Tidjani Belmansour

AUTOMATED TESTING OF YOUR CLOUD INFRASTRUCTURE

ConFoo.CA
DEVELOPER CONFERENCE

HELLO, MY NAME IS …

Tidjani Belmansour

Director, Cloud Architect @ Cofomo (https://www.cofomo.com)

Microsoft Azure MVP

Co-host « Communauté Azure de Québec » (https://meetup.com/azureqc)

@Tidjani_B /in/tidjani-belmansour

https://espacenuagic.com | https://dev.to/tidjani

Microsoft® MVP Most Valuable Professional

HOW MANY OF YOU ARE…

DEVELOPERS

HOW MANY OF YOU ARE…

WRITE (CODE) UNIT TESTS

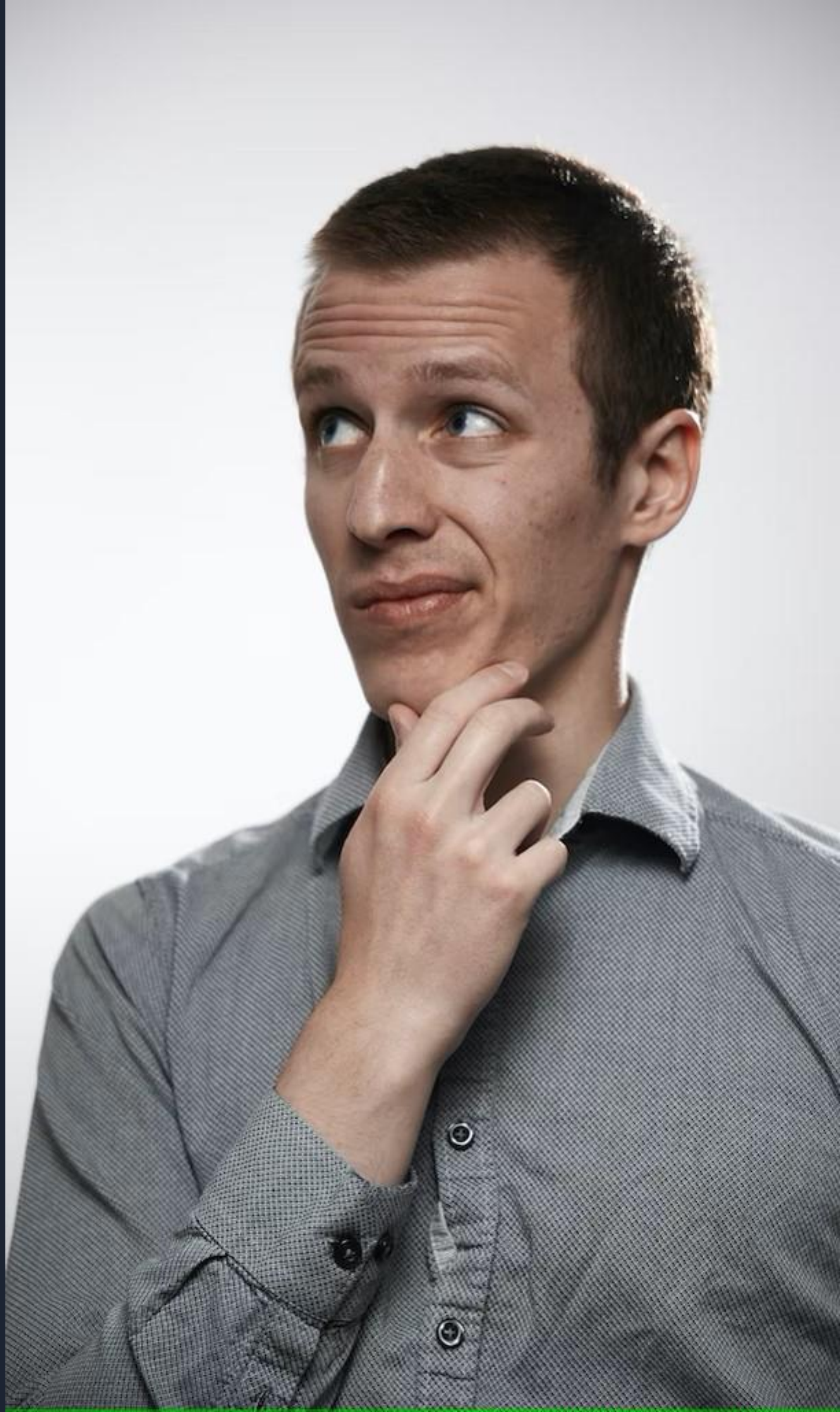
HOW MANY OF YOU ARE…

IT PROS / SYSADMINS

HOW MANY OF YOU ARE…

WRITE (INFRA) UNIT TESTS

WHY IS THAT ?!



Common answers

WAIT... WE CAN TEST OUR INFRASTRUCTURE ?!

HOW CAN WE DO THAT?

I’M NOT A DEVELOPER !

IT TAKES TIME…

WE DON’T HAVE IT !

WE HAVE INFRASTRUCTURE DIAGRAM, WE THUS KNOW OUR INFRASTRUCTURE

WHY SHOULD WE TEST IT?

Testing our infrastructure: Why ?

REASONS TO TEST OUR INFRASTRUCTURE

Cloud infrastructures are rarely finalized at day 1

They are built iteratively

They keep evolving during the whole lifecycle of the system

Mainly because of the nature of the Cloud

WITH EACH ITERATION OF THE INFRASTRUCTURE, WE WANT TO ENSURE THAT …

- We’ve built (and deployed) the right thing
- We didn’t break anything
  - i.e., no regressions

IS IT ONLY FOR CLOUD INFRASTRUCTURES?

- No !
  - Even though it has gain popularity with Cloud infrastructures
- The iterative, evolutive and ephemeral nature of the Cloud makes it crucial to test the infrastructures
- But we can also test on-premises infrastructures

Testing our infrastructure: What ?

WHAT SHALL WE TEST ?

01
Resources existence

02
Resource configuration

03
Naming conventions

04
Resources types

05
Resources SKUs

06
Instances count

07
Relationships between resources

08
Permissions

09
…

Testing our infrastructure: How ?

WHAT TOOL TO USE ?

