

Continuous Delivery to the PowerShell Gallery in 5 minutes

TODAY'S TO DO LIST

1. GET UP
2. Drink Coffee
3. Create Continuous Delivery
Process for Delivering Modules
to the PowerShell Gallery
4. In only 5 minutes 😊

Why Module?

- Ps1 scripts EVERYWHERE!!
- Use them in other scripts
- Share them
- Modularity
- Only export functions user needs

Why PowerShell Gallery?

- Public or Private Nuget Repository
- Share your Modules
- Centralise deployment of Modules
- PowerShell Gallery is safe and secure

What do we need?

- Plaster – Install-Module Plaster
- Pester – Install-Module Pester -Force
(even though it is installed by default)

What do we need?

- Plaster – Install-Module Plaster
- Pester – Install-Module Pester -Force
(even though it is installed by default)
- GitHub Account
- VSTS account

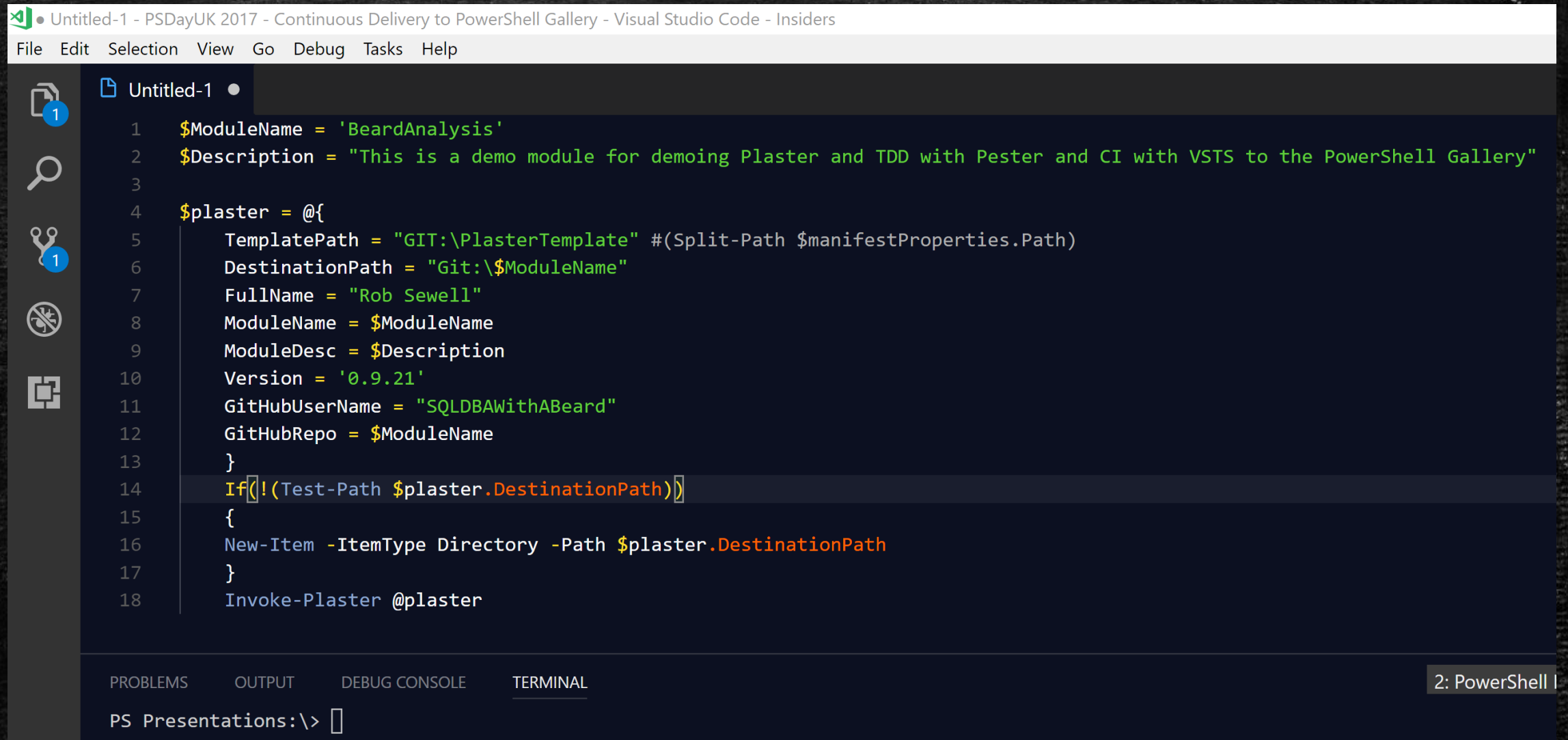
We won't have time for much detail!!

