

Boosting Test Speed with Parallel Testing and Targeted Automation



André Koene



Me

André Koene

sogeti newspark
Part of Capgemini

a BlueGroup IT company

FYNCH
EVERY TRIP COUNTS

Rabobank

roxit

vereniging
eigen huis | sta
sterker

 **kpn**

VEON



TS

JS



A





Me

André Koene



TRIMs

Navigation



What

Why

How

- T argeted
- R eliable
- I nformative
- M aintainable
- S peedy



What

“Parallel testing is a method used to run multiple **automated tests** at the same time, rather than one after another (sequentially).”



Order process

Endpoints related to the order process

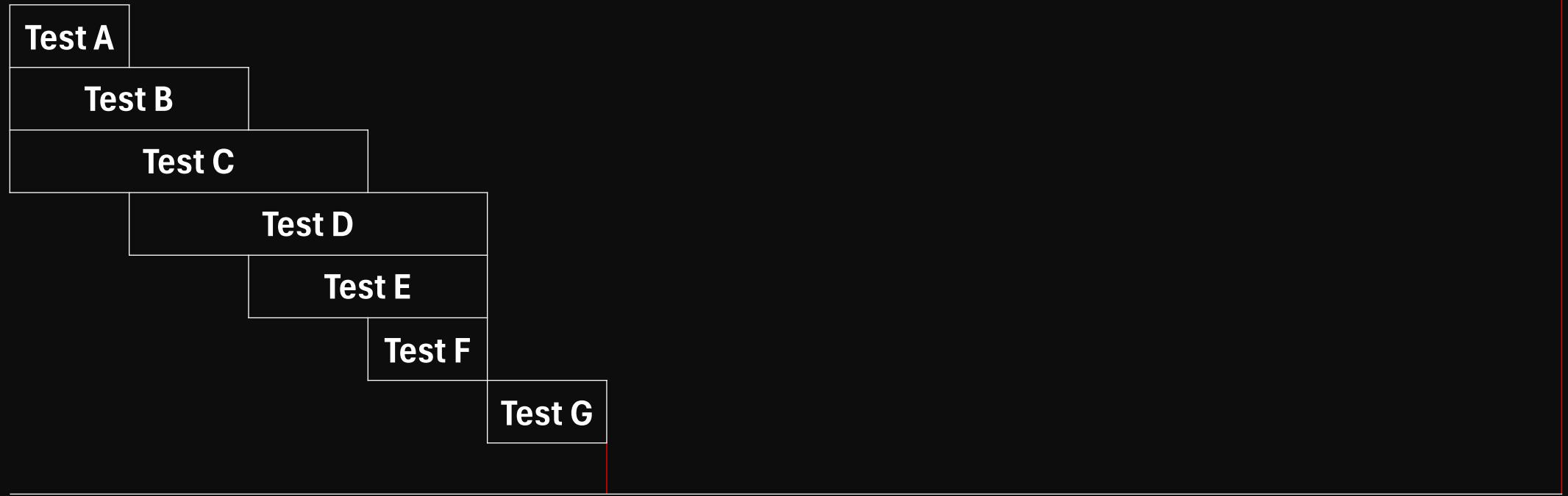
GET	/cart/{customerId}	Get the shopping cart contents
POST	/cart/{customerId}/addItem	Add an item to the shopping cart
POST	/cart/{customerId}/placeOrder	Place an order

Customer data

Endpoints related to the customer data

GET	/orders/{customerId}/history	Get customer order history
-----	------------------------------	----------------------------

