Stop manually publishing your PowerShell modules

Automate it in 2 minutes!

Daniel Schroeder

@deadlydog

CC BY SA

PURESTORAGE®

ScriptRunner®
The #1 for PowerShell Management

CC BY SA

Thank you iQmetrix

iQmetrix's purpose is to Create Great Experiences. We aim to enrich the lives of our employees, our partners, our clients, and their customers by enabling a wireless retail experience that's as human as our fundamental need for connection.

CC BY SA

About Dan

```powershell
$Dan = @{
    Name = 'Daniel Schroeder'
    Alias = 'deadlydog'
    Role = 'Individual Contributor'
    Company = 'iQmetrix'
    Address = @{
        City = 'Regina'
        Province = 'Saskatchewan'
        Country = 'Canada'
    }
    Experience = '2+ decades writing code'
    Blog = 'https://blog.danskingdom.com'
    Passions = @('Coding', 'Blogging', 'Automation', 'Dev Productivity Tools')
    Likes = @('.NET', 'PowerShell', 'Knowledge Sharing', 'Dogs', 'Roller Blading')
}
```

Agenda

- Why use modules?
- Why automate publishing?
- Ways to automate publishing
- Demo
- Wrap up
- Questions

CC BY SA

Why use modules?

- Easy to install — Install-Module (Install-PSResource)
- Easy to update — Update-Module (Update-PSResource)
- Easy to use — ~~& C:\Some\Path\Script.ps1~~

CC BY SA

Evolution to modules

1. Run commands in the terminal
2. Put commands into a script
3. Put Params on the script
4. Turn the script into a module

Not every script needs to be a module

- Good candidates: Used/updated frequently, shared with others

CC BY SA

Publishing a module is easy

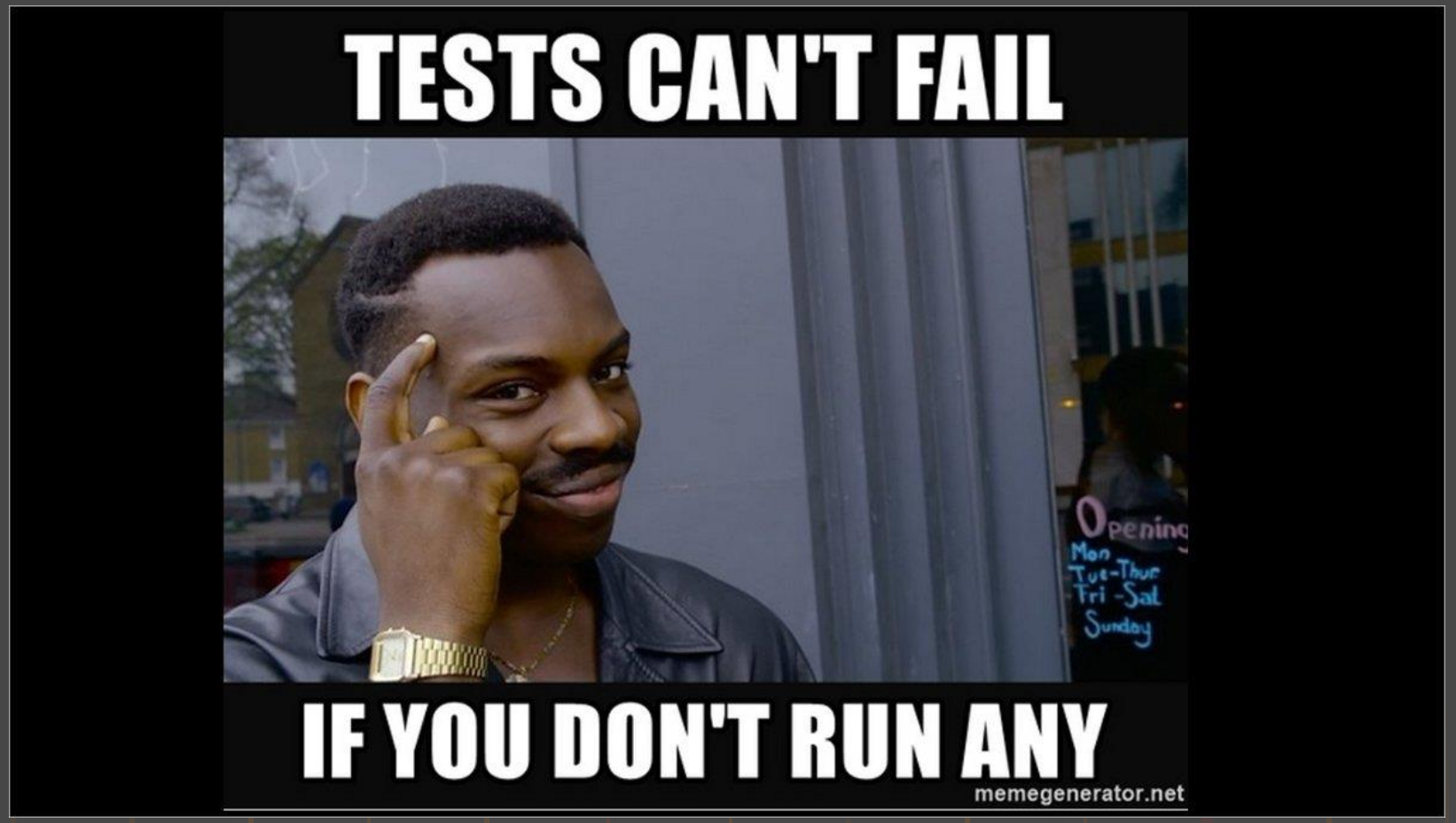
- Publish-Module (Publish-PSResource)
  - -Path C:\PathTo\MyModule –NuGetApiKey *****
  - -Repository MyInternalPowerShellFeed