Plaster

The image shows a Visual Studio Code window with the 'PlasterManifest.xml' file open. The Explorer on the left shows the project structure, including folders like 'docs', 'functions', 'internal', 'tests', and 'vscode'. The main editor displays the XML content of 'PlasterManifest.xml', which includes metadata, parameters, and content sections. The content section describes creating folder structure and deploying common files. The bottom status bar shows the current file is 'PlasterManifest.xml' and the editor is in 'XML' mode.

```
<?xml version="1.0" encoding="utf-8" ?>
<title>Full module template</title>
<description></description>
<author>Rob Sewell</author>
<tags></tags>
</metadata>
<parameters>
  <parameter name="FullName" type="text" prompt="Module author's name" />
  <parameter name="ModuleName" type="text" prompt="Name of your module" />
  <parameter name="ModuleDesc" type="text" prompt="Brief description on this module" />
  <parameter name="Version" type="text" prompt="Initial module version" default="0.0.1" />
  <parameter name="GitHubUserName" type="text" prompt="GitHub username" default="${PLASTER_PARAM_
  <parameter name="GitHubRepo" type="text" prompt="Github repo name for this module" default="${F
</parameters>
<content>
  <message>
    Creating folder structure
  </message>
  <file source='' destination='docs' />
  <file source='' destination='functions' />
  <file source='' destination='internal' />
  <file source='' destination='tests' />
  <message>
    Deploying common files
  </message>
  <file source='appveyor.yml' destination='' />
  <file source='contributing.md' destination='' />
</content>
</PlasterManifest>
```


Plaster



The screenshot shows the Visual Studio Code interface with a file named 'Untitled-1'. The editor contains a PowerShell script for creating a Plaster module. The script defines variables for the module name, description, and a hashtable for Plaster properties. It then checks if the destination path exists and creates it if necessary, before invoking the Plaster command.

```
1 $ModuleName = 'BeardAnalysis'
2 $Description = "This is a demo module for demoing Plaster and TDD with Pester and CI with VSTS to the PowerShell Gallery"
3
4 $plaster = @{
5     TemplatePath = "GIT:\PlasterTemplate" #(Split-Path $manifestProperties.Path)
6     DestinationPath = "Git:\$ModuleName"
7     FullName = "Rob Sewell"
8     ModuleName = $ModuleName
9     ModuleDesc = $Description
10    Version = '0.9.21'
11    GitHubUserName = "SQLDBAWithABeard"
12    GitHubRepo = $ModuleName
13 }
14 If(!(Test-Path $plaster.DestinationPath))
15 {
16     New-Item -ItemType Directory -Path $plaster.DestinationPath
17 }
18 Invoke-Plaster @plaster
```

The bottom of the window shows the 'TERMINAL' tab with the prompt 'PS Presentations:\>' and a cursor. The status bar on the right indicates '2: PowerShell'.