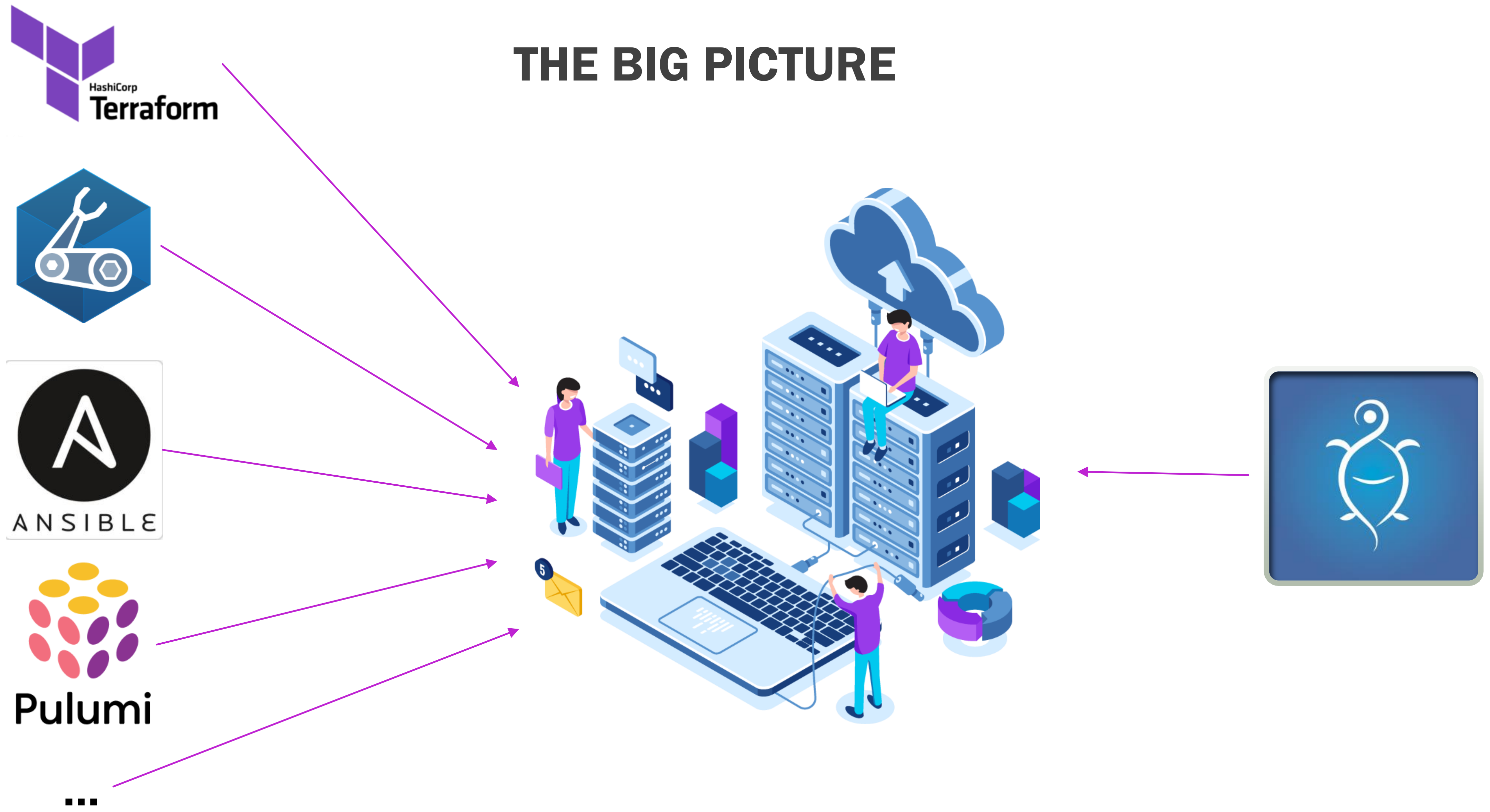
- We'll use Pester (*PowerShell Tester*)
- It's a testing Framework for (and written in) PowerShell
- It comes as a PowerShell module
- It is not specific to any given IaC tool nor any platform or Cloud

https://pester.dev

DISCLAIMER

- In this session, we’ll test an Azure infrastructure for convenience purposes
- However, the concepts presented here apply to other IT infrastructures

THE BIG PICTURE



USING PESTER

https://pester.dev

4 steps !

1. Import Pester module
2. Write tests
3. Define test parameters files (PowerShell Data File)
4. Run tests

STEP 0 - INSTALL PESTER MODULE

```
Install-Module Pester –Force -SkipPublisherCheck
```

Only once for each machine / server

STEP 1 - IMPORT PESTER MODULE

```
Import-Module Pester –Passthru
```

STEP 2 – WRITE TESTS | HARD-CODED PARAMETERS

Test code file: confoo.tests.ps1

```
Param(
    [string] [Parameter(Mandatory=$true)] $ResourceGroupName
)

# Get the resource we want to test (deployed to Azure)
$rg = Get-AzResourceGroup -ResourceGroupName $ResourceGroupName


# Define a test
Describe 'Validating the resource group' {
  It "ResourceGroupName is correct" {
    $rg.ResourceGroupName | Should –Be "confoo-rg"
  }
  It "RG location is correct" {
    $rg.Location | Should -Be "canadacentral"
  }
}
```

STEP 3 – RUN TESTS | HARD-CODED PARAMETERS

Pester v5.x

```
$container = New-PesterContainer `
-Path '.\Tests\confoo.tests.ps1' `
-Data @{ `
    ResourceGroupName = 'confoo-rg'; `
    ServiceBusName = 'confoo-pester'; `
    SkipConnection = 'Y'; `
}

$config = New-PesterConfiguration
$config.Run.Container = $container

Invoke-Pester -Configuration $config
```

WHAT IF WE HAVE MULTIPLE ENVIRONMENTS ?

- We obviously don't want to create one test file per environment…
  - Because we’d duplicate test code
- It would be nice to have test parameters files for each environment
  - We’d then have a single test code file but several test parameters files
- This is the appropriate approach to adopt !

STEP 2 – WRITE TESTS | USING PARAMETERS FILES

Test code file: confoo.tests.ps1

```
Param(
    [string] [Parameter(Mandatory=$true)] $ResourceGroupName,
    [string] [Parameter(Mandatory=$true)] $DataFilePath
)

# Load the test parameters files into the « expected » object:
$Expected = Import-PowerShellDataFile -Path $DataFilePath

# Get the resource we want to test (deployed to Azure)
$rg = Get-AzResourceGroup -ResourceGroupName $ResourceGroupName

# Define a test
Describe 'Validating the resource group' {
  It "ResourceGroupName is correct" {
    $rg.ResourceGroupName | Should -Be $Expected.ResourceGroup.Name
  }
  It "RG location is correct" {
    $rg.Location | Should -Be $Expected.ResourceGroup.Location
  }
}
```

STEP 2 – WRITE TESTS | USING PARAMETERS FILES

Test parameters file: confoo.prod.data.psd1

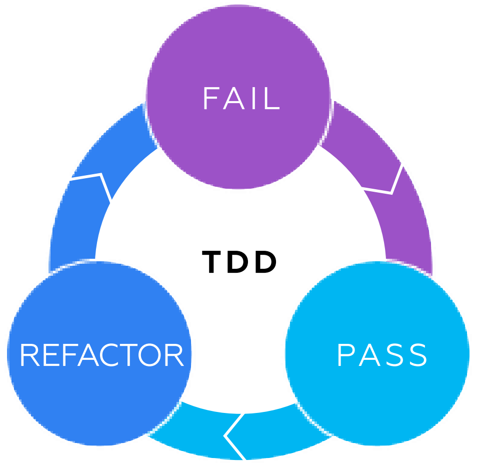
```
@{

   ResourceGroup = @{
        Name = 'confoo-rg'
        Location = 'canadacentral'
    }
}
```

STEP 3 – RUN TESTS | USING PARAMETERS FILES

Pester v5.x

```
$container = New-PesterContainer `
-Path '.\Tests\confoo.tests.ps1' `
-Data @{ `
    ResourceGroupName = 'confoo-rg'; `
    ServiceBusName = 'confoo-pester'; `
    DataFilePath = './Tests/Data/confoo.prod.data.psd1'; `
    SkipConnection = 'Y'; `
}

$config = New-PesterConfiguration
$config.Run.Container = $container

Invoke-Pester -Configuration $config
```

TDD



It is the recommended approach !

