

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

Author: Yorick Kuijs  
Cloud Solution Architect @ Microsoft  
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

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

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.

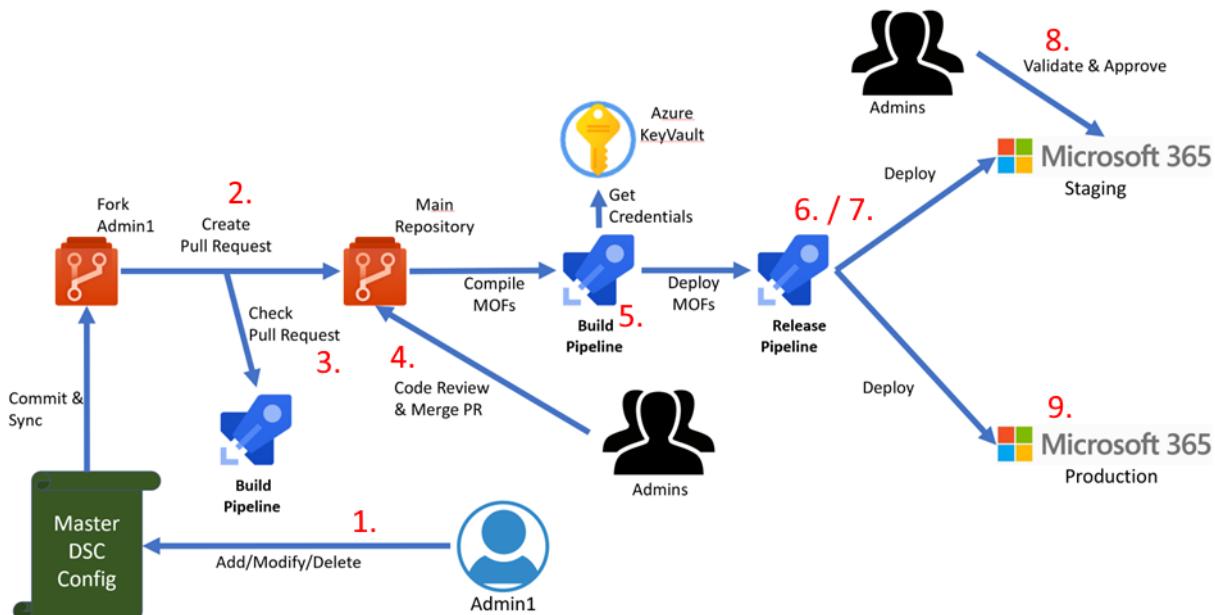
Many companies are adopting DevOps practices and are interested in applying them against Microsoft 365 as well. Infrastructure as Code and Continuous Deployment/Continuous Integration (CD/CI) are important concepts in DevOps.

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:

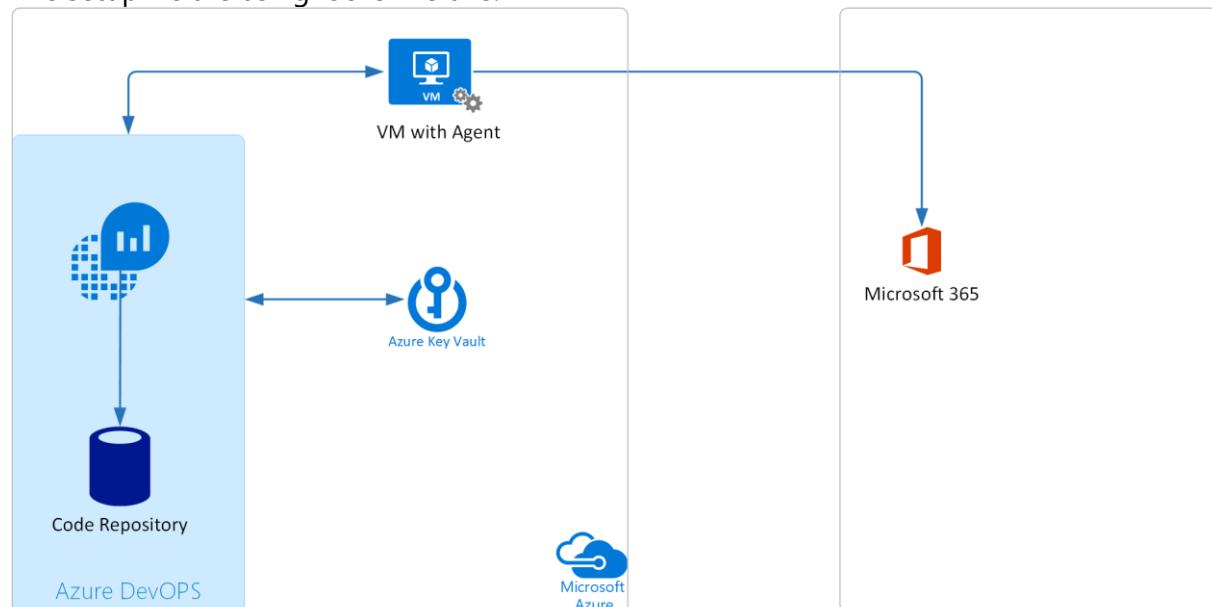
1. Admin1 updates the configuration in his personal copy (fork)
2. When done, Admin1 creates a Pull Request to have his changes merged into the Main repository

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.psd1 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:</p> <p><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></p> <p><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

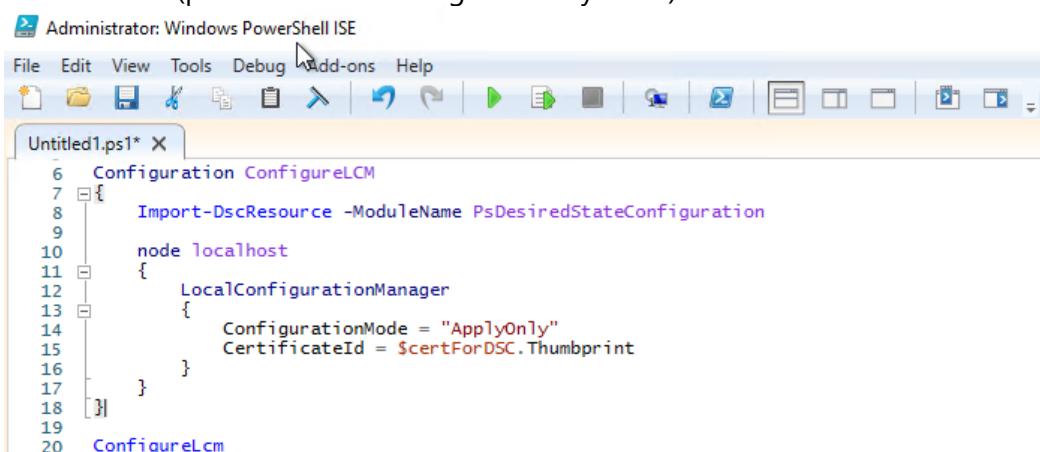
- Paste the following code in the white script pane:

```
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\ConfigureLCM -Verbose
VERBOSE: Performing the operation "Start-DsConfiguration: SendMetaConfigurationApply" on target "MSFT_DSCLoca
lConfigurationManager".
VERBOSE: Perform operation 'Invoke CimMethod' with following parameters: {'methodName' = SendMetaConfiguration
Apply,'className' = MSFT_DSCLocalConfigurationManager,'namespaceName' = root\Microsoft\Windows\DesiredStateCon
figuration'}.
VERBOSE: An LCM method call arrived from computer M365AUTOMATIONV with user sid S-1-5-21-773870280-229285111-1
404544293-500.
VERBOSE: [M365AUTOMATIONV]: LCM: [ Start Set ]
VERBOSE: [M365AUTOMATIONV]: LCM: [ Start Resource ] [MSFT_DSCLocalConfiguration]
VERBOSE: [M365AUTOMATIONV]: LCM: [ Start Set ] [MSFT_DSCLocalConfiguration]
VERBOSE: [M365AUTOMATIONV]: LCM: [ End Set ] [MSFT_DSCLocalConfiguration] in 0.0210 seconds.
VERBOSE: [M365AUTOMATIONV]: LCM: [ End Resource ] [MSFT_DSCLocalConfiguration]
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               : 6D094530ACB5F5E67EDFF394843941A43193EEE8
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      : {}
```

**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.

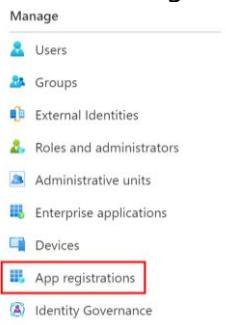
**Important:** Only perform the next steps if the script shows an error that consent could not be provided successfully:

- 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

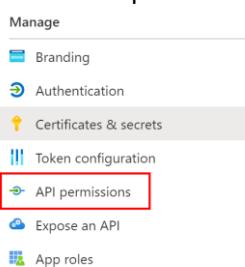


Azure Active  
Directory

- 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](#)

API / Permissions name	Type	Description	Admin consent req...	Status	...
<a href="#">Add a permission</a> <input checked="" type="checkbox"/> <a href="#">Grant admin consent for Microsoft</a>					
SharePoint (1)	Sites.FullControl.All	Application Have full control of all site collections	Yes	⚠️ Not granted for Microsoft	...

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

- 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>"

API / Permission name	Type	Description	Admin consent requ...	Status
Office 365 Exchange Online (1)				
Exchange.ManageAsApp	Application	Manage Exchange As Application	Yes	Granted for Contoso
SharePoint (1)				
Sites.FullControlAll	Application	Have full control of all site collections	Yes	Granted for Contoso

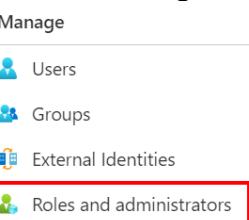
### 4.2.3 Add the App Registration to the Exchange Administrators role

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

The screenshot shows the 'All roles' section of the Microsoft 365 Admin center. On the left, there's a sidebar with links like 'All roles', 'Diagnose and solve problems', 'Activity', 'Access reviews', 'Audit logs', 'Troubleshooting + Support', and 'New support request'. The main area lists various roles with their descriptions. The 'Exchange Administrator' role is highlighted with a red box.

Role	Description
Compliance Data Administrator	Creates and manages compliance content.
Conditional Access Administrator	Can manage Conditional Access capabilities.
Customer LockBox Access Approver	Can approve Microsoft support requests to access customer organizational data.
Desktop Analytics Administrator	Can access and manage Desktop management tools and services.
Directory Readers	Can read basic directory information. Commonly used to grant directory read access.
Directory Writers	Can read and write basic directory information. For granting access to applications, r
Domain Name Administrator	Can manage domain names in cloud and on-premises.
Dynamics 365 Administrator	Can manage all aspects of the Dynamics 365 product.
Edge Administrator	Manage all aspects of Microsoft Edge.
Exchange Administrator	Can manage all aspects of the Exchange product.
Exchange Recipient Administrator	Can create or update Exchange Online recipients within the Exchange Online organiz

- Click the “Add assignments” button

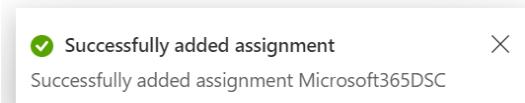
The screenshot shows the 'Add assignments' page. At the top, there are buttons for '+ Add assignments', 'Remove assignments', and 'Download assignments'. A message box says 'You can also assign built-in roles to groups now.' Below that are search and type filters, and a table showing 'No role assignments found'.

- 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”.

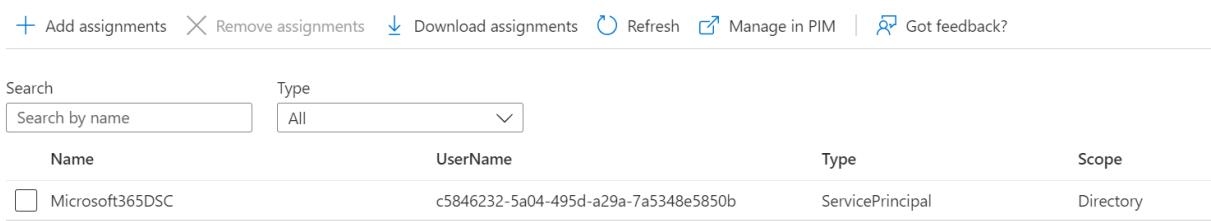
The screenshot shows the 'Add assignments' dialog. It has a search bar with 'microsoft365dsc' typed in. Below the search bar is a list of results, with one item ('Microsoft365DSC') selected and highlighted with a red box. This selected item is then shown in the 'Selected items' list below. At the bottom, there's a large red-bordered 'Add' button.

## 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



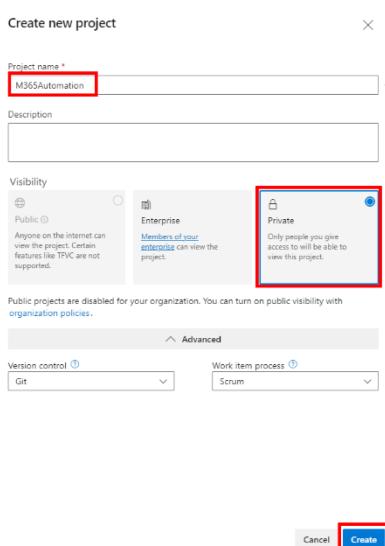
Name	UserName	Type	Scope
<input type="checkbox"/> Microsoft365DSC	c5846232-5a04-495d-a29a-7a5348e5850b	ServicePrincipal	Directory

### 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”



Project name \*  
M365Automation

Description

Visibility

Public (Anyone on the internet can view the project. Certain features like TFVC are not supported.)

Enterprise (Members of your enterprise can view the project.)

Private (Only people you give access to will be able to view this project.)

