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

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

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

Date: December 2nd 2020
Version: v1.1
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.

© 2020 Microsoft Corporation. All rights reserved.

Changelog

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>November 1st 2020</td>
<td>First release</td>
</tr>
<tr>
<td>1.0.1</td>
<td>November 3rd 2020</td>
<td>Updated incorrect links</td>
</tr>
<tr>
<td>1.1</td>
<td>December 2nd 2020</td>
<td>Incorporated feedback from Zaki Semar Shahul</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added Azure Conditional Access for the used service account</td>
</tr>
</tbody>
</table>
Table of Contents

1 Introduction........................................................................................................................................4
2 Prerequisites.........................................................................................................................................5
3 Preparation ..........................................................................................................................................6
   3.1 Create a DSC account in Microsoft 365.........................................................................................6
   3.2 Create a new project in Azure DevOps............................................................................................6
   3.3 Create an Agent Pool in Azure DevOps............................................................................................6
   3.4 Create Personal Access Token..........................................................................................................9
   3.5 Configure Azure DevOps Agent on the virtual machine.................................................................11
   3.6 Configure Azure Key Vault.............................................................................................................15
      3.6.1 Create Service Principle Name.................................................................................................15
      3.6.2 Create Azure KeyVault...........................................................................................................16
      3.6.3 Add secrets to your Vault .........................................................................................................20
      3.6.4 Adding Service Connection to the Azure DevOps project.........................................................21
   3.7 Configure the Local Configuration Manager....................................................................................26
4 Configuring Azure DevOps ..................................................................................................................29
   4.1 Populate scripts...............................................................................................................................29
   4.2 Configure Azure DevOps project ....................................................................................................34
      4.2.1 Create Build pipeline...............................................................................................................34
      4.2.2 Create Release pipeline ............................................................................................................36
      4.2.3 Validate that changes to the config are deployed successfully.................................................44
5 Security Enhancements ......................................................................................................................48
   5.1 Using Azure Conditional Access to secure service account.........................................................48
6 Learning materials ..............................................................................................................................52
   6.1 Desired State Configuration ............................................................................................................52
   6.2 Microsoft365Dsc.............................................................................................................................53
   6.3 Git....................................................................................................................................................53
7 Acronyms ...............................................................................................................................................54
1 Introduction

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

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

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

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

The setup we are using is:
2 Prerequisites

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

- Windows Server 2016 or above
- .Net Framework 4.7 or higher
- PowerShell v5.1
- Up to date PowerShellGet:
  ```
  Install-PackageProvider Nuget –Force
  Install-Module –Name PowerShellGet –Force
  ```
  **Note:** If you run into issues downloading these updates, check out the following article: [https://devblogs.microsoft.com/powershell/powershell-gallery-tls-support/](https://devblogs.microsoft.com/powershell/powershell-gallery-tls-support/)
- A local account with administrative privileges, to deploy configurations from Azure DevOps

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

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

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

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

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

3.2 Create a new project in Azure DevOps
- Log into the Azure DevOps portal
- Create new project

![Create a project to get started]

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

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

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

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

Enter a Name (for example: Microsoft365Dsc) and Description for the new pool and click "Create".
Click the newly created pool to open the pool
Click the "New agent" button to open the required information to add a new agent

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

Extract the downloaded zip file to the C:\Agent folder
3.4 Create Personal Access Token

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

- Click "New Token" to create a new token
• Enter a Name and select next year (not possible to select more than a year) as Expiration

Click "Show all scopes" and click "Create" to create the token
• IMPORTANT: Copy and store the generated token in a secure place. You cannot retrieve the token at a later point in time.

3.5 Configure Azure DevOps Agent on the virtual machine

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

Administrator: Command Prompt

```cmd
C:\Agent>config.cmd
```
Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps

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

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

- Paste the Personal Access Token and press [Enter]
• Enter "Microsoft365Dsc" (use the name specified earlier) as the Agent Pool and press [Enter]

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

• The Agent checks some prerequisites. Press [Enter] to use the default work folder
• If prompted: Press Enter to acknowledge “N” for “Perform an unzip for each step”
• Type “Y” to run the agent as a service

Enter the created service account credentials and press [Enter]

The agent is being configured and started automatically.
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 agents is present and Online

3.6 Configure Azure Key Vault

3.6.1 Create Service Principle Name
- Log into the Azure Portal
- Open Cloud Shell
• Make sure you select the Bash shell and enter:
  "az ad sp create-for-rbac -n Microsoft365Dsc"

  ![Bash Shell Output](image)

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

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

• Copy this information and store it in a secure place

### 3.6.2 Create Azure KeyVault

• Log into the Azure Portal

• Enter "Keyvault" in the top search bar and select "Key vaults"

  ![Azure KeyVault Search](image)

• Click "Add" to create a new Key Vault

  ![Add Key Vault](image)
• Enter the desired Resource group, Name and Region and then click “Review + create”

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

• The KeyVault will be created
Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps

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

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

• Select the "Select principal" option, enter the Service Principal Name you created earlier in the search box on the right, select your principal and click "Select".
• Open the "Secret permissions" and select the "Get" and "List" permissions

• Validate that everything is configured correctly and click "Add"
• Validate that the access policy has been added and click "Save" to store the new policies.

