒ Subscription  
☐ Management Group  
☐ Machine Learning Workspace

Subscription Id  
c37373a5-2499-4b46-9504-4e5486e1ff19  
Subscription ID from the publish settings file

Subscription Name  
Free Trial  
Subscription Name from the publish settings file

Authentication  
Service Principal Id  
b92bf63-7726-47ca-80b7-a0a7c3bffd50  
Client ID for connecting to the endpoint. Refer to Azure Service Principal link on how to create Azure Service Principal.

Credential  
☒ Service principal key ☐ Certificate

Service principal key  
\*\*\*\*\*  
Service Principal key for connecting to the endpoint. Refer to Azure Service Principal link on how to create Azure Service Principal. Ignore this field if the authentication type is spn/certificate.

Tenant ID  
323dabdf-4865-47f1-9d17-f0399521de7d  
Tenant ID for connecting to the endpoint. Refer to Azure Service Principal link on how to create Azure Service Principal.

Verify

Details  
Service connection name  
KeyVaultConnection

Description (optional)

[Learn more](#)  
[Troubleshoot](#)

Back Verify and save

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

**Note:** If you get an Access Denied error, check if you have added the App Registration to the Reader role in your Azure Subscription.

<sup>5</sup> Use App Registration nr 2. See paragraph 2.3 for more information.


- Make sure the "Grant access permission to all pipelines" is **not** checked and click "Verify and save"

Service connection name

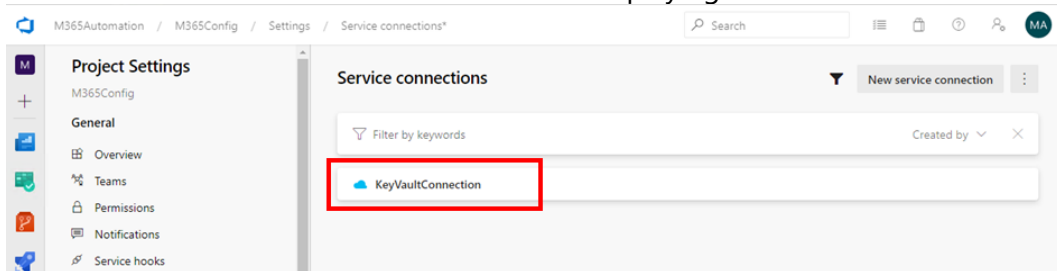
Description (optional)

Security

☐ Grant access permission to all pipelines

[Learn more](#) [Troubleshoot](#) [Back](#) [Verify and save](#) 

- The "Service connection" is now created and displaying



## 5 Configuring Azure DevOps

Now that all prerequisites have been created, we can fully configure the solution in Azure DevOps.

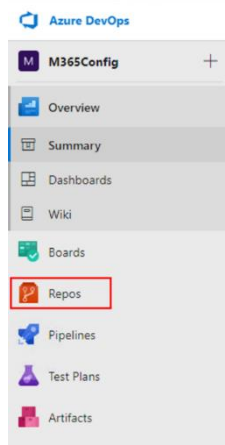
### 5.1 Populate scripts

The newly created Azure DevOps project contains a Git repository to which all scripts of this solution must be added. In this step we will upload the scripts of the solution to the repository in Azure DevOps:

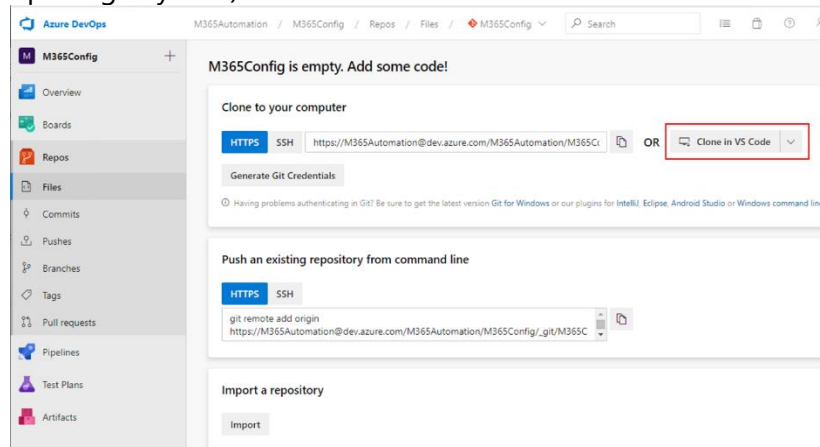
- Download and install Visual Studio Code from <https://code.visualstudio.com>
- Download and install Git from <https://git-scm.com>
  - Download the most recent version of Git by clicking the "Download" button



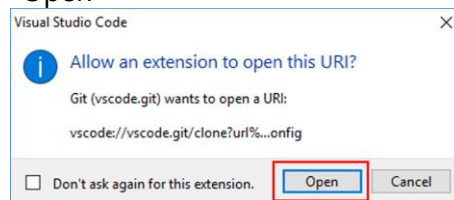
- Run the downloaded installer and use the default settings
- Download the DSC scripts from <https://aka.ms/M365DSCWhitepaper/Scripts>
  - This package contains several scripts, check chapter 8 "Script details" for more details.
- Upload the scripts to the DevOps repository
  - Open Azure DevOps Portal and browse to your project
  - Click the "Repos" icon in the left menu



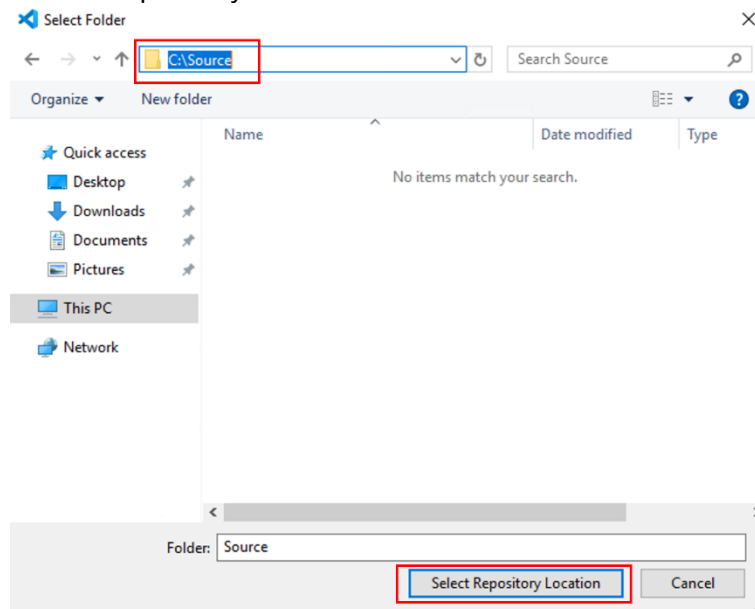
- Click on "Clone in VS Code" (acknowledge any browser notifications for opening any files)



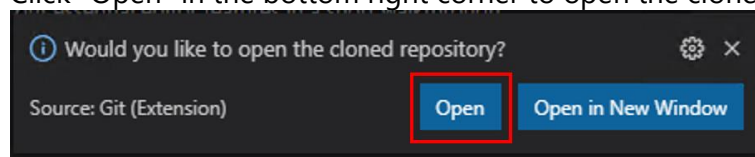
- Acknowledge that Visual Studio Code can open the external URL by clicking "Open"



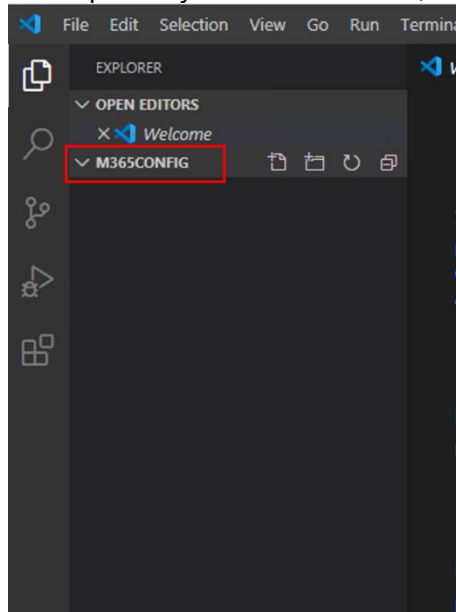
- Select "C:\Source" as the source folder (create if it does not exist) and select "Select Repository Location"



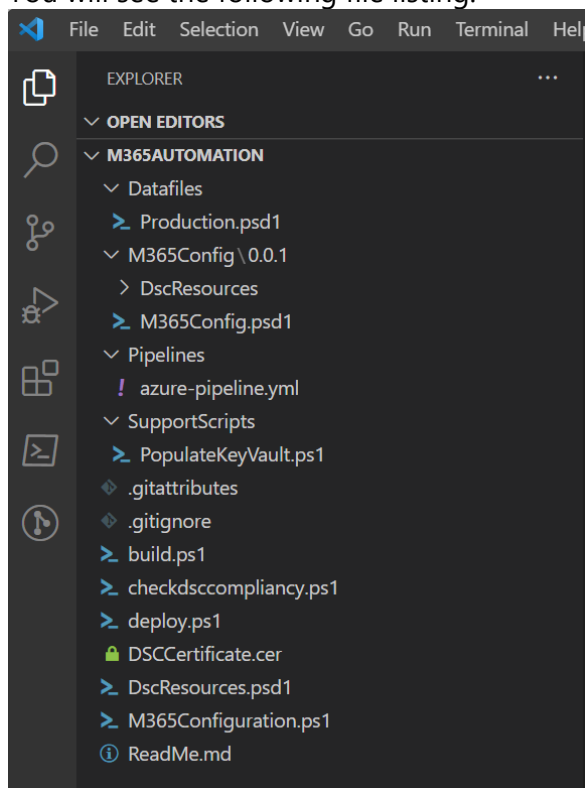
- Login with your Microsoft 365 admin account
- Click "Open" in the bottom right corner to open the cloned folder



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



- Open Windows 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 4.1.4 to the folder
- You will see the following file listing:



- Open the file "Datafiles\Production.psd1" and update:
  - The placeholder "<M365AdminAccount>" to the user principal name of the DSC admin user we created in paragraph 4.2.1
  - The placeholder "<appid>" to the application id created earlier in paragraph 4.2.2
  - The placeholder "<certthumb>" to the thumbprint of the Microsoft 365 authentication certificate created in paragraph 4.1.3
  - The placeholder "<tenantURL>" to the URL of your tenant, like "contoso.onmicrosoft.com"

```

10 NonNodeData = @{
11     Environment = @{
12         Name = 'Production'
13         ShortName = 'PRD'
14         TenantId = '<tenantURL>'
15         OrganizationName = '<tenantURL>'
16     }
17     Accounts = @(
18         @{
19             Workload = 'Exchange'
20             Account = '<M365AdminAccount>'
21         }
22         @{
23             Workload = 'Office365'
24             Account = '<M365AdminAccount>'
25         }
26         @{
27             Workload = 'PowerPlatform'
28             Account = '<M365AdminAccount>'
29         }
30         @{
31             Workload = 'SecurityCompliance'
32             Account = '<M365AdminAccount>'
33         }
34         @{
35             Workload = 'SharePoint'
36             Account = '<M365AdminAccount>'
37         }
38         @{
39             Workload = 'Teams'
40             Account = '<M365AdminAccount>'
41         }
42     )
43     AppCredentials = @(
44         @{
45             Workload = 'Exchange'
46             ApplicationId = '<appid>'
47             CertThumbprint = '<certthumb>'
48         }
49         @{
50             Workload = 'Office365'
51             ApplicationId = '<appid>'
52             CertThumbprint = '<certthumb>'
53         }
54         @{
55             Workload = 'PowerPlatform'
56             ApplicationId = '<appid>'
57             CertThumbprint = '<certthumb>'
58         }
59         @{
60             Workload = 'SecurityCompliance'
61             ApplicationId = '<appid>'
62             CertThumbprint = '<certthumb>'
63         }
64         @{
65             Workload = 'SharePoint'
66             ApplicationId = '<appid>'
67             CertThumbprint = '<certthumb>'
68         }
69         @{
70             Workload = 'Teams'
71             ApplicationId = '<appid>'
72             CertThumbprint = '<certthumb>'