Advanced

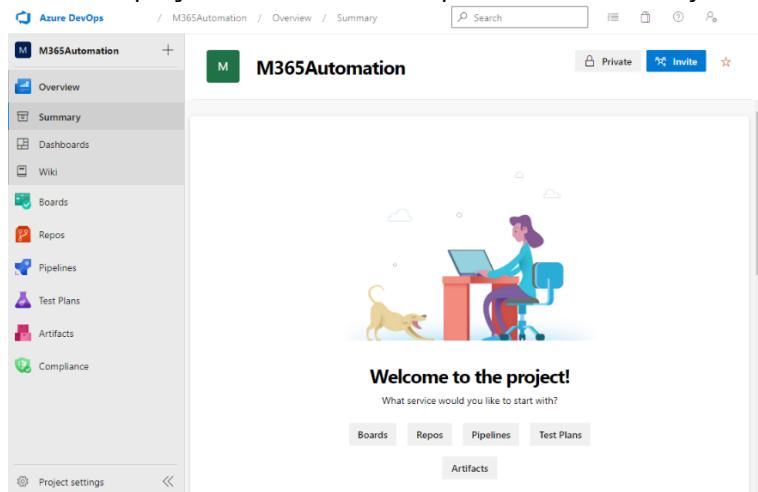
Version control (Git)

Work item process (Scrum)

Create

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

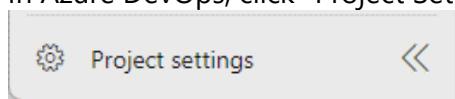
- Once the project is created, it is opened automatically



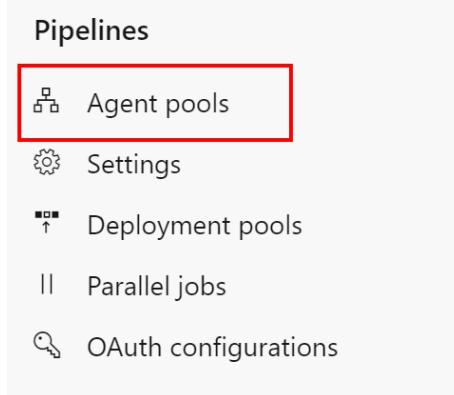
### 4.3.2 Create an Agent Pool in Azure DevOps

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:

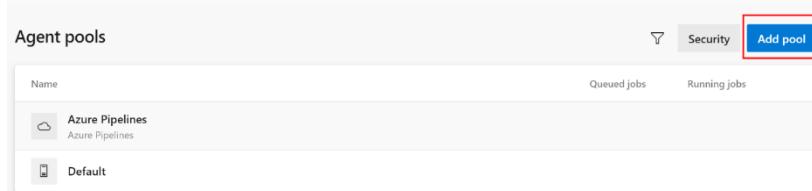
- Browse to the main Azure DevOps page
- Create a new Agent Pool
  - In Azure DevOps, click "Project Settings" in the lower left corner



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

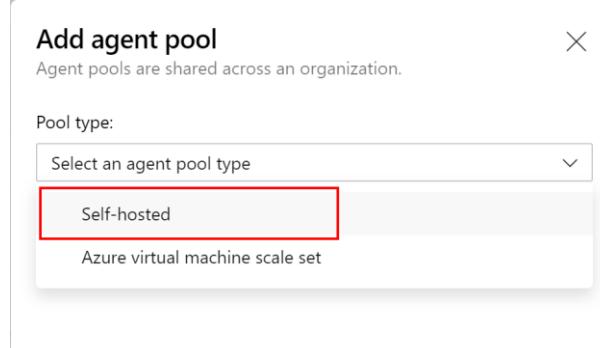


- Create a new Agent Pool by clicking the "Add pool" button in the upper right corner

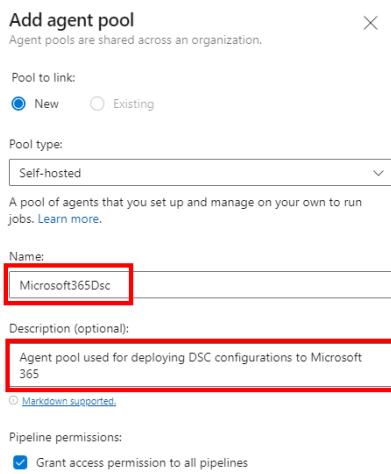


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

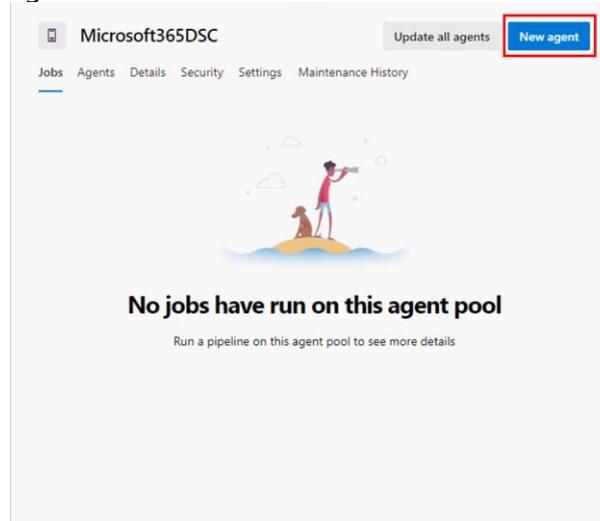
- Select "Self-hosted" as "Pool type"



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

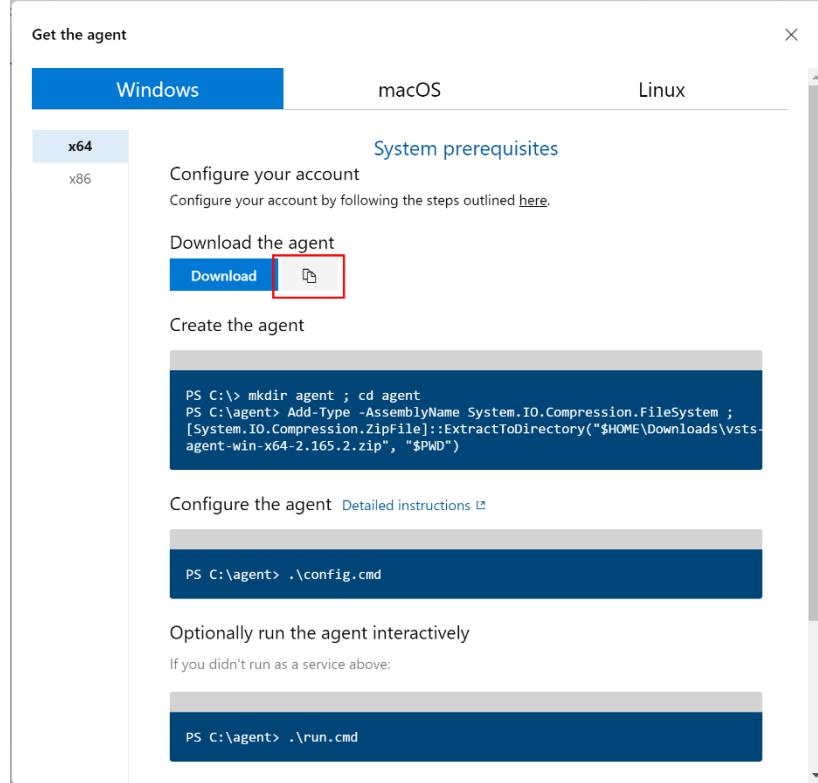


- Click the newly created pool to open the pool
- Click the "New agent" button to open the required information to add a new agent



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

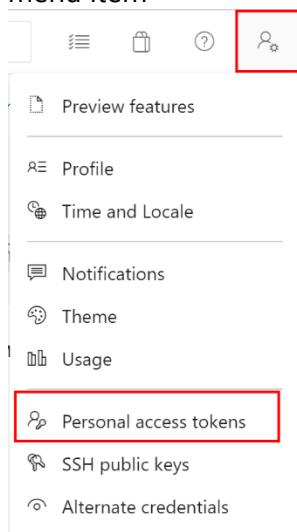
- Copy the download link to be used later in this document



### 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



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

- 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

A screenshot of the 'Personal Access Tokens' list page in Azure DevOps. At the top left is a red box around the '+ New Token' button. Below it are columns for 'Token name', 'Status', 'Organization', and 'Expires on'. A dropdown menu at the top right shows 'Active'.

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

**Create a new personal access token**

Name: DevOpsAgent

Organization: ykuujs

Expiration (UTC): Custom defined (4/16/2021)

**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

Show all scopes (27 more)

Create Cancel

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

**Create a new personal access token**

Name: DevOpsAgent

Organization: ykuujs

Expiration (UTC): Custom defined (4/16/2021)

**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**

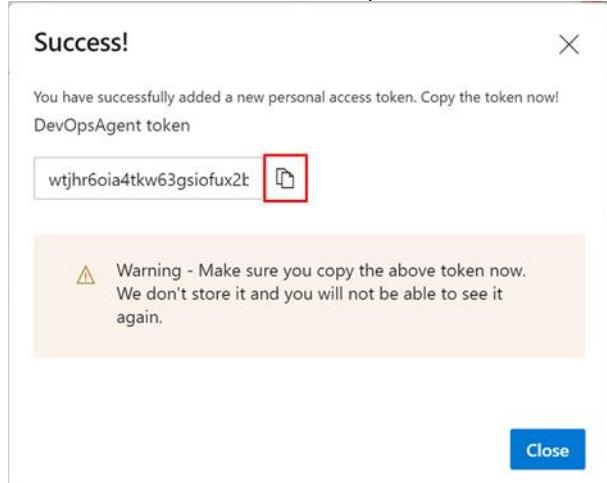
Read data from the analytics service

Read

Show less scopes

**Create** Cancel

- 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.

The screenshot shows the 'Personal Access Tokens' page in the Azure DevOps interface. It lists tokens with columns for Token name, Status, Organization, and Expires on. One token, 'DevOpsAgent' (Agent Pools (Read & manage)), is highlighted with a red border. The status is 'Active', the organization is 'ykuujs', and it expires on '4/16/2021'.

## 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

The screenshot shows an elevated Command Prompt window. The title bar says 'Administrator: Command Prompt'. The command 'c:\Agent>config.cmd' is typed into the prompt.

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

- 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

\Agent\Windows\PowerShell\agent.ps1

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

\Agent\Windows\PowerShell\agent.ps1

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

\Agent\Windows\PowerShell\agent.ps1

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 > *****
```

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

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

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

\Windows\system32\cmd.exe
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

\Windows\system32\cmd.exe
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

\Windows\system32\cmd.exe
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

```
c:\Agent>config.cmd

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]

```
>> 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) > M365AutomationV\M365ConfigAgentSvc
Enter Password for the account M365AutomationV\M365ConfigAgentSvc > *****
```

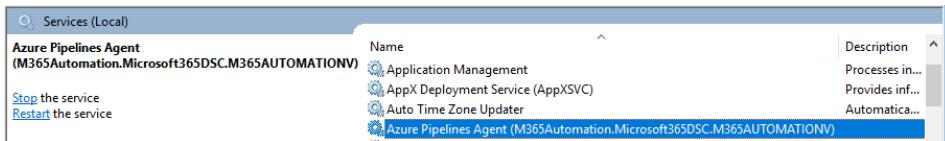
- The agent is being configured. Press Enter to start the service automatically.

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

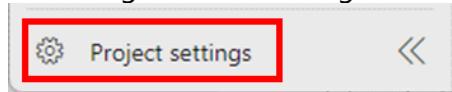
>> Register Agent:

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) > M365AutomationV\M365ConfigAgentSvc
Enter Password for the account M365AutomationV\M365ConfigAgentSvc > *****
Error reported in diagnostic logs. Please examine the log for more details.
- c:\Agent\_diag\Agent_20200423-083409-utc.log
Granting file permissions to 'M365AutomationV\M365ConfigAgentSvc'.
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
```

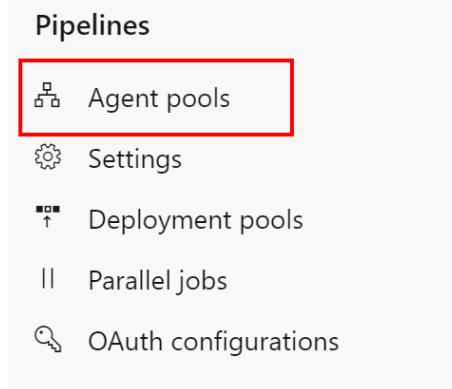
## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps



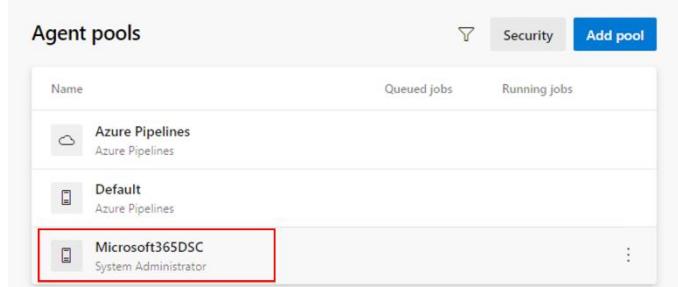
- Verify agent is successfully registered in Azure DevOps
  - Open the Azure DevOps portal
  - Click "Organization Settings" in the lower left corner



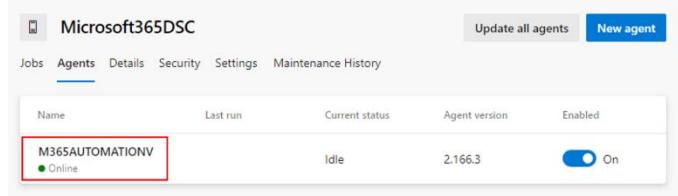
- Scroll down and under "Pipelines", click "Agent Pools"



- 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**

The screenshot shows the Azure Portal's search interface. A red box highlights the search bar containing "Azure Active Directory". Below the search bar, there are several tabs: "All", "Services (91)", "Marketplace (6)", and "Documentation". Under the "Services" heading, a list of services is shown, with "Azure Active Directory" highlighted by a red box. Other listed services include Security, Activity log, and Azure Cosmos DB.

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

The screenshot shows the "Microsoft | App reg" page under "Azure Active Directory". At the top, there is a navigation bar with "Home > Microsoft". Below it, the main title is "Microsoft | App reg" with a subtitle "Azure Active Directory". On the left, there is a sidebar with sections: "Overview", "Preview features", and "Diagnose and solve problems". The "Manage" section is expanded, showing options: "Users", "Groups", "External Identities", "Roles and administrators", "Administrative units", "Enterprise applications", "Devices", and "App registrations". A red box highlights the "App registrations" option.

<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 e.g. https://example.com/auth

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 [View](#)

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	: Microsoft365DSC_DevOpsPipeline	Client credentials	: 0 certificate, 1 secret
Application (client) ID	42e94ab0-f18d-43cc-83a4-2459c77ac958	Redirect URIs	: Add a Redirect URI
Object ID	7c0f7341-1534-4627-882e-1272bdca8898	Application ID URI	: Add an Application ID URI
Directory (tenant) ID	16b3c013-d300-468d-ac64-7eda0820b6d3	Managed application in ...	: Microsoft365DSC_DevOpsPipeline

Supported account types : My organization only

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

Delete Endpoints Preview features

^ Essentials

Display name	: Microsoft365DSC_DevOpsPipeline	Client credentials	: Add a certificate or secret
Application (client) ID	: 42e94ab0-f18d-43cc-83a4-2459c77ac958	Redirect URIs	: Add a Redirect URI
Object ID	: 7c0f7341-1534-4627-882e-1272bdca8898	Application ID URI	: Add an Application ID URI
Directory (tenant) ID	: 16b3c013-d300-468d-ac64-7eda0820b6d3	Managed application in ...	: Microsoft365DSC_DevOpsPipeline

Supported account types : My organization only

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

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

A secret string that the application uses to prove its identity when requesting

+ New client secret

Description	Expires	Value
No client secrets have been created for this application.		

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

Add a client secret

Description Microsoft365DSC

Expires 24 months

Add Cancel

- 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	[REDACTED]	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”

subscriptions

All Services (6) Marketplace (20)

Azure Active Directory (0)

Services

Subscriptions

Event Grid Subscriptions

Quotas

- 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.

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

- Then select the “Role assignments” tab

The screenshot shows the Microsoft 365 Admin Center interface. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Security, Events, Cost Management, and Cost analysis. The 'Access control (IAM)' link is highlighted with a red box. At the top, there's a navigation bar with 'Access control (IAM)' and other options like 'Add', 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and more. Below the navigation, there are tabs for 'Check access', 'Role assignments' (which is highlighted with a red box), 'Roles', 'Deny assignments', and 'Classic administrators'. A section titled 'Number of role assignments for this subscription' shows 1 item out of 2000. There are search and filter options: 'Search by name or email', 'Type : All', 'Role : All', 'Scope : All scopes', and 'Group by : Role'. A table below lists 11 items, including 4 users, 1 group, 5 service principals, and 1 managed identity. The columns are Name, Type, Role, Scope, and Condition.

- Click the “Add” button and then click “Add role assignment”

The screenshot shows a modal dialog for adding a role assignment. It has a header with '+ Add', 'Download role assignments', and 'Edit columns'. Below the header, there are three buttons: 'Add role assignment' (highlighted with a red box), 'Add co-administrator', and 'Add custom role'. A dropdown menu is open, showing 'this subscription' as the selected option. At the bottom of the dialog, there's a 'Got feedback?' link.

- Type “Reader” in the search bar and select the “Reader” role

The screenshot shows the 'Review + assign' step of the role assignment wizard. The 'Role' tab is selected. In the search bar, 'Reader' is typed and highlighted with a red box. Below the search bar, there are filters for 'Type : All' and 'Category : All'. A table lists roles, with 'Reader' being the first item, also highlighted with a red box. The table columns are Name (Name), Description (Description), and Category (Category). The 'Reader' row shows 'View all resources, but does not allow you to make any changes.' and 'acr quarantine data reader' under Category. Other rows listed are 'AcrQuarantineReader' and 'acr quarantine data reader'.

- Click “Next” to proceed to selecting the Members
- Click the “Select members” button

The screenshot shows the 'Select members' step of the role assignment wizard. The 'Members' tab is selected. It shows the 'Selected role' as 'Reader'. Under 'Assign access to', the 'User, group, or service principal' option is selected (indicated by a checked radio button). The 'Members' section contains a button '+ Select members' (highlighted with a red box) and a table with a single row for 'Name'.

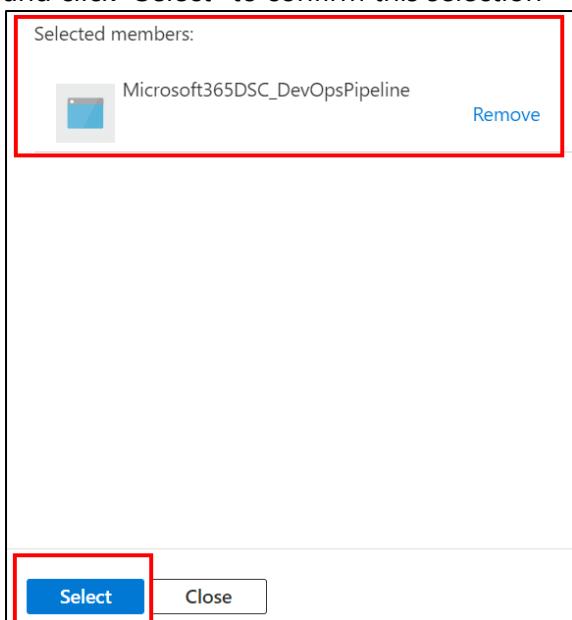
## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

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

### Select members



- 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

Reader				
<input type="checkbox"/>	App	Reader ⓘ	Management group (Inherited)	None
<input type="checkbox"/>	App	Reader ⓘ	Management group (Inherited)	None
<input type="checkbox"/>	App	Reader ⓘ	Management group (Inherited)	None
<input type="checkbox"/>	Group	Reader ⓘ	Management group (Inherited)	None
<input type="checkbox"/>	Microsoft365DSC_DevOps	App Reader ⓘ	This resource	None

### 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"



A screenshot of the Azure portal's search interface. The search bar at the top contains the text "keyva". Below the search bar, the "Services" section is visible, with "Key vaults" highlighted by a blue selection bar.

- Click "Create" to create a new Key Vault

#### Key vaults Microsoft Non-Production

+ Create  Manage deleted

Filter for any field...

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

Home > Key vaults > Create key vault

#### Create key vault

##### Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*

Free Trial

Resource group \*

(New) KeyVaultDsc

[Create new](#)

##### Instance details

Key vault name \*

M365DscAzureKeyVault

Region \*

(Europe) West Europe

Pricing tier \*

Standard

Soft delete

Enable Disable

Retention period (days) \*

90

Purge protection

Enable Disable

Review + create

< Previous

Next : Access policy >

<sup>4</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

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

The screenshot shows the 'Create key vault' wizard in the Azure portal. At the top, a green bar indicates 'Validation passed'. Below it, the 'Basics' section lists configuration details:

Subscription	Free Trial
Resource group	KeyVaultDsc
Key vault name	M365DscAzureKeyVault
Region	West Europe