CC BY SA

Oh right, version number

CC BY SA

Oh right, tests



CC BY SA

Oops, only I have the API key

CC BY SA

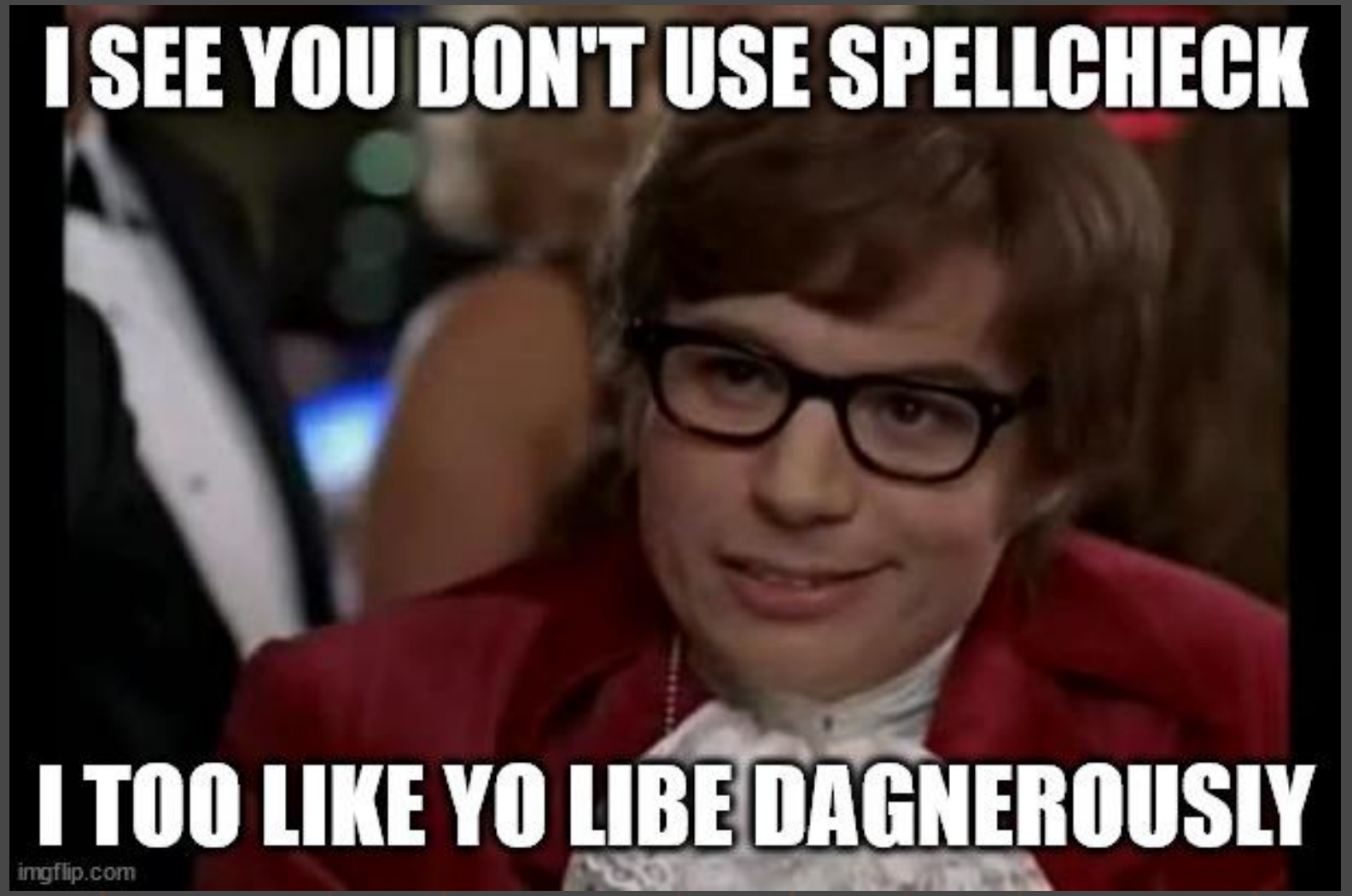
Time for deployment instructions

CC BY SA

Regenerate documentation

CC BY SA

Spellcheck code and docs

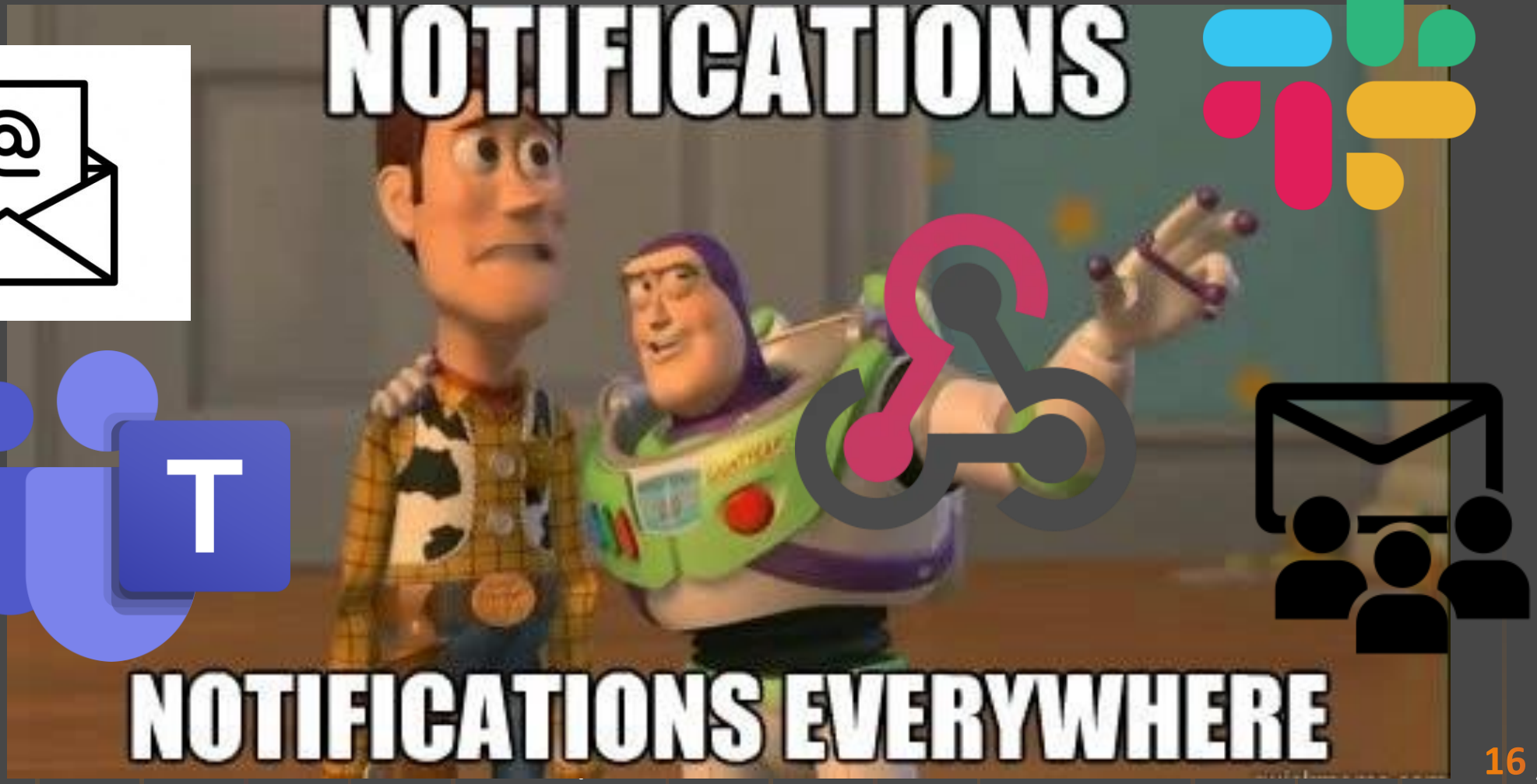


CC BY SA

Oh right, restrict access

CC BY SA

Send notifications



CC BY SA

Tag commit with the version number

CC BY SA

Let's write a deployment script!

PS> See-CheckList | Write-AutomationScript

An improvement, but we can do better with

Pipelines!