```

```
96 AcceptedDomains = @(
97     @{
98         Identity = '<tenantURL>'
99         DomainType = 'Authoritative'
100         MatchSubDomains = $false
101         OutboundOnly = $false
102         Ensure = 'Present'
103     }
104 )
105 DKIM = @(
106     @{
107         Identity = '<tenantURL>'
108         Enabled = $true
109         AdminDisplayName = ''
110         BodyCanonicalization = 'Relaxed'
111         HeaderCanonicalization = 'Relaxed'
112         KeySize = 1024
113     }
114 )
```

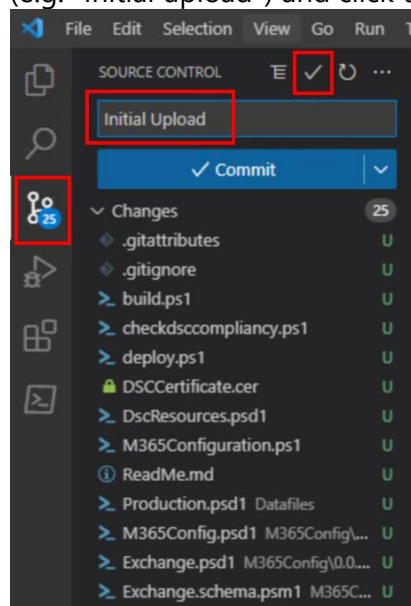
- Open the file "build.ps1" and update the \$VaultName variable on line 27 with the name of the Azure Key Vault you have created in paragraph 4.5.3

```
23 ##### SCRIPT VARIABLES #####
24 $dscScriptName = 'M365Configuration.ps1'
25
26 # Azure variables
27 $VaultName = '<Your KeyVault>'
28
29 ##### START SCRIPT #####
30
31 Write-Log -Message '*****'
```

- Open the file "deploy.ps1" and update the \$VaultName variable on line 33 with the name of the Azure Key Vault you have created in paragraph 4.5.3

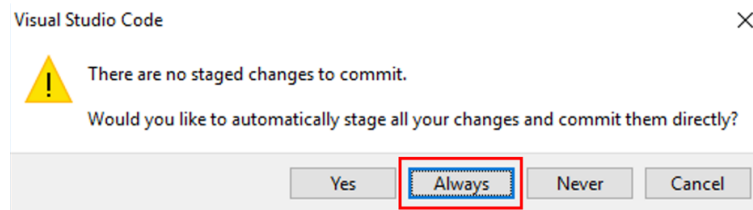
```
30 ##### SCRIPT VARIABLES #####
31
32 # Azure variables
33 $VaultName = '<Your KeyVault>'
34
```

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



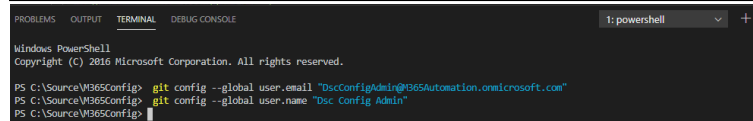


- Click "Always" if you get the message that there are no staged changes to commit.

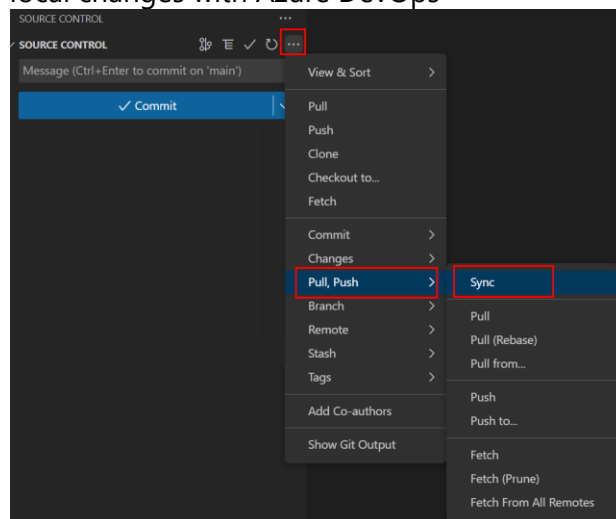


- If you get an error about an unknown e-mail address, run the following commands with your own information and retry the commit:

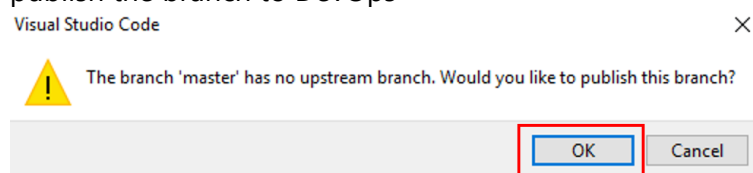
```
git config --global user.email <email>
git config --global user.name "<your_name>"
```



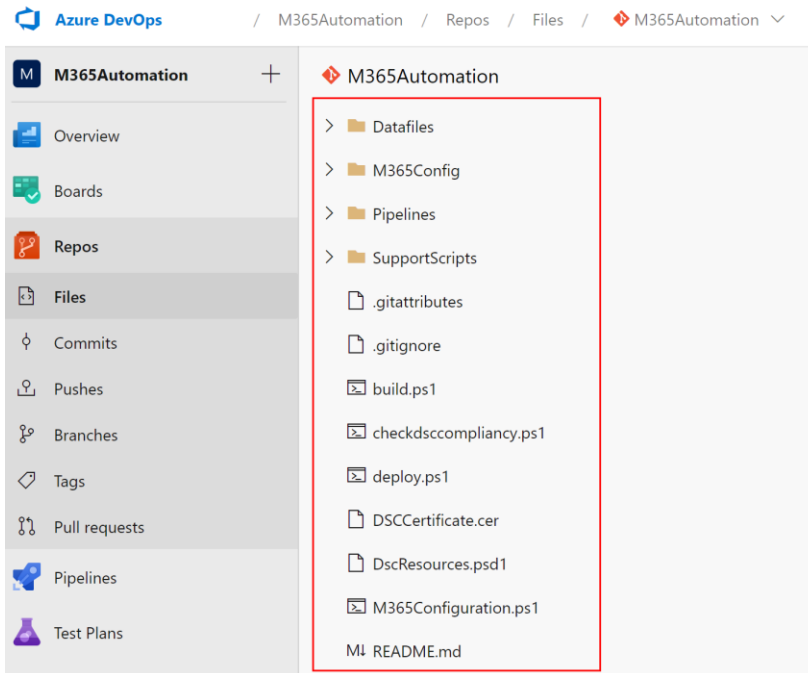
- Click the three dots icon and select "Push, Pull > Sync" to synchronize your local changes with Azure DevOps



- 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



### 5.2 Add secrets to your Key Vault

All the secrets and certificates used by the solution need to be added to the Azure Key Vault. The solution contains a script that simplifies this process. It reads all used accounts and certificates from the PowerShell data file you updated in the previous step (DataFiles\Production.ps1) and asks for the corresponding passwords. It then adds these to Azure Key Vault, using a specific naming standard.

In this step we are going to use this script to populate all required Key Vault items:

Run the provided script and follow the instructions:

- Log on / connect to the machine where you cloned your repository
- Open an elevated Windows PowerShell window
- Switch locations to the folder C:\Source\M365Automation\SupportScripts
- Run the following commands:

```
Install-Module Az.KeyVault
.\PopulateKeyVault.ps1 -VaultName <name_of_your_keyvault>
-DataFile Production
```

**Note:** Currently this solution contains just one data file: Production.ps1. You can extend the solution with additional environments, like Test and Acceptance. See paragraph 1.2 for more information.

When you do, you can simply use the name of the added data files for the DataFile parameter of the script.

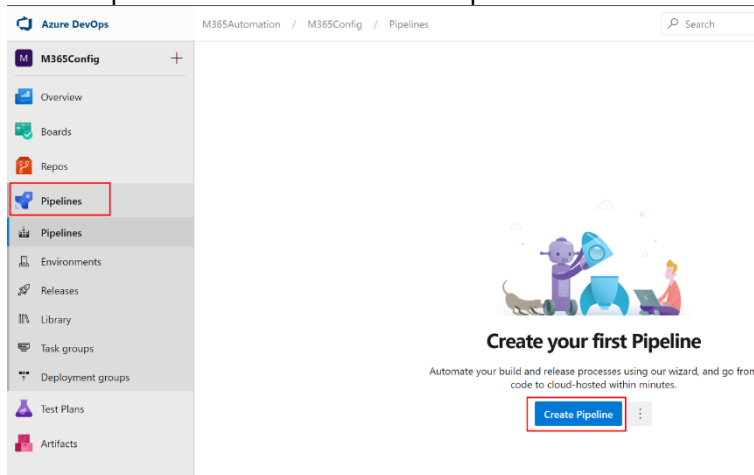
- This script will now read the data file and ask for all passwords and certificates it finds. If a secret is already present in the Key Vault, you are asked if you want to overwrite it or not.

### 5.3 Configure Azure DevOps project

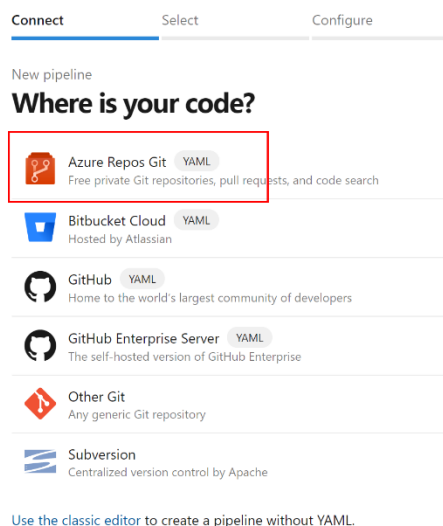
#### 5.3.1 Create Build pipeline

The Build pipeline will compile the DSC configurations into MOF files and create a deployment package. In this step we will create a new Build pipeline:

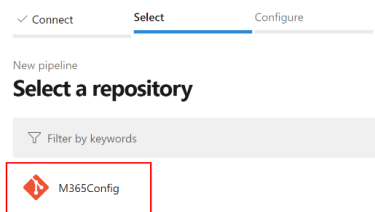
- Browse to the Azure DevOps Portal
- Click "Pipelines" and click "Create Pipeline"



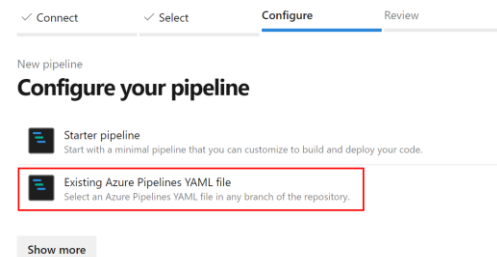
- Select "Azure Repos Git"