Pricing tier	Standard
Enable soft delete	Enabled
Enable purge protection	Enabled
Retention period (days)	90 days

The 'Access policy' section shows:

Azure Virtual Machines for deployment	Disabled
Azure Resource Manager for template deployment	Disabled
Azure Disk Encryption for volume encryption	Disabled
Permission model	Vault access policy
Access policies	1

The 'Networking' section shows:

Connectivity method	Public endpoint (all networks)
---------------------	--------------------------------

At the bottom, there are 'Previous' and 'Next' buttons, and a prominent blue 'Review + create' button.

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

The screenshot shows the 'M365DscAzureKeyVault | Overview' page in the Azure portal. A message box says 'Your deployment is complete'. Deployment details are listed:

Deployment name	M365DscAzureKeyVault	Start time	4/23/2020, 2:15:08 AM
Subscription	Free Trial	Correlation ID	0ef7c008-0087-47f1-a062-61e312b321ac
Resource group	KeyVaultDsc		

Under 'Next steps', there is a blue 'Go to resource' button.

- Click "Access policies" and click "Create"

The screenshot shows the 'Access policies' blade in the Azure portal. On the left, 'Access policies' is highlighted with a red box. At the top, there is a blue '+ Create' button.

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

- 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
<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".

① Permissions    ② 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-e694e127d70c
--

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

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

The screenshot shows the 'Review + create' step of creating a new Azure Key Vault. At the top, there are four status indicators: 'Permissions' (green checkmark), 'Principal' (green checkmark), 'Application (optional)' (green checkmark), and 'Review + create' (blue outline). Below these, the 'Key Permissions' section lists four operations: 'Key Management Operations' (None selected), 'Cryptographic Operations' (None selected), 'Privileged Key Operations' (None selected), and 'Rotation Policy Operations' (None selected). The 'Secret Permissions' section lists two operations: 'Secret Management Operations' (None selected) and 'Privileged Secret Operations' (None selected). The 'Certificate Permissions' section lists two operations: 'Certificate Management Operations' (None selected) and 'Privileged Certificate Operations' (None selected). The 'Principal' section shows the principal name as 'Microsoft365DSC\_DeVopsPipeline' and the object ID as 'fa1360ae-a384-441f-88d8-53bd5307a393'. The 'Application' section shows the authorized application as 'None selected' and the object ID as 'None selected'. At the bottom, there are 'Previous' and 'Create' buttons, with the 'Create' button being highlighted with a red box.

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

The screenshot shows the Azure portal interface with the URL 'Home > M365DscAzureKeyVault | Overview > M365DscAzureKeyVault | Access policies'. A success message box is displayed in the center of the page, stating: 'Updating the key vault 'M365DscAzureKey... 2:34 AM. The key vault 'M365DscAzureKeyVault' has been successfully updated.' The message box has a red border around it.

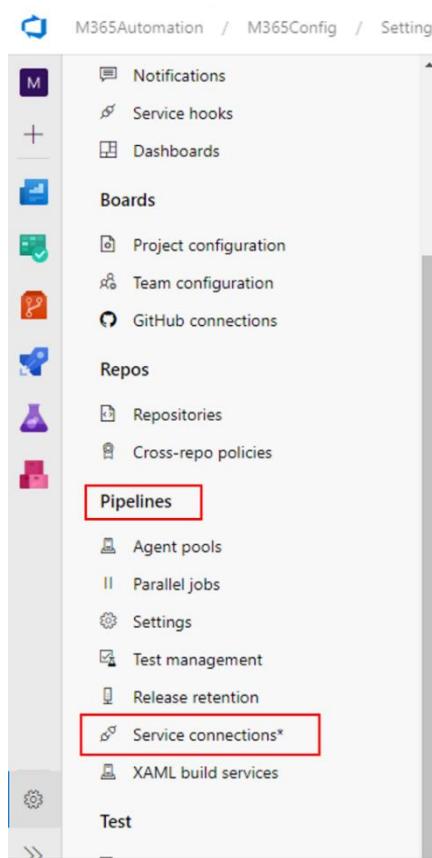
### 4.5.4 Adding a Service Connection to Azure to the Azure DevOps project

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:

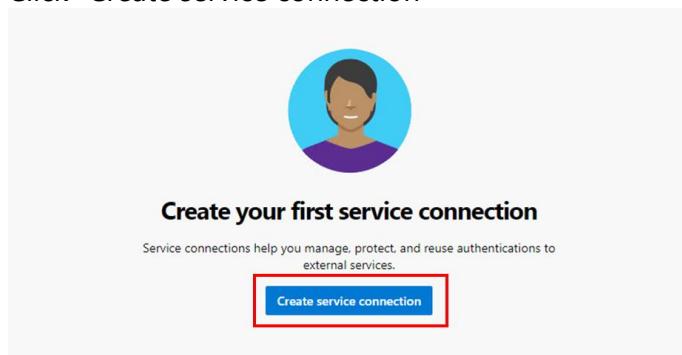
- Open the Azure DevOps Portal
- Browse to your project
- Click "Project Settings" in the lower left corner

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

- Scroll to the "Pipelines" section and select "Service connections\*"

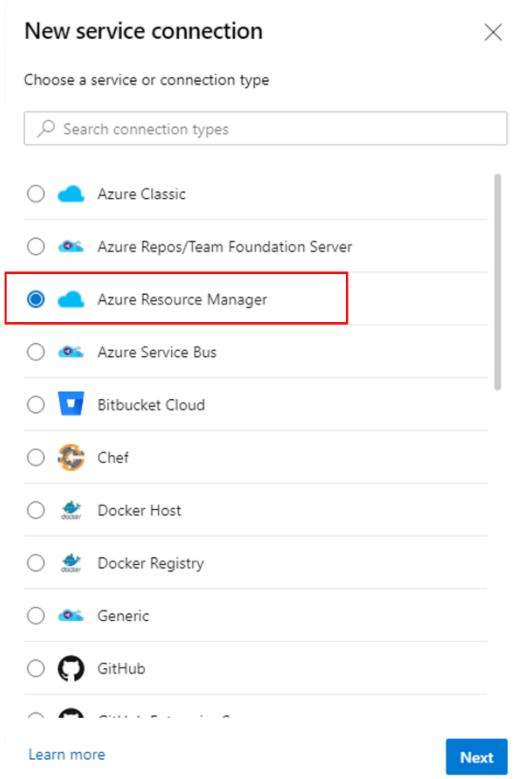


- Click "Create service connection"

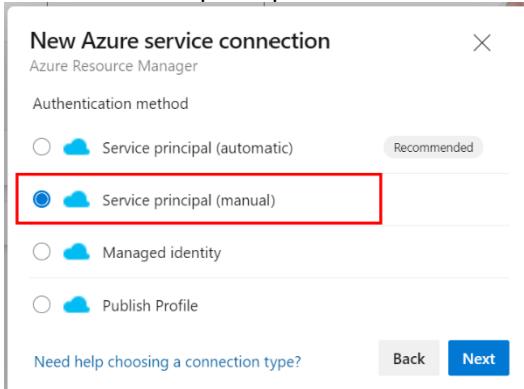


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

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



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



NOTE: We are using the already created Service Principal Name

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

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



- 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.

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

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

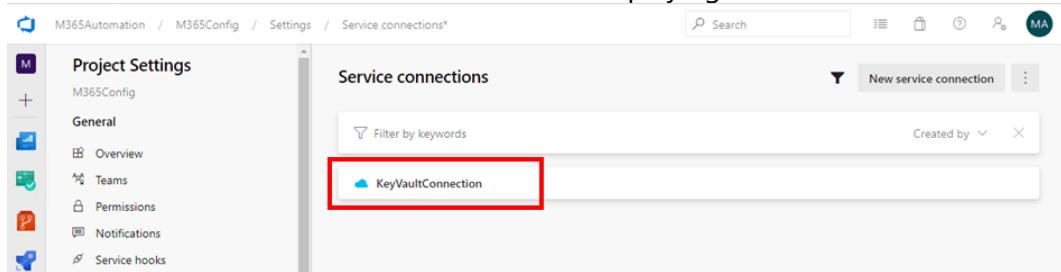
Service connection name  
KeyVaultConnection

Description (optional)

Security  
 Grant access permission to all pipelines

Learn more    Back    **Verify and save**    ▾

- The "Service connection" is now created and displaying



The screenshot shows the 'Service connections' page in the Azure DevOps interface. On the left, there's a sidebar with 'Project Settings' for 'M365Config'. The main area displays a list of service connections under 'Service connections'. A red box highlights the 'KeyVaultConnection' entry, which has a blue icon next to it. The page includes standard navigation elements like 'Search', 'Back', and 'Verify and save'.

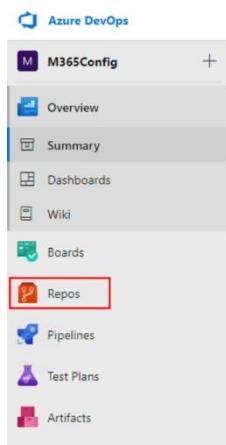
## 5 Configuring Azure DevOps

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

### 5.1 Populate scripts

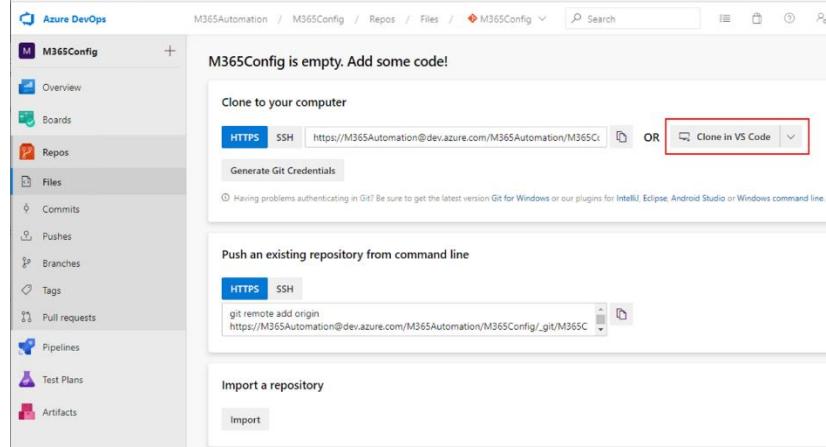
The newly created Azure DevOps project contains a Git repository to which all scripts of this solution must be added. In this step we will upload the scripts of the solution to the repository in Azure DevOps:

- 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

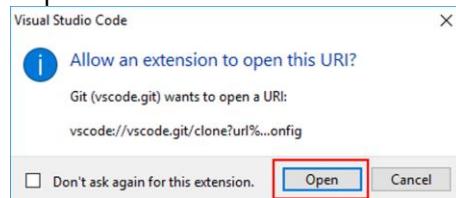


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

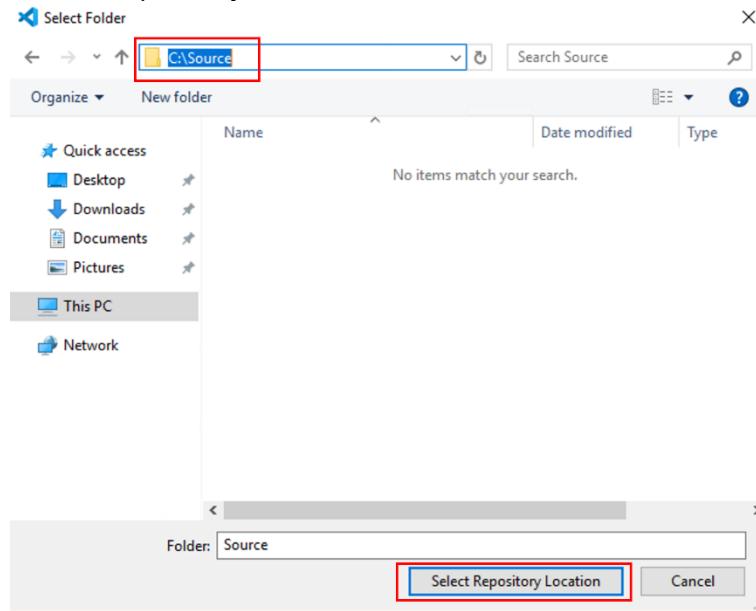
- Click on "Clone in VS Code" (acknowledge any browser notifications for opening any files)



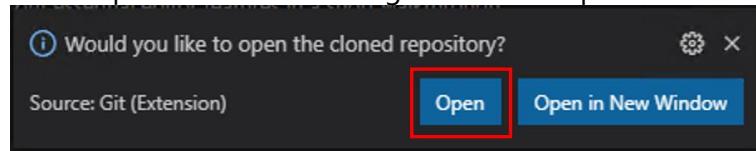
- Acknowledge that Visual Studio Code can open the external URL by clicking "Open"



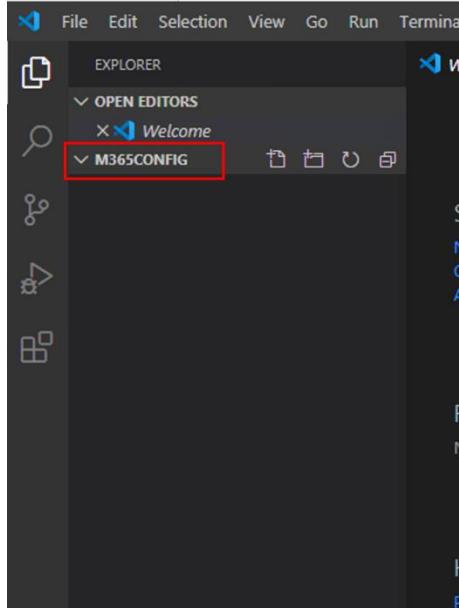
- Select "C:\Source" as the source folder (create if it does not exist) and select "Select Repository Location"



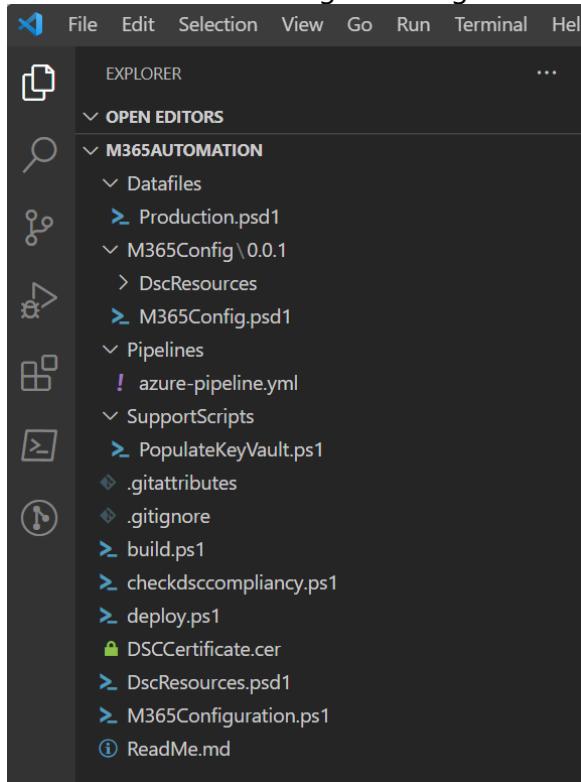
- Login with your Microsoft 365 admin account
- Click "Open" in the bottom right corner to open the cloned folder



- 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.ps1” 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>' [Placeholder]
15             OrganizationName = '<tenantURL>' [Placeholder]
16         }
17         Accounts        = @(
18             @{
19                 Workload = 'Exchange'
20                 Account  = '<M365AdminAccount>' [Placeholder]
21             }
22             @{
23                 Workload = 'Office365'
24                 Account  = '<M365AdminAccount>' [Placeholder]
25             }
26             @{
27                 Workload = 'PowerPlatform'
28                 Account  = '<M365AdminAccount>' [Placeholder]
29             }
30             @{
31                 Workload = 'SecurityCompliance'
32                 Account  = '<M365AdminAccount>' [Placeholder]
33             }
34             @{
35                 Workload = 'SharePoint'
36                 Account  = '<M365AdminAccount>' [Placeholder]
37             }
38             @{
39                 Workload = 'Teams'
40                 Account  = '<M365AdminAccount>' [Placeholder]
41             }
42         )
43         AppCredentials = @{
44             @{
45                 Workload      = 'Exchange'
46                 ApplicationId = '<appid>' [Placeholder]
47                 CertThumbprint = '<certthumb>' [Placeholder]
48             }
49             @{
50                 Workload      = 'Office365'
51                 ApplicationId = '<appid>' [Placeholder]
52                 CertThumbprint = '<certthumb>' [Placeholder]
53             }
54             @{
55                 Workload      = 'PowerPlatform'
56                 ApplicationId = '<appid>' [Placeholder]
57                 CertThumbprint = '<certthumb>' [Placeholder]
58             }
59             @{
60                 Workload      = 'SecurityCompliance'
61                 ApplicationId = '<appid>' [Placeholder]
62                 CertThumbprint = '<certthumb>' [Placeholder]
63             }
64             @{
65                 Workload      = 'SharePoint'
66                 ApplicationId = '<appid>' [Placeholder]
67                 CertThumbprint = '<certthumb>' [Placeholder]
68             }
69             @{
70                 Workload      = 'Teams'
71                 ApplicationId = '<appid>' [Placeholder]
72                 CertThumbprint = '<certthumb>' [Placeholder]
73             }
74         )
75     )
76 }
```

