PowerShell Conference Europe 2019

Hannover, Germany

June 4-7, 2019

# Custom Pester assertions are the vocabulary of ~~your~~ our tests

Jakub Jares

Platinum Sponsor

aws

# This Session

- Learn how to write your own assertions and integrate them with Pester's Should.


nohwnd

# Agenda

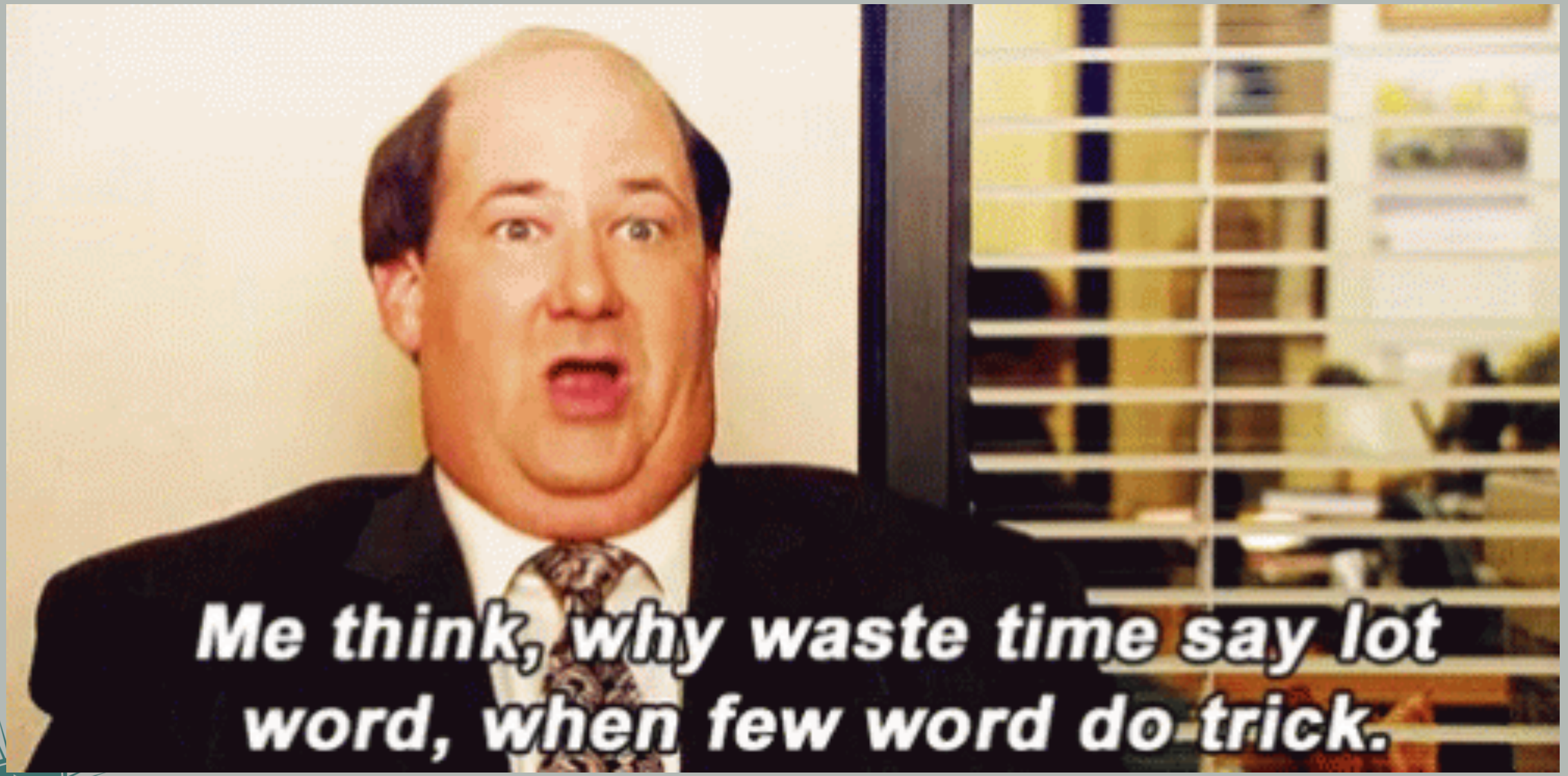
- Why custom assertions
- Extending Should
- Testing your assertions

- Good practice

- Sharing your assertions

# Why custom assertions?


nohwnd




PSCONF.EU

nohwnd

# Built-in assertions

- Pester assertions are low level (very generic).
- Should -Be (-eq), -BeTrue (-eq $true), -Match (-match), -Throw (try catch)  etc.