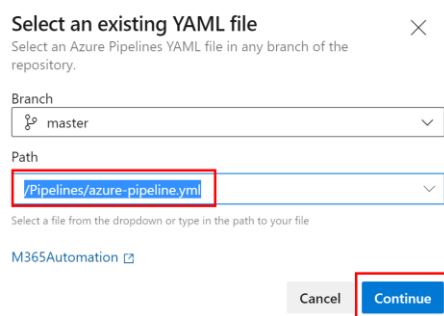
- Click the name of your Project



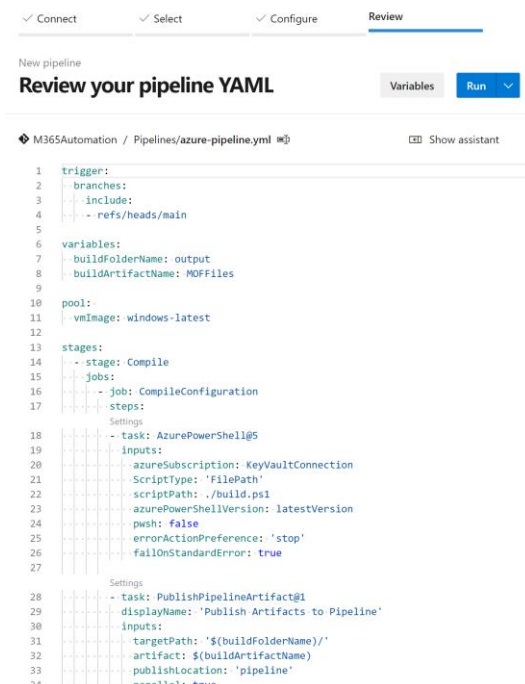
- Select the "Existing Azure Pipelines YAML file"



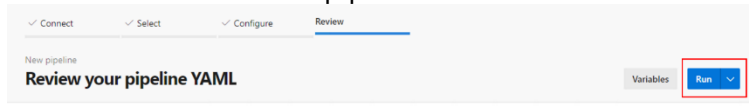
- Select the file "azure-pipelines.yml" and click "Continue"



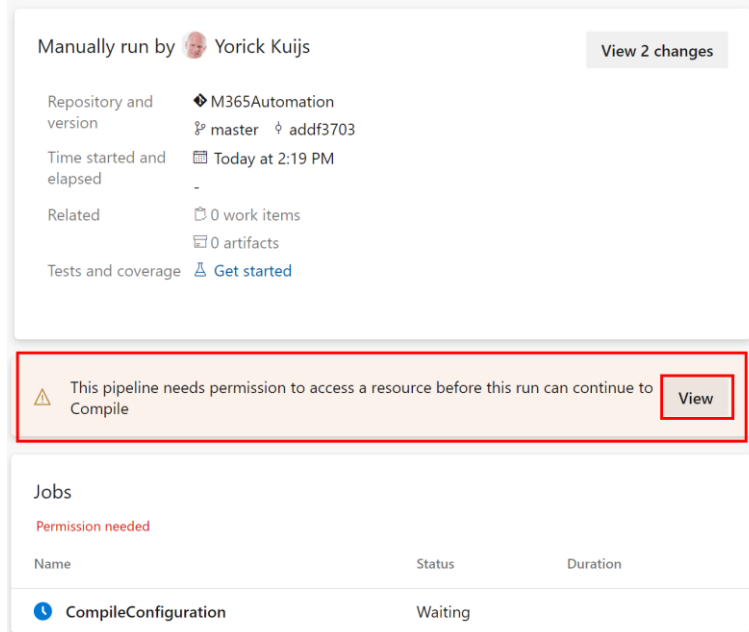
- The pipeline then shows you the azure-pipelines.yml file you uploaded in a previous step



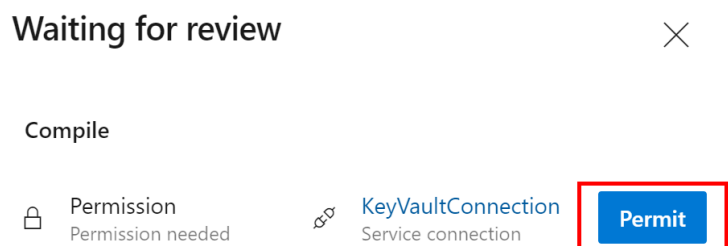
- Select "Run" to start the pipeline



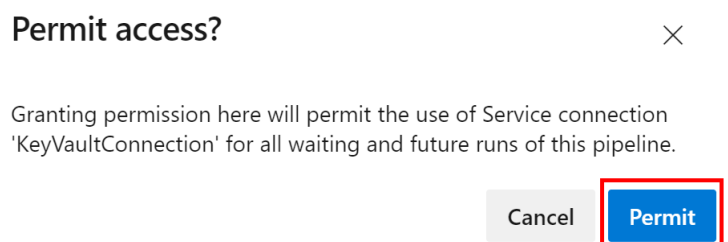
- The pipeline is created and started. On the page that is opened, you see the details of that pipeline run. However, it will not yet start.
- If you wait a couple of seconds, the page is refreshed, and you can see that you first need to provide permissions on the Service Connection. Click on the "View" button



- Click on the "Permit" button to provide permissions to the "KeyVaultConnection" Service Connection

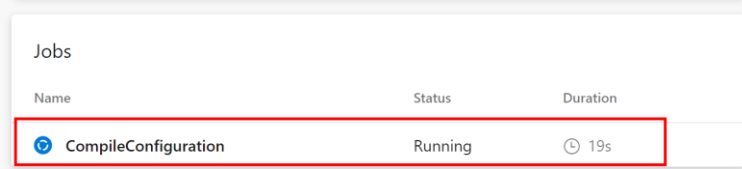


- On the dialog that appears, click "Permit" once more



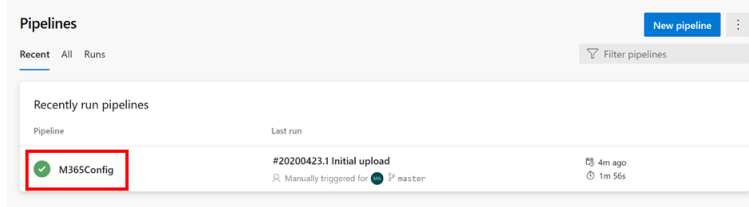
## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

- After a couple of seconds, the pipeline will start running and the job is executed



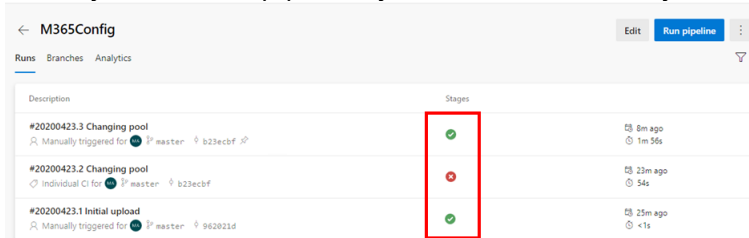
Name	Status	Duration
CompileConfiguration	Running	19s

- Check if the pipeline has completed successfully



Pipeline	Last run
M365Config	#20200423.1 Initial upload Manually triggered for master

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



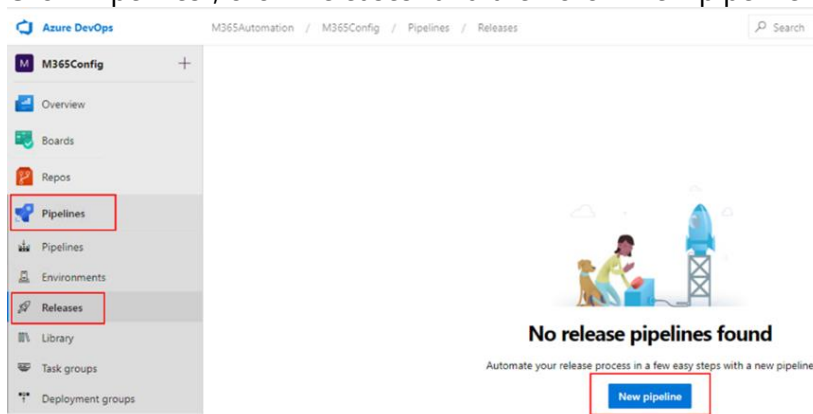
Description	Stages	Time
#20200423.3 Changing pool Manually triggered for master	✓	8m ago 1m 56s
#20200423.2 Changing pool Individual CI for master	✗	23m ago 54s
#20200423.1 Initial upload Manually triggered for master	✓	25m ago +1s

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

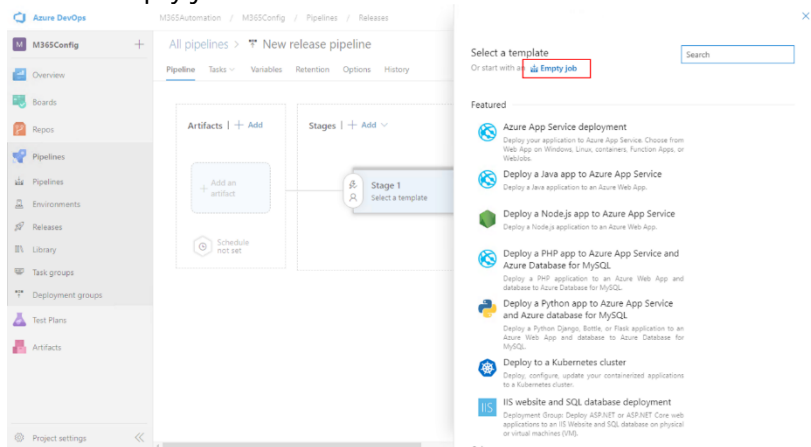
### 5.3.2 Create the Deployment Release pipeline

The solution uses a release pipeline to deploy new configurations to the target environments. Currently we are only deploying to one environment (Production), but you can easily extend the number of environments by adding a stage for each environment. In this step we are going to configure this release pipeline:

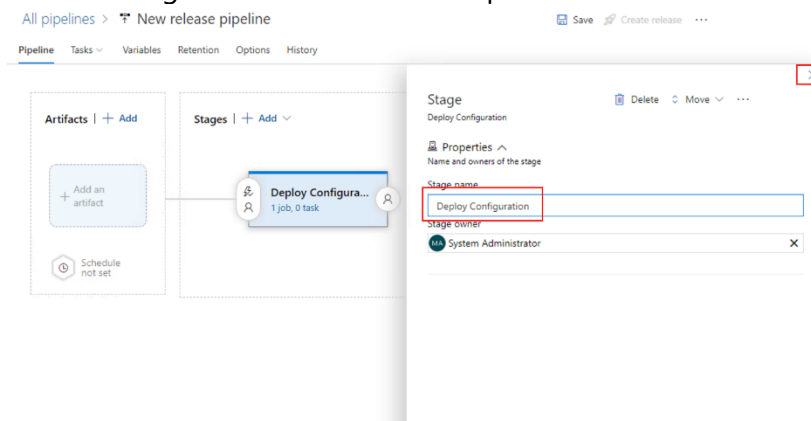
- Go to the Azure DevOps Portal
- Click "Pipelines", click "Releases" and then click "New pipeline"



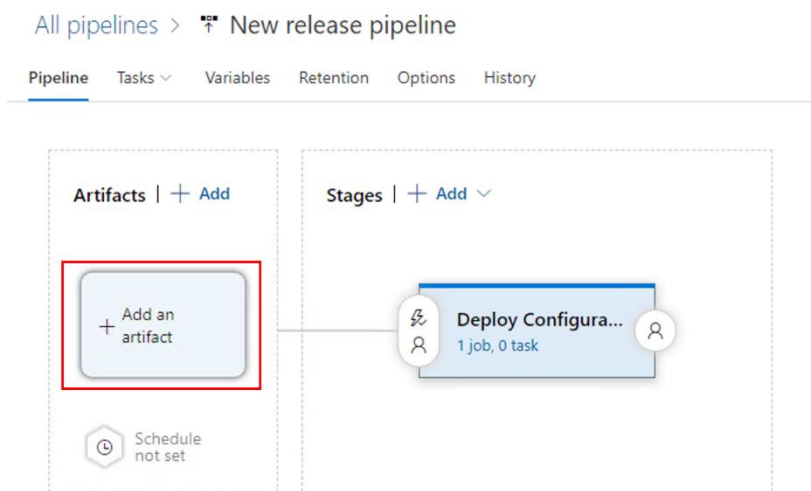
- Select "Empty job"



