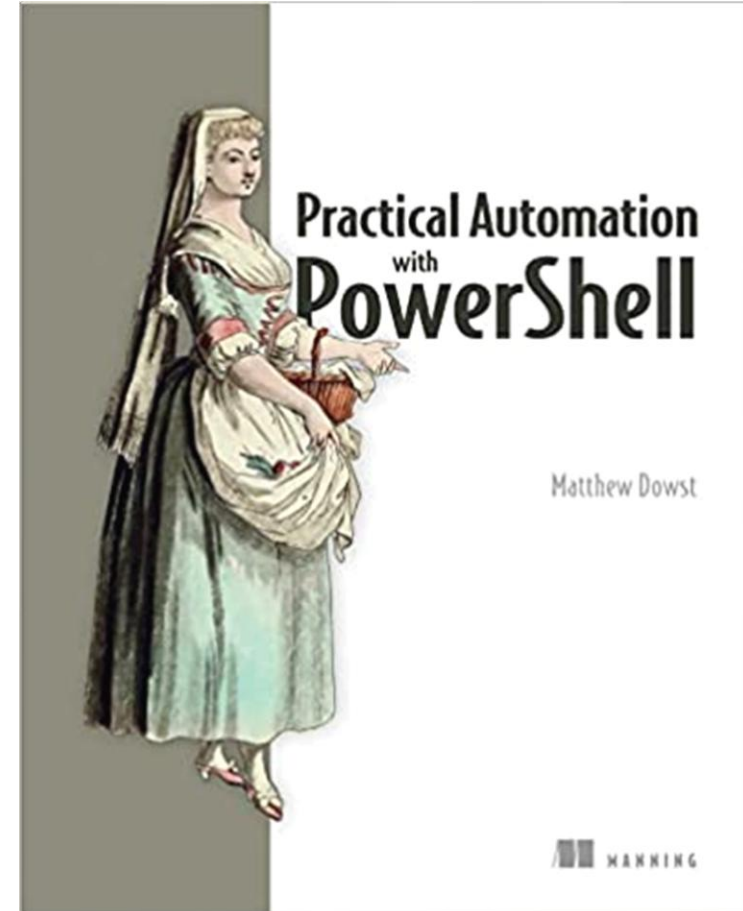




Taking Pester Beyond the Code

Matthew Dowst

- Principal Consultant
- Managed automation architect
- Author



Why Pester?



- ✓ Open-source testing framework for PowerShell
- ✓ Built specifically for PowerShell
- ✓ Intuitive Syntax
- ✓ Work with PowerShell 3, 4, 5, 6, and 7
- ✓ Hundreds of contributors

Key Features



ASSERTIONS



TEST DISCOVERY



SETUP AND TEARDOWN

How it works

Pester Syntax

- Describe
- Context
- It
- Should

```
Describe "Get-Sum function" {  
    Context "when input is positive numbers" {  
        It "returns the correct sum" {  
            $result = Add-Numbers -Number 2 -Plus 3  
            $result | Should -Be 5  
        }  
    }  
    Context "when input includes negative numbers" {  
        It "returns the correct sum" {  
            $result = Add-Numbers -Number 2 -Plus -3  
            $result | Should -Be -1  
        }  
    }  
}
```

Common Gotchas

- Scoping
- Test Pollution
- Overreliance on set up and tear down
- Not using the correct -Be

● Pester v5.3.1

```
Starting discovery in 1 files.  
Discovery found 2 tests in 4ms.  
Running tests.
```

```
Running tests from 'D:\Projects\Pester_Infrastructure_Test'
```

```
Describing Get-Sum function
```

```
Context when input is positive numbers
```

```
  [+] returns the correct sum 18ms (1ms|17ms)
```

```
Context when input includes negative numbers
```

```
  [+] returns the correct sum 3ms (1ms|2ms)
```

```
Tests completed in 72ms
```

```
Tests Passed: 2, Failed: 0, Skipped: 0 NotRun: 0
```

Thank You

- All presentation materials are available at: github.com/mdowst
- Blog: dowst.dev
- PowerShell Weekly: psweekly.dowst.dev
- Find me: linktr.ee/mdowst

