PowerShell Desired State Configuration (DSC)

DSC is a management platform in PowerShell that enables to manage IT and development infrastructure with configuration as code.

There are three versions of DSC available:

DSC 1.1 is the legacy version of DSC that originally shipped in Windows PowerShell 5.1.

DSC 2.0 is the version of DSC that shipped in PowerShell 7.

With the release of PowerShell 7.2, the PSDesiredStateConfiguration module is no longer included in the PowerShell package. Separating DSC into its own module allows us to invest and develop DSC independent of PowerShell and reduces the size of the PowerShell package. Users of DSC will enjoy the benefit of upgrading DSC without the need to upgrade PowerShell, accelerating the time to deployment of new DSC features. Users that want to continue using DSC v2 can download PSDesiredStateConfiguration 2.0.5 from the PowerShell Gallery.

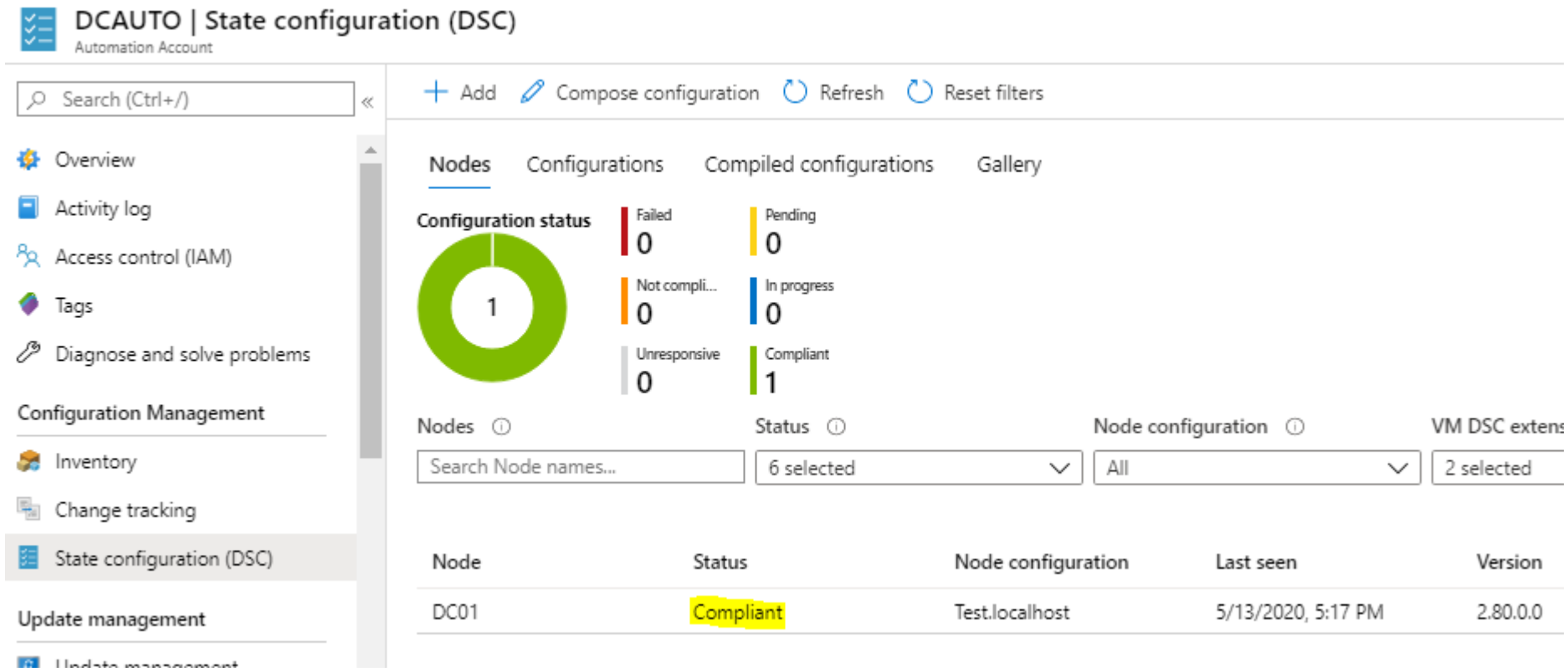
DSC 3.0 is the new version of DSC. This version is a preview release that is still being developed. Users working with non-Windows environments can expect cross-platform features in DSC 3.0. DSC 3.0 is the version that is supported by the machine configuration feature of Azure Automanage.

PowerShellGet and the PowerShell Gallery

- The **PowerShell Gallery** is the central repository for PowerShell content.
- In it, we can find PowerShell scripts, modules containing PowerShell cmdlets and Desired State Configuration (DSC) resources.
- Some of these packages are authored by Microsoft, and others are authored by the PowerShell community.
- The **PowerShellGet** module contains cmdlets for discovering, installing, updating, and publishing PowerShell packages from the PowerShell Gallery.
- These packages can contain artifacts such as Modules, DSC Resources, Role Capabilities, and Scripts. Make sure you have the latest version of PowerShellGet and PackageManagement installed.
- The **Microsoft.PowerShell.PSResourceGet** module replaces the PowerShellGet and PackageManagement modules.
- Microsoft.PowerShell.PSResourceGet version 1.0.1 ships in PowerShell 7.4.0. The latest version is available in the PowerShell Gallery and can be installed on any supported version of PowerShell.