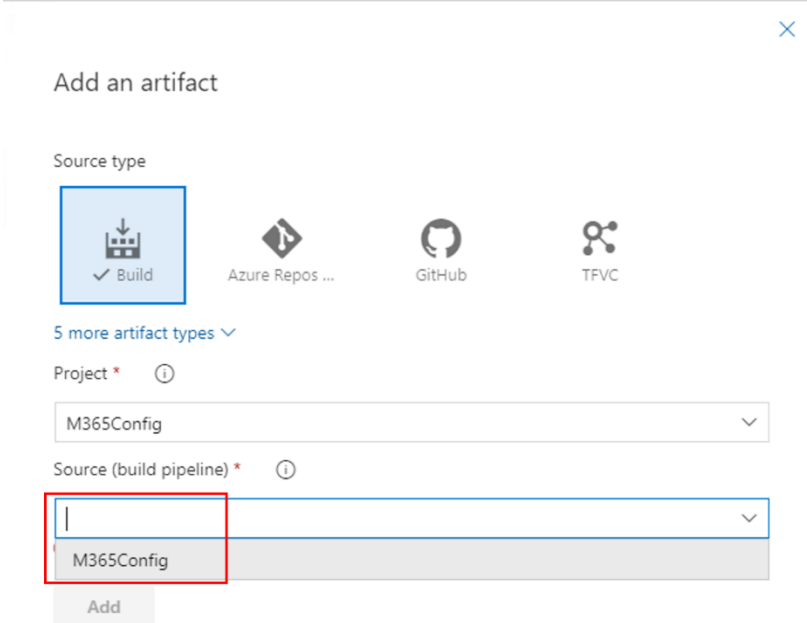
- Give the Stage a name and close the pane



- Click "Add an artifact"



- Under "Source" select the build pipeline



Add an artifact

Source type

✓ Build Azure Repos ... GitHub TFVC

5 more artifact types

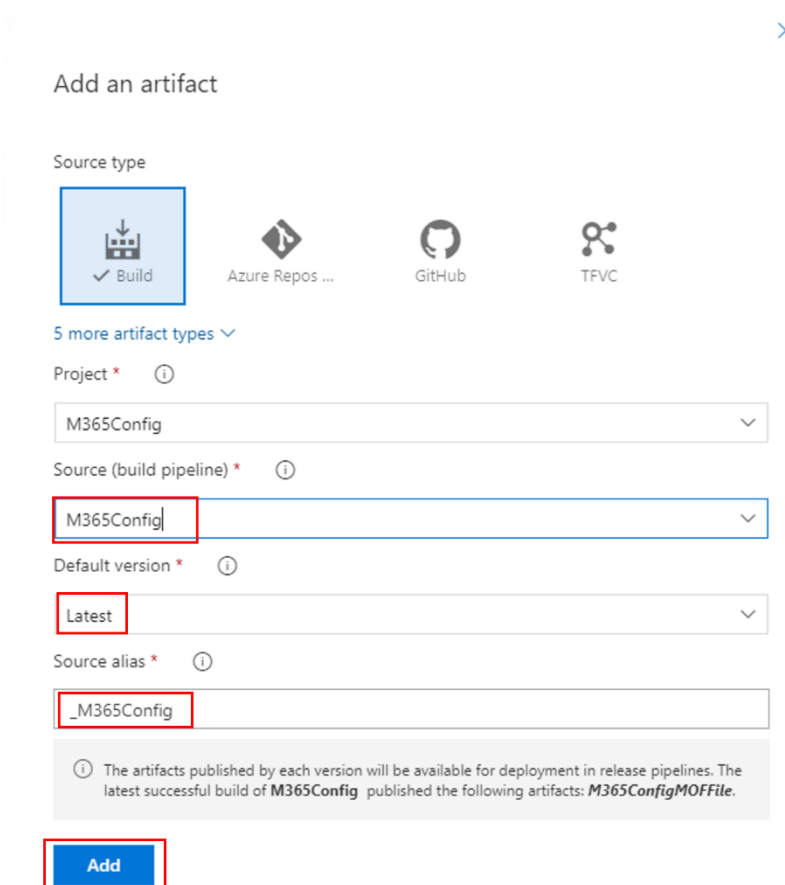
Project \* M365Config

Source (build pipeline) \* M365Config

Add

- After selecting the Source, more options will appear. Leave them as default and click "Add".

**NOTE:** Notice the "Source alias" value. We need this value in a subsequent step.



Add an artifact

Source type

✓ Build Azure Repos ... GitHub TFVC

5 more artifact types

Project \* M365Config

Source (build pipeline) \* M365Config

Default version \* Latest

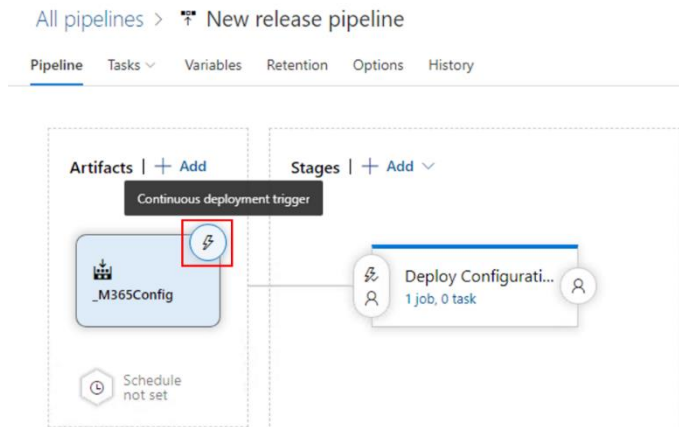
Source alias \* \_M365Config

The artifacts published by each version will be available for deployment in release pipelines. The latest successful build of **M365Config** published the following artifacts: **M365ConfigMOFFile**.

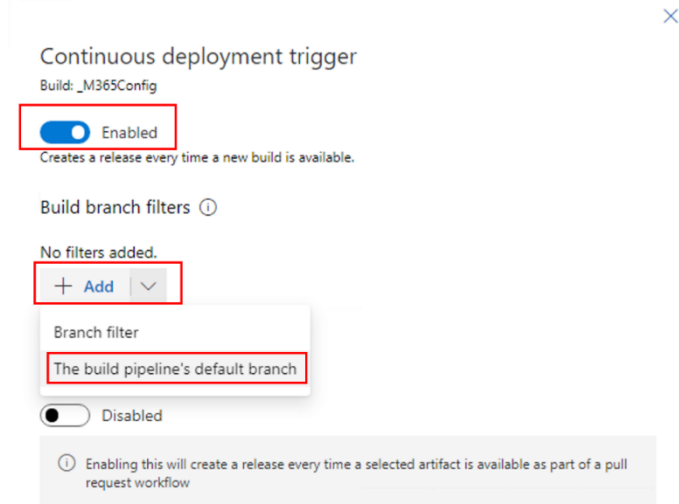
Add



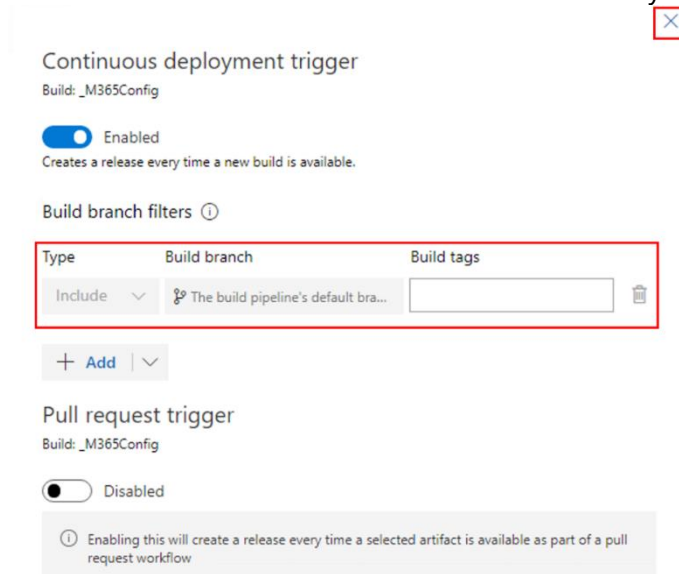
- Click 'Add'
- Configure the Release pipeline triggers by clicking the Lightning icon next to Artifacts



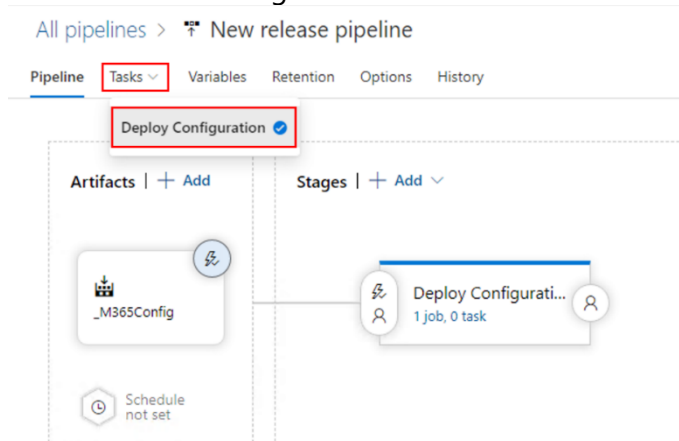
- Enable the "Continuous deployment trigger", under "Build branch filters" click the drop-down next to "Add" and select "The build pipeline's default branch"



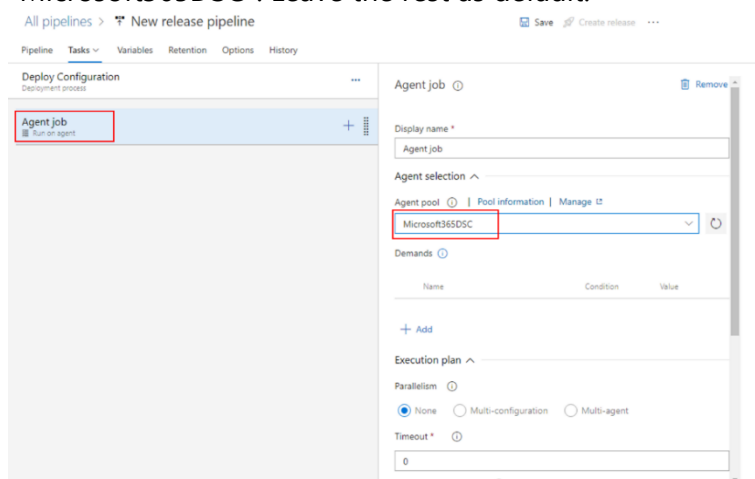
- Make sure the branch has been added successfully and close the pane



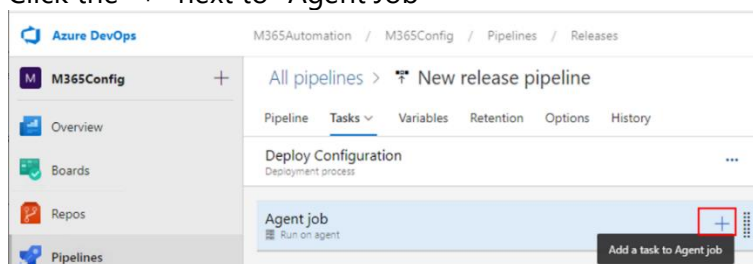
- Select "Tasks > <Stage name>"



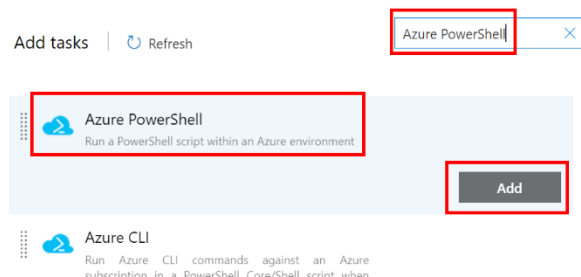
- Select the task "Agent job" in the left part of the pane and change the "Agent pool" to "Microsoft365DSC". Leave the rest as default.