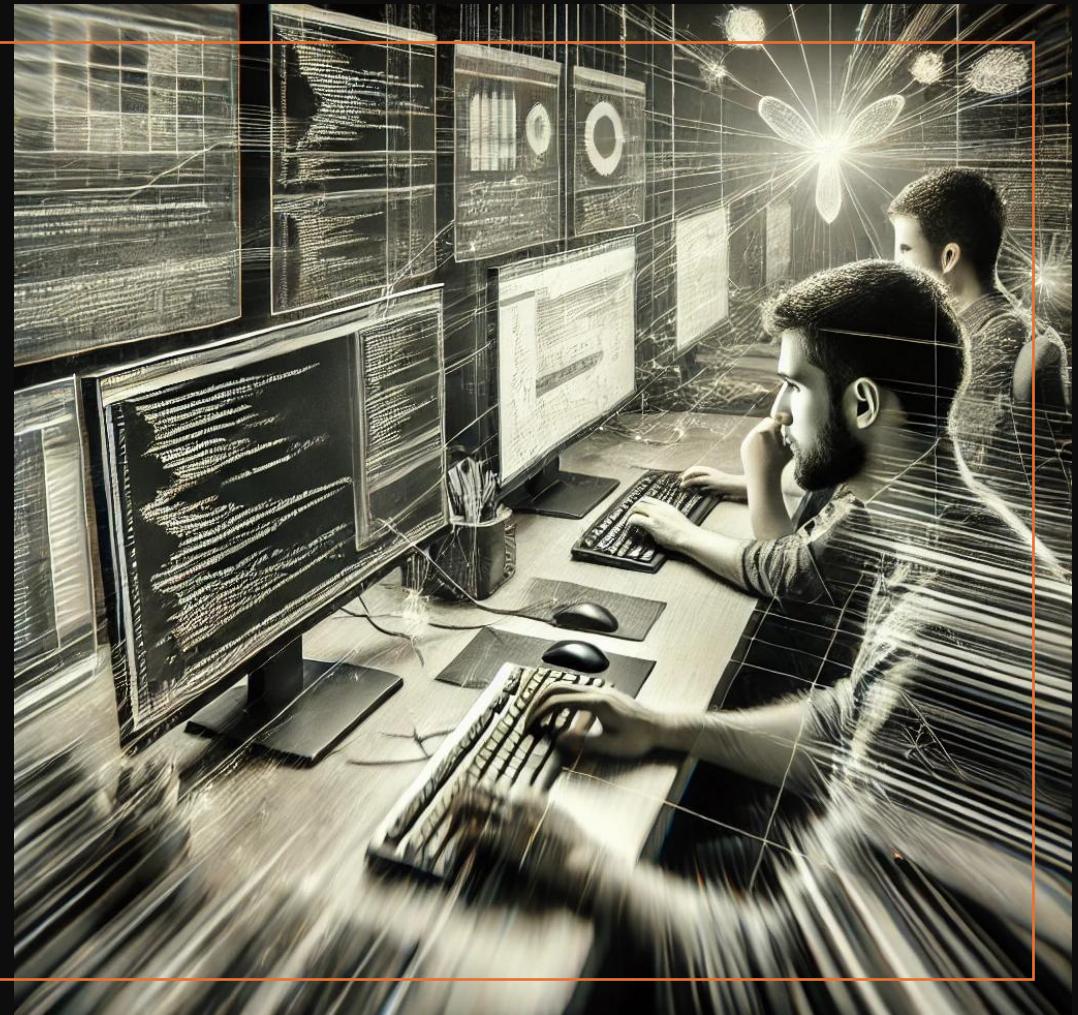
Test A	Test B	Test C	Test D	Test E	Test F	Test G
--------	--------	--------	--------	--------	--------	--------