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

```
96     AcceptedDomains      = @(
97         @{
98             Identity      = '<tenantURL>'
99             DomainType   = 'Authoritative'
100            MatchSubDomains = $false
101            OutboundOnly   = $false
102            Ensure        = 'Present'
103        }
104    )
105
106    DKIM          = @{
107        @{
108            Identity      = '<tenantURL>'
109            Enabled       = $true
110            AdminDisplayName = ''
111            BodyCanonicalization = 'Relaxed'
112            HeaderCanonicalization = 'Relaxed'
113            KeySize      = 1024
114        }
115    }
```

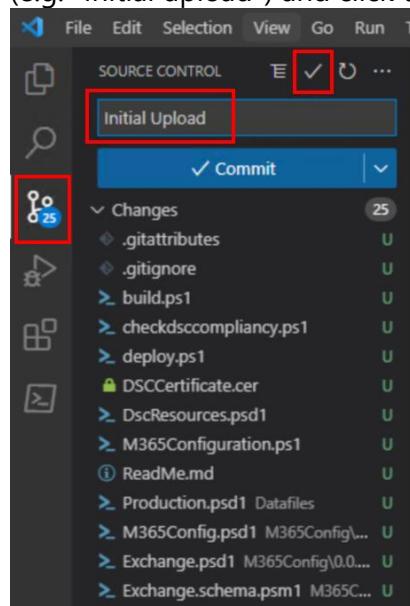
- Open the file "build.ps1" and update the \$VaultName variable on line 27 with the name of the Azure Key Vault you have created in paragraph 4.5.3

```
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

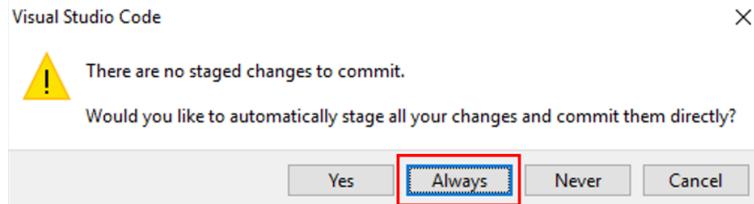
```
30 ##### SCRIPT VARIABLES #####
31
32 # Azure variables
33 $VaultName = '<Your KeyVault>'
```

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



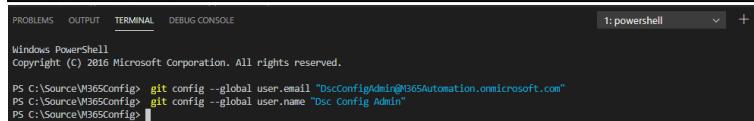
## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

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

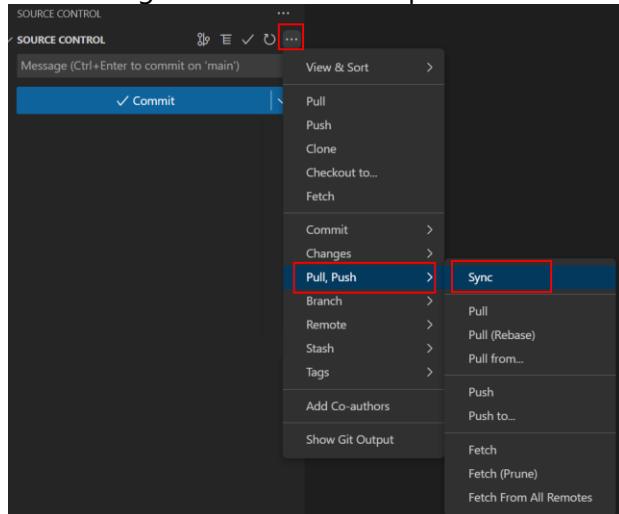


- If you get an error about an unknown e-mail address, run the following commands with your own information and retry the commit:

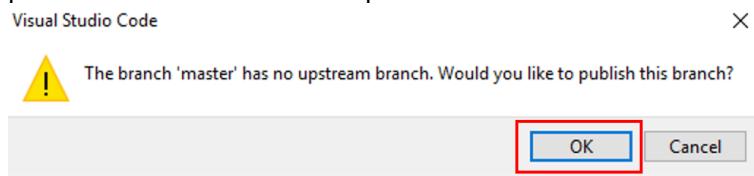
```
git config --global user.email <email>
git config --global user.name "<your_name>"
```



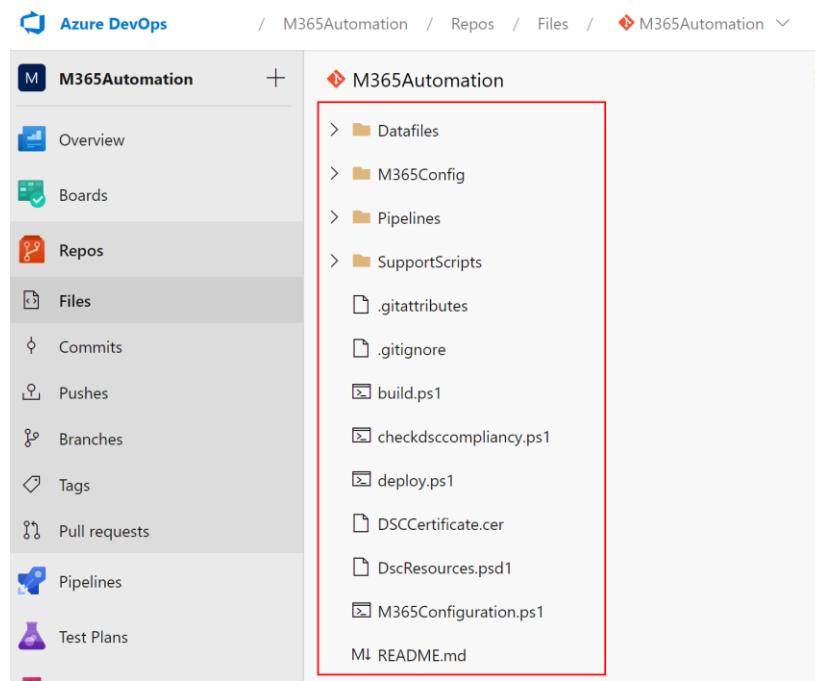
- Click the three dots icon and select "Push, Pull > Sync" to synchronize your local changes with Azure DevOps



- 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.psd1) 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.psd1. 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.

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

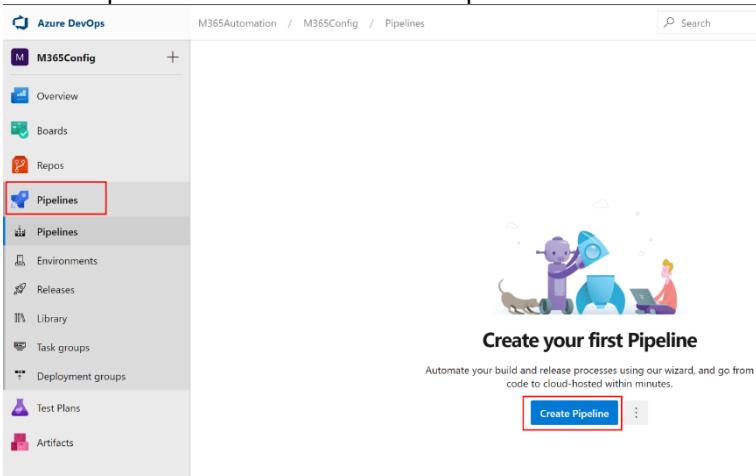
- This script will now read the data file and ask for all passwords and certificates it finds. If a secret is already present in the Key Vault, you are asked if you want to overwrite it or not.

### 5.3 Configure Azure DevOps project

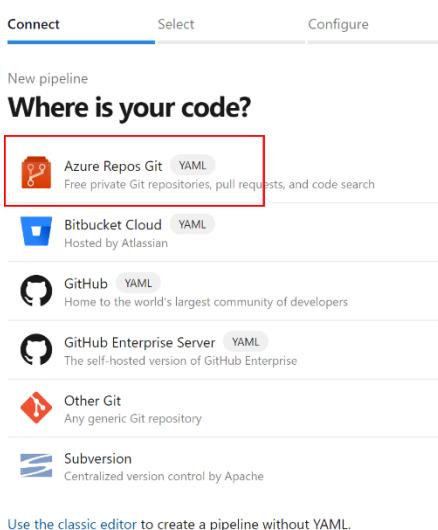
#### 5.3.1 Create Build pipeline

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"

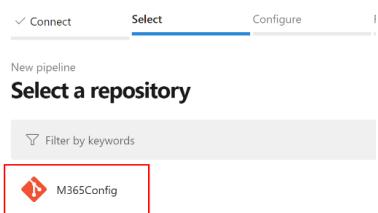


- Select "Azure Repos Git"

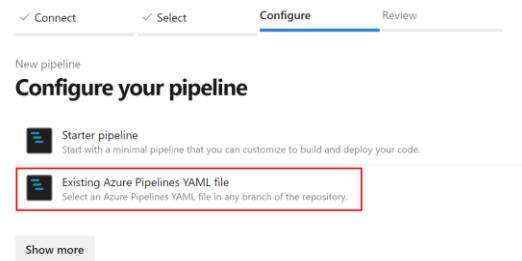


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

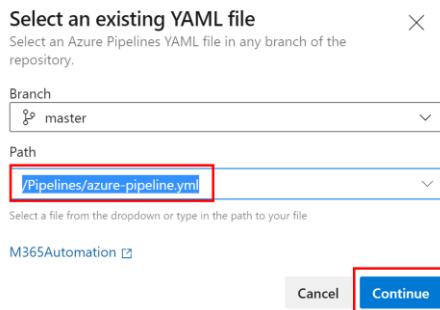
- Click the name of your Project



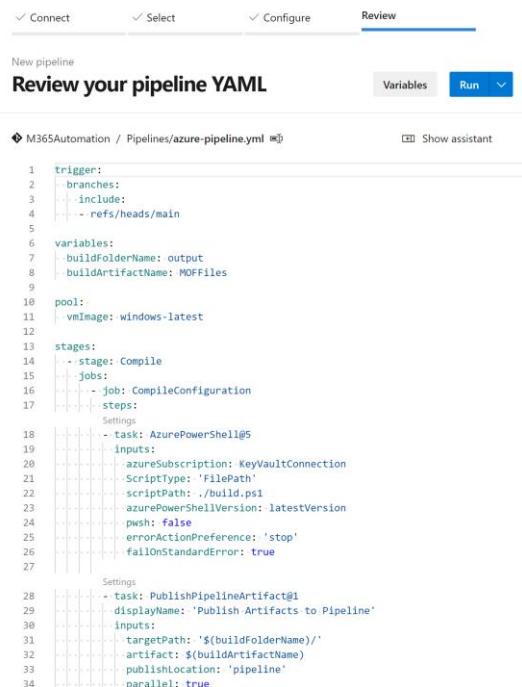
- Select the "Existing Azure Pipelines YAML file"



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

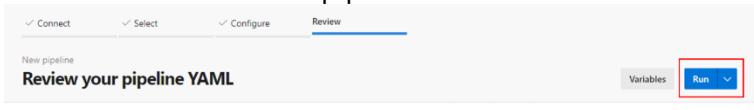


- The pipeline then shows you the azure-pipelines.yml file you uploaded in a previous step



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

- 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

A screenshot of the Azure DevOps pipeline run details screen. At the top, it says 'Manually run by Yorick Kuijs' and 'View 2 changes'. Below that, it shows repository and version information: 'M365Automation' at 'master' commit 'addf3703'. It also shows the time started and elapsed: 'Today at 2:19 PM'. Under 'Related', it lists '0 work items' and '0 artifacts'. Under 'Tests and coverage', it has a 'Get started' link. A red box highlights a warning message: '⚠ This pipeline needs permission to access a resource before this run can continue to Compile' with a 'View' button next to it. Below this, there's a 'Jobs' section with a 'Permission needed' message, a table with one row ('CompileConfiguration' status 'Waiting'), and a 'Cancel' button.

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

Waiting for review X

Compile



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

Permit access? X

Granting permission here will permit the use of Service connection 'KeyVaultConnection' for all waiting and future runs of this pipeline.



## 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

A screenshot of the Azure DevOps 'Jobs' page. It shows a table with columns 'Name', 'Status', and 'Duration'. A single row is highlighted with a red border, representing a pipeline run named 'CompileConfiguration' which is currently 'Running' for 195 seconds.

- Check if the pipeline has completed successfully

A screenshot of the Azure DevOps 'Pipelines' page. It shows a list of recently run pipelines. One pipeline, 'M365Config', is highlighted with a red box. Below it, a table shows three specific runs with their details: '#20200423.1 Initial upload' (4m ago), '#20200423.2 Changing pool' (23m ago), and '#20200423.3 Changing pool' (8m ago).

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

A screenshot of the Azure DevOps 'Runs' page for the 'M365Config' pipeline. It lists three runs. The third run, '#20200423.3 Changing pool', is selected and shown in more detail. A red box highlights the 'Stages' column, which displays three stages: 'b23ecbf' (green circle), 'b23ecbf' (red circle), and '962021d' (green circle). Each stage has a timestamp indicating when it was completed.