- Click the "+" next to "Agent Job"



- Search for "Azure PowerShell", select "Azure PowerShell" and click "Add"



- Select the "Azure PowerShell" task



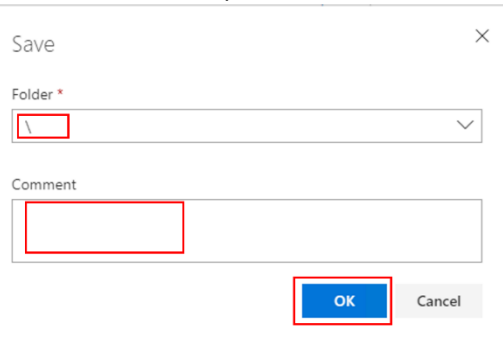
- In the configuration window for the Azure PowerShell task, configure the following settings:
  - Azure Subscription: "KeyVaultConnection" or whatever name you gave the service connection
  - Select "Script File Path" under "Script Type" and browse to the "deploy.ps1" file by clicking the "..." button
  - Enter "-Environment Production" as Script Arguments
  - Select "Stop" as ErrorActionPreference and check the "Fail on Standard Error" checkbox
  - Change the 'Azure PowerShell Version' to "Latest installed version"

A screenshot of the 'Azure PowerShell' task configuration window. The window has a title bar with 'Azure PowerShell' and a help icon. Below the title bar, there are links for 'View YAML' and 'Remove'. The 'Task version' is set to '5.\*'. The 'Display name' is 'Azure PowerShell script: FilePath'. The 'Azure Subscription' is set to 'KeyVaultConnection'. The 'Script Type' is 'Script File Path'. The 'Script Path' is '\$(System.DefaultWorkingDirectory)/\_M365Automation/MOFiles/deploy.ps1'. The 'Script Arguments' are '-Environment "Production"'. The 'ErrorActionPreference' is 'Stop'. The 'Fail on Standard Error' checkbox is checked. The 'Azure PowerShell version options' are expanded, showing 'Latest installed version' selected and 'Specify other version' as an option.

- Open the "Advanced" section and make sure the Working Directory matches the path of the previous step (copy/paste if required).


A screenshot of the 'Advanced' section of the Azure PowerShell task configuration. The 'Advanced' section is expanded. The 'Use PowerShell Core' checkbox is unchecked. The 'Working Directory' is set to '\$(System.DefaultWorkingDirectory)/\_M365Automation/MOFiles'.

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





A "Save" dialog box with a close button (X) in the top right corner. It contains a "Folder" field with a dropdown arrow, a "Comment" text area, and "OK" and "Cancel" buttons at the bottom. Red boxes highlight the "Folder" field, the "Comment" text area, and the "OK" button.

- Hover over the "New release pipeline" name and click on the pen icon that appears behind the title

All pipelines >  New release pipeline 

- Enter a name for the pipeline, like "Deploy Configurations" and click "Save"

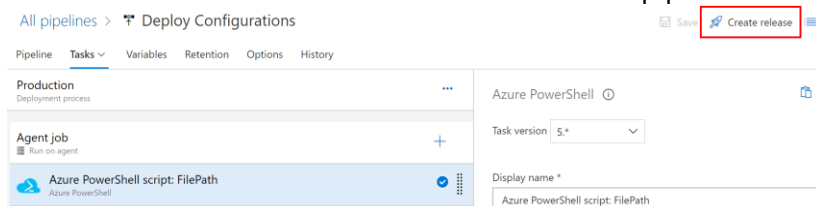
All pipelines >  Deploy Configurations  Save ...

- Add a change comment if you want and click "OK"



A "Save" dialog box with a close button (X) in the top right corner. It contains a "Comment" text area and "OK" and "Cancel" buttons at the bottom. Red boxes highlight the "Comment" text area and the "OK" button.

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



The Azure DevOps Release pipeline interface for a pipeline named "Deploy Configurations". The top bar shows "All pipelines > Deploy Configurations" and buttons for "Save", "Create release", and a help icon. Below the bar are tabs for "Pipeline", "Tasks", "Variables", "Retention", "Options", and "History". The "Tasks" tab is active, showing a task named "Azure PowerShell script: FilePath" under the "Agent job" section. The right sidebar shows the task configuration for "Azure PowerShell", including "Task version" (5.0) and "Display name" (Azure PowerShell script: FilePath).

- Leave everything as default and click "Create"

The screenshot shows the 'Create a new release' dialog box. At the top, it says 'Create a new release' and 'New release pipeline'. Below this, there's a 'Pipeline' section with a dropdown menu currently showing 'Deploy Configuration'. A note says 'Click on a stage to change its trigger from automated to manual.' Below that is a 'Stages for a trigger change from automated to manual.' section with a dropdown menu. The 'Artifacts' section shows a table with 'Source alias' and 'Version' columns. The table has one row: '\_M365Config' and '20200423.3'. Below the table is a 'Release description' text area. At the bottom, there are 'Create' and 'Cancel' buttons. The 'Create' button is highlighted with a red box.

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

All pipelines > New release pipeline

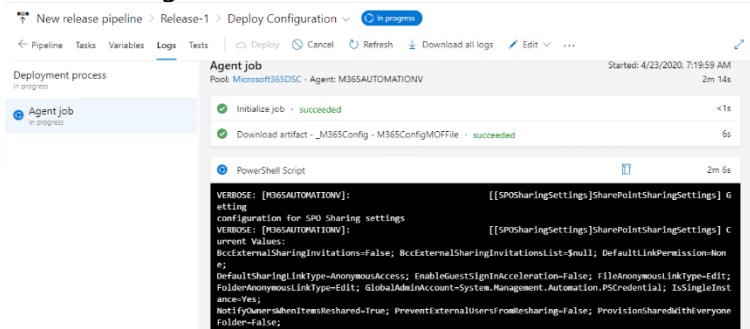


- Review the progress

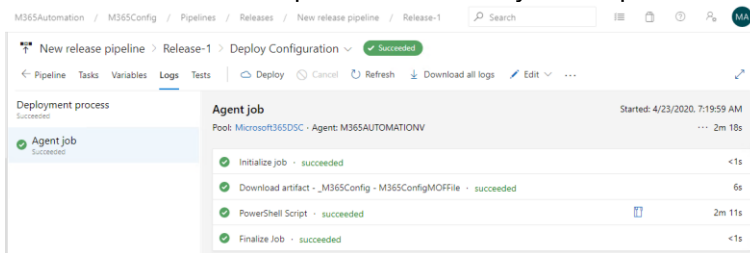
The screenshot shows the 'Release-1' page in Azure DevOps. The breadcrumb navigation at the top reads 'M365Automation / M365Config / Pipelines / Releases / New release pipeline / Release-1'. Below this is a header 'New release pipeline > Release-1'. There are tabs for 'Pipeline', 'Variables', and 'History'. The 'Pipeline' tab is active. Below the tabs are buttons: '+ Deploy', 'Cancel', 'Refresh', 'Edit', and a menu icon. The main content area is divided into two sections: 'Release' and 'Stages'. The 'Release' section shows 'Manually triggered' by 'System Administrator' on '4/23/2020, 7:18 AM'. It also shows 'Artifacts' with a table: '\_M365Config' and '20200423.3'. The 'Stages' section shows a 'Deploy Configuration' stage that is 'In progress for 1m'. It has a progress indicator showing '3/3 TASKS' and a timer '01:06'.

## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

- Click the stage for more details

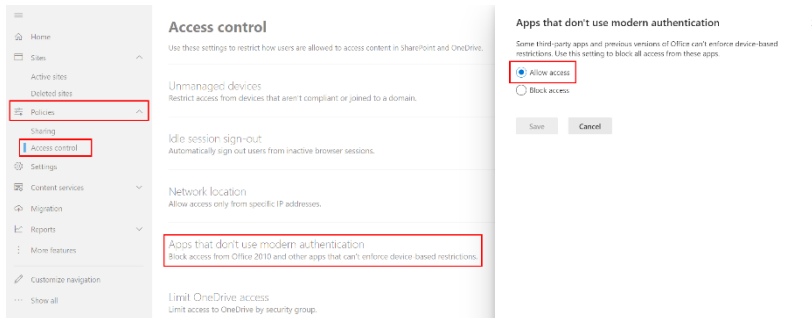


- When the release completes successfully, all steps should have green check marks.



### 5.3.3 Validate that changes to the config are deployed successfully

- Make sure the following setting is configured:  
SharePoint Admin Center > Policies > Access control > Apps that don't use modern authentication



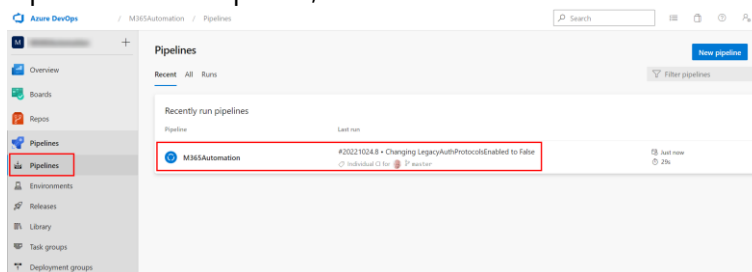
- The above setting is configured by the "LegacyAuthProtocolsEnabled" DSC setting that can be found in "M365Config\0.0.1\DscResources\SharePoint\SharePoint.schema.psm1" in the repository:

```
SPOTenantSettings 'TenantSettings'
{
    IsSingleInstance = "Yes"
    ApplyAppEnforcedRestrictionsToAdHocRecipients = $true
    ConditionalAccessPolicy = "AllowFullAccess"
    FilePickerExternalImageSearchEnabled = $true
    HideDefaultThemes = $false
    LegacyAuthProtocolsEnabled = $true
    MarkNewFilesSensitiveByDefault = "AllowExternalSharing"
    MaxCompatibilityLevel = 15
    MinCompatibilityLevel = 15
    NotificationsInSharePointEnabled = $true
    OfficeClientADALDisabled = $false
    OwnerAnonymousNotification = $true
    PublicCdnAllowedFileTypes = "CSS,EOT,GIF,ICO,JPEG,JPG,JS,MAP,PNG,SVG,TTF,WOFF"
    PublicCdnEnabled = $false
    SearchResolveExactEmailOrUPN = $false
    SignInAccelerationDomain = ""
    UseFindPeopleInPeoplePicker = $false
    UsePersistentCookiesForExplorerView = $false
    UserVoiceForFeedbackEnabled = $true
    ApplicationId = $ApplicationId
    TenantId = $TenantId
    CertificateThumbprint = $Thumbprint
}
```