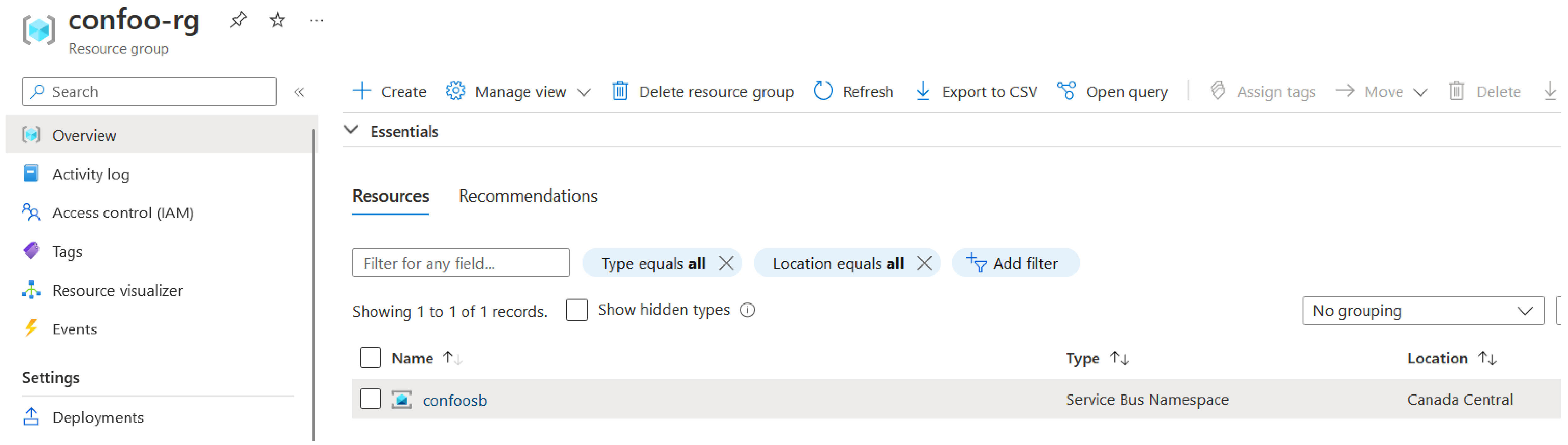
1. We start by writing the tests representing the expected infrastructure
2. We run the tests. They should fail
3. We write the IaC code that will deploy the expected infrastructure
4. We rerun the tests. They should succeed
5. We refactor the IaC code if needed

A MORE COMPLETE EXAMPLE ...

- Existence / Inexistence of a resource
- Tags
- Naming convention
- Parameter values (SKU, resource name, location)
- Service Bus: check that the expected queues have been created (count and names)
- Feature activation (e.g., RequiresDuplicateDetection)
- Resource status (e.g., Active)

https://github.com/BelRarr/confoo2023-infratesting

WHAT IT LOOKS LIKE ...

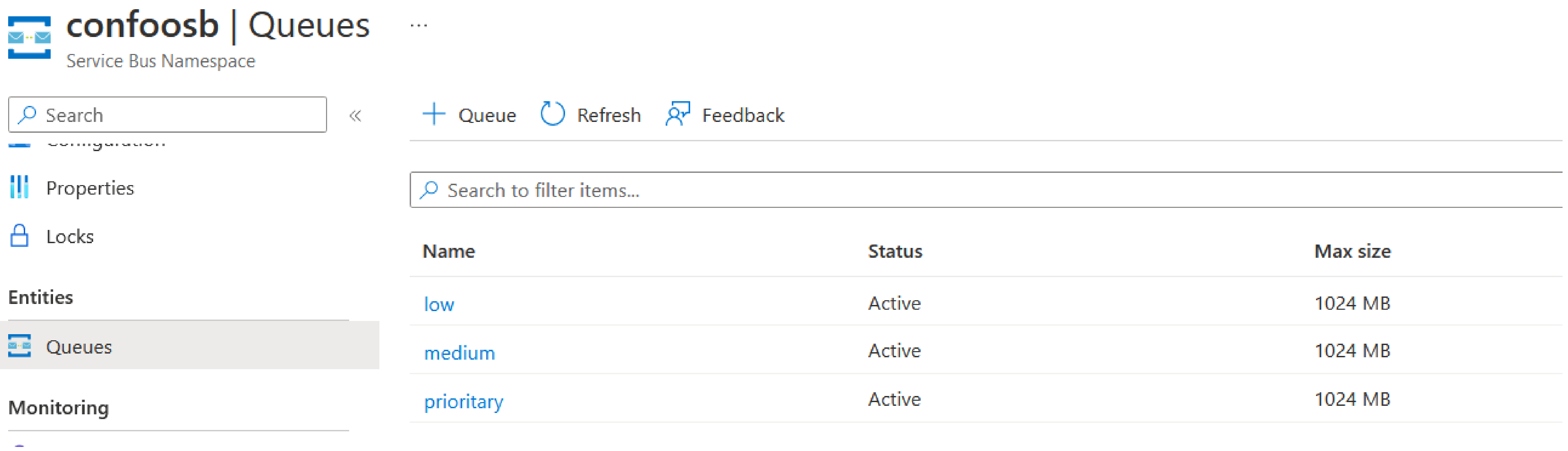

confoo-rg

Resource group

Showing 1 to 1 of 1 records.