Plaster

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS Presentations:\> Invoke-Plaster @plaster
```


Pester

Write Test

Run Test

Fail Test

Write Code

Run Test

Pass Test

Repeat

Pester

The screenshot shows the Visual Studio Code interface with two PowerShell files open. The left file, `Invoke-BeardAnalysis.Tests.ps1`, contains Pester test definitions. The right file, `Invoke-BeardAnalysis.ps1`, contains a function definition. The terminal at the bottom shows the command to run the tests.

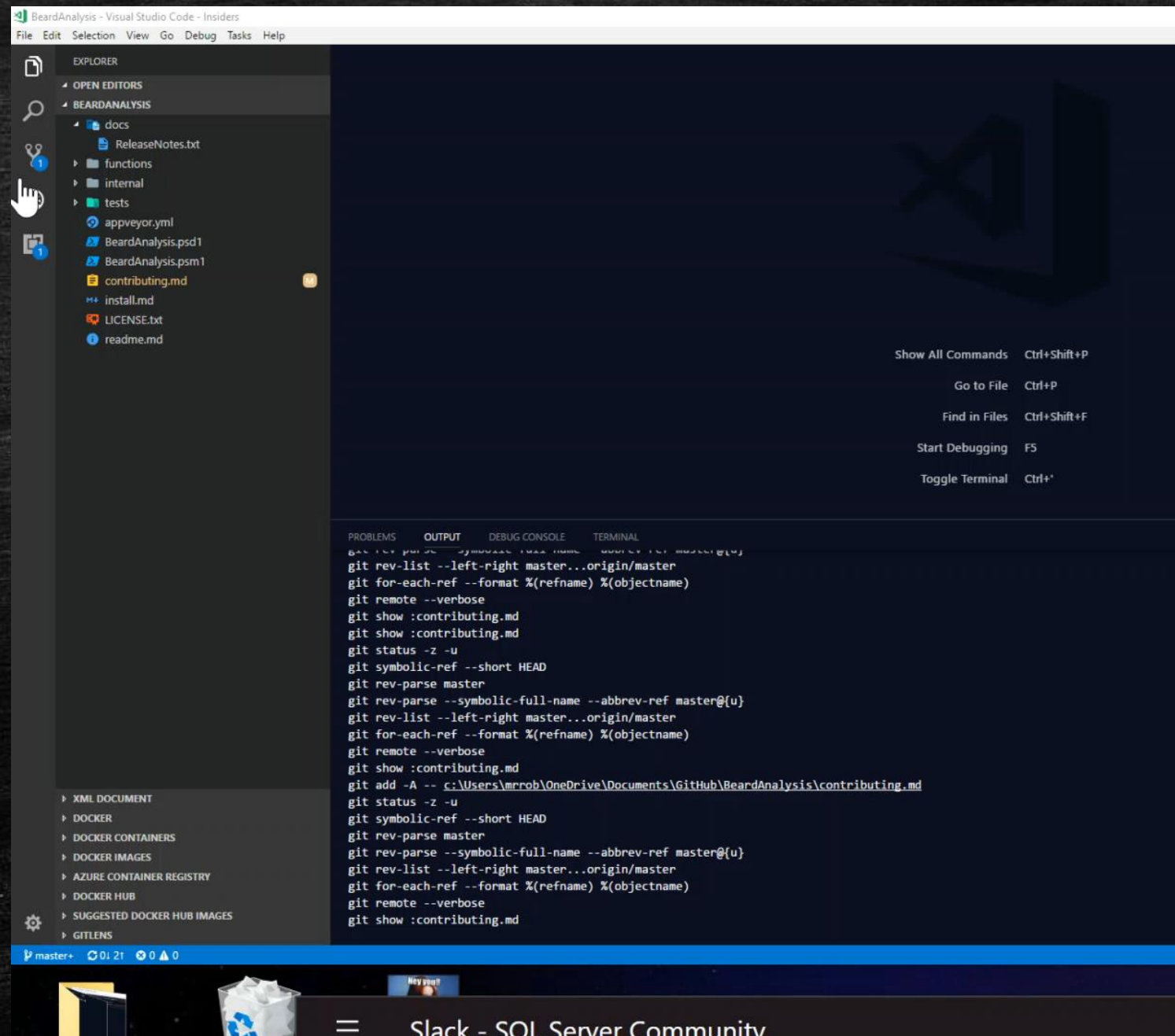
```
5 Describe "Testing - Invoke-BeardAnalysis" {
6     Context "Input" { }
7     Context "Execution" { }
8     Context "Output" {
9         BeforeAll {
10             Mock Start-Process {}
11         }
12         It "Should Return Nothing" {
13             Invoke-BeardAnalysis | Should BeNullOrEmpty
14         }
15         It "Should call Start-Process with no parameters" {
16             Assert-MockCalled -CommandName Start-Process -Times
17         }
18     }
19 }
```

