PowerShell Desired State Configuration - Anthony Adams

Imagine you have 2 computers, it would be pretty easy to configure them even if you had to manage them separately, you could configure them to be very similar if not totally the same one by one.

But what if you had 10? 50? 100? Also, in these cases, it is very common for someone to change a configuration on one of the machines making that one different from the others, what of we want to prevent this? What if there was a way to make sure all the machines on our network were consistent?

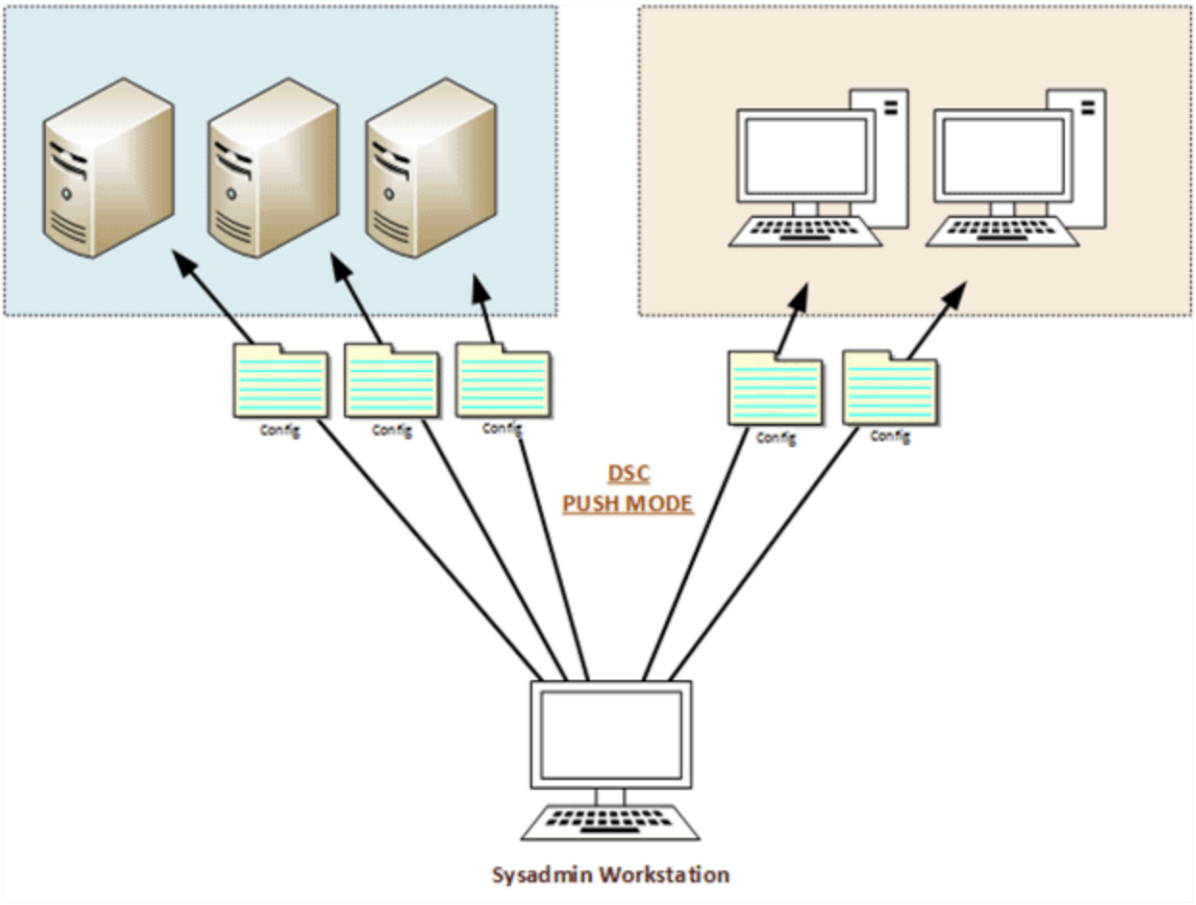
PowerShell DSC is a tool to achieve this. Starting at PowerShell 4.0 and up, this feature helps SysAdmins

standardize the automation and configuration of both windows and Linux operating systems.