- Pester is often used for specialized stuff (high-level):
  - testing environments
  - validating deployments
  - making sure you databases are in check
  - testing help, documentation
  - etc.

# Some examples

```
Describe "Alllow incoming traffic to web server" {
    It "https is allowed" {
        $firewallRule = Get-NetFirewallRule -DisplayName 'Web server https'

        $firewallRule.Enabled | Should -BeTrue
        $firewallRule.Direction | Should -Be 'Inbound'
        $firewallRule.Action | Should -Be 'Allow'
        $firewallrule.Profile.HasFlag(
 [Microsoft.PowerShell.Cmdletization.GeneratedTypes.NetSecurity.Profile]::Domain ) |
            Should -BeTrue -Because "the firewall rule should be in the Domain profile"

        $portRule = $firewallRule | Get-NetFirewallPortFilter
        $portRule.Protocol | Should -Be 'TCP'
        $portRule.LocalPort | Should -Be 443
    }
}
```

# Problems

- first assertion that fails, will fail the whole test
- test will fail with non-descriptive error messages
- invariants are hard to enforce, e.g. we must not forget to test if the rule is enabled
- there is some hidden knowledge that can't be easily documented, e.g. checking the flag is not straightforward

- overall it's complex code
- we cannot test this

```powershell
# custom assertion
$firewallRule = Get-NetFirewallRule -DisplayName
$firewallRule |
      Should -BeFirewallRule `
             -Allow `
             -Protocol TCP `
             -Port 443 `
             -Profile Domain
```

- hides complexity
- fails with all available information and descriptive error message
- ensures invariants are not forgotten (rule is inbound, rule is enabled)

- easy to read
- easy to re-use
- easy to test

```powershell
$hashtable | Should -HaveKey "Name" -Value "Jakub"

$person | Should -HaveProperty "Name" -Value "Jakub" -Type "string"

$value | Should -BeInRange -Min 1 -MaxInclusive 26

{ Reverse-Array $arr } | Should -BeFaster -Than 500ms -Repeat 1000

$command | Should -HaveParameter Name -Mandatory -Type ([int])

$response | Should -BeHttp Ok -ContentType Json -Body '{ "name" : "Jakub" }'

$certificate | Should -Not -ExpireBefore ([DateTime]::Now.AddDays(7))

$ip | Should -ListenOn 443 -Protocol TCP -TimeOut 3

$ip | Should -BeInSubnet "192.168.1.0/24"

# anything else you want :)
```

# Why custom assertions?

- your tests become easier to read
- your error messages will give you more information
- you can test your assertions to make sure they work
- you are likely testing something that someone else is also testing, so you can share 🙂

# Should -HaveParameter

Oliver wrote something useful.

Oliver made a PR.

Should -HaveParameter shipped in 4.6.0.

**Be like Oliver.** 🙂

https://github.com/lipkau


nohwnd

# Extending should


nohwnd

# Basic building blocks

- a should function
- required parameters
- result object

- operator registration

# Should -BeAwesome

to help me ensure the facts of life


nohwnd

# Ensuring the facts of life...

TOP DEFINITION

## Jaap

Jaap is the name given to the most awesome people. They rock at everything they do, making **everybody else** love them beyond limits. Jaap is a God **amongst men**, everything he touches turns to gold and everything he says is true.. Jaaps tend to be the best looking people alive (or dead), and have very large male **productive** organs. Jaap is (the) Future.