```
0 references
1 function Invoke-BeardAnalysis {
2
3 }
4
```

2: PowerShell Integrated

```
PS C:\temp> Invoke-Pester .\Invoke-BeardAnalysis.Tests.ps1 -Show Fails
```


Git



VSTS Build

Builds Releases Library Task Groups Deployment Groups*

Build 394


Phase 1

- Initialize Job
- Get Sources
- Pester Test Runner
- Publish Test Results
- Copy Files to Staging Directory
- Publish Artifact: BeardAnalysis
- Post Job Cleanup

PowerShell_Gallery_Build / Build 394 / Phase 1

Edit build definition Cancel Queue new build... Download all logs as zip Release

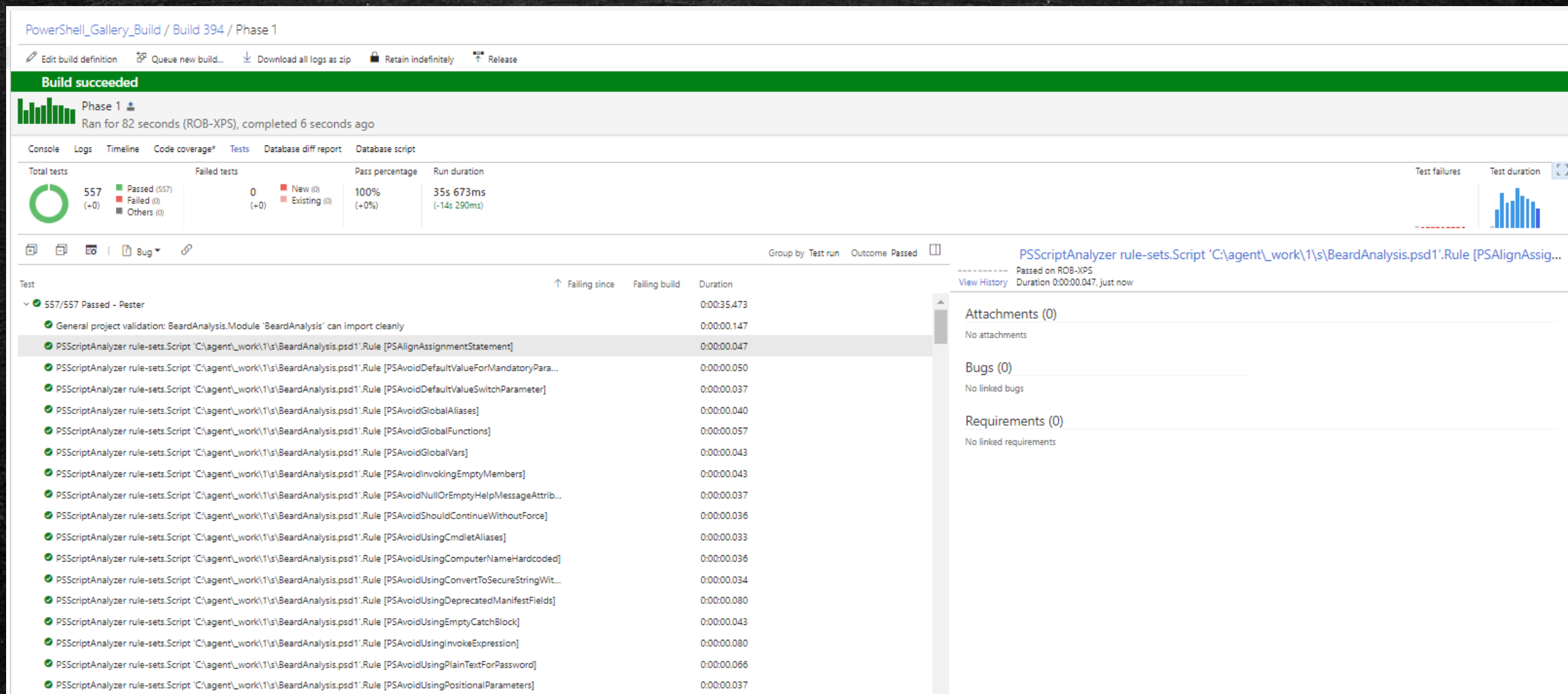
Build Started

Phase 1  Running for 60 seconds (ROB-XPS)

Console Timeline Code coverage* Tests Database diff report Database script