Actions / Workflows

CC BY SA

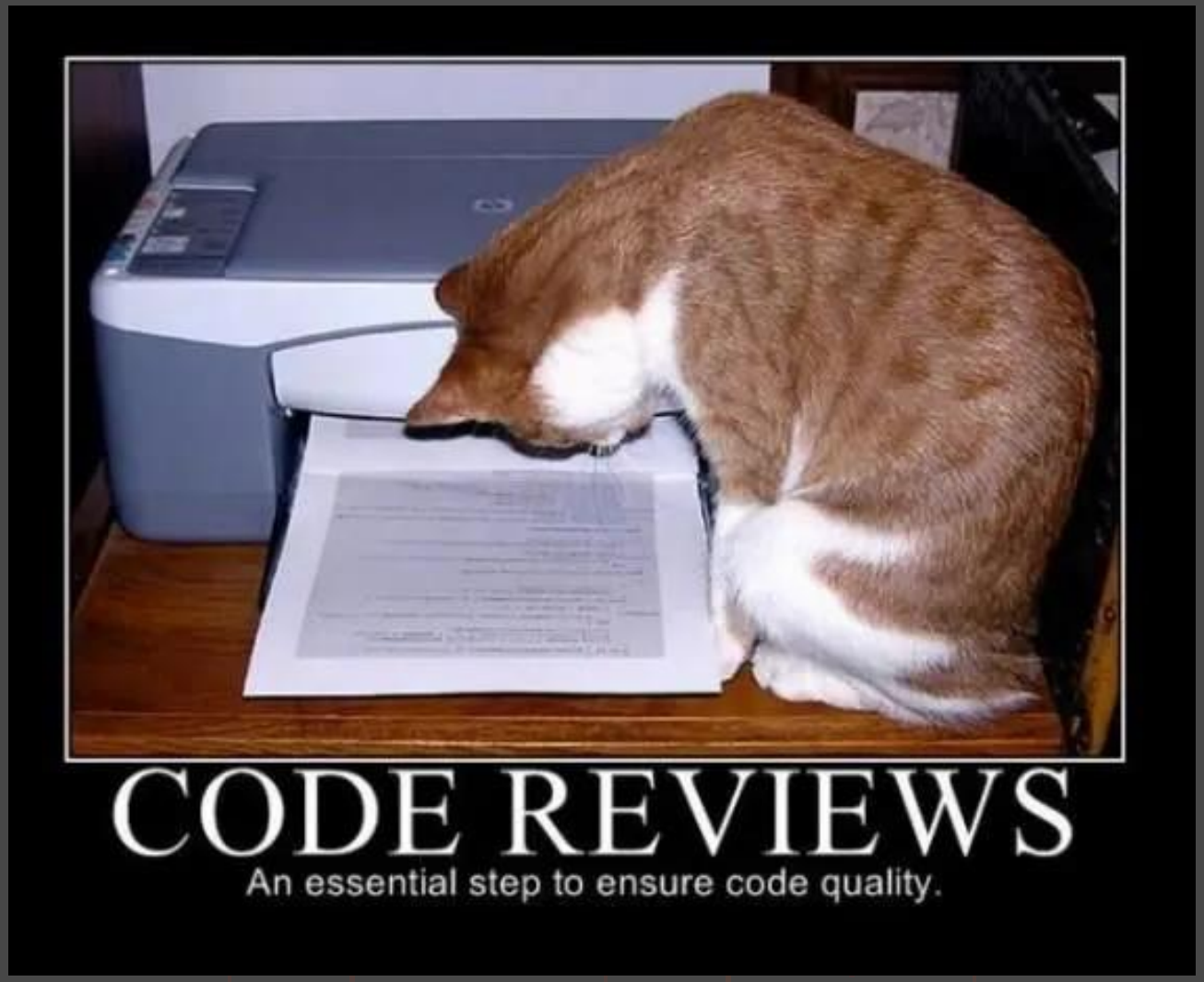
Use source control

sre-one-off-scripts

- src
  - Copy-ADComputerGroupMembershipToAnotherComputer
  - DisableWindowsServiceOnServers
  - Get-AllAzureResources
  - Get-IisEnvironmentVariablesOnServers
  - Get-ReginaServerPatchInfoFromPuppet
  - Get-ScheduledTasksFromServers
  - Get-WindowsVersionAndStatsOfAllServers
  - RemoveApplicationInsightsResources
  - Template.NewScript
  - Template.RunCodeOnEveryActiveDirectoryVirtualMachine
  - Template.RunCodeOnEveryDataCenterVirtualMachine
  - Watch-GitHubUsersRateLimit
- .editorconfig
- .gitignore
- ReadMe.md