Name	Type	Location
confoosb	Service Bus Namespace	Canada Central

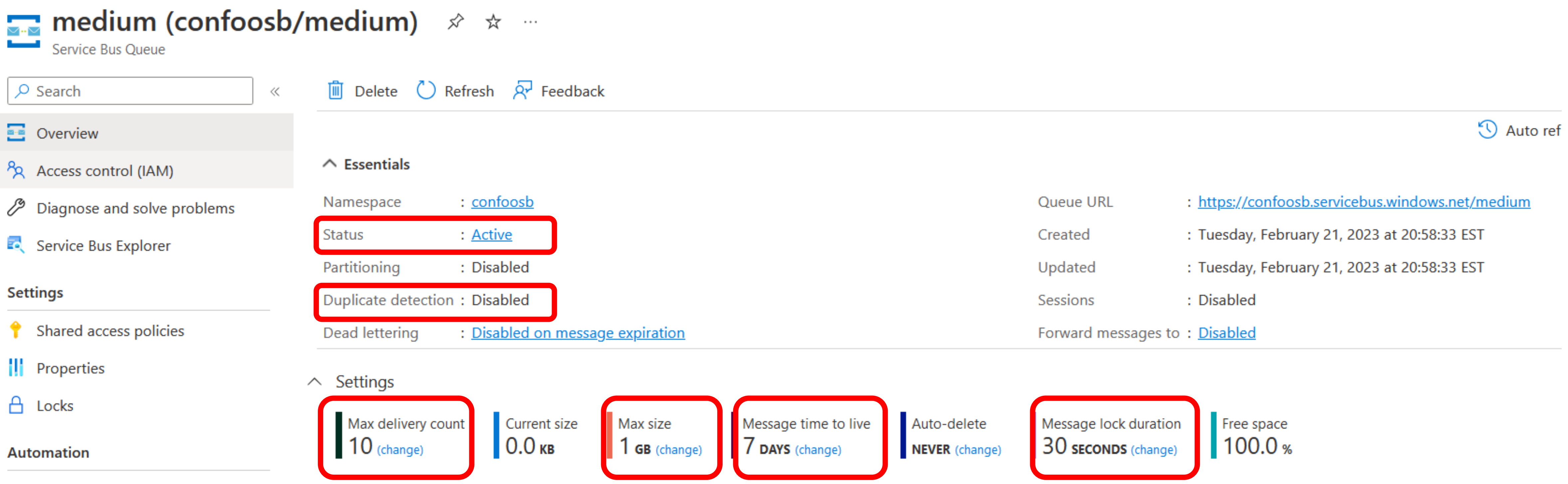
WHAT IT LOOKS LIKE …


confoosb | Queues ...

Service Bus Namespace

Name	Status	Max size
low	Active	1024 MB
medium	Active	1024 MB
prioritary	Active	1024 MB

WHAT IT LOOKS LIKE …



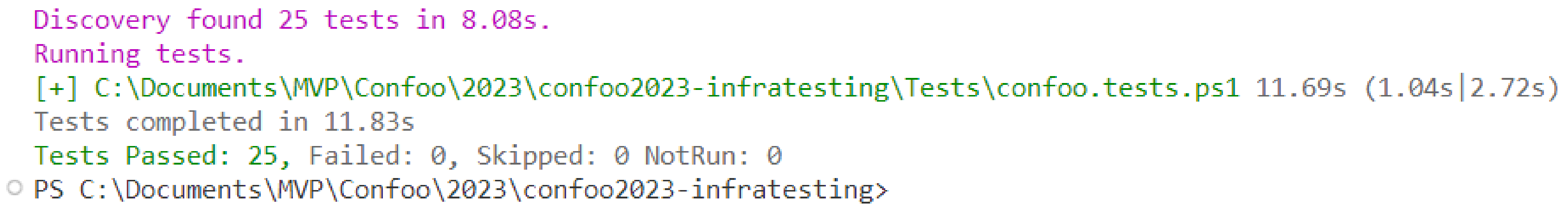
Essentials

Namespace	: confoosb	Queue URL	: https://confoosb.servicebus.windows.net/medium
Status	: Active	Created	: Tuesday, February 21, 2023 at 20:58:33 EST
Partitioning	: Disabled	Updated	: Tuesday, February 21, 2023 at 20:58:33 EST
Duplicate detection	: Disabled	Sessions	: Disabled
Dead lettering	: Disabled on message expiration	Forward messages to	: Disabled

Settings

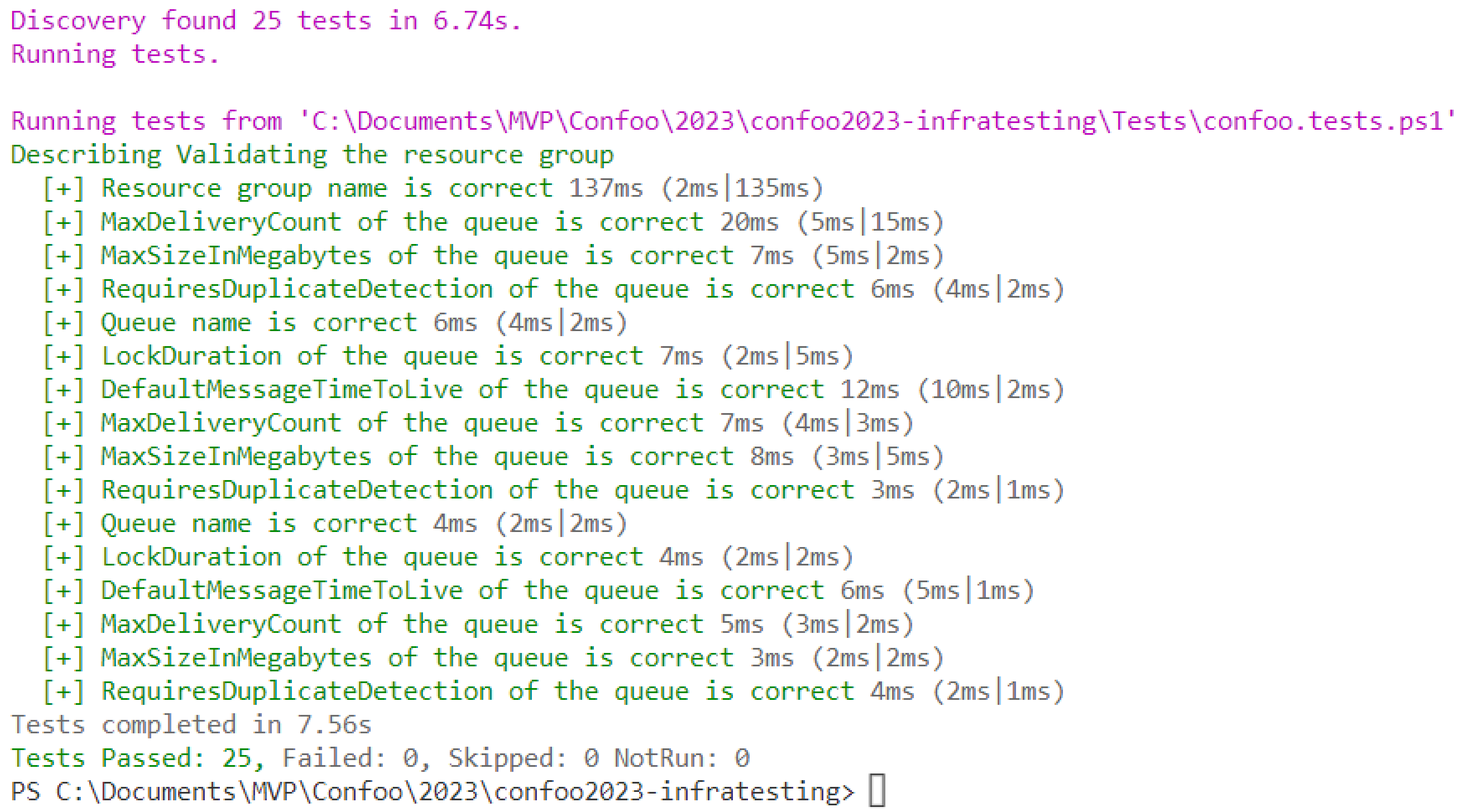
Max delivery count	Current size	Max size	Message time to live	Auto-delete	Message lock duration	Free space
10 (change)	0.0 KB	1 GB (change)	7 DAYS (change)	NEVER (change)	30 SECONDS (change)	100.0 %