```
[+] Opens the image if ShowImage switch used and Detailed Switch
67ms
[+] Checks the Mock was called for Speaker Face
28ms
[+] Checks the Mock was called for Start-Process
20ms
Context Get-SpeakerBeard Output
[+] Should Return the Beard Value for a Speaker
124ms
[+] Should Return Speaker Name, Beard Value and URL if Detailed Specified
24ms
[+] Checks the Mock was called for Speaker Face
11ms
[+] Returns the Top 1 Ranked Beards
43ms
[+] Returns the Bottom 1 Ranked Beards
25ms
[+] Returns the Top 5 Ranked Beards
20ms
[+] Returns the Bottom 5 Ranked Beards
29ms
Tests completed in 35.64s
Tests Passed: 557,
Failed: 0,
Skipped: 0,
Pending: 0,
Inconclusive: 0
Finishing: Pester Test Runner
Starting: Publish Test Results
=====
Task       : Publish Test Results
Description : Publish Test Results to VSTS/TFS
Version    : 2.0.1
Author     : Microsoft Corporation
Help       : [More Information](https://go.microsoft.com/fwlink/?LinkID=613742)
=====
Async Command Start: Publish test results
Publishing test results to test run '66'
Test results remaining: 557. Test run id: 66
```


VSTS Build



VSTS Release

The screenshot displays the VSTS Release interface for a release named 'Release-88'. The left sidebar shows the 'Release Definitions' section with 'Master' selected, indicating 5/5 tasks enabled. The main area shows the 'Agent phase' with a list of tasks: 'Update Version in Manifest File', 'Commit Version Number Change', 'Update Release Notes in Manifest File', 'Commit Release Notes Change', and 'Publish Package to PowerShell Gallery'. The 'Update Version in Manifest File' task is selected, and its configuration is shown on the right. The task is of type 'Script' and is configured to update the version in the manifest file. The script content is as follows:

```
# Set location to module home path in artifacts directory
try {
    Set-Location $(System.DefaultWorkingDirectory)\PowerShell_Gallery_Build\BeardAnalysis
}
catch {
    Write-Error "Failed to set location"
}