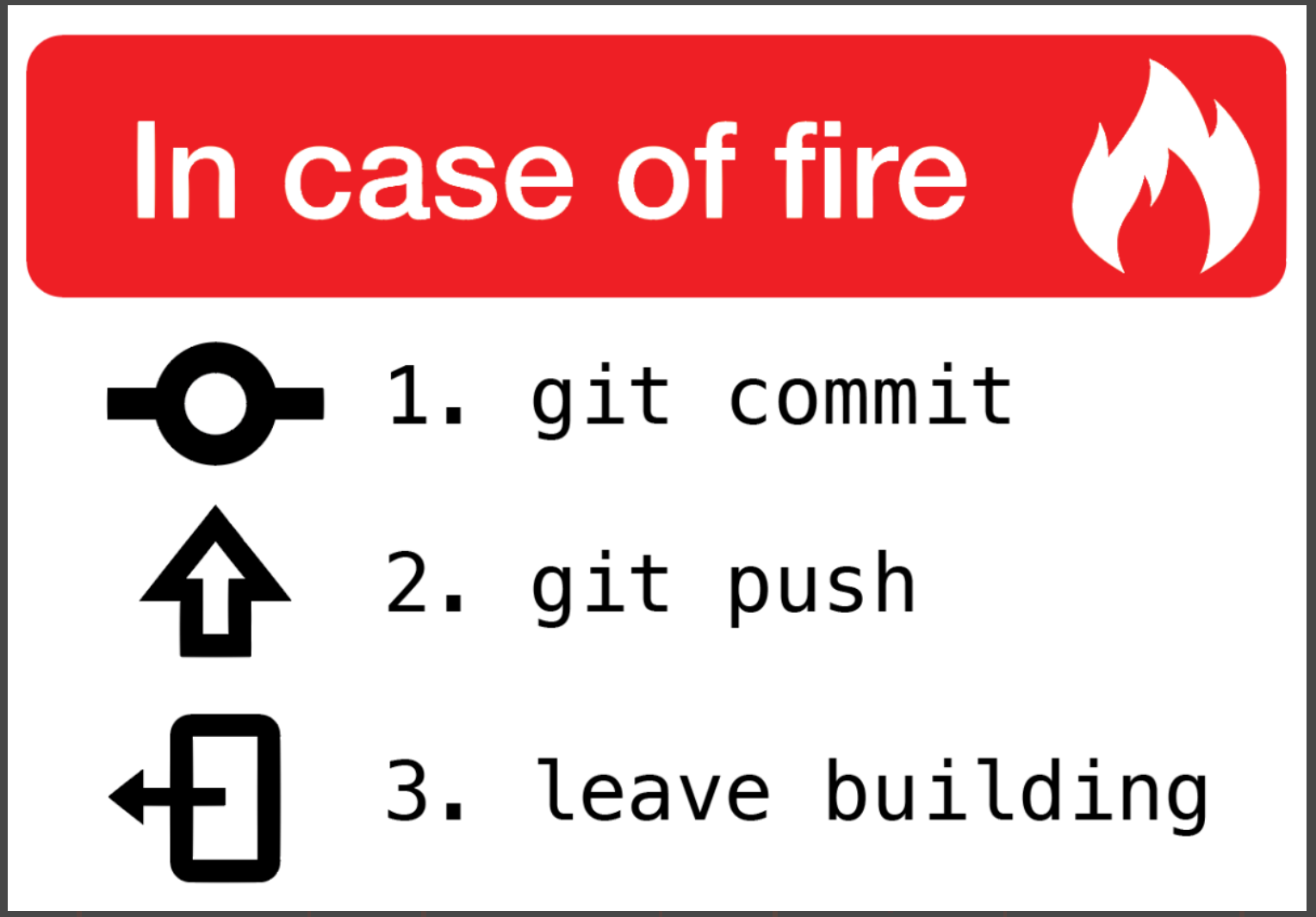
CC BY SA

Don’t forget code reviews



CC BY SA

Oh right, push your code up



CC BY SA

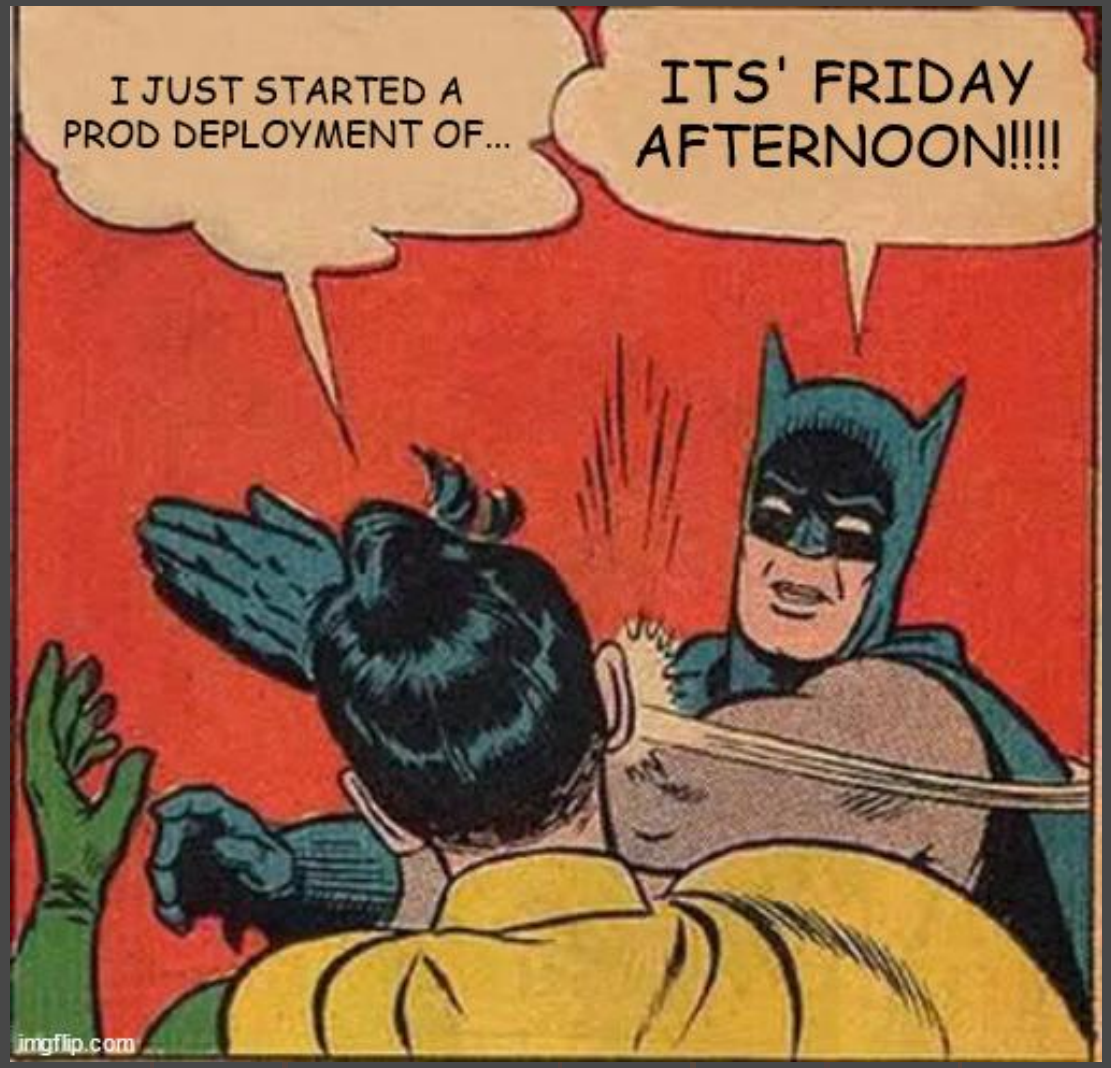
Publish from the main branch

CC BY SA

Get approval

CC BY SA

Deployment schedules

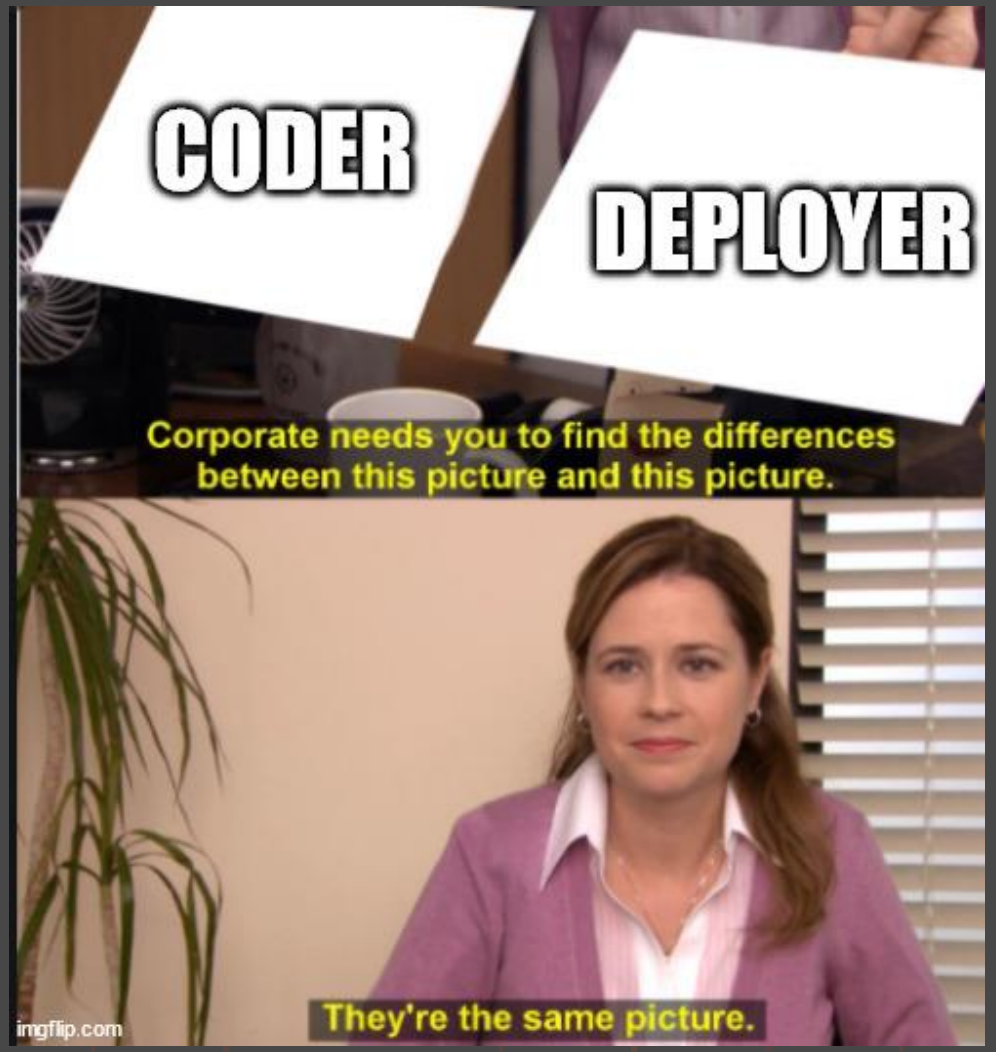


CC BY SA

Baking new versions

CC BY SA

Compliance