DSC can also simplify and automate the configuration process of your system by allowing the SysAdmin to specify a goal for the configuration and allow DSC to do the intricate configuration process on its own.

There are 2 types of architecture when it comes to DSC

Push Mode



the configurations are sent (“pushed”) manually towards

one or more units that we call “node”. This action is done by an administrator.

This is a 1 way communication model. The admin sends configuration settings to the node machines.

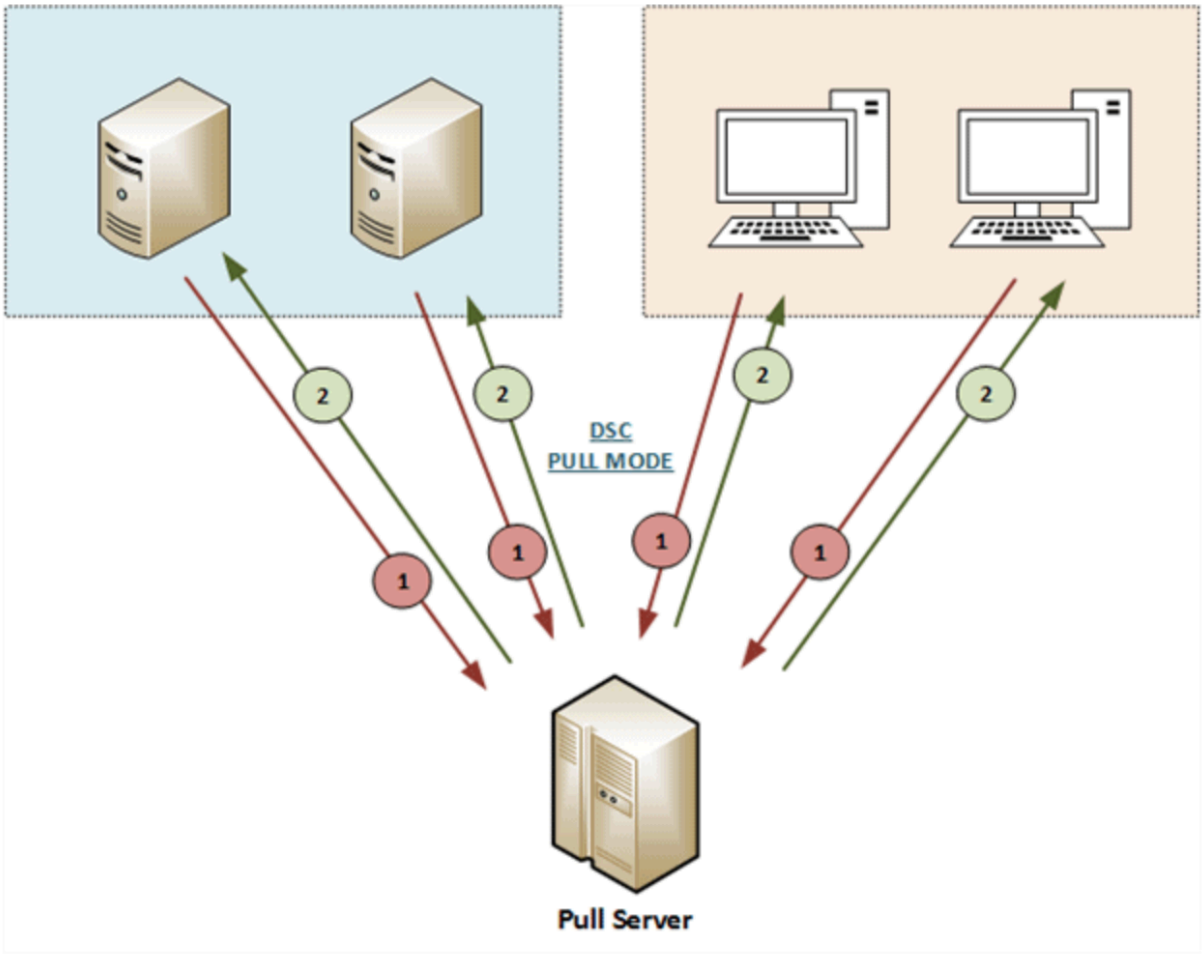
Advantages:

- Set up costs. It is not necessary to invest in a new server because the configurations are pushed from your workstation.
- The simplicity of the architecture because all configurations are stocked on your workstation.
- More decentralized

The disadvantages:

- The complexity required to manage the machines connected. Because laptops are not always connected to the network, the sending of the configuration may fail.
-

Pull Mode



A Pull Server is created and the nodes check with this server regularly to pull configuration settings at a predetermined time interval.

In this architecture, the nodes themselves that check in with the parent server and request new updates if they are available. If one is available the “pull server” will send the updated configuration settings to the compatible machine(s). The default setting time interval is 15 min.

The advantages of this scenario are:

- Automation of the deployment of the configurations
- The management of numerous machines, connected or not. As soon as the machine is connected to the network, it asks to the “Pull Server” for its configuration.

The disadvantages:

- You need to deploy one more server

PowerShell Desired State Configuration

• Management platform in Windows PowerShell v4

• Enables deploying and managing configuration data for software services and managing the environment in which services run.

• “Provides a set of Windows PowerShell language extensions, new Windows PowerShell cmdlets, and resources that you can use to declaratively specify how you want your software environment to be configured.”

• It also provides a means to maintain and manage existing configurations.

• Various ‘Providers’ enable built-in functionality for configuring specific resources, including but not limited to:

Provider	Description
File Resource	Managing files and folders
WindowsFeature Resource	Managing server features and roles
Package Resource	Managing Windows installer and setup.exe packages
Registry Resources	Managing registry settings
Services Resource	Manages services

DSC Local Configuration Manager

• The Local Configuration Manager is the PowerShell Desired State Configuration (DSC) engine.
• Runs on all target nodes, and it is responsible for calling the configuration resources specified in the configuration script.
• See Built-in Windows PowerShell Desired State Configuration Resources for a full list of DSC Providers: https://learn.microsoft.com/en-us/powershell/dsc/resources/authoringresource?view=dsc-1.1

Sample Configuration MyWebConfig

```
$WebsiteFilePath=”\\acmeserver\DSC\content”

Configuration MyWebConfig
{
   Node "WIN-GEHPBE9SEH0"
   {
     WindowsFeature MyRoleExample
      {
          Ensure = "Present"
          Name = "Web-Server"
      }

     File MyWebsiteContent
      {
         Ensure = "Present"
         Type = "Directory“
         Recurse = $true
         SourcePath = $WebsiteFilePath
         DestinationPath = "C:\inetpub\wwwroot"
         DependsOn = "[WindowsFeature]MyRoleExample"
      }
   }
}
```

The above case there is a configuration named MyWebConfig ensuring IIS is running There is a target node names WIN-GEHPBE9SEH0" and this node is to run a feature named MyRole example. This feature is doing 2 things;

Ensuring a feature named “Web-Server” is present

Ensuring a file is present. This file is located in a directory located at C:\inetpub\wwwroot

The second resource should only be applied after ‘web-server’ feature is installed

Managed Object Format (MOF) File

• The configuration script is run as a function.
• Calling the function generates a MOF schema file that is used to deploy the configuration on the target nodes
• The configuration is applied by using the DSC cmdlets

• Get-DscConfiguration – gets the current configuration
• Start-DscConfiguration – Applies the configuration
• Test-DscConfiguration – Tests if the actual matches the desired

In the lab

• Configure and shared folder AcmeServer domain controller to house the source files of a web server deployment
• Create a Desired State Configuration script to configure the Windows Core member server
• Install IIS and deploy the content from the AcmeServer share
• Apply the configuration to the Windows Core Member server