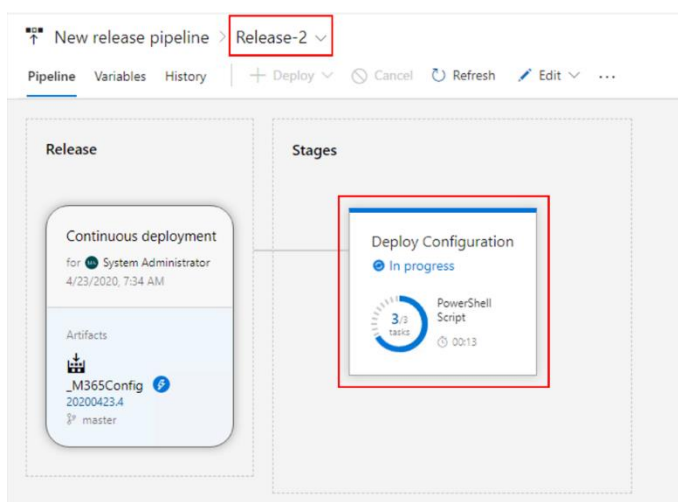
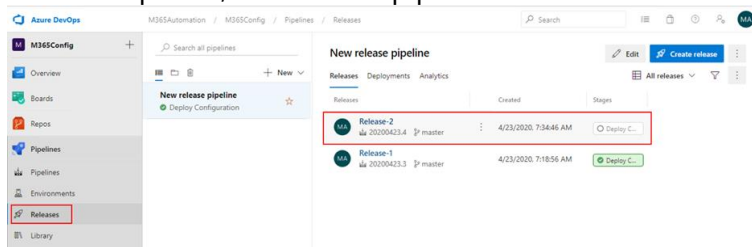
- Change this setting from "\$true" to "\$false"

```
SPOTenantSettings 'TenantSettings'
{
    IsSingleInstance = "Yes"
    ApplyAppEnforcedRestrictionsToADRecipients = $true
    ConditionalAccessPolicy = "AllowFullAccess"
    FilePickerExternalImageSearchEnabled = $true
    HideDefaultThemes = $false
    LegacyAuthProtocolsEnabled = $false
    MarkNewFilesSensitiveByDefault = "AllowExternalSharing"
    MaxCompatibilityLevel = 15
    MinCompatibilityLevel = 15
    NotificationsInSharePointEnabled = $true
    OfficeClientADALDisabled = $false
    OwnerAnonymousNotification = $true
    PublicCdnAllowedFileTypes = "CSS,EOT,GIF,ICO,JPEG,JPG,JS,MAP,PNG,SVG,TTF,WOFF"
    PublicCdnEnabled = $false
    SearchResolveExactEmailUPN = $false
    SigninAccelerationDomain = ""
    UseFindPeopleInPeoplePicker = $false
    UsePersistentCookiesForExplorerView = $true
    UserVoiceForFeedbackEnabled = $true
    ApplicationId = $ApplicationId
    TenantId = $TenantId
    CertificateThumbprint = $Thumbprint
}
```

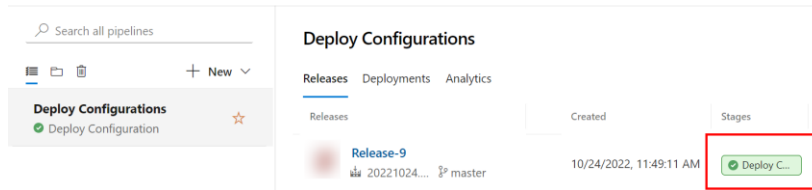
- Save the file, go to the Git Source Control section, enter a commit description, commit the change and sync the repository with Azure DevOps
- Open the Build Pipeline, which should have started



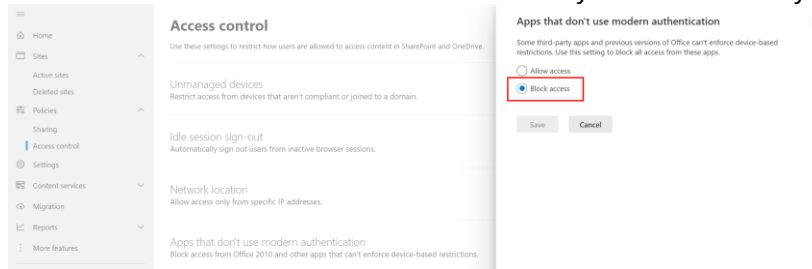
- Once completed, the Release pipeline should automatically start



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



- Go to the SharePoint Admin Portal and verify if that is actually the case



### 5.3.4 Create a scheduled Compliance Test Release pipeline

The solution uses a scheduled pipeline to periodically check if the environments are still in the desired state and sends a notification of the results to either an email address or a Teams channel. In this step we are going to configure this test pipeline:

- First decide if you want to send a message to a Teams channel or an e-mail message:
  - Teams: Make sure you set up a webhook on the Teams channel you want to use. To learn how to do this, check-out this article: <https://learn.microsoft.com/en-us/microsoftteams/platform/webhooks-and-connectors/how-to/add-incoming-webhook#create-an-incoming-webhook>
  - E-mail: Make sure you have an App Registration<sup>6</sup> with App Secret configured in Azure Active Directory, which has the "Graph > Mail.Send" permissions granted: <https://learn.microsoft.com/en-us/graph/auth-register-app-v2> <https://learn.microsoft.com/en-us/graph/auth-v2-service#2-configure-permissions-for-microsoft-graph>
- Update the values in the "checkdsccompliance.ps1" script with the correct values:
  - When using Teams, set the useTeams variable to \$true and update the "teamsWebhook" variable with the URL created when configuring the webhook on the Teams channel

<sup>6</sup> Use App Registration nr 3. See paragraph 2.3 for more information.

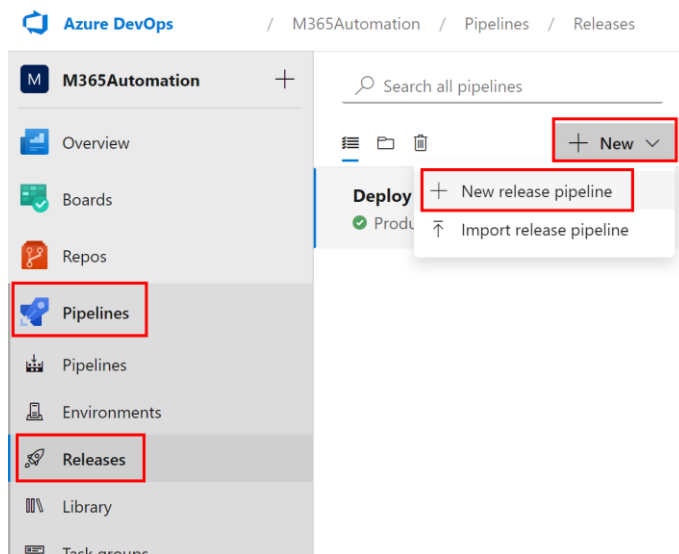


- When using E-mail, set the useMail variable to \$true and update the following variables:
  - mailAppId: The Application ID of the App Registration
  - mailAppSecret: The generated secret of the App Registration
  - mailTenantId: The Tenant ID of the App Registration
  - mailFrom: The user principal name of a user in the tenant
  - mailTo: The mail address of the user the mail should be sent to.

```
##### GENERIC VARIABLES #####  
$workingDirectory = $PSScriptRoot  
$encounteredError = $false  
  
$useMail = $true  
$mailAppId = '<APPID>  
$mailAppSecret = '<SECRET>  
$mailTenantId = '<TENANTID>  
$mailFrom = '<FROM>  
$mailTo = '<TO>  
  
$useTeams = $true  
$teamsWebhook = '<WEBHOOK>
```

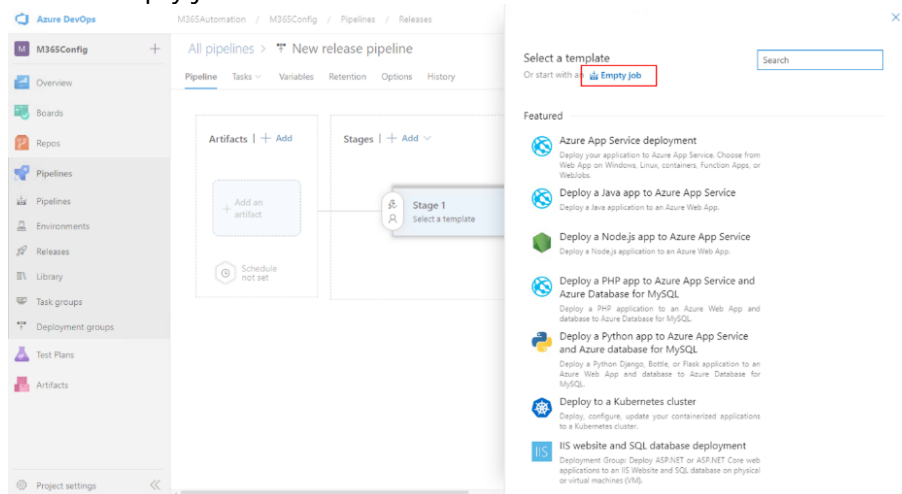
Now configure the Test release pipeline:

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

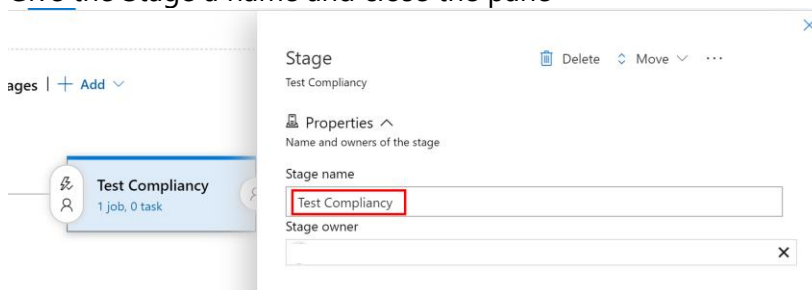


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

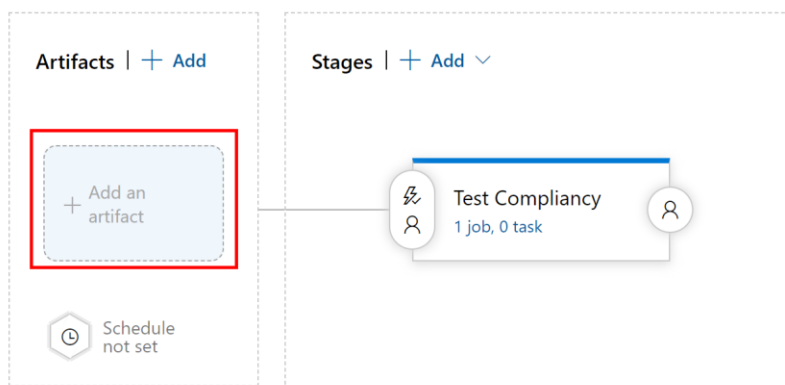
- Select "Empty job"



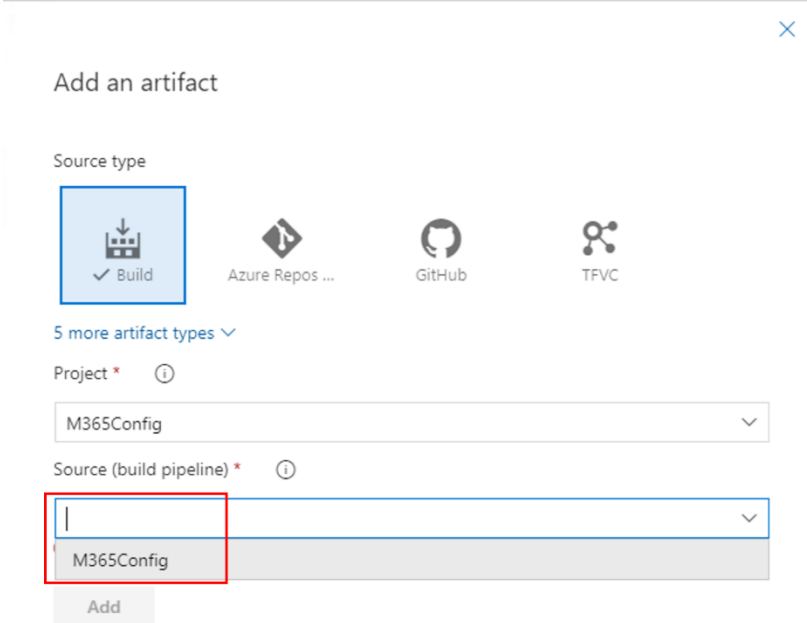
- Give the Stage a name and close the pane



- Click "Add an artifact"