CC BY SA

Why use a pipeline?

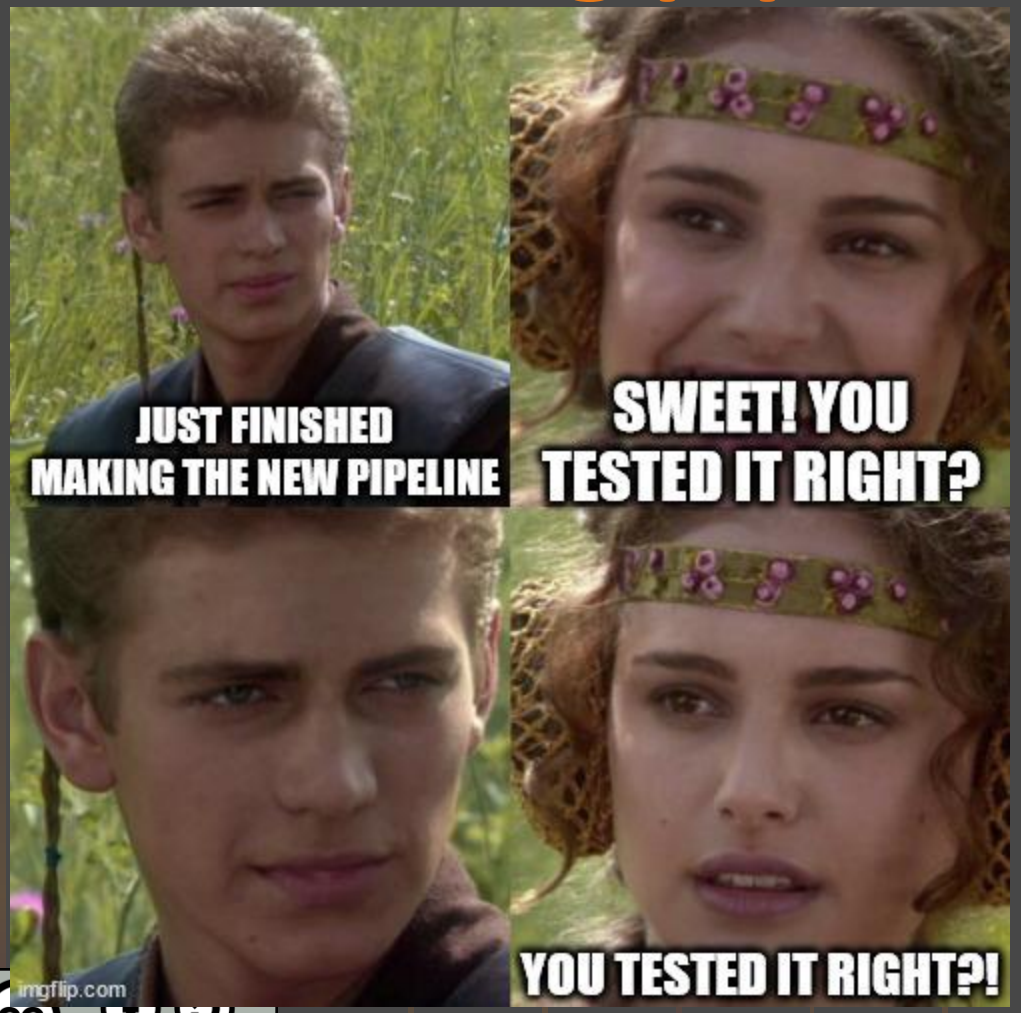
- Helps enforce good practices:
  - Source control
  - Enables branch policies for code reviews
  - Code must be pushed to remote repo
  - Only publish from the main branch by default
  - Enables approvals and deployment schedules
  - Long-running deployment processes
- Pit of Success