• Next you should see the message that the KeyVault was updated successfully.

3.6.3 Add secrets to your Vault
• Click "Secrets" in the left menu.
• Click "Generate/Import" to create a new secret

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

3.6.4 Adding Service Connection to the Azure DevOps project

• Open the Azure DevOps Portal
• Browse to your project
Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps

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

![Project Settings and Pipelines section](image)

• Click "Create service connection"

![Create service connection](image)
• Select "Azure Resource Manager"

• Click "use the full version of the service connection dialog" link

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 stored when creating the Service Principal Name
  
  a. Enter a "Connection name"
  
  b. Enter the "appId" as "Service principal client ID"
  
  c. Enter the "password" as the "Service principal key"
  
  d. Enter the "tenant" as the "Tenant ID" (potentially already populated)
Click “Verify connection” to validate the entered information. The status should change to “Verified”
• Select "Allow all pipelines to use this connection" and click "OK"

![Image of Azure Resource Manager service connection]

• The "Service connection" is now created and displaying

![Image of Microsoft 365 DSC service connection]

### 3.7 Configure the Local Configuration Manager

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

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

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

○ Run the following command and store the value:
• Export public certificate (required for MOF compilation)
  o Export the public certificate to a CER file by running the following command:

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

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

  ```powershell
  Configuration ConfigureLCM
  {
    Import-DscResource -ModuleName PsDesiredStateConfiguration
    node localhost
    {
      LocalConfigurationManager
      {
        CertificateId = "<thumbprint>"
      }
    }
  }
  ConfigureLcm
  ```

  o Update the "<thumbprint>" with your own certificate thumbprint and run the code (F5)
Run the following command to deploy the Local Configuration Manager config:

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

The output should look like this:

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

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

Optional: Secure your certificate

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

4.1 Populate scripts

- Download and install Visual Studio Code from [https://code.visualstudio.com](https://code.visualstudio.com)
- Download and install Git from [https://git-scm.com](https://git-scm.com)
  - Download the most recent version of Git by clicking the "Download" button
- Run the downloaded installer and use the default settings
- Download the DSC scripts from [https://microsoft365dsc.com/Pages/Resources/Whitepapers/M365Automation.zip](https://microsoft365dsc.com/Pages/Resources/Whitepapers/M365Automation.zip)
  - This package contains several scripts:

<table>
<thead>
<tr>
<th>Script Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure-pipelines.yml</td>
<td>The configuration file for the Azure DevOps Build Pipeline. This file defines which steps are required to build the DSC MOF file.</td>
</tr>
<tr>
<td>Build.ps1</td>
<td>The script that is responsible for retrieving the credentials and building the DSC MOF file.</td>
</tr>
<tr>
<td>DSCCertificate.cer</td>
<td>The DSC Signing Certificate, which is used to encrypt the used passwords in the DSC MOF file.</td>
</tr>
<tr>
<td>M365Configuration.ps1</td>
<td>The Microsoft 365 configuration file which defines the target state.</td>
</tr>
<tr>
<td>M365ConfigurationData.psd1</td>
<td>The parameter file used to make sure the configuration can be used across multiple environments.</td>
</tr>
<tr>
<td>Prep.ps1</td>
<td>The script that is responsible for preparing the environment for building the DSC MOF file. This script for example downloads the Microsoft365Dsc module.</td>
</tr>
<tr>
<td>ReadMe.md</td>
<td>A project description file in Markdown format. This will be displayed when opening the repository in Azure DevOps.</td>
</tr>
</tbody>
</table>
Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps

- Upload the scripts to the DevOps repository
  - Open Azure DevOps Portal and browse to your project
  - Click the "Repos" icon in the left menu
- Click on "Clone in VS Code" (acknowledge any browser notifications for opening any files)
Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps

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

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

- Login with your Microsoft 365 admin account
- Click "Open" in the bottom right corner to open the cloned folder
○ The repository is now available (but still empty) in Visual Studio Code

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

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

```powershell
NonNodeData = @{}
HRSiteCol = @{}

Url1 = 'https://M365Automation.sharepoint.com/sites/hr'
Owner = 'admin@M365Automation.onmicrosoft.com'
```

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

```powershell
$password = ConvertTo-SecureString -String $env:SAPW -AsPlainText -Force
$username = 'admin@M365Automation.onmicrosoft.com'
```

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

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

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

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

○ You might get the below message when running the Sync. Click "OK" to publish the branch to DevOps.
Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps

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

![Azure DevOps Portal](image)

4.2 Configure Azure DevOps project

4.2.1 Create Build pipeline

- Go to the Azure DevOps Portal
- Click "Pipeline" and click "Create Pipeline"
Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps

• Select "Azure Repos Git"

