[Skip to end of metadata](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes" \l "page-metadata-end)

* Created by [Babbitt, Ricc](https://wiki.ercot.com/display/~rbabbitt), last modified on [Feb 14, 2021](https://wiki.ercot.com/pages/diffpagesbyversion.action?pageId=145787262&selectedPageVersions=12&selectedPageVersions=13)

[Go to start of metadata](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#page-metadata-start)

* [Overview](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-Overview)
  + [Purpose](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-Purpose)
  + [Problem](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-Problem)
  + [Benefits](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-Benefits)
  + [Terms](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-Terms)
* [Conceptual](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-Conceptual)
  + [Current State](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-CurrentState)
  + [Future State](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-FutureState)
* [The Pipeline](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-ThePipeline)
  + [Overview](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-Overview.1)
  + [Triggering a Pipeline](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-TriggeringaPipeline)
  + [Feedback Loops](https://wiki.ercot.com/pages/viewpage.action?spaceKey=CIA&title=PowerShell+Development+Processes#PowerShellDevelopmentProcesses-FeedbackLoops)

Overview

Purpose

This document prescribes methods for maintaining and promoting PowerShell code changes, refactoring our current PowerShell scripts into parameterized PowerShell modules, introduce methods for automating code analysis and unit tests, and publishing PowerShell artifacts for use in ERCOT's config and automation pipelines.

Problem

Today, we employ many static PowerShell scripts to carry out different processes for our build automation, however any static code analysis and/or unit testing is a slow manual process, prone to errors and inconsistency.

Benefits

Following are benefits to using strategies described in this document

* Testing is performed frequently and is visible, allowing easier identification of any changes that break a pipeline.
* Non-Monolithic scripts are easier for developers to work with and maintain.
* The published artifact is a monolithic PowerShell module that is optimized for production use.
* In addition to being consumed by our automation processes, published PowerShell modules are easily reusable by both development and operations teams for use when manual intervention is needed (eg IAM team for creating infrastructure access groups; DBA team for automated SQL installs)

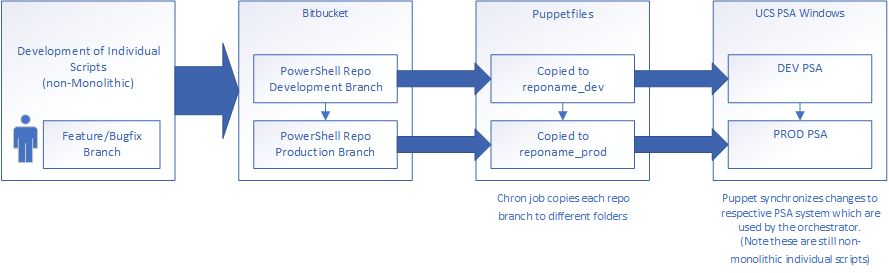
Terms

CI/CD terms can mean different things to different people therefore it is important to understand what they mean in this document.

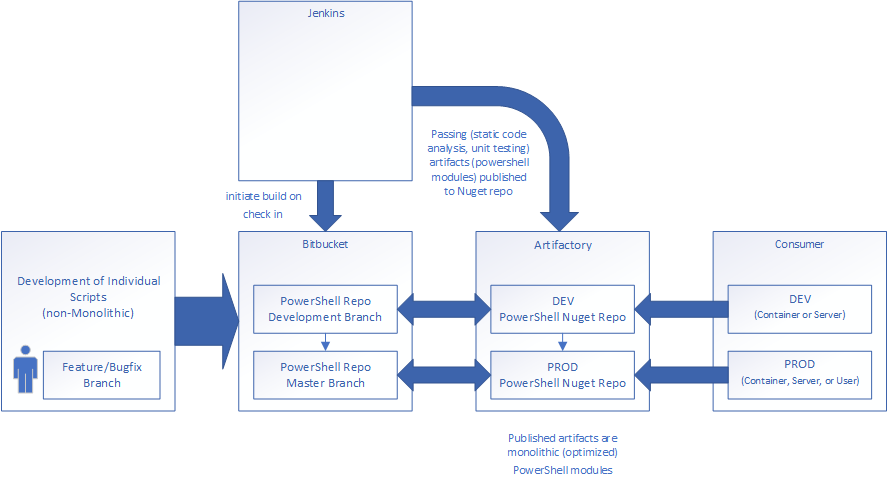
* Build:  Compiling source PowerShell project into a new artifact. For PowerShell, this artifact will be a new module containing various parameterized functions to be used by machine or person
* Test:  Code analysis and unit tests performed against a PowerShell project before building a new artifact.
* Deploy:  Publishing the artifact to the artifact repository ready for consumption.
* Release:  Deploying the newly version of the artifact to be used in the environment or higher environment

Conceptual

Current State



Future State



The Pipeline

Overview

Every check-in should lead to a potential release. A change in source code via git commit should trigger a notification to the CI/CD tool (Jenkins) to try and build a new artifact.

|  |  |
| --- | --- |
| Build | A webhook can be used to trigger Jenkins to initiate a build and potentially deploy a new artifact.  If tests are passing, this process will combine individual scripts into a packaged PowerShell module |
| Test | PSScriptAnalyzer static code analysis and Pester unit tests will be performed.  Only if these are passing will a new module be built |
| Deploy | Built modules will be published to a Nutget repository.  Version bumping is required |
| Release | Different strategies for releasing higher version modules into our build environment   * Use Requires version statement in any script that uses a module * Use -RequiredVersion parameter when importing or installing the module   The release pipeline can be further automated using Pester for Infrastructure Validation Testing |

Triggering a Pipeline

It is important to understand the following components listed below since introducing changes to any one of them has potential to break our deployment pipeline. Every change made to PowerShell source code, its data, PowerShell's configuration and/or environment, should trigger a new instance of the pipeline.  
Any one of the following potentially can change the behavior of the system and therefore should be tested.

|  |  |
| --- | --- |
| Executable Code | PowerShell Modules - Packages that contain PowerShell scripts and/or compiled DLLs that extend PowerShell's capabilities.  PSSnapins - .NET Framework assemblies that contain PowerShell providers and/or cmdlets.  PowerShell Scripts – Files that contain a series of PowerShell commands. These may be build scripts used to initiate a new PowerShell module build from source repository, or iterative controller scripts used to invoke PowerShell modules wherever they are used in other processes. |
| Environment / Configuration | Host environment – Including operating system version and/or patch level.  WMF - Upgrading Windows Management Framework includes updates to Windows PowerShell, Windows PowerShell Desired State Configuration (DSC), Windows Remote Management (WinRM), and Windows Management Instrumentation (WMI). |
| Data Structure | Changes to any structured data feeding into a pipeline should trigger a new feedback loop. |

Feedback Loops

A feedback process involves automated testing of every change, including:

* The process to build a new PowerShell module artifact.
* Static Code analysis – PSscriptAnalyzer built-in and custom rules should pass. We should also run compatibility checks to ensure module will be compatible above a minimum version of PowerShell present in the environment.
* Unit tests – All Pester unit tests should pass.
* Test coverage – A minimum percentage code coverage for unit tests should be met.