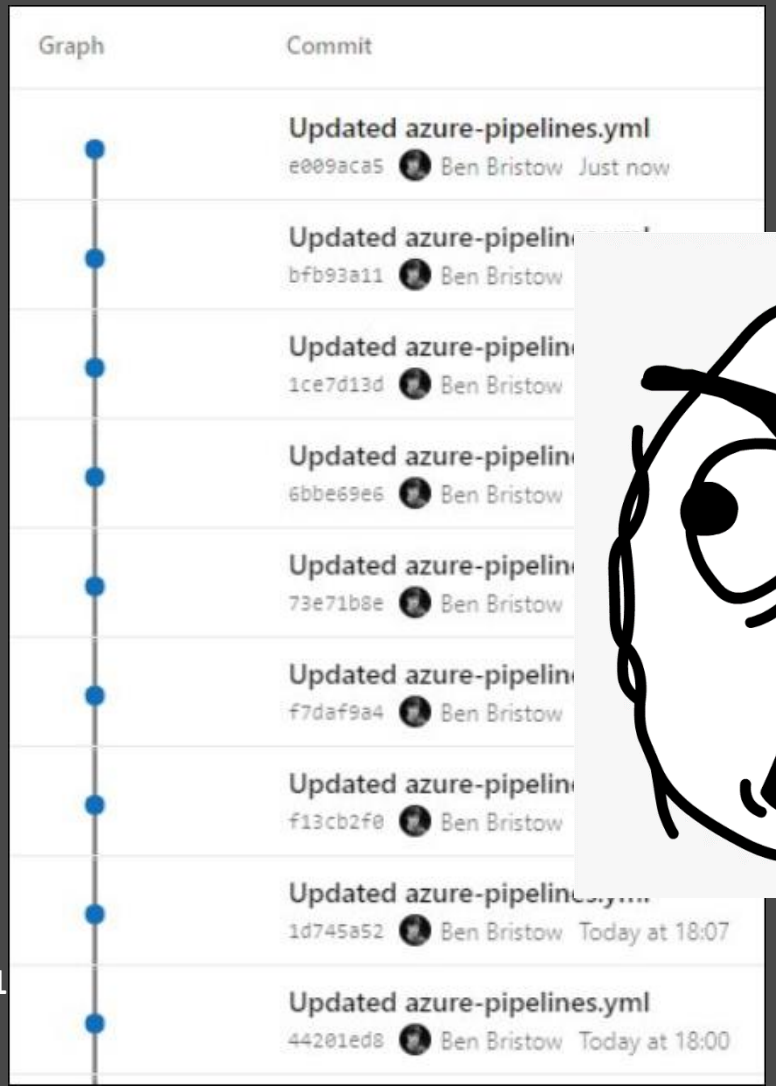
CC BY SA

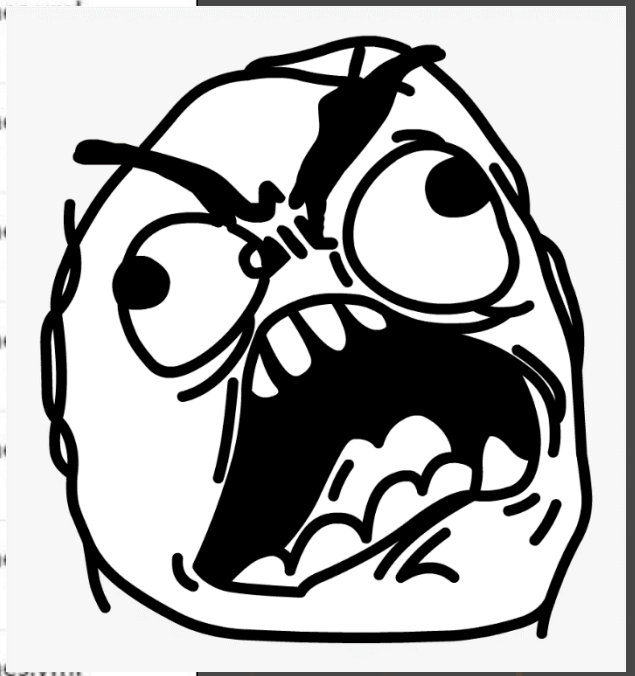
Why use a pipeline?

CC BY SA

Testing pipelines can be..... frustrating







CC BY SA

Demo

Let's publish a new module!

github.com/deadlydog/PowerShell.ScriptModuleRepositoryTemplate

CC BY SA

Other templates?

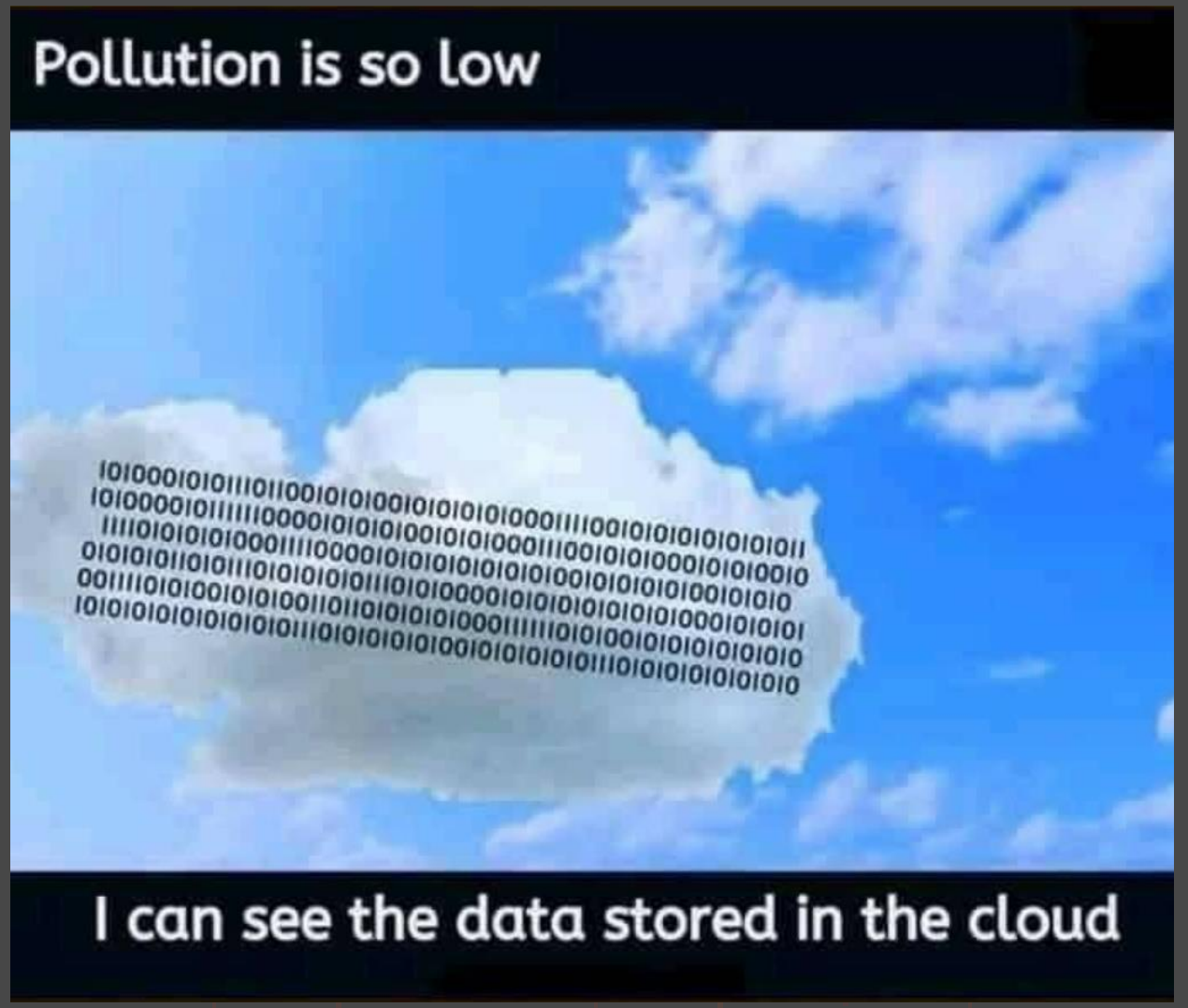
- PSModuleTemplate
- ModuleTemplate
  - No docs

