PowerShell Core
Benefits and challenges

Aleksandar Nikolić | Microsoft MVP

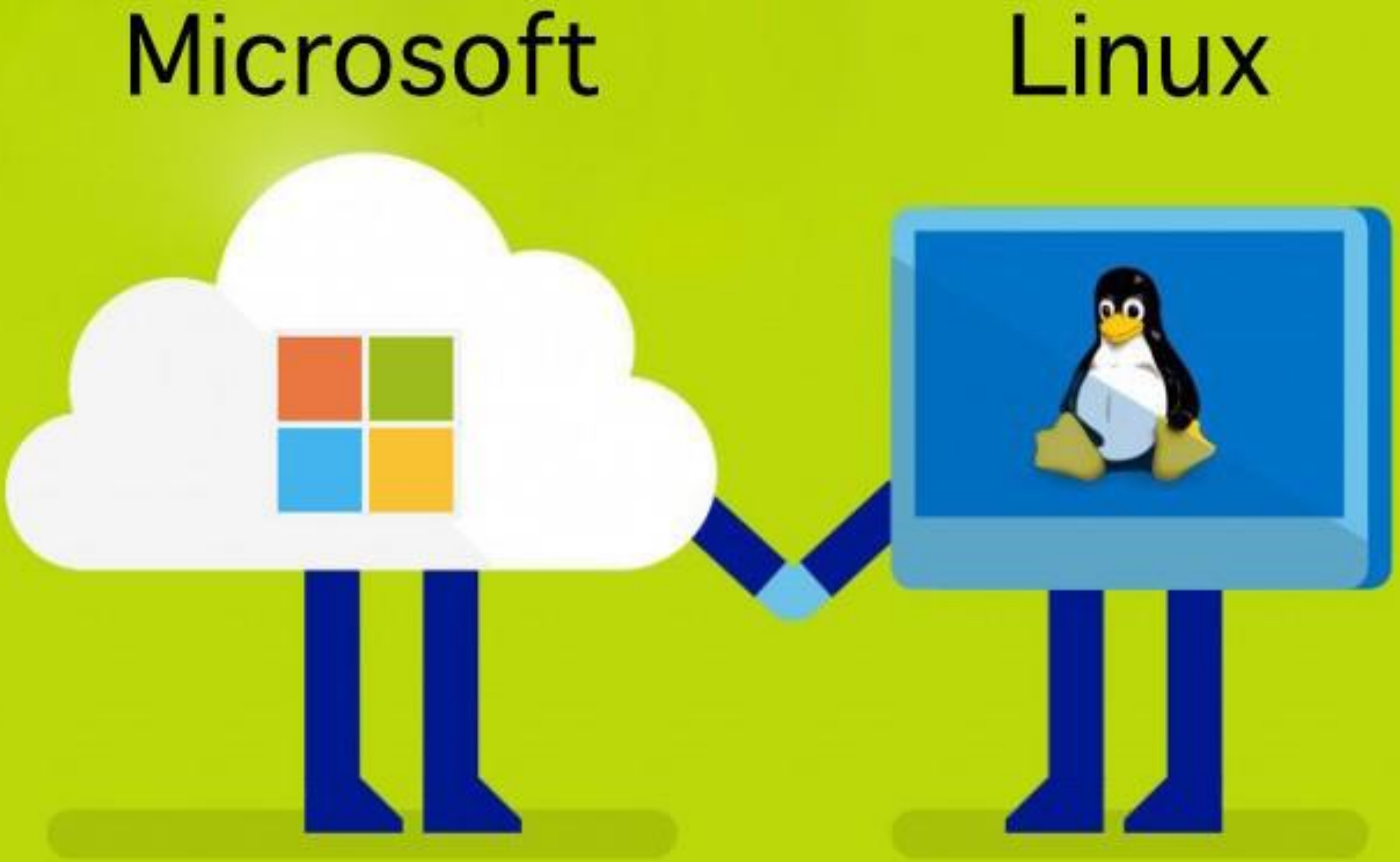
PS> whoami

Aleksandar Nikolić

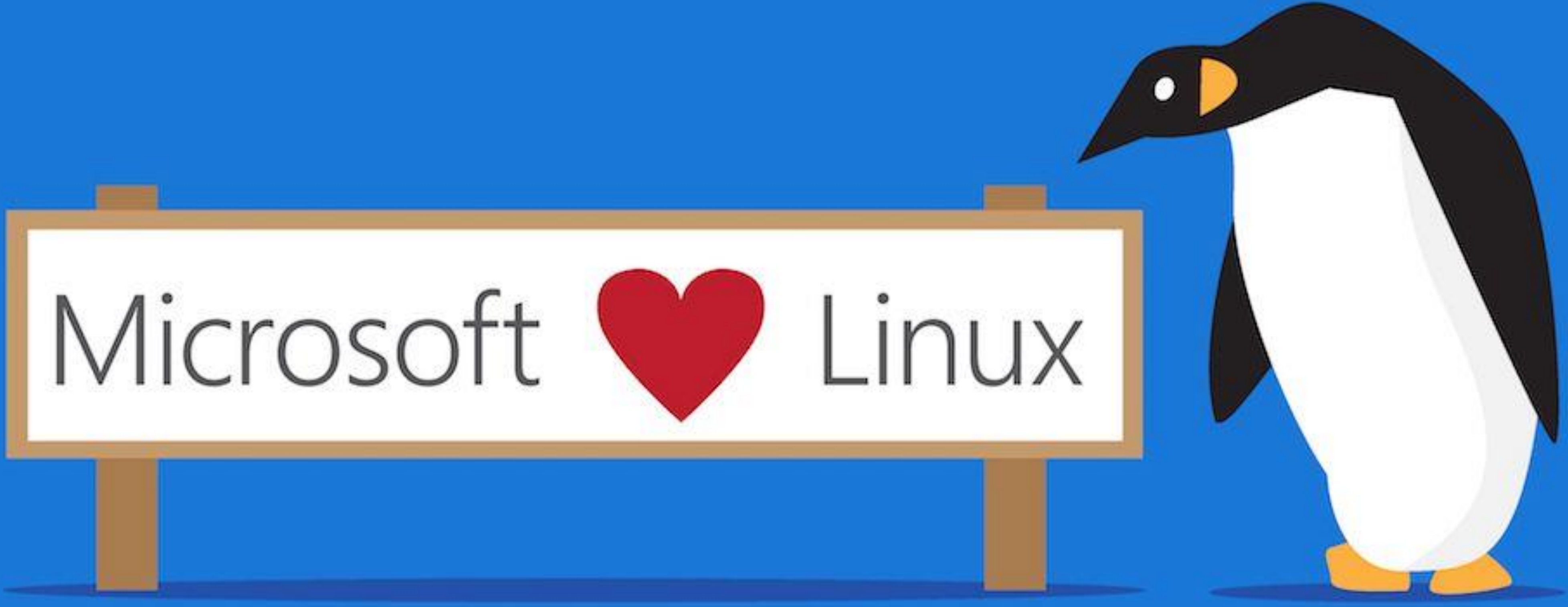
- PowerShell and Azure trainer
- Microsoft Azure MVP
- Cloud and Datacenter Management MVP
- Co-founder of PowerShellMagazine.com
- @alexandair

Microsoft ❤ Linux

Microsoft Azure Cloud



Microsoft



PowerShell Core

- Introduced in August 2016
- Current stable version: 6.2.1

- PowerShell Core 6.0 released on January 10, 2018
- PowerShell Core 6.1 released on September 13, 2018
- PowerShell Core 6.2.0.-preview.1 released on October 17, 2018
- PowerShell Core 6.2.1 released yesterday