- Under "Source" select the build pipeline



Add an artifact

Source type

✓ Build Azure Repos ... GitHub TFVC

5 more artifact types

Project \* ⓘ

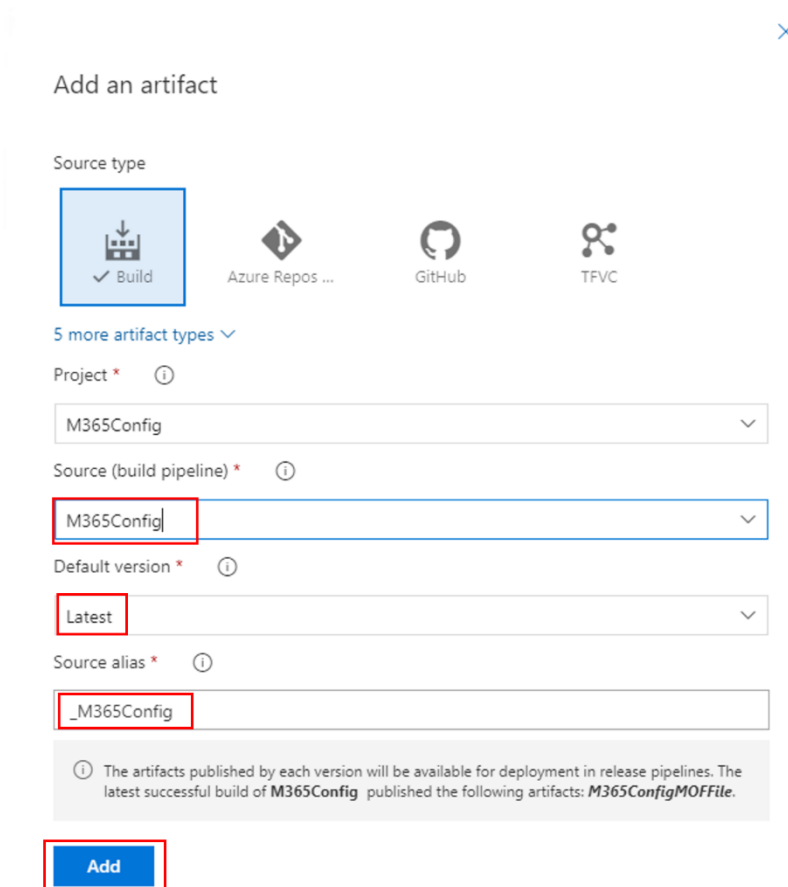
M365Config

Source (build pipeline) \* ⓘ

M365Config

Add

- After selecting the Source, more options will appear. Leave them as default and click "Add".



Add an artifact

Source type

✓ Build Azure Repos ... GitHub TFVC

5 more artifact types

Project \* ⓘ

M365Config

Source (build pipeline) \* ⓘ

M365Config

Default version \* ⓘ

Latest

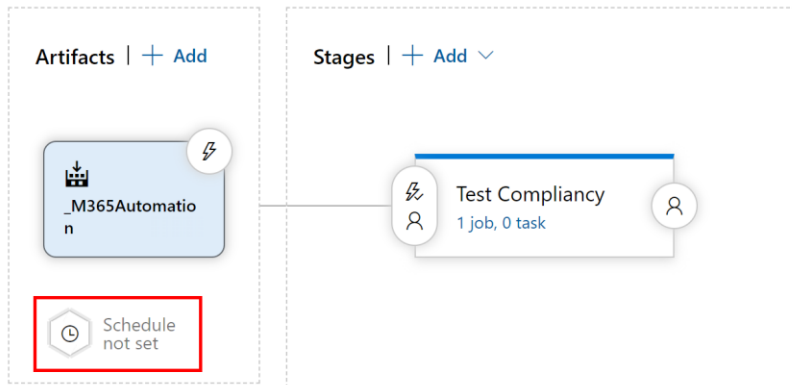
Source alias \* ⓘ

\_M365Config

ⓘ The artifacts published by each version will be available for deployment in release pipelines. The latest successful build of **M365Config** published the following artifacts: **M365ConfigMOFFile**.

Add

- Configure the Release pipeline triggers by clicking the "Schedule not set" below Artifacts



- Enable the schedule by moving the slider to "Enabled" and configure the schedule shown below (or your own schedule). When done, close the window by clicking the "x" in the upper right corner

**Scheduled release trigger**  
Define schedules to trigger releases

☒ Enabled  
Create a new release at the specified times

Mon through Sun

☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat ☒ Sun  
00h 00m (UTC) Coordinated Universal Time

☐ Only schedule releases if the source or pipeline has changed

Mon through Sun at 6:00

☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat ☒ Sun  
06h 00m (UTC) Coordinated Universal Time

☐ Only schedule releases if the source or pipeline has changed

Mon through Sun at 12:00

☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat ☒ Sun  
12h 00m (UTC) Coordinated Universal Time

☐ Only schedule releases if the source or pipeline has changed

Mon through Sun at 18:00

☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat ☒ Sun  
18h 00m (UTC) Coordinated Universal Time

☐ Only schedule releases if the source or pipeline has changed

+ Add a new time

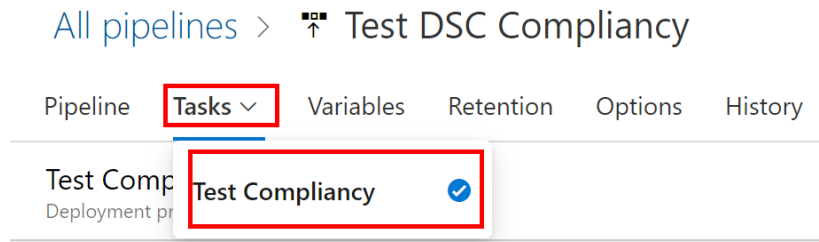
- Edit the pipeline name by clicking the "New release pipeline" in the top of the screen and entering "Test DSC Compliancy"

[All pipelines](#) > [New release pipeline](#) 

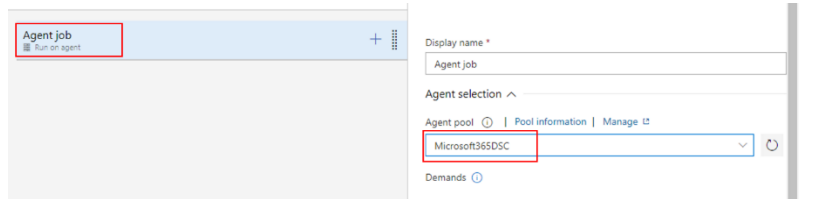
New name:

[All pipelines](#) > [Test DSC Compliancy](#)

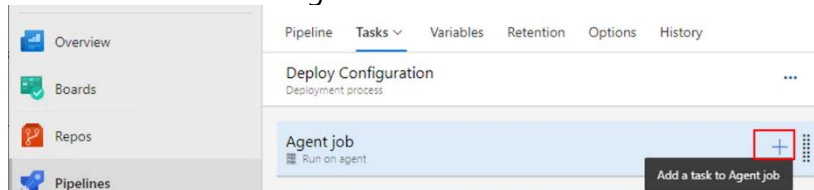
- Select "Tasks > <Stage name>"



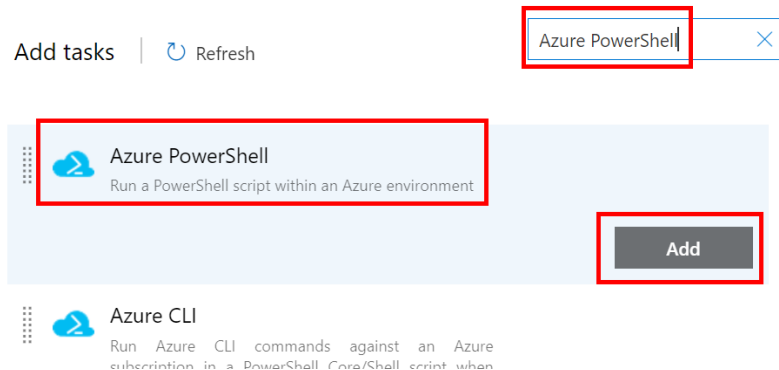
- Select the task "Agent job" in the left part of the pane and change the "Agent pool" to "Microsoft365DSC". Leave the rest as default.



- Click the "+" next to "Agent Job"



- Search for "Azure PowerShell", select "Azure PowerShell" and click "Add"



- Select the "Azure PowerShell" task



- In the configuration window for the Azure PowerShell task, configure the following settings:
  - Azure Subscription: "KeyVaultConnection" or whatever name you gave the service connection
  - Select "Script File Path" as "Type" and browse to the "checkdscompliance.ps1" file by clicking the "..." button
  - Select "Stop" as ErrorActionPreference and check the "Fail on Standard Error" checkbox
  - Ensure 'Latest installed version' is selected under Azure PowerShell Version

Azure PowerShell ⓘ [View YAML](#) [Remove](#)

Task version 5.\* ▼

Display name \*  
Azure PowerShell script: FilePath

Azure Subscription \* ⓘ | [Manage](#) [🔗](#)  
KeyVaultConnection ▼ ↻

Script Type ⓘ  
☒ Script File Path ☐ Inline Script

Script Path ⓘ  
\$(System.DefaultWorkingDirectory)/\_M365Automation/MOFFiles/checkdscompliance.ps1 ...

Script Arguments ⓘ  
...

ErrorActionPreference ⓘ  
Stop ▼  
☒ Fail on Standard Error ⓘ

Azure PowerShell version options ^

Azure PowerShell Version ⓘ  
☒ Latest installed version ☐ Specify other version

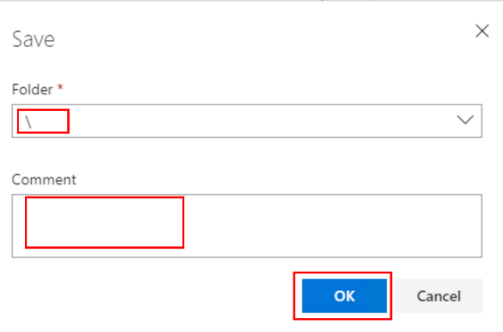
- Open the "Advanced" section and make sure the Working Directory is matching the path of the previous step.

Advanced ^

☐ Use PowerShell Core ⓘ

Working Directory ⓘ  
\$(System.DefaultWorkingDirectory)/\_M365Automation/MOFFiles ...

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



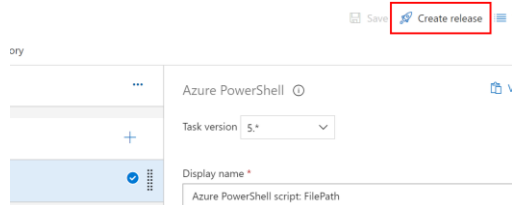
Save

Folder \*

Comment

OK Cancel

- Click "Create release" to test the created Release pipeline, which will run a DSC Compiancy check. Leave options as default and click "Create".



Save Create release

ory

Azure PowerShell

Task version 5.1

Display name \*

Azure PowerShell script: FilePath

- When done and all is configured correctly, you will receive a notification either via Teams or via E-mail.

## 6 Troubleshooting

### 6.1 Error: Service connection could not be found

When running a Release pipeline, you might run into the following error:



If this is the case, please check with which name you have created the service connection in paragraph 4.5.4 and the used "Azure Subscription" property selected while creating the Release pipelines in paragraphs 5.3.2 and 5.3.4.

### 6.2 Error: Release pipeline throws an error about the PSGallery not found

When you run a Release pipeline and are receiving an error about the PSGallery not being found, please log onto the VM with the account of the DevOps agent. That will run the Out-of-the-Box Experience wizard and configure the PSGallery for you.