- Module.Template
  - AppVeyor for CI/CD

- Stucco
- PSModuleDevelopment

CC BY SA

Pipelines aren't all sunshine



CC BY SA

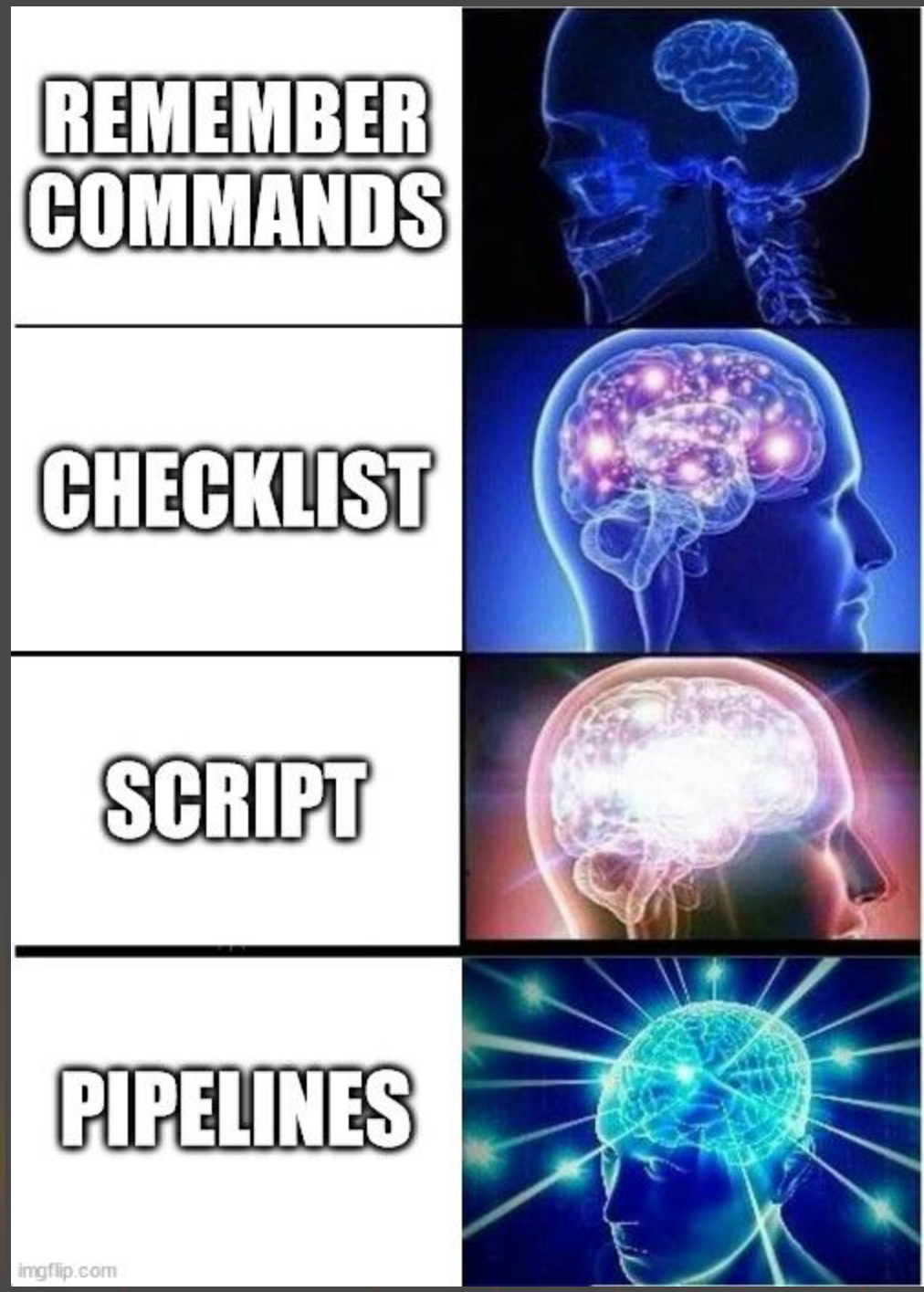
Scripts vs. Marketplace

CC BY SA

Recap

Pipelines over manual script running:

- Consistency and repeatability
- Logging (persisted) and Visibility
- Security (permissions and API keys)
- Auditing
- Notifications
- Approvals and schedules
- Compliance



CC BY SA

THANK YOU

Please review this session

Demo module/template:
ScriptModuleRepositoryTemplate

Session Review

twitter.com/deadlydog

github.com/deadlydog

blog.danskingdom.com

CC BY SA

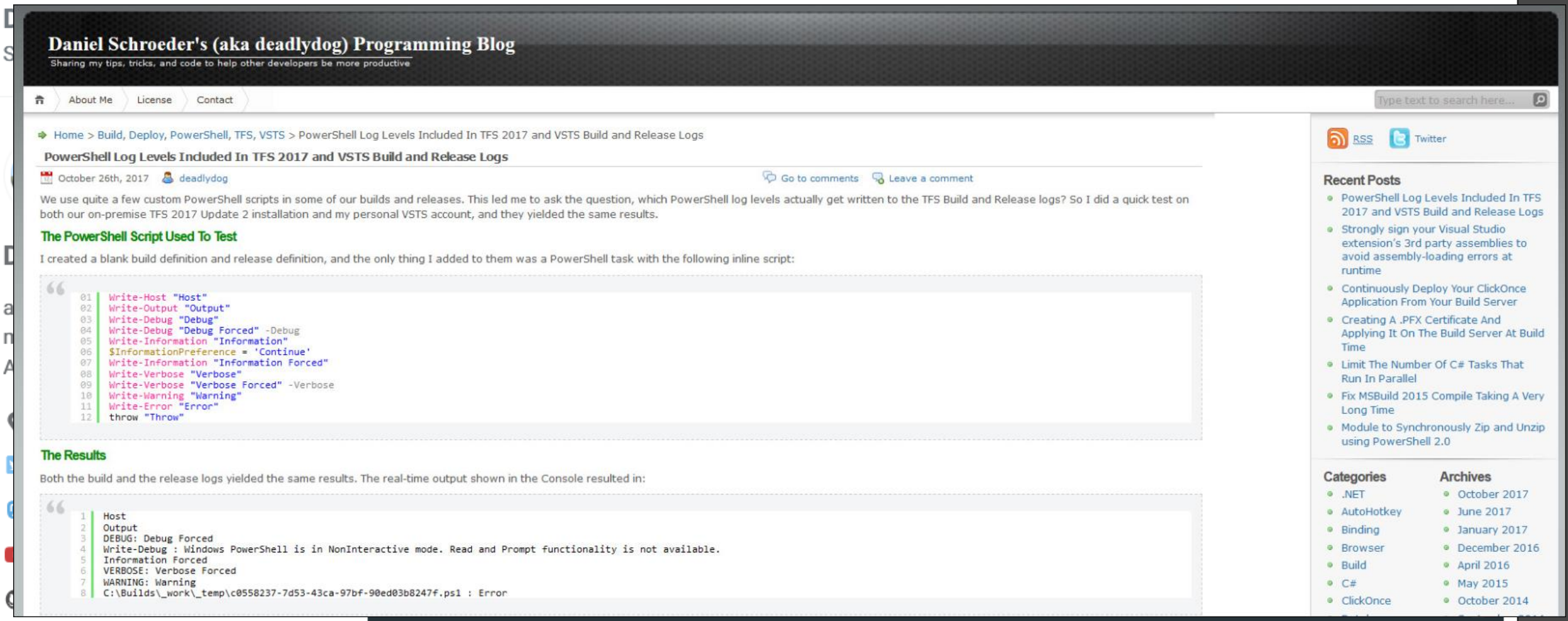
Heading

CC BY SA

Pipeline YAML

- Web editor is king
- VS Code extensions
  - GitHub Actions
  - Azure Pipelines
- Syntax highlighting, autocomplete, etc.