DSC in Azure Automation

https://tkolber.medium.com/configuring-azure-dsc-automation-with-powershell-in-5-steps-454fbef9457b



Syntax

```
Configuration <ConfigurationName>
{
   Import-DscResource -ModuleName '<ModuleName>'

   Node '<NodeName>'
   {
      <ResourceName> <ResourceInstanceName>
      {
         Property1 = 'Value1'
         Property2 = 'Value2'
         # Add more properties as needed
      }
   }
}

# Apply the configuration
<ConfigurationName> -OutputPath '<OutputPath>'
Start-DscConfiguration -Path '<OutputPath>' -Wait -Verbose
```

```
Configuration ServiceManagement
{
   Import-DscResource -ModuleName 'PSDesiredStateConfiguration'

   Node 'localhost'
   {
      Service WindowsUpdateService
      {
         Name       = ‘winrm'
         Ensure     = 'Present'
         State      = 'Running'
         StartupType = 'Automatic'
      }
   }
}

# Apply the configuration
ServiceManagement -OutputPath "C:\DSC\ServiceManagement"
Start-DscConfiguration -Path "C:\DSC\ServiceManagement" -Wait -Verbose
```

To ensure that the IIS web server feature is installed on a Windows server:

```
Configuration WebServer
{
   Import-DscResource -ModuleName 'PSDesiredStateConfiguration'

   Node  'Server1', 'Server2'
   {
      WindowsFeature IIS
      {
         Name   = 'Web-Server'
         Ensure = 'Present'
      }
   }
}

# Apply the configuration
WebServer -OutputPath "C:\DSC\WebServer"
Start-DscConfiguration -Path "C:\DSC\WebServer" -Wait -Verbose -ComputerName ‘Server1’, 'Server2’
```

Commonly used DSC module

- PSDesiredStateConfiguration: Contains built-in DSC resources like File, Service, WindowsFeature, Environment, etc.
- xPSDesiredStateConfiguration: Contains experimental and community-contributed DSC resources.
- xNetworking: Contains resources for managing network settings.
- xWebAdministration: Contains resources for managing IIS and web applications.
- xActiveDirectory: Contains resources for managing Active Directory.
- xSQLServer: Contains resources for managing SQL Server.
- xComputerManagement: Contains resources for managing computer settings like users, groups, and time zones.
- xStorage: Contains resources for managing storage settings like disks, volumes, and partitions.
- xHyper-V: Contains resources for managing Hyper-V virtual machines and settings.
- xWindowsUpdate: Contains resources for managing Windows Update settings.

SQL

```
# Step 1: Install the SqlServer module (if not already installed)
Install-Module -Name SqlServer -Force -Scope CurrentUser

# Step 2: Define the connection string
$serverName = "YourServerName"
$databaseName = "YourDatabaseName"
$connectionString = "Server=$serverName;Database=$databaseName;Integrated Security=True;"

# Step 3: Define the SQL query
$sqlQuery = "SELECT TOP 10 * FROM YourTableName"

# Step 4: Execute the SQL query and retrieve results
try {
    $results = Invoke-Sqlcmd -ConnectionString $connectionString -Query $sqlQuery
    # Step 5: Display the results
    $results | Format-Table -AutoSize
} catch {
    Write-Error "An error occurred: $_"
}
```

PowerShell is a versatile scripting language and automation tool that can be used for a variety of real-time use cases in IT administration, DevOps, and development environments.

Here are some common real-time use cases for PowerShell:

Automating System Administration Tasks

User Management: Create, modify, and delete user accounts in Active Directory.

Service Management: Start, stop, and restart Windows services.

Scheduled Tasks: Create and manage scheduled tasks on Windows systems.

Monitoring and Reporting

System Health Monitoring: Monitor CPU, memory, disk usage, and other system metrics.

Event Log Monitoring: Continuously monitor Windows Event Logs for specific events.

Network Monitoring: Check the status of network connections and interfaces.

Configuration Management

Desired State Configuration (DSC): Ensure systems are configured to a desired state.

Configuration Files: Manage and update configuration files across multiple systems.

DevOps and CI/CD Pipelines

Build Automation: Automate build processes using PowerShell scripts.

Deployment Automation: Deploy applications and services to various environments.

Integration with CI/CD Tools: Integrate PowerShell scripts with tools like Jenkins, Azure DevOps, and GitHub Actions.

Cloud Management

Azure Automation: Manage Azure resources using Azure PowerShell.

AWS Management: Manage AWS resources using AWS Tools for PowerShell.

VMware Automation: Manage VMware infrastructure using PowerCLI.

Security and Compliance

Patch Management: Automate the installation of Windows updates.

Security Auditing: Audit system security settings and configurations.

Compliance Reporting: Generate compliance reports for regulatory requirements.

Data Management

Database Management: Execute SQL queries and manage databases.

File Management: Automate file operations like copying, moving, and deleting files.

Log Management: Aggregate and analyze log files from multiple sources.