NORMAL VERBOSITY



```
Discovery found 25 tests in 8.08s.
Running tests.
[+] C:\Documents\MVP\Confoo\2023\confoo2023-infratesting\Tests\confoo.tests.ps1 11.69s (1.04s|2.72s)
Tests completed in 11.83s
Tests Passed: 25, Failed: 0, Skipped: 0 NotRun: 0
PS C:\Documents\MVP\Confoo\2023\confoo2023-infratesting>
```

DETAILED VERBOSITY



```
Discovery found 25 tests in 6.74s.
Running tests.

Running tests from 'C:\Documents\MVP\Confoo\2023\confoo2023-infratesting\Tests\confoo.tests.ps1'
Describing Validating the resource group
  [+] Resource group name is correct 137ms (2ms|135ms)
  [+] MaxDeliveryCount of the queue is correct 20ms (5ms|15ms)
  [+] MaxSizeInMegabytes of the queue is correct 7ms (5ms|2ms)
  [+] RequiresDuplicateDetection of the queue is correct 6ms (4ms|2ms)
  [+] Queue name is correct 6ms (4ms|2ms)
  [+] LockDuration of the queue is correct 7ms (2ms|5ms)
  [+] DefaultMessageTimeToLive of the queue is correct 12ms (10ms|2ms)
  [+] MaxDeliveryCount of the queue is correct 7ms (4ms|3ms)
  [+] MaxSizeInMegabytes of the queue is correct 8ms (3ms|5ms)
  [+] RequiresDuplicateDetection of the queue is correct 3ms (2ms|1ms)
  [+] Queue name is correct 4ms (2ms|2ms)
  [+] LockDuration of the queue is correct 4ms (2ms|2ms)
  [+] DefaultMessageTimeToLive of the queue is correct 6ms (5ms|1ms)
  [+] MaxDeliveryCount of the queue is correct 5ms (3ms|2ms)
  [+] MaxSizeInMegabytes of the queue is correct 3ms (2ms|2ms)
  [+] RequiresDuplicateDetection of the queue is correct 4ms (2ms|1ms)
Tests completed in 7.56s
Tests Passed: 25, Failed: 0, Skipped: 0 NotRun: 0
PS C:\Documents\MVP\Confoo\2023\confoo2023-infratesting>
```

GENERATING A TESTS REPORT

```powershell
$config = New-PesterConfiguration
$config.Output.Verbosity = "Detailed"
$config.Run.Container = $container

###################################################
$config.TestResult.Enabled = $true
$config.TestResult.OutputFormat = "NUnitXml"
$config.TestResult.OutputPath = "results.xml"
###################################################

Invoke-Pester -Configuration $config
```

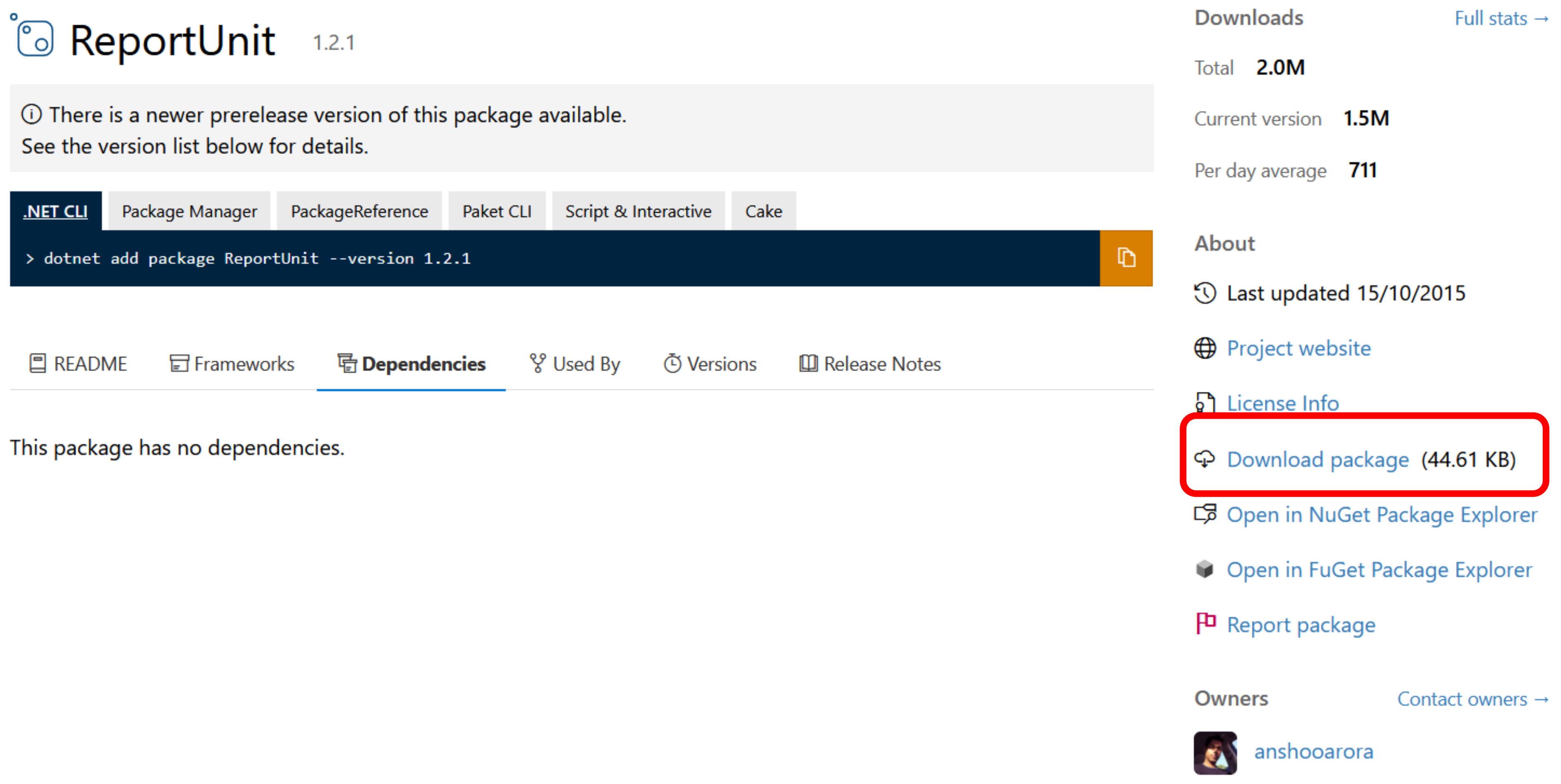
GENERATING A TESTS REPORT

We then need « ReportUnit »