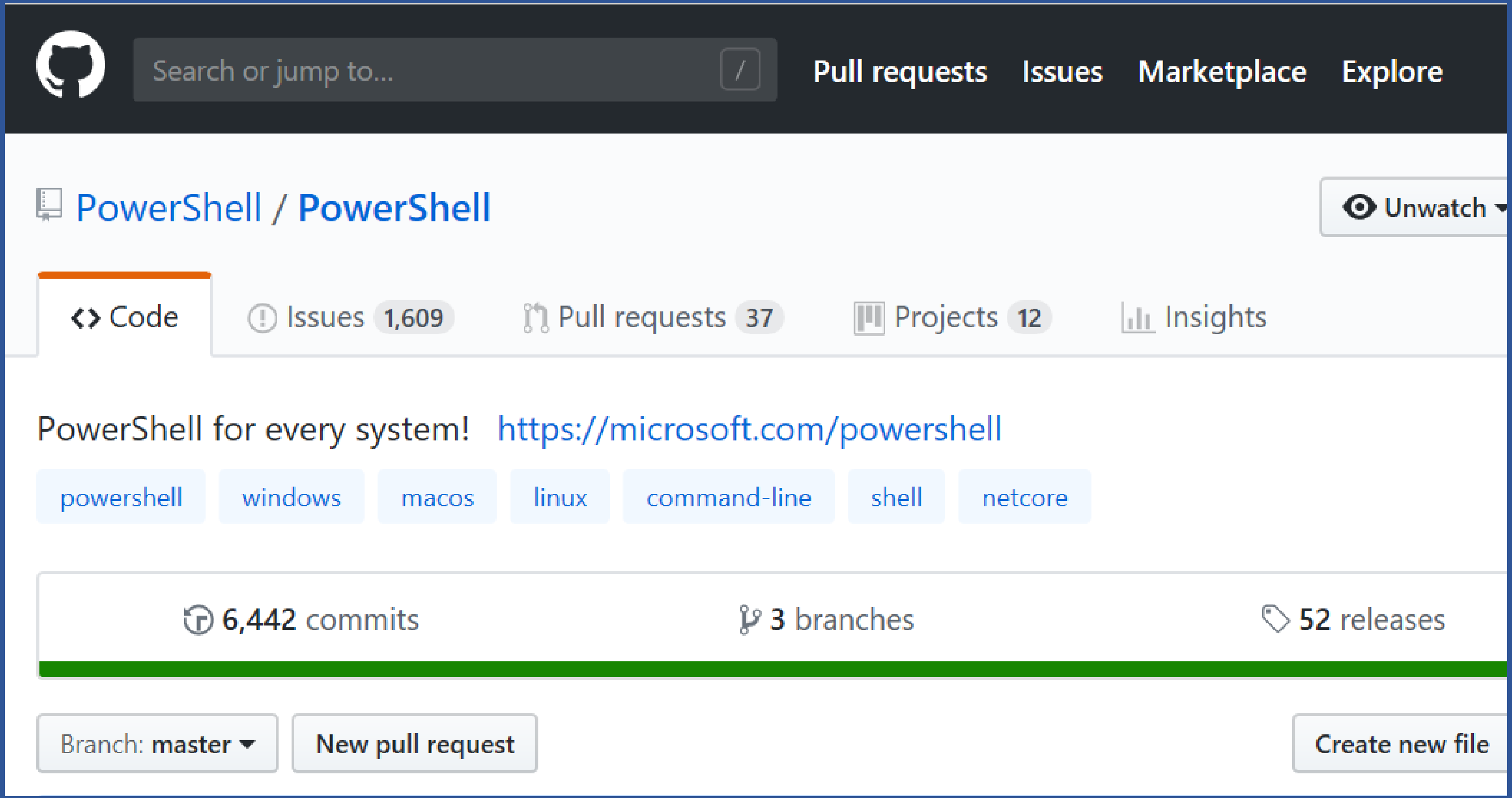
- 64 releases so far

- Built on the .NET Core

Why do we need PowerShell Core?

- Manage our heterogenous environments in the hybrid cloud
- “Run anywhere, manage anything”
- A list of supported operating systems https://aka.ms/pslifecycle
- Installing PowerShell Core:
- https://docs.microsoft.com/powershell/scripting/setup/installing-powershell

PowerShell Core on GitHub



PowerShell / PowerShell

PowerShell for every system! https://microsoft.com/powershell

6,442 commits | 3 branches | 52 releases

Power BI analysis https://aka.ms/PSGitHubBI

Main features

- Cross-platform: Windows, macOS, and Linux
- Side-by-side and portable
- SSH-based PowerShell remoting

DEMO

PowerShell Core on Windows

PowerShell Core on Linux (WSL)

PowerShell Core in the Azure Cloud Shell

PowerShell Core in Docker containers

What about Windows PowerShell?

- Still fully supported and serviced

- Will not be “replaced” by PowerShell Core within Windows
- Remaining a stable platform for existing workloads

- No new feature innovation planned
- Includes PSRP over SSH

What about PowerShell ISE?

- No new feature innovation planned
- Only bug and security fixes

- Future PowerShell editor
- Visual Studio Code + PowerShell extension

- On Windows: Install-Script -Name Install-VSCode

DEMO

PowerShell extension for Visual Studio Code

Limitations of PowerShell Core

- Some modules are incompatible with .NET Core
- A few “built-in” cmdlets are missing from PowerShell Core
  - WMI v1 cmdlets, PerfCounter, EventLog, LocalAccounts
  - On non-Windows platforms, these modules are missing:
    - CimCmdlets
    - Microsoft.WSMan.Management
    - PSDiagnostics
- Removed snap-ins and workflow

Module coverage for PowerShell Core

- The success of PowerShell Core depends on your success
- Working with Azure, Windows, and Office to increase coverage
  - Azure PowerShell “Az” module
  - AzureAD.Standard.Preview
- WindowsCompatibility module
  - Transparent implicit remoting to local Windows PowerShell
- PowerShell Core 6.1 on Windows 10 version 1809 and Windows Server 2019
  - Compatibility with 1900+ existing cmdlets

DEMO

Windows Compatibility

PowerShell Core on Linux: Differences

Automatic variables

```
Name                          Value
----                          -----
EnabledExperimentalFeatures {}
HOME                          /home/aleksandar
IsCoreCLR                     True
IsLinux                       True
IsMacOS                       False
IsWindows                     False
OutputEncoding                System.Text.UTF8Encoding
PROFILE                      /home/aleksandar/.config/powershell/Microsoft.PowerShell_profile.ps1
PSEdition                     Core
PSHOME                        /opt/microsoft/powershell/6
```