- 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.

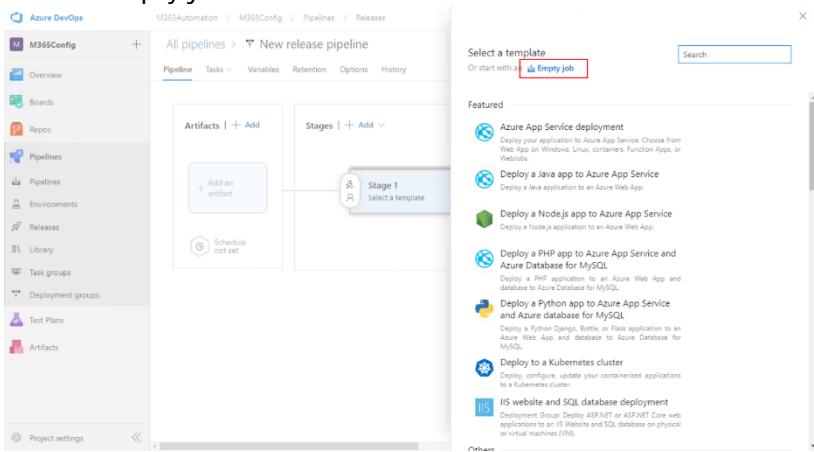
In this step we are going to configure this release pipeline:

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

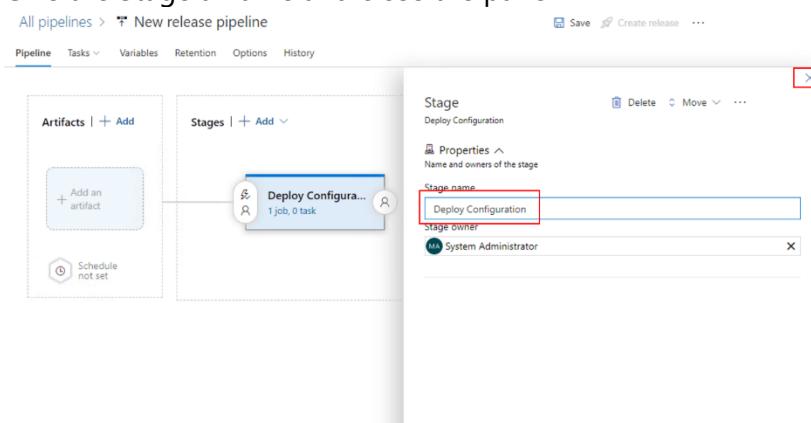
A screenshot of the Azure DevOps 'Pipelines' page. The left sidebar shows 'Releases' selected with a red box. The main area displays a message 'No release pipelines found' with a 'New pipeline' button highlighted with a red box. The URL in the browser bar is 'M365Automation / M365Config / Pipelines / Releases'.

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

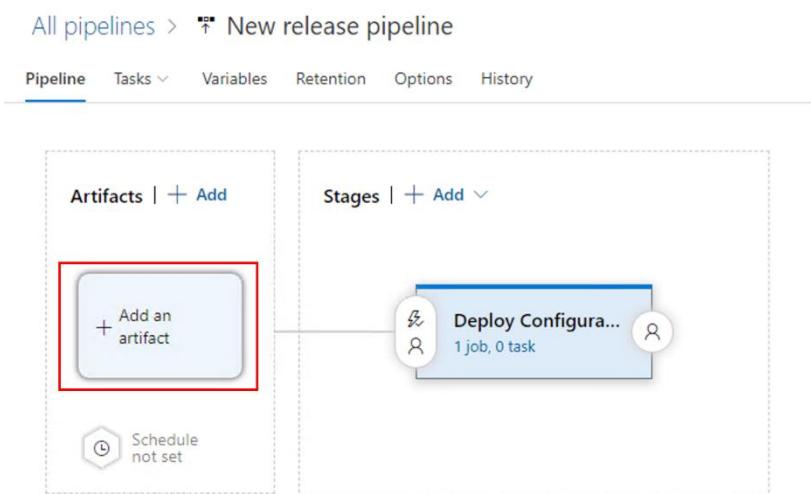
- Select "Empty job"



- Give the Stage a name and close the pane

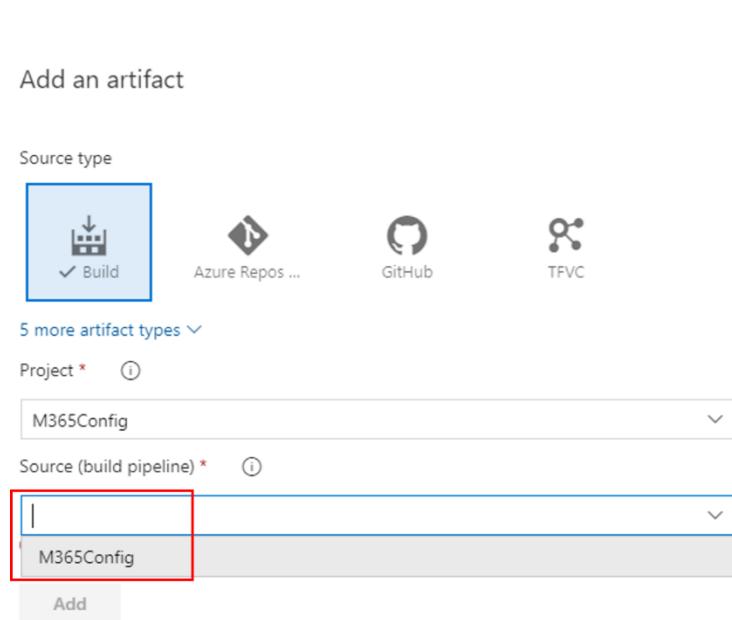


- Click "Add an artifact"



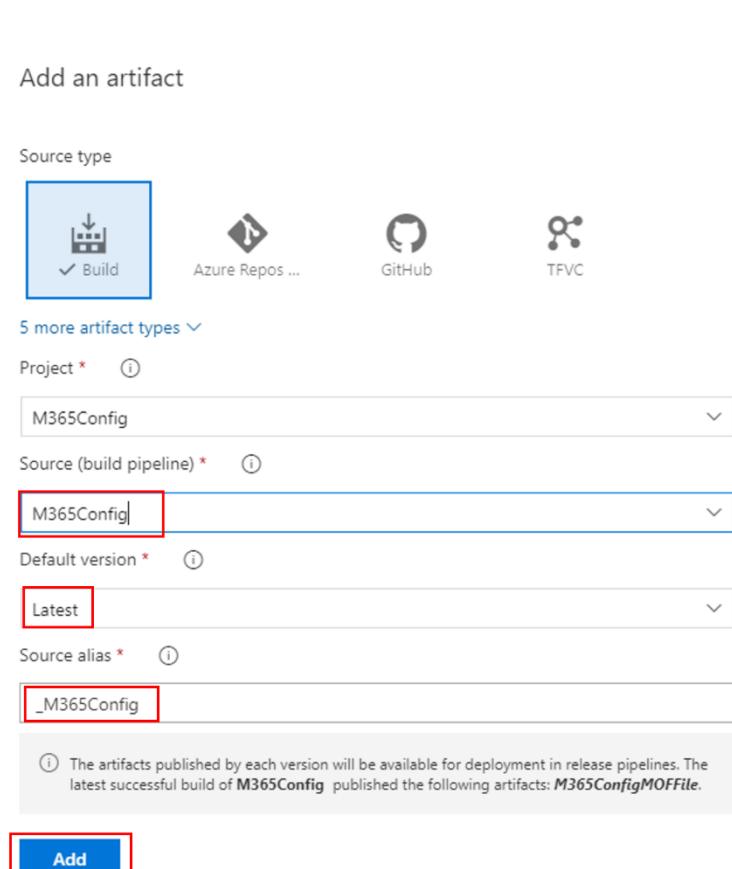
## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

- Under "Source" select the build pipeline



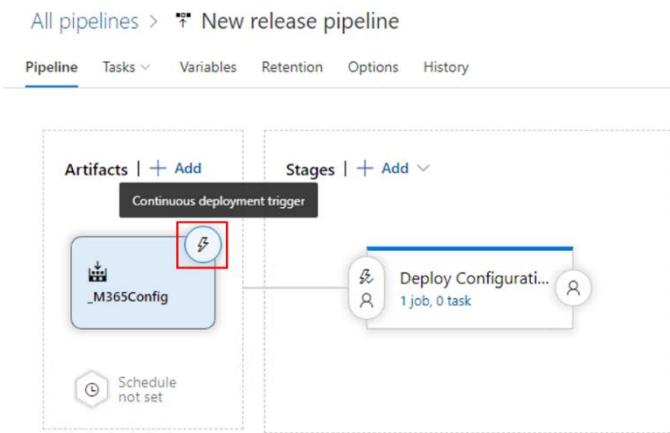
- After selecting the Source, more options will appear. Leave them as default and click "Add".

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

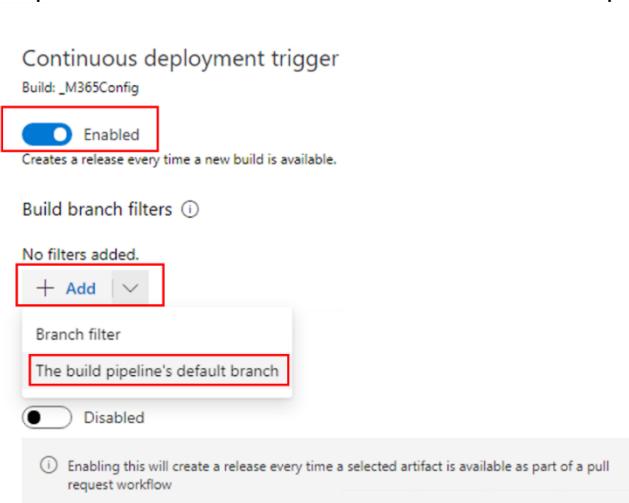


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

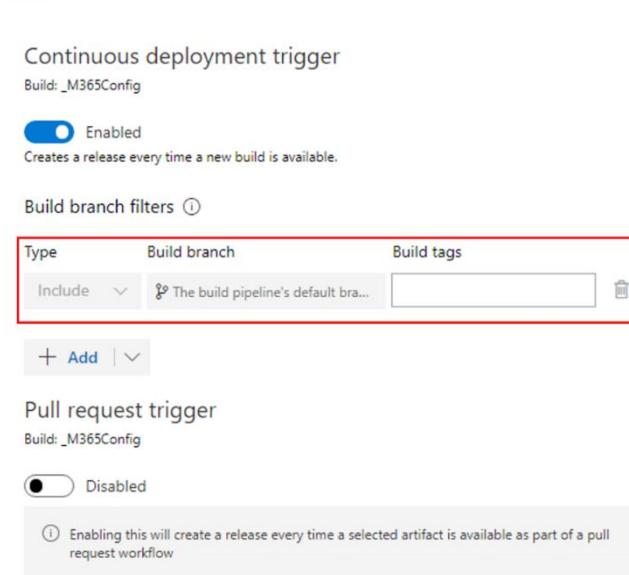
- Click 'Add'
- Configure the Release pipeline triggers by clicking the Lightning icon next to Artifacts



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

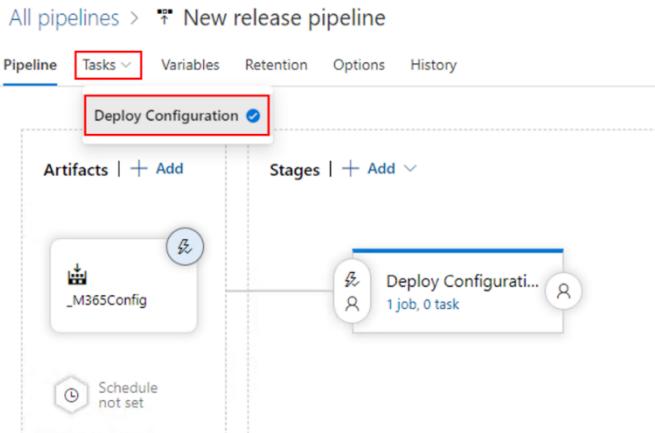


- Make sure the branch has been added successfully and close the pane

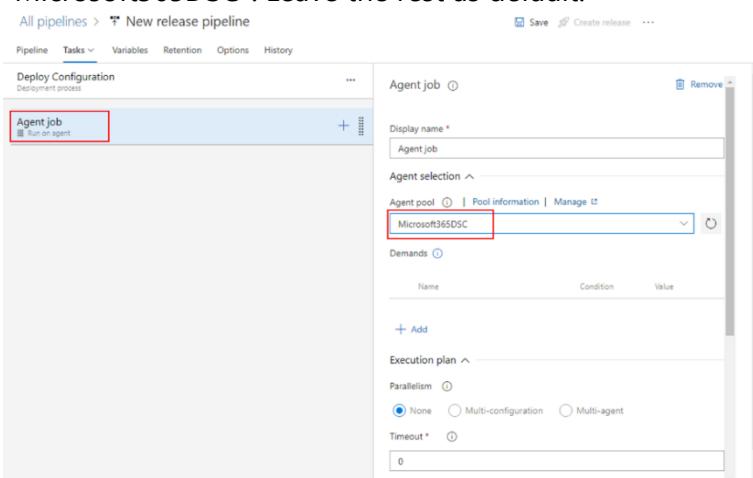


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

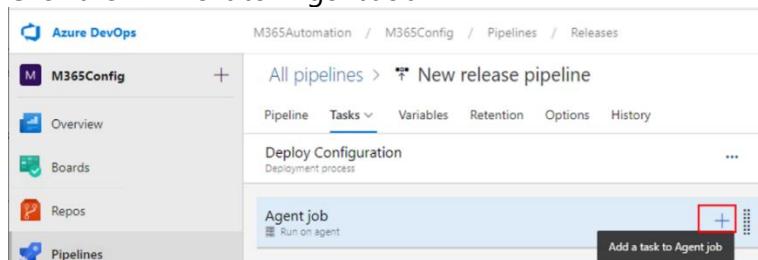
- Select "Tasks > <Stage name>"



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



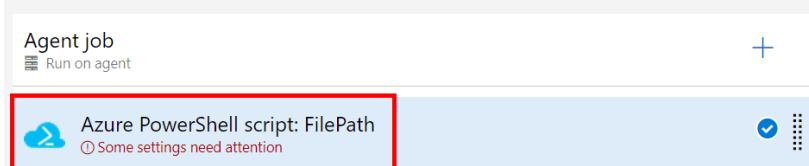
- Click the "+" next to "Agent Job"



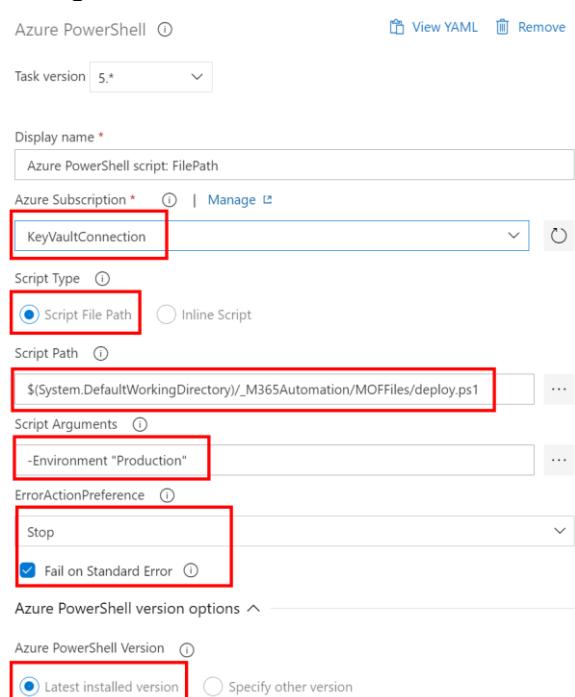
- Search for "Azure PowerShell", select "Azure PowerShell" and click "Add"



- 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"

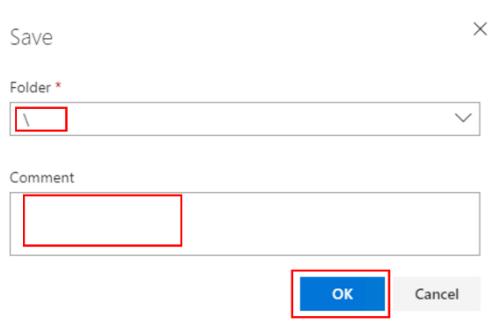


- Open the "Advanced" section and make sure the Working Directory matches the path of the previous step (copy/paste if required).



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

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



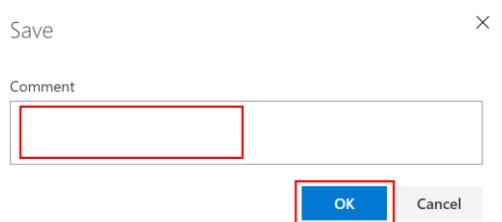
- Hover over the "New release pipeline" name and click on the pen icon that appears behind the title



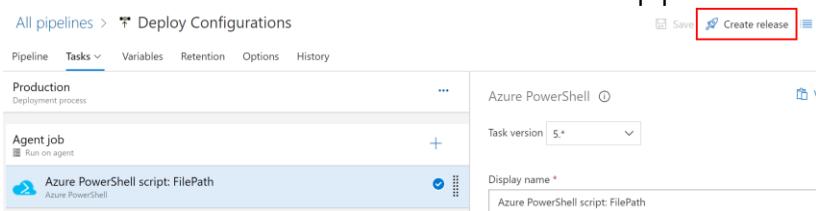
- Enter a name for the pipeline, like "Deploy Configurations" and click "Save"

All pipelines > Deploy Configurations  ...

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

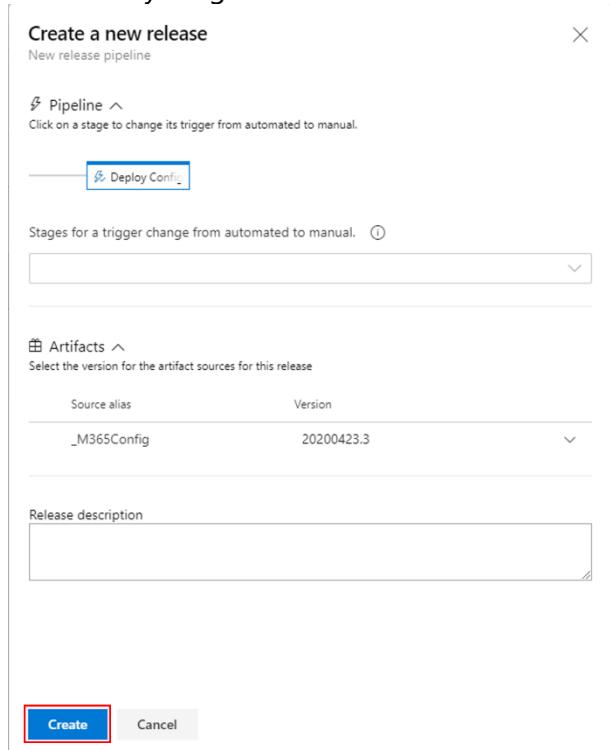


- Click "Create release" to test the created Release pipeline.



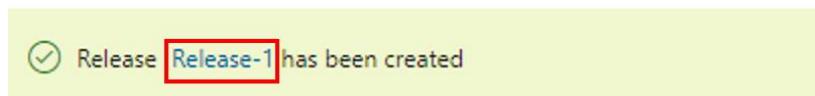
## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

- Leave everything as default and click "Create"



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

All pipelines > New release pipeline

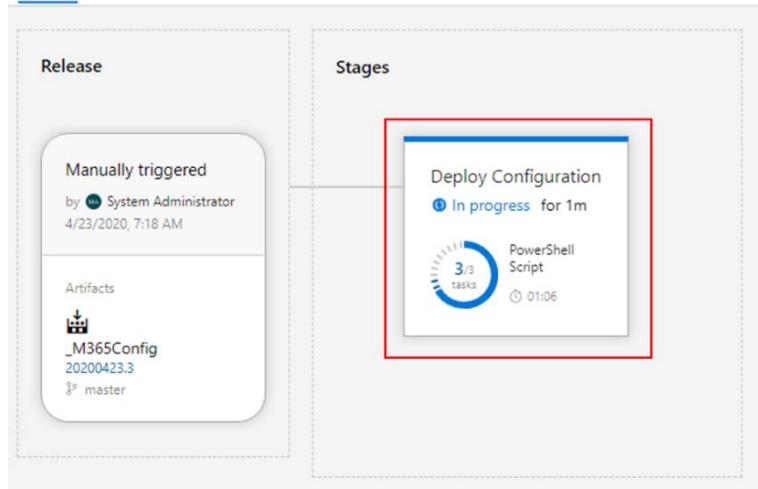


- Review the progress

M365Automation / M365Config / Pipelines / Releases / New release pipeline / Release-1

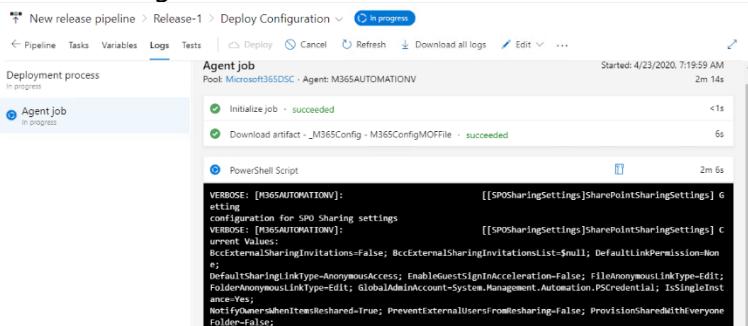
New release pipeline > Release-1

Pipeline Variables History | + Deploy Cancel Refresh Edit ...

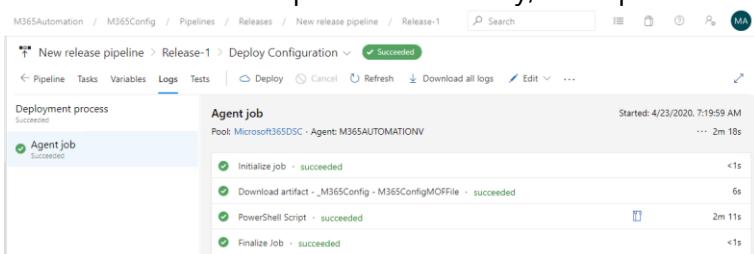


# Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

- 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

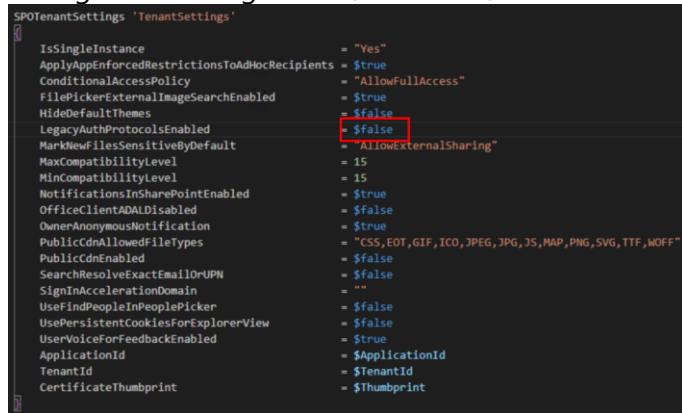
A screenshot of the SharePoint Admin Center's 'Access control' page. The 'Policies' section is selected. A red box highlights the 'Access control' link under 'Sharing'. On the right, a modal dialog titled 'Apps that don't use modern authentication' is open, showing two options: 'Allow access' (selected) and 'Block access'. Below the dialog, there are sections for 'Unmanaged devices', 'Idle session sign-out', 'Network location', and 'Limit OneDrive access'. A red box highlights the 'Allow access' button in the dialog.

- 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"
    ApplyAppForcedRestrictionsToAdHocRecipients = $true
    ConditionalAccessPolicy = "AllowFullAccess"
    FilePickerExternalImageSearchEnabled = $true
    HideDefaultThemes = $false
    LegacyAuthProtocolsEnabled = $true
    MarkNewFilesSensitiveByDefault = "AllowExternalSharing"
    MaxCompatibilityLevel = 15
    MinCompatibilityLevel = 15
    NotificationsInSharePointEnabled = $true
    OfficeClientAtALDDisabled = $false
    OwnerAnonymousNotification = $true
    PublicCdnAllowedFileTypes = "CSS,EOT,GIF,ICO,JPEG,JPG,JS,MAP,PNG,SVG,TTF,WOFF"
    PublicCdnEnabled = $false
    SearchResolveExactEmailOrUPN = ""
    SignInAccelerationDomain = $false
    UseFindPeopleInPeoplePicker = $true
    UsePersistentCookiesForExplorerView = $true
    UserVoiceForFeedbackEnabled = $true
    ApplicationId = $ApplicationId
    TenantId = $TenantId
    CertificateThumbprint = $Thumbprint
}
```