1. Get it from nuget.org (https://www.nuget.org/packages/ReportUnit)
2. Rename « .nupkg » to « .zip » and extract it
3. Execute « ReportUnit.exe » (from the « tools » folder)

```
ReportUnit.exe  <folder containing the test results> <folder where to generate the report>
```

GENERATING A TESTS REPORT



ReportUnit 1.2.1

There is a newer prerelease version of this package available.
See the version list below for details.

```
> dotnet add package ReportUnit --version 1.2.1
```

This package has no dependencies.

Downloads

Full stats →

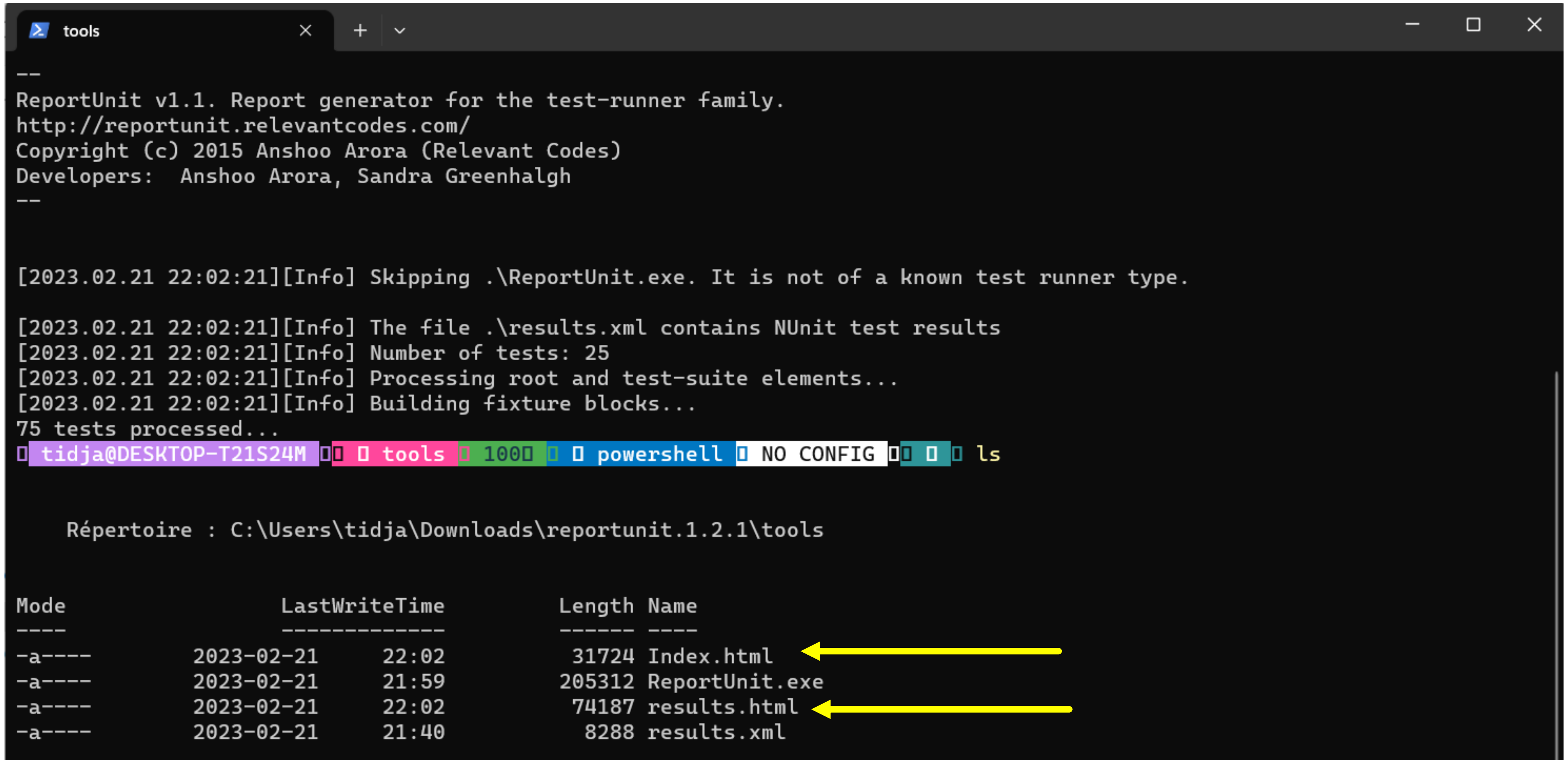
Total	2.0M
Current version	1.5M
Per day average	711

About

Last updated 15/10/2015

Project website
License Info
Download package (44.61 KB)
Open in NuGet Package Explorer
Open in FuGet Package Explorer
Report package
Owners
Contact owners →
anshooarora