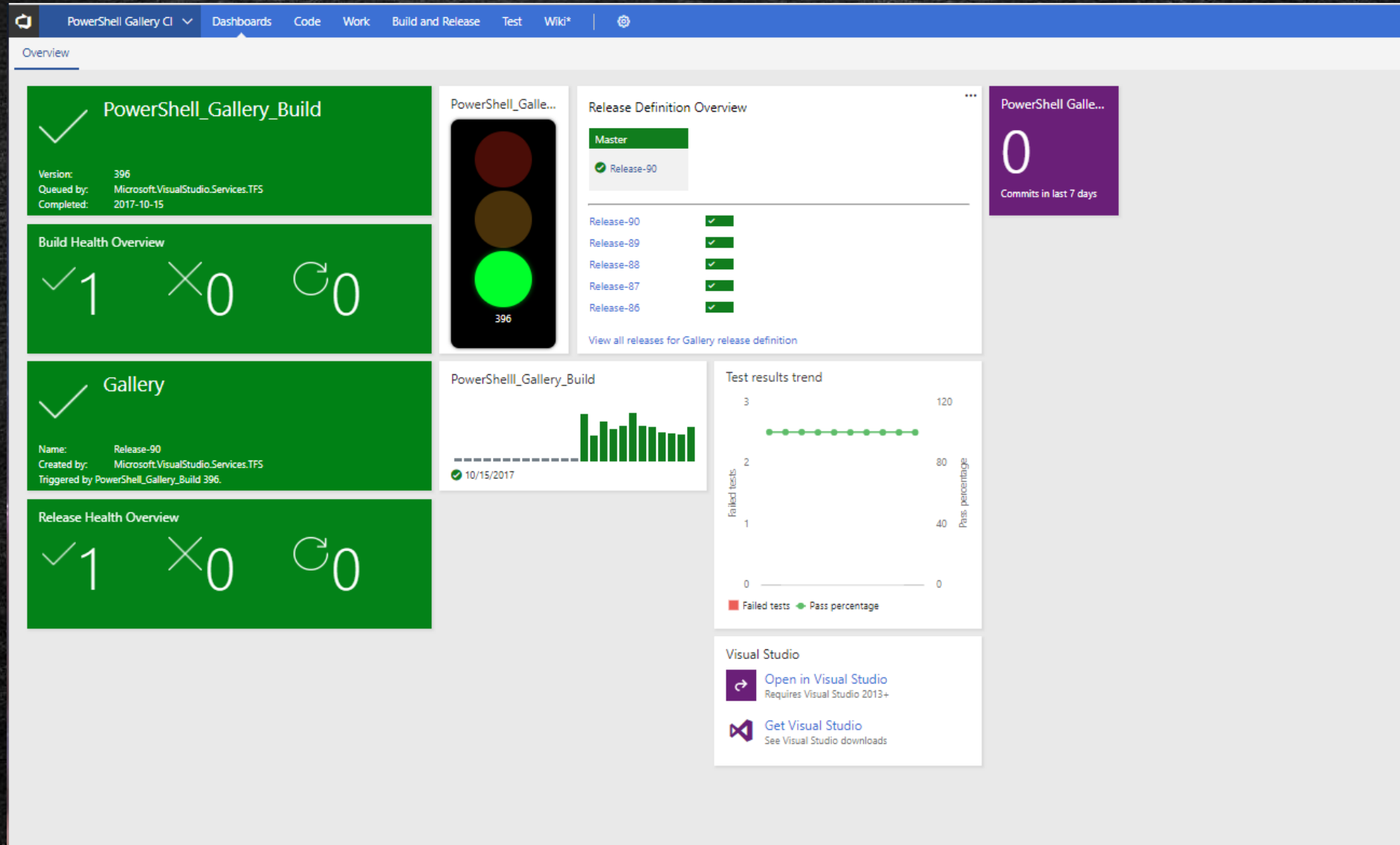
# get the contents of the module manifest file
```

The 'ErrorActionPreference' is set to 'Stop'. The 'Advanced' section shows 'Control Options' with 'Enabled' checked, 'Continue on error' unchecked, 'Always run' unchecked, and 'Timeout' set to 0. The 'Version' dropdown is set to '2* (preview)'.


VSTS Release

The screenshot displays the VSTS Release interface for a build named 'Release-88'. The left sidebar shows the 'Releases' tab with a search bar and a list of release definitions. The main area is divided into two panes. The left pane shows the build steps, with the 'Agent phase' expanded, listing tasks such as 'Initialize Job', 'Download Artifacts', 'Update Version in Manifest File', 'Commit Version Number Change', 'Update Release Notes in Manifest File', 'Commit Release Notes Change', 'Publish Package to PowerShell Gallery', and 'Post-deployment approval'. The right pane shows the 'Logs' tab for the 'Agent phase', displaying a detailed log of the build process. The log includes various environment variables and build parameters, such as [AGENT_ID], [AGENT_JOBNAME], [AGENT_MACHINE_NAME], [AGENT_NAME], [AGENT_OS], [AGENT_RELEASEDIRECTORY], [AGENT_ROOTDIRECTORY], [AGENT_SERVEROMDIRECTORY], [AGENT_TEMPDIR], [AGENT_TOOLSDIRECTORY], [AGENT_VERSION], [AGENT_WORKFOLDER], [AZURE_HTTP_USER_AGENT], [BUILD_BUILDID], [BUILD_BUILDNUMBER], [BUILD_BUILDURI], [BUILD_DEFINITIONID], [BUILD_DEFINITIONNAME], [BUILD_PROJECTID], [BUILD_PROJECTNAME], [BUILD_REPOSITORY_PROVIDER], [BUILD_REQUESTEDFOR], [BUILD_REQUESTEDFORID], [BUILD_SOURCEBRANCH], [BUILD_SOURCEBRANCHNAME], [BUILD_SOURCEVERSION], [BUILD_TYPE], [MSDEPLOY_HTTP_USER_AGENT], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_BUILDID], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_BUILDNUMBER], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_BUILDURI], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_DEFINITIONID], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_DEFINITIONNAME], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_PROJECTID], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