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

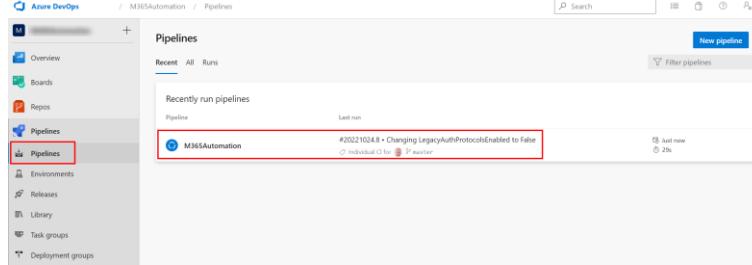
- Change this setting from "\$true" to "\$false"



```
SPOTenantSettings 'TenantSettings'
{
    IsSingleInstance = "Yes"
    ApplyAppForcedRestrictionsToAdHocRecipients = $true
    ConditionalAccessPolicy
    FilepickerExternalImageSearchEnabled
    HideDefaultThemes
    LegacyAuthProtocolsEnabled
    MarkNewFilesSensitiveByDefault
    MaxCompatibilityLevel
    MinCompatibilityLevel
    NotificationsInSharePointEnabled
    OfficeClientAddDisabled
    OwnerAnonymousNotification
    PublicCdnAllowedFileTypes
    PublicCdnEnabled
    SearchResolveExactEmailOrUPN
    SignInAccelerationDomain
    UseFindPeopleInPeoplePicker
    UsePersistentCookiesForExplorerView
    UserVoiceForFeedbackEnabled
    ApplicationId
    TenantId
    CertificateThumbprint
```

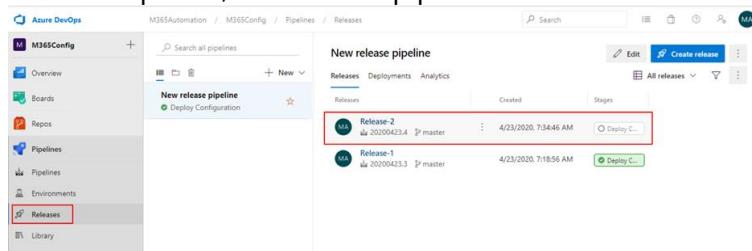
The screenshot shows a PowerShell script block named 'SPOTenantSettings' with a single parameter 'TenantSettings'. Inside the block, there is a line of code: '\$LegacyAuthProtocolsEnabled = \$true'. A red box highlights the '\$true' value, indicating it needs to be changed to '\$false'.

- 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

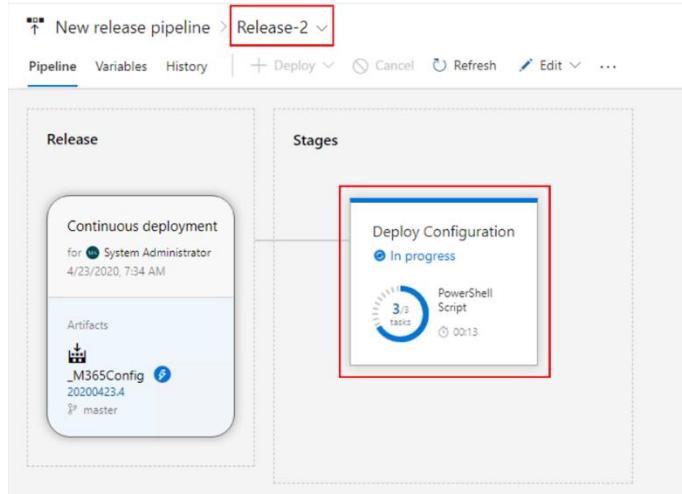


The screenshot shows the 'Pipelines' page in Azure DevOps. The left sidebar has 'Pipelines' selected. In the main area, under 'Recently run pipelines', there is a card for 'M365Automation' with a red box around it. The card shows the commit hash '#2021024.8 + Changing LegacyAuthProtocolsEnabled to False' and the status 'Just now'.

- Once completed, the Release pipeline should automatically start



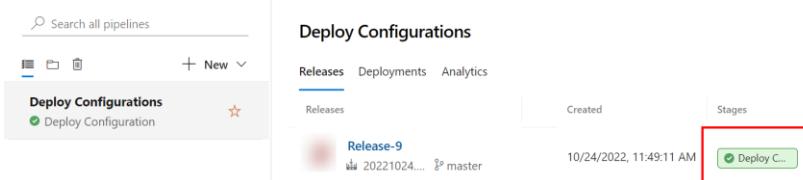
The screenshot shows the 'Releases' page in Azure DevOps. The left sidebar has 'Releases' selected. In the main area, there are two release cards: 'Release-2' (status 'In progress') and 'Release-1' (status 'Completed').



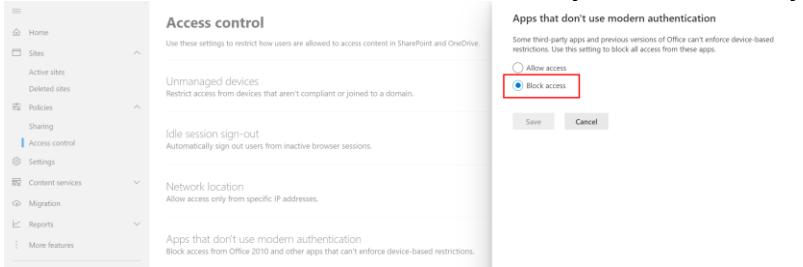
The screenshot shows the 'New release pipeline' page for 'Release-2'. The top navigation bar has 'Release-2' selected. The main area is divided into 'Release' and 'Stages' sections. The 'Release' section shows a 'Continuous deployment' for 'System Administrator' on '4/23/2020, 7:34 AM'. The 'Stages' section shows a 'Deploy Configuration' stage with a status of 'In progress'. A red box highlights this stage. Below it, there is a 'PowerShell Script' step with a duration of '00:13'.

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

- 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 Compliancy 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.

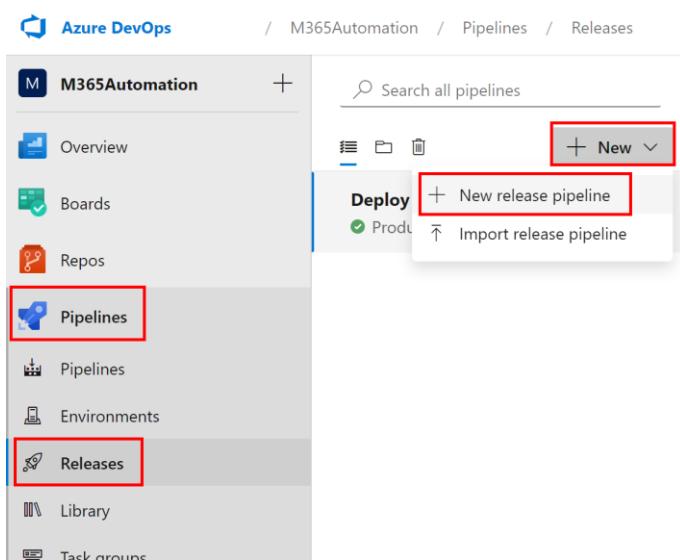
```
##### GENERIC VARIABLES #####
$workingDirectory = $PSScriptRoot
$encounteredError = $false

$useMail = $true
$mailAppId = '<APPID>'
$mailAppSecret = '<SECRET>'
$mailTenantId = '<TENANTID>'
$mailFrom = '<FROM>'
$mailTo = '<TO>'

$useTeams = $true
$teamsWebhook = '<WEBHOOK>'
```

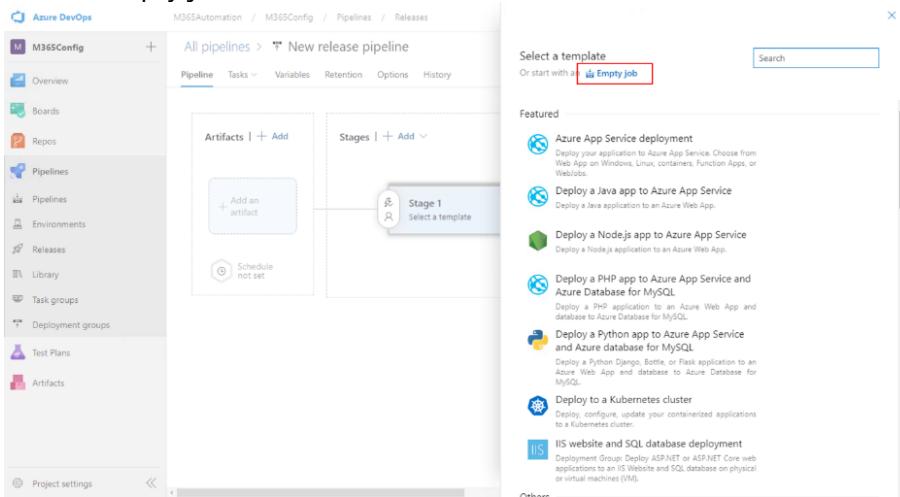
Now configure the Test release pipeline:

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

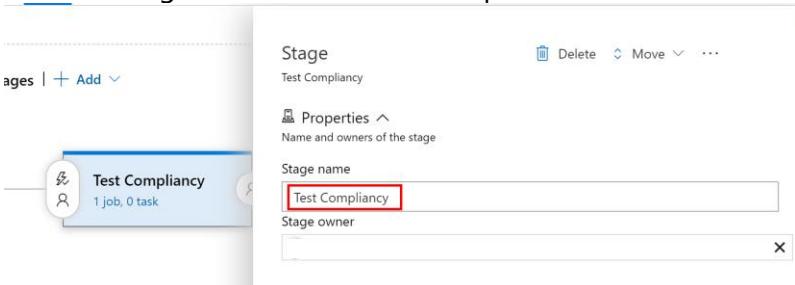


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

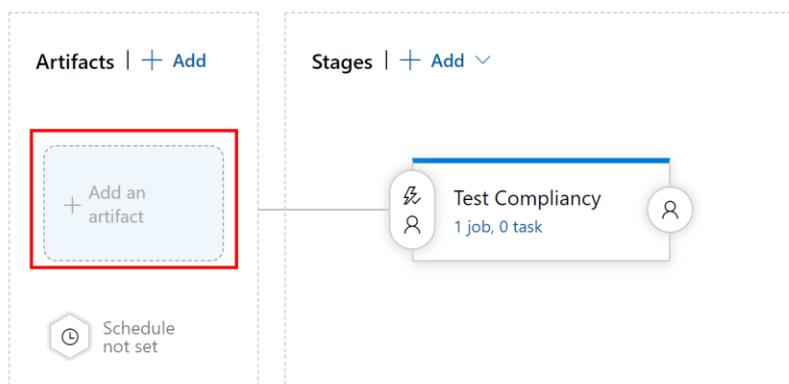
- Select "Empty job"