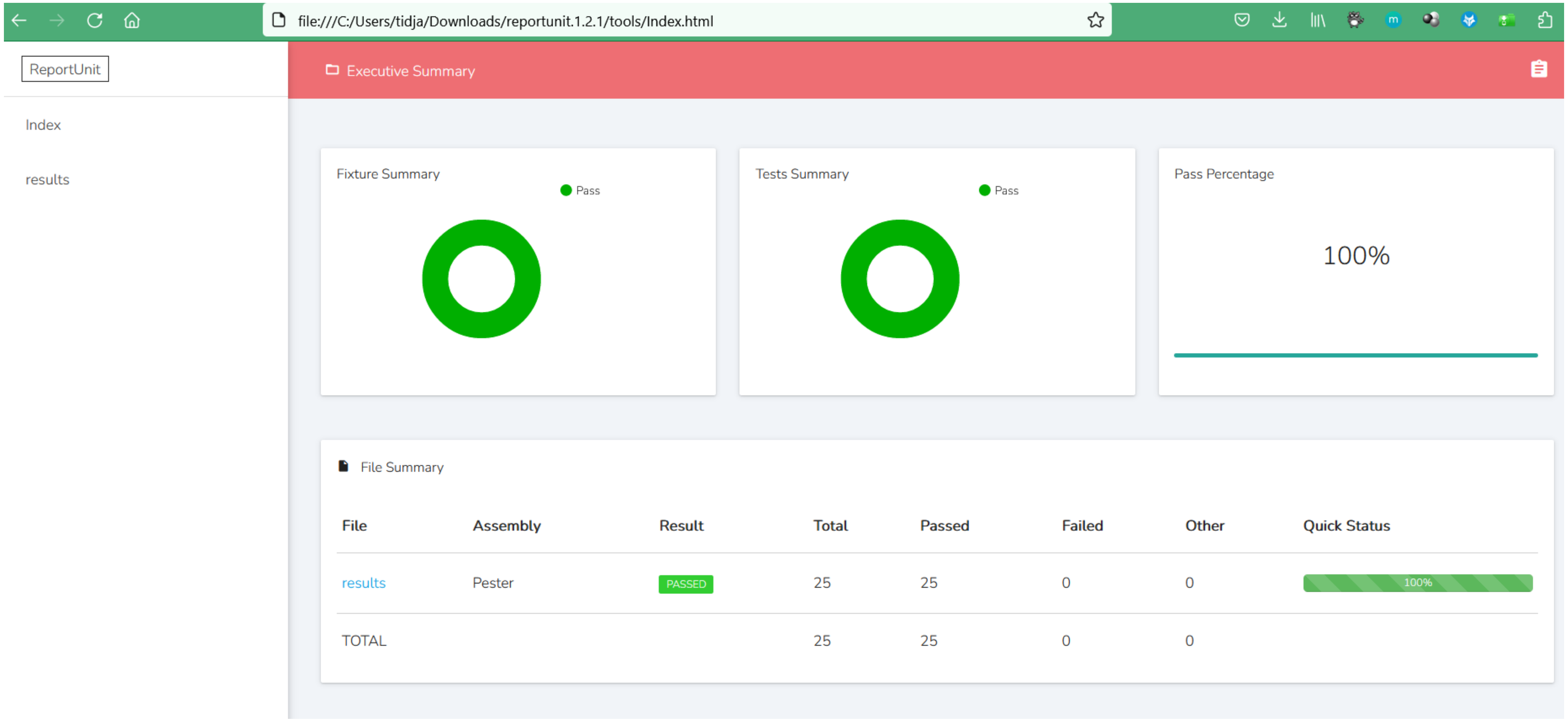
GENERATING A TESTS REPORT



tools

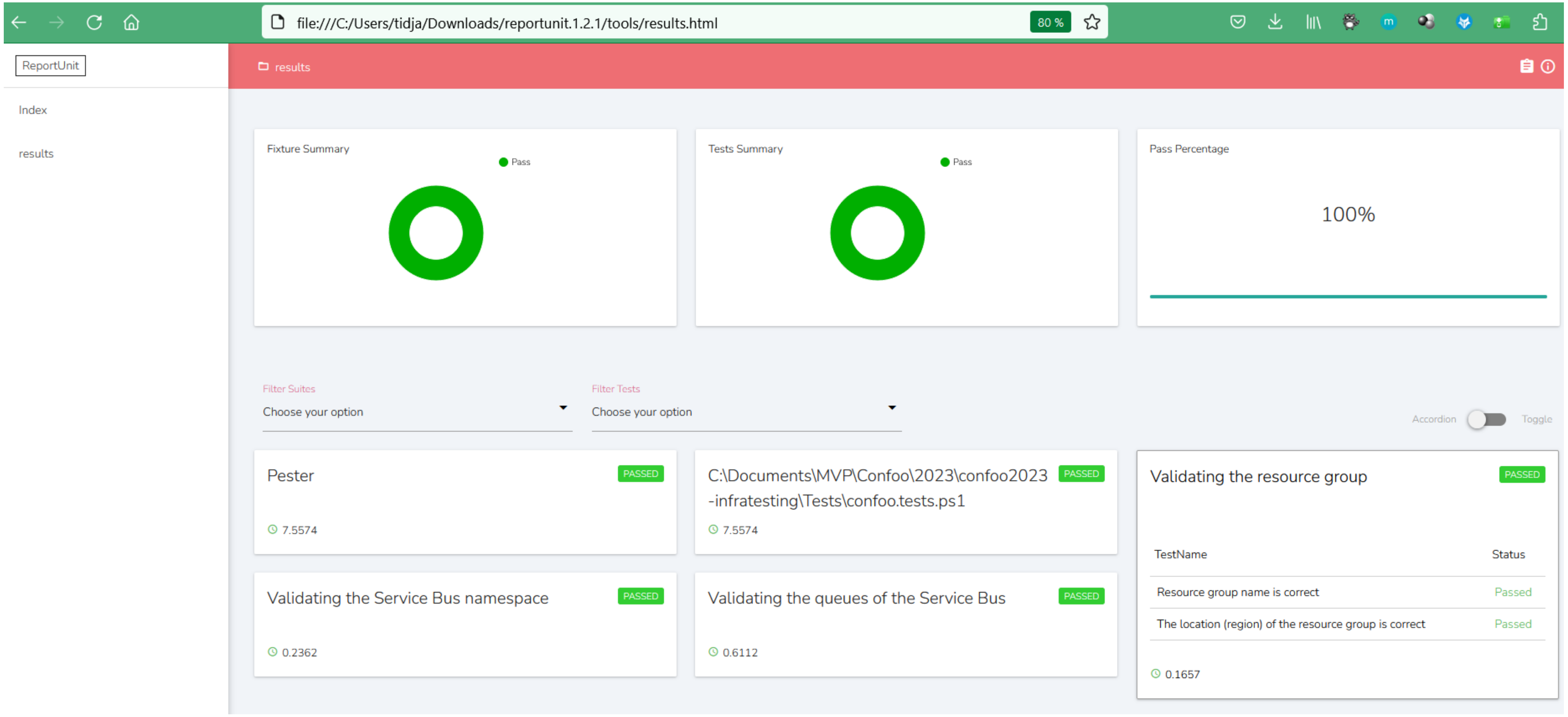
```
--
ReportUnit v1.1. Report generator for the test-runner family.
http://reportunit.relevantcodes.com/
Copyright (c) 2015 Anshoo Arora (Relevant Codes)
Developers:  Anshoo Arora, Sandra Greenhalgh
--


[2023.02.21 22:02:21][Info] Skipping .\ReportUnit.exe. It is not of a known test runner type.

[2023.02.21 22:02:21][Info] The file .\results.xml contains NUnit test results
[2023.02.21 22:02:21][Info] Number of tests: 25
[2023.02.21 22:02:21][Info] Processing root and test-suite elements...
[2023.02.21 22:02:21][Info] Building fixture blocks...
75 tests processed...
 tidja@DESKTOP-T21S24M   tools  100   powershell  NO CONFIG     ls


    Répertoire : C:\Users\tidja\Downloads\reportunit.1.2.1\tools


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        2023-02-21     22:02          31724 Index.html
-a----        2023-02-21     21:59         205312 ReportUnit.exe
-a----        2023-02-21     22:02          74187 results.html
-a----        2023-02-21     21:40           8288 results.xml
```