_PROJECTNAME], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_REPOSITORY_PROVIDER], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_REQUESTEDFOR], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_REQUESTEDFORID], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_SOURCEBRANCH], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_SOURCEBRANCHNAME], [RELEASE_ARTIFACTS_POWERSHELL_GALLERY_BUILD_SOURCEVERSION], [RELEASE_ATTEMPTNUMBER], [RELEASE_DEFINITIONENVIRONMENTID], [RELEASE_DEFINITIONID], [RELEASE_DEFINITIONNAME], [RELEASE_DEPLOYMENT_REQUESTEDFOR], [RELEASE_DEPLOYMENT_REQUESTEDFOREMAIL], [RELEASE_DEPLOYMENT_REQUESTEDFORID], [RELEASE_DEPLOYMENTID], [RELEASE_DEPLOYPHASEID], [RELEASE_ENVIRONMENTID], and [RELEASE_ENVIRONMENTNAME].

VSTS DashBoard




PowerShell Gallery

 PowerShell Gallery

Register | Sign in

Home Items Publish Statistics Documentation Support

Search Items





61
Downloads


0
Downloads of 0.9.23

2017-10-15
Last published

[Contact Owners](#)
[Report Abuse](#)
[How to Download](#)
[Module Statistics](#)

 0

 0

 0

BeardAnalysis 0.9.23

This is a demo module for demoing Plaster and TDD with Pester and CI with VSTS to the PowerShell Gallery


Inspect

PS> Save-Module -Name BeardAnalysis -Path <path>

Install

PS> Install-Module -Name BeardAnalysis

Deploy

 Deploy to Azure Automation

See [Documentation](#) for more details.

Release Notes


These are the new release notes for this amazing module

There are a number of important changes in this release as this time it is

- Much Newer
- Extra amazingly More Magical
- Just So much better than the previous version - Really it is
- It has many new features and everything
- and it was demoed in 5 minutes

Was first demonstrated in Singapore at DevOps Days

Owners

 SQLDBAWithABeard

Authors

Rob Sewell

Copyright

(c) 2017 Rob Sewell. All rights reserved.

Summary

Continuous Delivery of PowerShell Module to the Gallery is easy

Plaster can template your module framework

Pester is a Unit Testing Framework for PowerShell

VSTS is free

Have Fun – Create some modules today