## 7 Security Enhancements

### 7.1 Using Azure Conditional Access to secure service account

Azure Conditional Access<sup>7</sup> 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 addresses of all the VMs
- Name of DSC service account created in paragraph 3.1, e.g., "DscConfigAdmin"

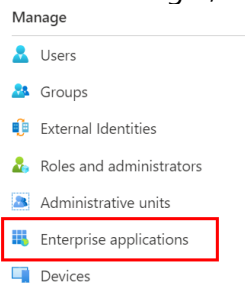
#### Steps

- Open the Azure Portal (<https://portal.azure.com>)

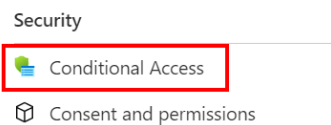
- Go to Azure Active Directory



- Under "Manage", click "Enterprise applications"

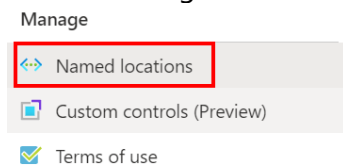


- Under "Security", click "Conditional Access"



- First, we are going to create a Named Location

- Under "Manage", click "Named locations"



<sup>7</sup> Azure AD Premium P1 license required

- Click "New location"

[+ 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

[Home](#) > [Contoso](#) > [Enterprise applications](#) > [Conditional Access](#) >

### New named location

[↑](#) Upload [↓](#) Download

Name \*

Azure Self Hosted VMs ✓

Define the location using:

☒ IP ranges

☐ Countries/Regions

☐ Mark as trusted location ⓘ

IP ranges

123.123.123.123/32 ✓ ...

- Click "Create" to create the Named location
- Next, select "Policies" and click "New policy"

[Home](#) > [Contoso](#) > [Enterprise applications](#) >

### Conditional Access | Policies

Azure Active Directory

[+ New policy](#) [What If](#) [Refresh](#) [Got feedback?](#)

[Policies](#) [Insights and reporting](#) [Diagnose and solve problems](#)

**New Conditional Access policies now apply to all client app types when n**

Policy Name

- 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 "Select users and groups"
    - Check "Users and groups"
    - Search and select the DSC Service Account

[Home](#) > [Conditional Access](#) >

### New

Conditional access policy

Control user access based on conditional access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name \*

Conditional Access for DSC Service Acco... ✓

Assignments

Users and groups ⓘ

Specific users included

Cloud apps or actions ⓘ

No cloud apps or actions selected

Conditions ⓘ

0 conditions selected

Access controls

Grant ⓘ

0 controls selected

Session ⓘ

0 controls selected

Control user access based on users and groups assignment for all users, specific groups of users, directory roles, or external guest users [Learn more](#)

**Include** Exclude

☐ None

☐ All users

☒ Select users and groups

☐ All guest and external users ⓘ

☐ Directory roles ⓘ

☒ Users and groups

Select

1 user

**DS** DSC Service Account  
dsc@M365x812127.onmicros... ⋮

- Cloud apps or actions: Select "All cloud apps"

[Home](#) > [Conditional Access](#) >

### New

Conditional access policy

Control user access based on conditional access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name \*

Conditional Access for DSC Service Acco... ✓

Assignments

Users and groups ⓘ

Specific users included

Cloud apps or actions ⓘ

All cloud apps

Conditions ⓘ

0 conditions selected

Access controls

Grant ⓘ

Control user access based on all or specific cloud apps or actions. [Learn more](#)

Select what this policy applies to

**Cloud apps** User actions

**Include** Exclude

☐ None

☒ All cloud apps

☐ Select apps

⚠ Don't lock yourself out! This policy impacts the Azure portal. Before you continue, ensure that you or someone else will be able to get back into the portal. Disregard this warning if you are configuring persistent browser session policy that works correctly only if "All cloud apps" are selected.

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

Home > Conditional Access > New

Conditional access policy

Control user access based on conditional access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name \*

Conditional Access for DSC Service Acco... ✓

Assignments

Users and groups ⓘ

Specific users included >

Cloud apps or actions ⓘ

All cloud apps >

Conditions ⓘ

1 condition selected >

User risk ⓘ

Not configured >

Sign-in risk ⓘ

Not configured >

Device platforms ⓘ

Not configured >

Locations ⓘ

Any location >

Client apps ⓘ

Not configured >

Configure ⓘ

Yes No

Include

Exclude

Select the locations to exempt from the policy

○ All trusted locations

● Selected locations

Select

None >

Select

Locations

Location type: All types Trusted type: All types

Search names

Name	Location type	Trusted
MFA Trusted IPs	IP ranges	Yes
Azure Self Hosted VMs	IP ranges	No

- Access controls > Grant
  - Select "Block access"

Home > Conditional Access > New

Conditional access policy

Control user access based on conditional access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name \*

Conditional Access for DSC Service Acco... ✓

Assignments

Users and groups ⓘ

Specific users included >

Cloud apps or actions ⓘ

All cloud apps >

Conditions ⓘ

1 condition selected >

Access controls

Grant ⓘ

Block access >

Session ⓘ

0 controls selected >

Grant

Control user access enforcement to block or grant access. [Learn more](#)

● Block access

○ Grant access

☐ Require multi-factor authentication ⓘ

☐ Require device to be marked as compliant ⓘ

☐ Require Hybrid Azure AD joined device ⓘ

☐ Require approved client app ⓘ

[See list of approved client apps](#)

☐ Require app protection policy ⓘ

[See list of policy protected client apps](#)

☐ Require password change (Preview) ⓘ

For multiple controls

● Require all the selected controls

○ Require one of the selected controls

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

[Home](#) > [Conditional Access](#) >

### New

Conditional access policy

Control user access based on conditional access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name \*

Conditional Access for DSC Service Acco... ✓

#### Assignments

Users and groups ⓘ  
Specific users included >

Cloud apps or actions ⓘ  
All cloud apps >

Conditions ⓘ  
1 condition selected >

#### Access controls

Grant ⓘ  
Block access >

Session ⓘ  
0 controls selected >

Enable policy

Report-only On Off

Create

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

## 8 Script details

This whitepaper uses 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://aka.ms/M365DSCWhitepaper/Scripts>

The package contains these files:

File name	Description
<b>.gitattributes</b>	File used by Git, which specifies how each type of file should be handled. Usually there is no need to update this file.
<b>.gitignore</b>	File used by Git, which specifies all files and folders Git must ignore. Usually there is no need to update this file.
<b>build.ps1</b>	The script that is responsible for configuring the Microsoft hosted agent, retrieving the service account password from Azure Key Vault and compiling the DSC MOF file.
<b>Checkdsccompliance.ps1</b>	The script that used by the Test DSC Compliance pipeline to check all environment on compliance with the Desired State and send the results via Email or Teams channel message.
<b>deploy.ps1</b>	The script that is responsible for configuring the self-hosted agent and deploying the DSC MOF file to the LCM of the virtual machine.
<b>DscResources.psd1</b>	Data file that specifies the version of Microsoft365DSC to be used. If you want to use a different version of Microsoft365DSC, just update this file.
<b>M365Configuration.ps1</b>	The master DSC configuration file that orchestrates the various composite resources and passes the provided credentials/app registration <sup>8</sup> info to those resources.
<b>ReadMe.md</b>	A project description file in Markdown format. This will be displayed when opening the repository in Azure DevOps.
Folder: <b>DataFiles</b>	PowerShell data files for each environment that should be managed. The solution only contains one file but can be extended when required. See paragraph 1.2 for more info.
<b>Production.psd1</b>	Data file with all information for the environment called "Production".

<sup>8</sup> Use App Registration nr 1. See paragraph 2.3 for more information.

Folder: <b>M365Config</b>	<b>Composite DSC resource used by the solution</b>
Folder: <b>DscResources</b>	
Folder: <b>Exchange</b>	
<b>Exchange.psd1</b>	Resource data file, which specifies the details of the resource
<b>Exchange.schema.psm1</b>	Composite resource code. This file defines the desired state for the Exchange resource. Add Exchange configurations to this file.
Folder: <b>Office365</b>	
<b>Office365.psd1</b>	Resource data file, which specifies the details of the resource
<b>Office365.schema.psm1</b>	Composite resource code. This file defines the desired state for the Office365 resource. Add Office365 configurations to this file.
Folder: <b>PowerPlatform</b>	
<b>PowerPlatform.psd1</b>	Resource data file, which specifies the details of the resource
<b>PowerPlatform.schema.psm1</b>	Composite resource code. This file defines the desired state for the Power Platform resource. Add Power Platform configurations to this file.
Folder: <b>SecurityCompliance</b>	
<b>SecurityCompliance.psd1</b>	Resource data file, which specifies the details of the resource
<b>SecurityCompliance.schema.psm1</b>	Composite resource code. This file defines the desired state for the SecurityCompliance resource. Add Security & Compliance configurations to this file.
Folder: <b>Teams</b>	
<b>Teams.psd1</b>	Resource data file, which specifies the details of the resource
<b>Teams.schema.psm1</b>	Composite resource code. This file defines the desired state for the Teams resource. Add Teams configurations to this file.
Folder: <b>SharePoint</b>	
<b>SharePoint.psd1</b>	Resource data file, which specifies the details of the resource
<b>SharePoint.schema.psm1</b>	Composite resource code. This file defines the desired state for the SharePoint resource. Add SharePoint configurations to this file.
<b>M365Config.ps1</b>	Module manifest file for the composite resource
Folder: <b>Pipelines</b>	The configuration file for the Azure DevOps Build Pipeline. This file defines which steps are required to build the DSC MOF file.
<b>azure-pipeline.yml</b>	The Build pipeline definition used in this solution. The file defines to first run the Build script and when

	that completes successfully, package the results as an artifact.
Folder: <b>SupportScripts</b>	Scripts used during configuration of the solution. Not used during by any of the pipelines.
<b>PopulateKeyVault.ps1</b>	Script used to populate all required items in Azure Key Vault. It is using the specified data file to determine what items to create.



## 9 Learning materials

### 9.1 Desired State Configuration

- Microsoft Learn: "Getting Started with PowerShell Desired State Configuration"
  - <https://docs.microsoft.com/en-us/shows/getting-started-with-powershell-dsc/>
- Microsoft Learn: "Advanced PowerShell Desired State Configuration"
  - <https://docs.microsoft.com/en-us/shows/advanced-powershell-dsc-and-custom-resources/>
- Desired State Configuration Overview for Engineers
  - <https://learn.microsoft.com/en-us/powershell/dsc/overview/DscForEngineers>
- Creating configurations
  - Configurations: <https://learn.microsoft.com/en-us/powershell/dsc/configurations/configurations>
  - Write, Compile, and Apply a Configuration: <https://learn.microsoft.com/en-us/powershell/dsc/configurations/write-compile-apply-configuration>
  - DependsOn: <https://learn.microsoft.com/en-us/powershell/dsc/configurations/resource-depends-on>
  - DSC Resources: <https://learn.microsoft.com/en-us/powershell/dsc/resources/resources>
- Using configuration data in DSC
  - <https://learn.microsoft.com/en-us/powershell/dsc/configurations/configData>
  - <https://learn.microsoft.com/en-us/powershell/dsc/configurations/separatingEnvData>
- Composite resources
  - <https://learn.microsoft.com/en-us/powershell/dsc/resources/authoringresourcecomposite>
- Secure the MOF file
  - <https://learn.microsoft.com/en-us/powershell/dsc/pull-server/secureMOF>
  - <https://learn.microsoft.com/en-us/powershell/dsc/configurations/configDataCredentials>
- Local Configuration Manager
  - Configuring: <https://learn.microsoft.com/en-us/powershell/dsc/managing-nodes/metaConfig>
  - Push/Pull model: <https://learn.microsoft.com/en-us/powershell/dsc/pull-server/enactingConfigurations>
- Apply, Get, and Test Configurations on a Node
  - <https://learn.microsoft.com/en-us/powershell/dsc/managing-nodes/apply-get-test>

- Debugging DSC
  - <https://learn.microsoft.com/en-us/powershell/dsc/troubleshooting/debugResource>

## 9.2 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>

## 9.3 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>

## 10 Acronyms

Acronym	Meaning
<b>CD/CI</b>	Continuous Development / Continuous Integration
<b>DSC</b>	Desired State Configuration
<b>LCM</b>	Local Configuration Manager
<b>MFA</b>	Multi-Factor Authentication
<b>MOF</b>	Managed Object Format
<b>VM</b>	Virtual Machine