*Check out what Jaap is doing, he is **my hero**!*

***Ooh**, Jaap is soo hot, I must have his **babies**.*

#jaap #awesome #rock #future #dutch

**by TomorrowTime May 13, 2009**

129 51

**Get a Jaap mug for your mate Manafort.**


# DEMO

Should -BeAwesome

# Should –BeInSubnet


nohwnd

# Should -BeInSubnet is our example

```
$ip | Should -BeInSubnet "192.168.1.0/24"
```

```
     192 .      168 .        1 .      170
11000000 . 10101000 . 00000001 . 10101010 <- host

     192 .      168 .        1 .        0
11000000 . 10101000 . 00000001 . 00000000 <- subnet

     255 .      255 .      255 .        0
11111111 . 11111111 . 11111111 . 00000000 <- mask
```

http://cidr.xyz/


nohwnd

# DEMO

## Should -BeInSubnet

# Should –All

Consuming array input


nohwnd

# DEMO

Should -All


nohwnd

# Good practice


nohwnd

# Good practice

- Name your function **Should-***
- Provide comment based help with examples so it can be looked up by **Get-ShouldOperator**
- Use **-Be** or **-Have** prefix if it fits
- Use 'Expected <expected value>, but got <actual value>.'
- Support **-Not**, or throw and exception when used
- Support **-Because**
- When checking for failure check for '**PesterAssertionFailed**'
- Write tests!

# Where do I start?


nohwnd

# Where do I start?

Look for **Should -BeTrue** or **Should -Be $true**

Look for tests that are complex or have more than one assertions

Add **-Not** to some of your tests and look at the errors, are they helpful? Can you easily improve them by adding **-Because**?

# Sharing your assertions


nohwnd

# Sharing your assertions

- Simply pack them in a module
- You can avoid publishing the Should- functions by using **`Export-ModuleMember -Function @()`**

# Other approaches

# Function that throws

easy to start with

no special knowledge is necessary

absolute freedom of what you do

easy to test

harder to discover

does not look native to Pester

See https://github.com/nohwnd/Assert for inspiration

# DEMO

## Custom assertion function

# DEMO

## Assert module

# Summary

- Adding your own assertions is simple with just a few steps to follow
- Test your assertions to make sure they work as they should

- If you make something generally useful consider sharing it!


nohwnd

I have Pester stickers come say hi and grab some.

nohwnd

# Come join discussion about Pester v5

Today, Track 2, 12:25

Please read Pester -> issues -> Community talk about Pester 5

tiny.cc/talkv5


nohwnd

# Slides and demo code

```
Start-Process -FilePath https://github.com/psconfeu/2019
```

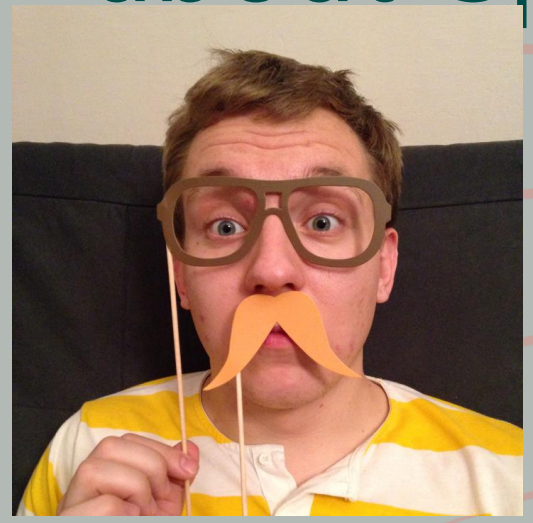
nohwnd

# Questions?

Use the conference app to vote for this session:
https://my.eventraft.com/psconfeu

nohwnd

# about Speaker



Jakub Jares

@nohwnd

me@jakubjares.com

I am Jakub Jares and enjoy coding and architecting solutions. For the past few years I work as a .NET Developer for cngroup.dk in Prague, Czech Republic. I mainly work with C# and F#, during the day. But in the evenings I often go back to PowerShell and improve and maintain Pester, or try to find out if something crazy, like doing MVVM in PowerShell, is possible.