Configuration as code: Automating Windows Server 2016 configuration with PowerShell and DSC

Microsoft IT supports approximately 36,000 servers across our cloud and on-premises infrastructure. Managing these servers requires hundreds of thousands of operating system configuration changes over the server lifecycle. In the past, we used a process that was originally developed for Windows Server 2003.

With Windows Management Foundation 5.0, introduced in Windows Server 2016, we benefit from enhancements to Desired State Configuration (DSC) and how it interacts with PowerShell. With DSC, we can define configuration as code, which means that configuration logic and optional tasks can be performed programmatically using PowerShell.

Understanding configuration change and management requirements

Our configuration process used to rely on scripts that made changes by following a set of steps. Typically, a script would be implemented by:

1. Determining and documenting required changes.
2. Developing a new script or modifying an existing script to change the configuration.
3. Testing the script for functionality in a test environment.
4. Submitting the script for approval as part of the change process.
5. Approving the script and putting it into the production environment.
6. Having several users run the script once or twice a year.

A team managed the script environment and ensured that scripts were run in the correct sequence, if multiple scripts were needed. It was a tedious manual process that had several problems:

- Significant IT human resources were required to manage processes.
- Making the changes caused server downtime.
- The script-based solution was not scalable or elastic.
- We experienced configuration drift because different versions of the scripts were being used and people ran them on different schedules.

Implementing PowerShell and DSC for server configuration

With DSC, we can define configuration as code, so configuration logic and optional configuration tasks can be performed programmatically with PowerShell. DSC also gives us the ability to create reusable DSC resources that can be configured on a granular level and then combined with other DSC resources to form a complete server configuration. Because PowerShell and DSC management is declarative and configuration settings are controlled individually, it is easier to use across a large group of servers or across varying server configurations.

Extending a consistent configuration environment

When we implemented PowerShell and DSC, we focused on Windows Server 2016 servers first, both on-premises and in Microsoft Azure. In Azure, we could use Azure Automation to build an even more streamlined process. We installed Windows Management Framework (WMF) 5.0 on servers running Windows Server 2012 R2, Windows Server 2012, and Windows Server 2008 R2 to include them in the new configuration process and enable Azure automation.

Creating a flexible development process

With Windows Server 2016 and the WMF 5.0, we reduced the effort needed to reconfigure changes in our environment using PowerShell and DSC. We used DSC resources to build small, modular configuration change actions. Once in place, those DSC resources can be combined and manipulated with PowerShell to make changes on one or more servers. Change deployment in PowerShell enabled us to be both specific and scalable for pushing out configuration changes. What was once a multi-level process that required numerous people and teams, is now a process that a single engineer can manage.

PowerShell and DSC make it simple for developers to interact with the configuration process, too. Instead of asking for changes from the configuration team, developers can use PowerShell and DSC to reconfigure as their solution requires. This streamlines development and gets new features and functionality into production faster.

Conclusion

PowerShell and DSC allow us to configure servers in an agile way. Because it is easy to do, a specific node configuration using PowerShell DSC can be managed by a single engineer for our entire server environment. The reusable, flexible application of DSC ensures that only necessary configuration changes are made, and it allows our DevOps team to quickly implement changes and fix configuration issues quickly, while still moving forward. We significantly reduced the opportunity for configuration errors, and PowerShell and DSC have eliminated thousands of hours of downtime for server configuration changes.

For more information

Windows PowerShell Desired State Configuration overview

Built-in Windows PowerShell Desired State Configuration resources

Azure Automation DSC overview

How to install and configure Azure PowerShell

The what, why, and how of Azure Automation Desired State Configuration (DSC)

DSC resource style guidelines

Azure Automation DSC PowerShell ISE Plugin

Windows Management Framework (WMF) 5.0 release notes overview

Microsoft IT

microsoft.com/ITShowcase