CC BY SA



Daniel Schroeder's (aka deadlydog) Programming Blog

Sharing my tips, tricks, and code to help other developers be more productive

About Me | License | Contact

Type text to search here...

Home > Build, Deploy, PowerShell, TFS, VSTS > PowerShell Log Levels Included In TFS 2017 and VSTS Build and Release Logs

PowerShell Log Levels Included In TFS 2017 and VSTS Build and Release Logs

October 26th, 2017 deadlydog

Go to comments Leave a comment

We use quite a few custom PowerShell scripts in some of our builds and releases. This led me to ask the question, which PowerShell log levels actually get written to the TFS Build and Release logs? So I did a quick test on both our on-premise TFS 2017 Update 2 installation and my personal VSTS account, and they yielded the same results.

The PowerShell Script Used To Test

I created a blank build definition and release definition, and the only thing I added to them was a PowerShell task with the following inline script:

```
Write-Host "Host"
Write-Output "Output"
Write-Debug "Debug"
Write-Debug "Debug Forced" -Debug
Write-Information "Information"
$InformationPreference = 'Continue'
Write-Information "Information Forced"
Write-Verbose "Verbose"
Write-Verbose "Verbose Forced" -Verbose
Write-Warning "Warning"
Write-Error "Error"
throw "Throw"
```

The Results

Both the build and the release logs yielded the same results. The real-time output shown in the Console resulted in:

```
Host
Output
DEBUG: Debug Forced
Write-Debug : Windows PowerShell is in NonInteractive mode. Read and Prompt functionality is not available.
Information Forced
VERBOSE: Verbose Forced
WARNING: Warning
C:\Builds\_work\_temp\c0558237-7d53-43ca-97bf-90ed03b8247f.ps1 : Error
```

RSS Twitter

Recent Posts

- PowerShell Log Levels Included In TFS 2017 and VSTS Build and Release Logs
- Strongly sign your Visual Studio extension's 3rd party assemblies to avoid assembly-loading errors at runtime
- Continuously Deploy Your ClickOnce Application From Your Build Server
- Creating A .PFX Certificate And Applying It On The Build Server At Build Time
- Limit The Number Of C# Tasks That Run In Parallel
- Fix MSBuild 2015 Compile Taking A Very Long Time
- Module to Synchronously Zip and Unzip using PowerShell 2.0

Categories

- .NET
- AutoHotkey
- Binding
- Browser
- Build
- C#
- ClickOnce

Archives

- October 2017
- June 2017
- January 2017
- December 2016
- April 2016
- May 2015
- October 2014

LinkedIn

Stack Overflow

Email

Blog RSS Feed

```
Add-TiPSImportToPowerShellProfile

Set-TiPSConfiguration -AutomaticallyWritePowerShellTip Daily -AutomaticallyUpdateModule Weekly
```

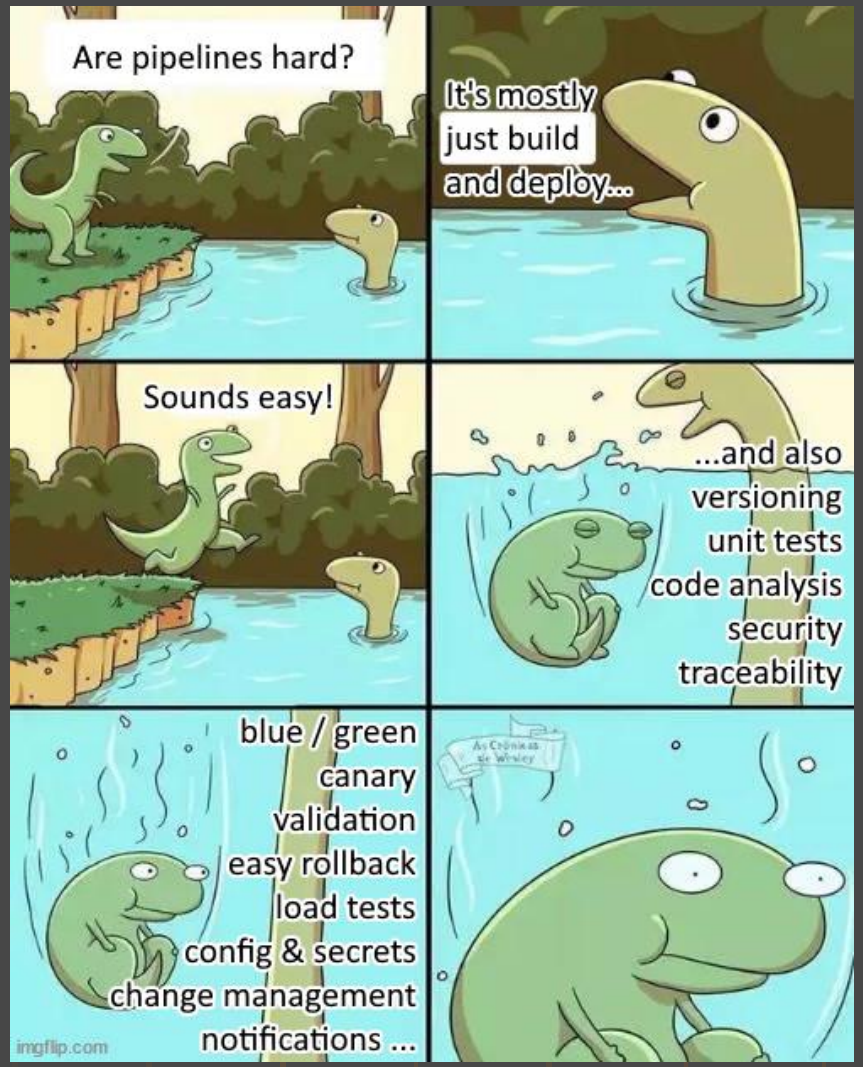
Display a tip

To display a PowerShell tip, simply run the `Write-PowerShellTip` command, or use its alias `tips`.

Why use pipelines?

CC BY SA

CC BY SA


Are pipelines hard?
It's mostly just build and deploy...
Sounds easy!
...and also
versioning
unit tests
code analysis
security
traceability
blue / green
canary
validation
easy rollback
load tests
config & secrets
change management
notifications ...
imgflip.com

CC BY SA