ENVIRONMENT variables

```
Name                           Value
----                           -----
_                              /usr/bin/pwsh
DOCKER_HOST                    tcp://0.0.0.0:2375
HOME                           /home/aleksandar
HOSTTYPE                       x86_64
LANG                           en_US.UTF-8
LESSCLOSE                      /usr/bin/lesspipe %s %s
LESSOPEN                       | /usr/bin/lesspipe %s
LOGNAME                        aleksandar
LS_COLORS                      rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd...
NAME                           DESKTOP-MC2AESS
PATH                           /opt/microsoft/powershell/6:/home/aleksandar/bin:/home/aleksandar/.loc...
PSModulePath                   /home/aleksandar/.local/share/powershell/Modules:/usr/local/share/powe...
PWD                            /home/aleksandar
…
```

$PROFILE

```
PS C:\> $PROFILE | Get-Member -Type NoteProperty | ft definition -Wrap

Windows
AllUsersAllHosts=C:\Program Files\PowerShell\6\profile.ps1
AllUsersCurrentHost=C:\Program Files\PowerShell\6\Microsoft.PowerShell_profile.ps1
CurrentUserAllHosts=C:\Users\aleksandar\Documents\PowerShell\profile.ps1
CurrentUserCurrentHost=C:\Users\aleksandar\Documents\PowerShell\Microsoft.PowerShell_profile.ps1

Linux
AllUsersAllHosts=/opt/microsoft/powershell/6/profile.ps1
AllUsersCurrentHost=/opt/microsoft/powershell/6/Microsoft.PowerShell_profile.ps1
CurrentUserAllHosts=/home/aleksandar/.config/powershell/profile.ps1
CurrentUserCurrentHost=/home/aleksandar/.config/powershell/Microsoft.PowerShell_profile.ps1
```

PSReadLine history file

```
Default value:

On Windows:
$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\$($
host.Name)_history.txt
On Linux:
$HOME/.local/share/powershell/PSReadLine/$($host.Name)_h
istory.txt
On macOS:
$XDG_DATA_HOME/powershell/PSReadLine/$($host.Name)_histo
ry.txt
```

#Requires statement

```
-PSEdition <PSEdition-Name>
```

Specifies a PowerShell edition that the script requires. Valid values are Core for PowerShell Core and Desktop for Windows PowerShell.

For example:

```
    #Requires -PSEdition Core
```

Help

```
~/.local/share/powershell/Help

 # Respect PAGER, use more on Windows, and use less on Linux
        $moreCommand,$moreArgs = $env:PAGER -split '\s+'
        if ($moreCommand) {
            $help | & $moreCommand $moreArgs
        } elseif ($IsWindows) {
            $help | more.com
        } else {
            $help | less
        }
```

less – CLI text viewer

```
Page Up, Page Down, arrows, Spacebar
/keyword to search (case-sensitive)
     n – next instance
     p – previous instance
q – quit
LESS env. variable (define in .bashrc)
      LESS='-C -M -I -j 10 -# 4'
PAGER=less
```

.NET Core to the rescue!

```
Windows: $env:PSModulePath -split ';'
Linux:   $env:PSModulePath -split ':'

$env:PSModulePath -split [IO.Path]::PathSeparator

[IO.Path] | Get-Member -Static
```

How to create a temp file

```
X-plat:
     [IO.Path]::GetTempFileName()
     New-TemporaryFile

Linux:
     tempfile
```

DEMO

.NET Core to the rescue!

How to interact with PowerShell Core in WSL

Start your WSL distro and run “pwsh”

Run “ubuntu run pwsh” from Windows PowerShell or PowerShell Core on windows

Install PSWsl module on Windows PowerShell or PowerShell Core on Windows

```
Install-Module PSWsl –Scope CurrentUser

Enter-WslDistribution -DistributionName ubuntu
Invoke-WslCommand -DistributionName ubuntu -Scriptblock {hostname}
```

DEMO

WSL and PowerShell Core

Remoting on PowerShell Core

- Three ways to remote:
  - PowerShell remoting over WSMan/WinRM
  - PowerShell remoting over SSH
  - Plaintext SSH remoting

- Limitations
  - WSMan PSRP server is experimental on non-Windows platforms
  - WSMan PSRP client doesn’t support Kerberos on non-Windows platforms

- Hypothesis
  - SSH is the future and everyone should use it
  - WSMan/WinRM is still very important and we should continue to support it

DEMO

SSH-based PowerShell Remoting

BONUS DEMO

Windows Terminal