  Select a repository

  Filter by keywords

  M365Config

• Click the name of your Project

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

• Select "Run" to start the pipeline
• Check if the pipeline completes successfully

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

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

4.2.2 Create Release pipeline

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

• Select “Empty job”
• Give the Stage a name and close the pane

• Click "Add an artifact"

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

• Configure the Release pipeline triggers by clicking the Lightning icon next to Artifacts
• Enable the “Continuous deployment trigger”, under "Build branch filters" click "Add" and select "The build pipeline's default branch"

• Make sure the branch has been added successfully and close the pane
• Select "Tasks > <Stage name>"

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

• Click the "+" behind "Agent Job"
Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps

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

- Select the “PowerShell” task

- Select “Inline” as “Type” and paste the following code in the “Script” window:

```powershell
Start-DscConfiguration -Path "$(System.ArtifactsDirectory)_M365Config\M365ConfigMOFFi le" -Verbose -Wait -Force
if ($error.Count -gt 0)
{
    write-host "the following errors occurred during dsc configuration";
    write-host ($error | fl * | out-string );
    throw $error[-1];
}
```

**NOTE:** You might have to update the value for the Path parameter: "$(System.ArtifactsDirectory)\<source_alias>\<artifact_name>"
  - source_alias: This comes from the value to noted in a previous step
  - artifact_name: This name is specified in the azure-pipelines.yml file.
• Open the “Advanced” section and select “Fail on Standard Error”

Advanced

- Fail on Standard Error
- Ignore $LASTEXITCODE
- Use PowerShell Core

• Click “Save”. Use “\" as folder and add a comment if you prefer. Click “OK”

• Click "Create release" to test the created Release pipeline.
• Leave everything default and click “Create”

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

• Review the progress
• Click the stage for more details

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

4.2.3 Validate that changes to the config are deployed successfully

• Make sure the following setting is configured:
  SharePoint Admin Center > Policies > File and folder links > Only people in your organization
The above setting is configured by the "DefaultSharingLinkType" DSC setting:

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

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, that should have started.
- Once completed, the Release pipeline should automatically start.

- When the Release pipeline completes, the setting should have been changed in the SharePoint Admin Portal.
Managing Microsoft 365 in true DevOps style with Microsoft365Dsc and Azure DevOps
5 Security Enhancements

5.1 Using Azure Conditional Access to secure service account

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

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

Steps
- Open the Azure Portal (https://portal.azure.com)
- Go to Azure Active Directory
- Under “Manage”, click “Enterprise applications”
- First we are going to create a Named Location
- Under “Manage”, click “Named locations”
- Click “New location”
- Enter the required information:
  - Name: “Azure Self Hosted VMs” (or any other name you want to use)
  - Define the location using: “IP Ranges”
  - IP ranges: The public IP address of the VM in the “123.123.123.123/32” format

\(^1\) Azure AD Premium P1 license required
Click “Create” to create the Named location

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

Create a new policy, using the following settings:

- Name: “Conditional Access for DSC Service Account” (or the name you would like to use)
- Users and groups > Include
  - Select “Selected users and groups”
  - Check “Users and groups”
  - Search and select the DSC Service Account
• **Cloud apps or actions:** Select “All cloud apps”

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

• **Access controls > Grant**
  - Select “Block access”
• Under “Enable policy”, select “On” to activate the policy and click “Create”

• The DSC service account can now only be used to authenticate from the Azure DevOps Self Hosted VMs
6 Learning materials

6.1 Desired State Configuration

- Channel9: “Getting Started with PowerShell Desired State Configuration”
- Channel9: “Advanced PowerShell Desired State Configuration”
- Desired State Configuration Overview for Engineers
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/overview/dscforengineers
- Creating configurations
  - Configurations: https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/configurations
  - DependsOn: https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/resource-depends-on
  - DSC Resources: https://docs.microsoft.com/en-us/powershell/scripting/dsc/resources/resources
- Using configuration data in DSC
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/configdata
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/separatingenvdata
- Secure the MOF file
  - https://docs.microsoft.com/en-us/powershell/scripting/dsc/configurations/configdatacredentials
- Local Configuration Manager
  - Push/Pull model: https://docs.microsoft.com/en-us/powershell/scripting/dsc/pull-server/enactingconfigurations
- Apply, Get, and Test Configurations on a Node
- Debugging DSC
6.2 Microsoft365Dsc

- Microsoft365dsc.com
  - https://microsoft365dsc.com/
- Microsoft365Dsc promotion video
  - https://aka.ms/m365dcpromo
- 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/UCveScabVT6pxqYgGRu17iw
- PluralSight: “SharePoint Conference ’19: IT Pros, Get Relevant! Upskilling for Today’s Cloud” (subscription required)

6.3 Git

- Git manual
- PluralSight: "How Git Works" (subscription required)
- PluralSight: "Mastering Git" (subscription required)
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSC</td>
<td>Desired State Configuration</td>
</tr>
<tr>
<td>LCM</td>
<td>Local Configuration Manager</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
</tbody>
</table>