- Give the Stage a name and close the pane

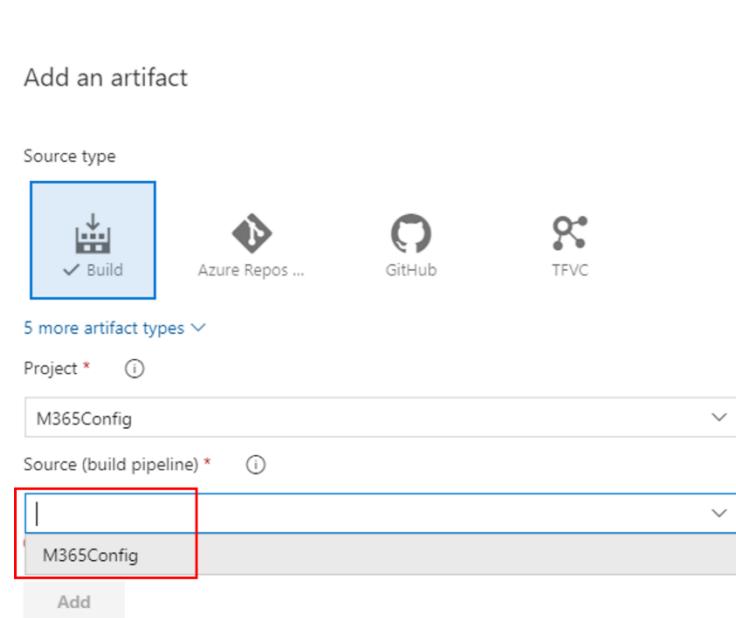


- Click "Add an artifact"

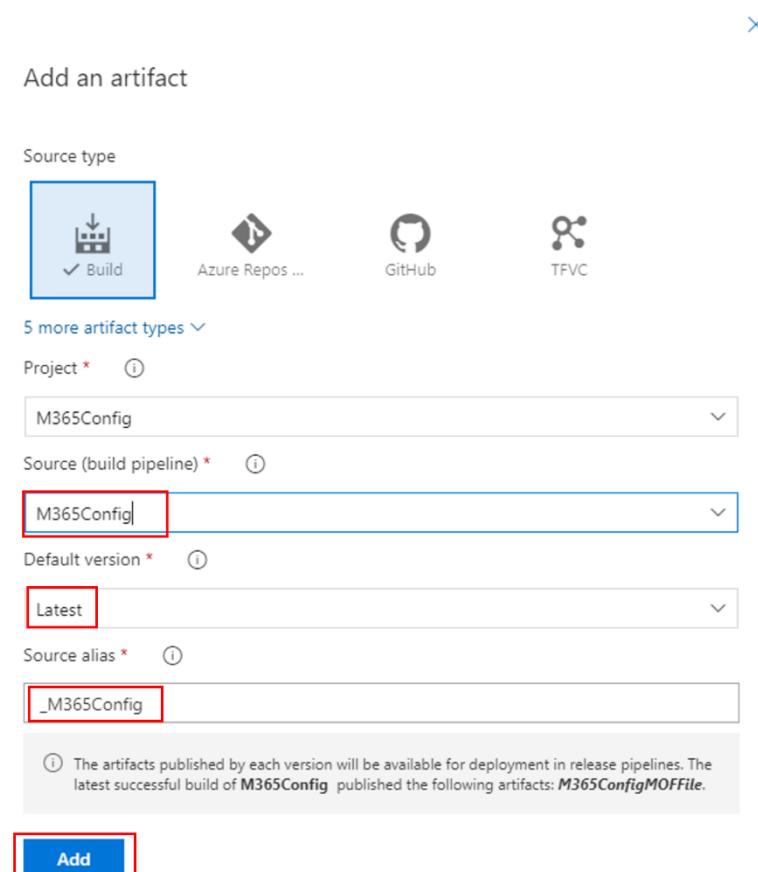


## Managing Microsoft 365 in true DevOps style with Microsoft365DSC and Azure DevOps

- Under "Source" select the build pipeline

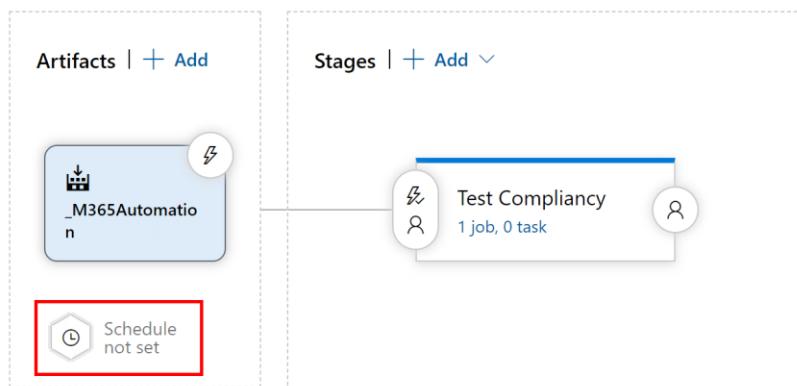


- After selecting the Source, more options will appear. Leave them as default and click "Add".



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

- 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

00h 00m (UTC) Coordinated Universal Time

Only schedule releases if the source or pipeline has changed

Mon through Sun at 6:00

06h 00m (UTC) Coordinated Universal Time

Only schedule releases if the source or pipeline has changed

Mon through Sun at 12:00

12h 00m (UTC) Coordinated Universal Time

Only schedule releases if the source or pipeline has changed

+ Add a new time

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

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

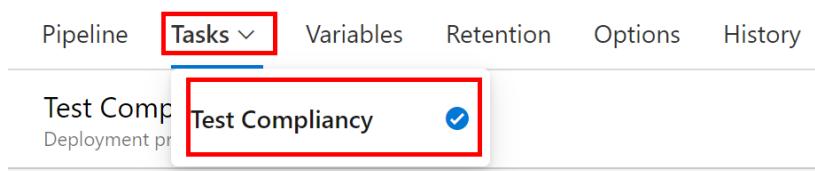
All pipelines > New release pipeline

New name:

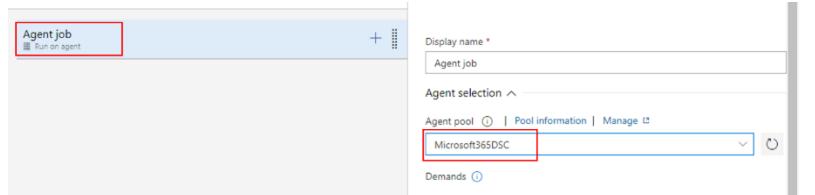
All pipelines > Test DSC Compliancy

- Select "Tasks > <Stage name>"

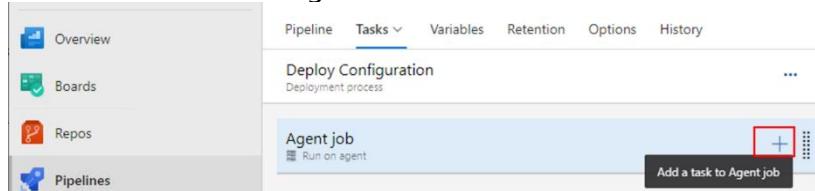
All pipelines > Test DSC Compliancy



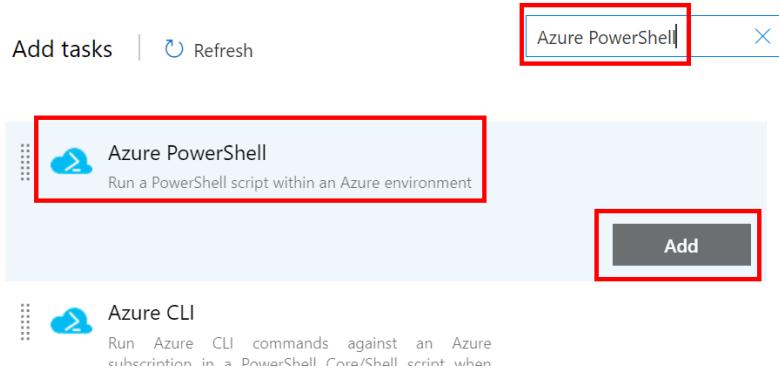
- Select the task "Agent job" in the left part of the pane and change the "Agent pool" to "Microsoft365DSC". Leave the rest as default.



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

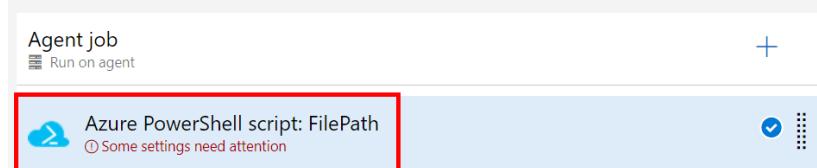


- Search for "Azure PowerShell", select "Azure PowerShell" and click "Add"



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

- Select the "Azure PowerShell" task



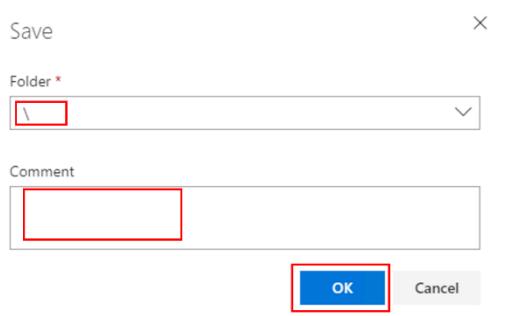
- In the configuration window for the Azure PowerShell task, configure the following settings:
  - Azure Subscription: "KeyVaultConnection" or whatever name you gave the service connection
  - Select "Script File Path" as "Type" and browse to the "checkdsccompliance.ps1" file by clicking the "..." button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

A screenshot of the Azure PowerShell task configuration window. The 'Task version' dropdown is set to 5.\*. The 'Display name' field contains 'Azure PowerShell script: FilePath'. The 'Azure Subscription' dropdown is set to 'KeyVaultConnection'. The 'Script Type' radio button is selected for 'Script File Path'. The 'Script Path' input field contains the path '\$(System.DefaultWorkingDirectory)/\_M365Automation/MOFiles/checkdsccompliance.ps1'. The 'ErrorActionPreference' dropdown is set to 'Stop' with the 'Fail on Standard Error' checkbox checked. Under 'Azure PowerShell version options', the 'Azure PowerShell Version' dropdown is set to 'Latest installed version'.

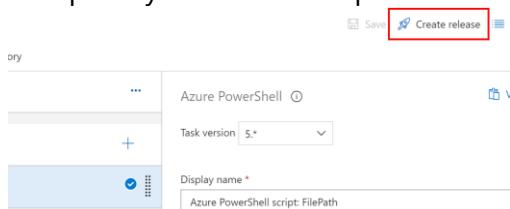
- Open the "Advanced" section and make sure the Working Directory is matching the path of the previous step.

A screenshot of the 'Advanced' section of the Azure PowerShell task configuration. The 'Working Directory' input field contains the path '\$(System.DefaultWorkingDirectory)/\_M365Automation/MOFiles'.

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



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

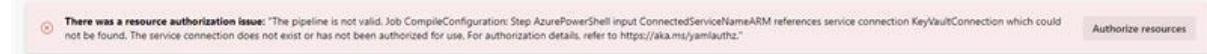


- When done and all is configured correctly, you will receive a notification either via Teams or via E-mail.

## 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"

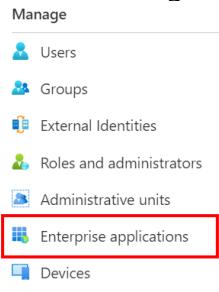
Steps

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

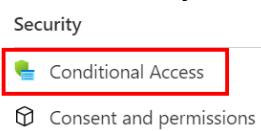


Azure Active  
Directory

- Under "Manage", click "Enterprise applications"

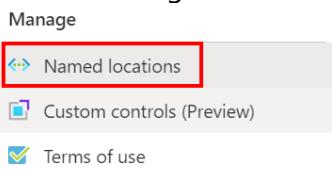


- Under "Security", click "Conditional Access"



- First, we are going to create a Named Location

- Under "Manage", click "Named locations"



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

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

- 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

Policies

[+ New policy](#)

[What If](#)

[Refresh](#)

[Got feedback?](#)

Insights and reporting

Diagnose and solve problems

Policy Name

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

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

- 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](#)

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

Name \*

Conditional Access for DSC Service Acco... ✓

Include

Exclude

None

All users

Select users and groups

Assignments

Users and groups ⓘ >  
Specific users included

All guest and external users ⓘ

Cloud apps or actions ⓘ >  
No cloud apps or actions selected

Directory roles ⓘ

Conditions ⓘ >  
0 conditions selected

Users and groups

Select

>

1 user

DSC Service Account  
dsc@M365x812127.onmicrosoft.com ... ⋮

Access controls

Grant ⓘ >  
0 controls selected

Session ⓘ >  
0 controls selected

- 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](#)

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

Select what this policy applies to

Cloud apps

User actions

Name \*

Conditional Access for DSC Service Acco... ✓

Include

Exclude

None

All cloud apps

Select apps

Assignments

Users and groups ⓘ >  
Specific users included

Cloud apps or actions ⓘ >  
All cloud apps

Conditions ⓘ >  
0 conditions selected

Access controls

Grant ⓘ >

**⚠** 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.



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

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

The screenshot shows the 'Conditional Access' policy creation interface. In the main pane, under 'Conditions', the 'Locations' section is expanded, showing 'Any location' selected. A red box highlights this selection. To the right, a 'Select' dialog box is open, titled 'Select Locations'. It shows a list of locations with 'Azure Self Hosted VMs' selected, indicated by a checked checkbox. A red box highlights this selection.

- Access controls > Grant
  - Select "Block access"

The screenshot shows the 'Conditional Access' policy creation interface. In the main pane, under 'Access controls', the 'Grant' section is expanded, showing 'Block access' selected. A red box highlights this selection. To the right, a 'Grant' dialog box is open, titled 'Grant'. It shows two options: 'Block access' (selected) and 'Grant access'. Below these, there are several optional checkboxes for requirements like MFA, device compliance, and app protection policies. A red box highlights the 'Block access' radio button in the dialog.

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

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

The screenshot shows the 'New Conditional access policy' wizard. At the top, there's a breadcrumb navigation: Home > Conditional Access >. The title 'New' is displayed above the main form area. Below the title, a subtitle reads 'Conditional access policy'. A descriptive text explains that the policy controls user access based on conditional access policy signals to make decisions and enforce organizational policies, with a 'Learn more' link.

**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**

A red box highlights the 'On' button under 'Enable policy', and another red box highlights the 'Create' button at the bottom.

- 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 is used by the Test DSC Compliancy 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.

<b>Folder: M365Config</b>	<b>Composite DSC resource used by the solution</b>
<b>Folder: DscResources</b>	
<b>Folder: 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.
<b>Folder: 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.
<b>Folder: 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.
<b>Folder: 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.
<b>Folder: 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.
<b>Folder: 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
<b>Folder: 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.
<b>Folder: 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