GENERATING A TESTS REPORT | INDEX.HTML



File	Assembly	Result	Total	Passed	Failed	Other	Quick Status
results	Pester	PASSED	25	25	0	0	100%
TOTAL			25	25	0	0	

GENERATING A TESTS REPORT | RESULTS.HTML



file:///C:/Users/tidja/Downloads/reportunit.1.2.1/tools/results.html

ReportUnit

Index

results

Fixture Summary

● Pass

Tests Summary

● Pass

Pass Percentage

100%

Filter Suites
Choose your option

Filter Tests
Choose your option

Accordion Toggle

Pester PASSED
7.5574

C:\Documents\MVP\Confoo\2023\confoo2023-infratesting\Tests\confoo.tests.ps1 PASSED
7.5574

Validating the Service Bus namespace PASSED
0.2362

Validating the queues of the Service Bus PASSED
0.6112

Validating the resource group PASSED

TestName	Status
Resource group name is correct	Passed
The location (region) of the resource group is correct	Passed

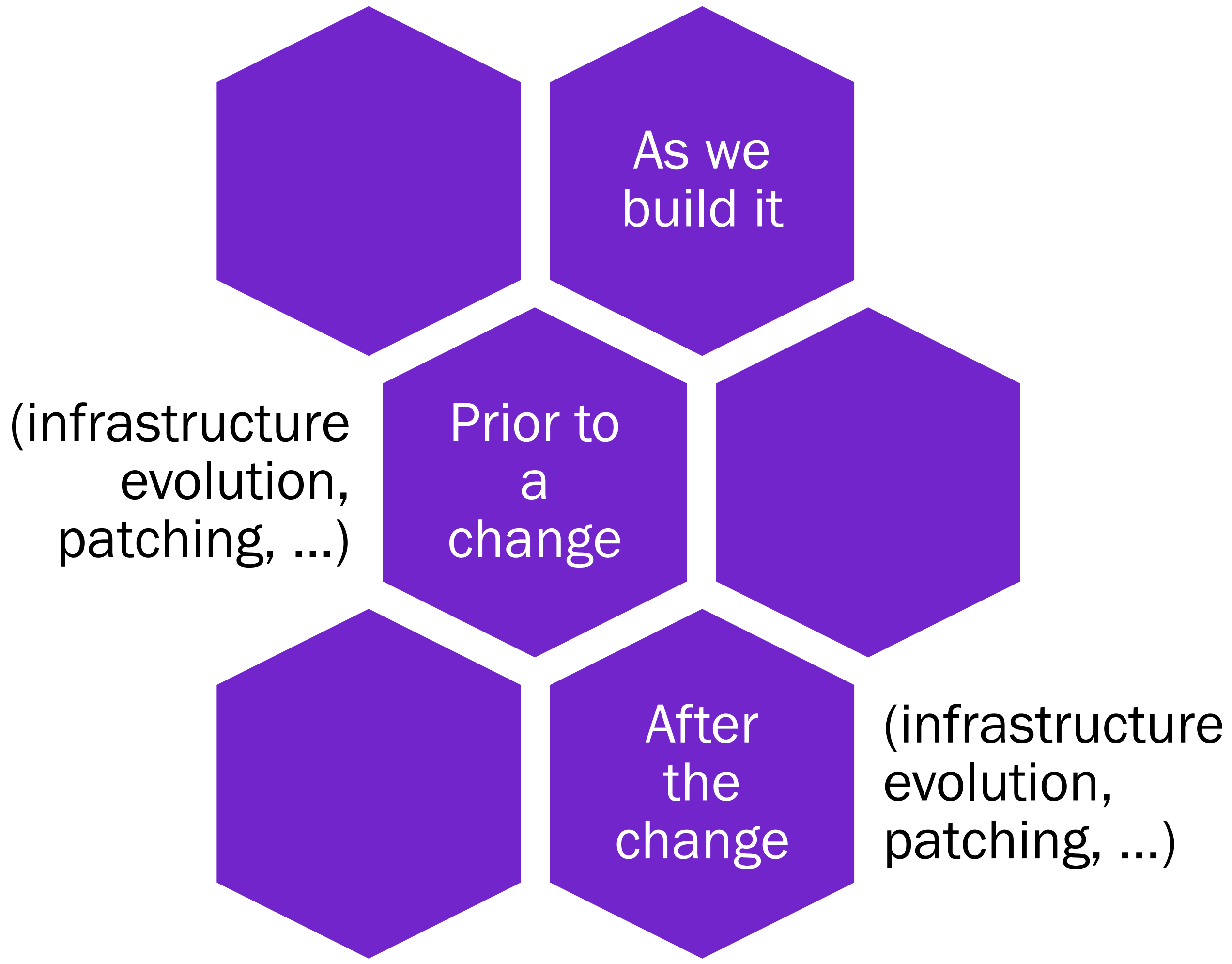
0.1657

Where can I find the code for this session?

https://github.com/BelRarr/confoo2023-infratesting

Testing our infrastructure: When ?

WHEN TO TEST OUR INFRASTRUCTURES ?


As we build it
(infrastructure evolution, patching, …)
Prior to a change
After the change
(infrastructure evolution, patching, …)

MANUALLY EXECUTING THE TESTS

- We can, of course, execute the tests manually but it won’t be the best approach …
- Do this when writing / updating your infrastructure code

```
$container = New-PesterContainer `
-Path './Tests/confoo.tests.ps1' `
-Data @{ `
    ResourceGroupName = 'confoo-rg'; `
    ServiceBusName = 'confoo-pester'; `
    DataFilePath = './Tests/Data/confoo.prod.data.psd1'; `
    SkipConnection = 'Y'; `
}

$config = New-PesterConfiguration
$config.Run.Container = $container

Invoke-Pester -Configuration $config
```

AUTOMATICALLY EXECUTING THE TESTS

- When in release mode, we should use the automated approach
- We run our infrastructure tests in the CI/CD pipeline
  - Just like we would do with application tests
- We therefore add this step in the YAML code of the CI/CD pipeline

AUTOMATICALLY EXECUTING THE TESTS

```yaml
# /pipelines/azure-pipelines.yml
pool:
  vmImage: windows-2019

steps:
  - task: PowerShell@2
    displayName: "Run Pester tests"
    inputs:
      targetType: "inline"
      script: |
        Set-Location ./tests
        Invoke-Pester -CI
      ignoreLASTEXITCODE: true

  - task: PublishTestResults@2
    inputs:
      testResultsFormat: "NUnit"
      testResultsFiles: "**/Test*.xml"
      failTaskOnFailedTests: true
      testRunTitle: "Validate Task Files"
```

Is that all about Pester ?

GOING FURTHER

https://pester.dev

Other features include:

- BeforeAll
- AfterAll
- BeforeEach
- AfterEach
- Mock
- …

MVP Microsoft® Most Valuable Professional

@tidjani_b

/in/tidjani-belmansour

ConFoo.CA
DEVELOPER CONFERENCE