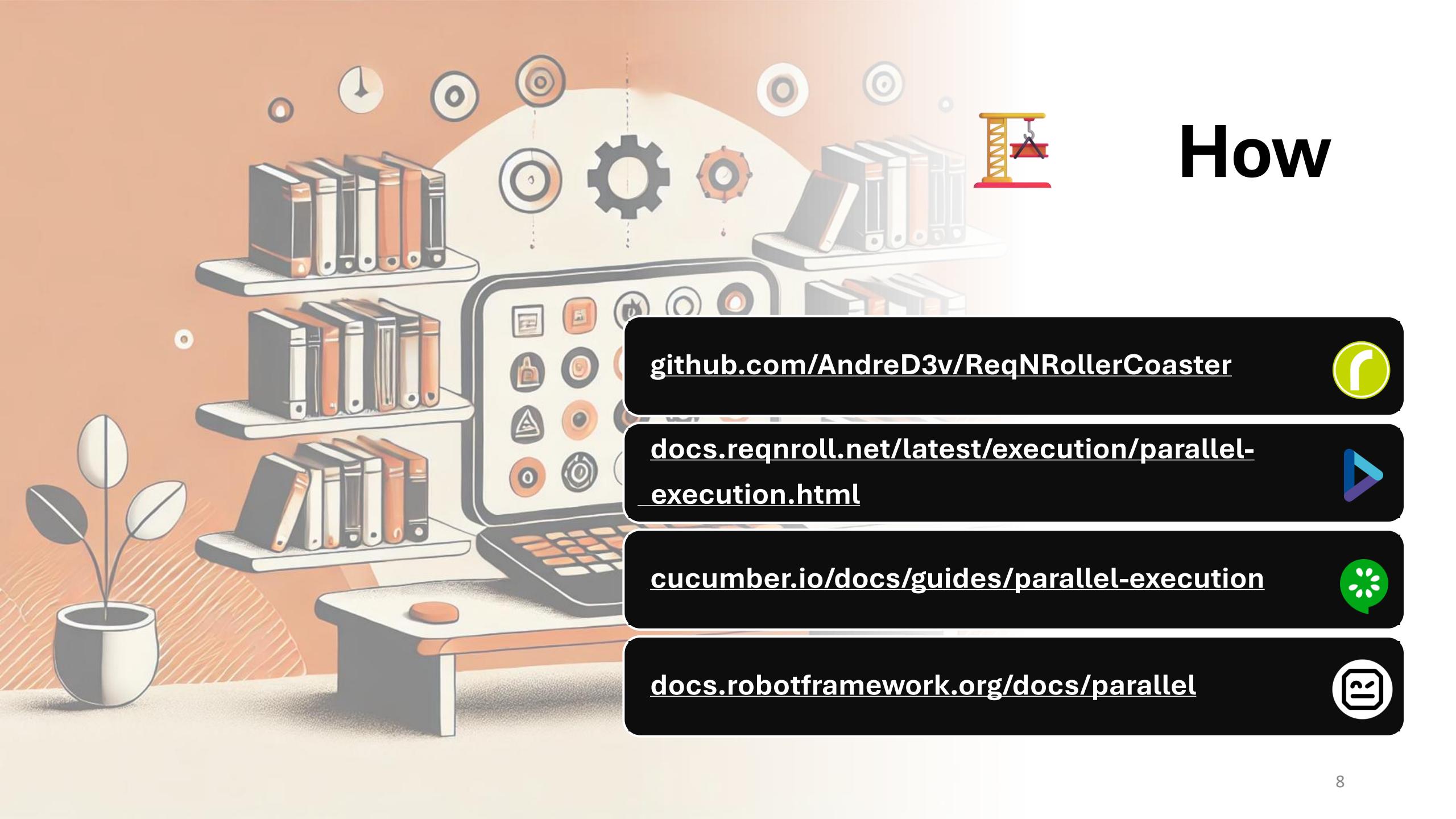


? Why

Context switches are killing

“Fast feedback from tests enables quicker bug fixes and immediate retesting while the details are still fresh in mind.”





How

github.com/AndreD3v/ReqNRollerCoaster



docs.reqnroll.net/latest/execution/parallel-execution.html



cucumber.io/docs/guides/parallel-execution



docs.robotframework.org/docs/parallel





TRIMS

“TRIMS is a mnemonic for creating valuable automation that truly supports your testing.”

Richard Bradshaw





TRIMS

“Every principle plays a crucial role in accelerating test feedback”



T argeted



R eliable



I nformative



M aintainable



S peedy

$$\text{Risk} = \text{Probability} \times \text{Severity}$$

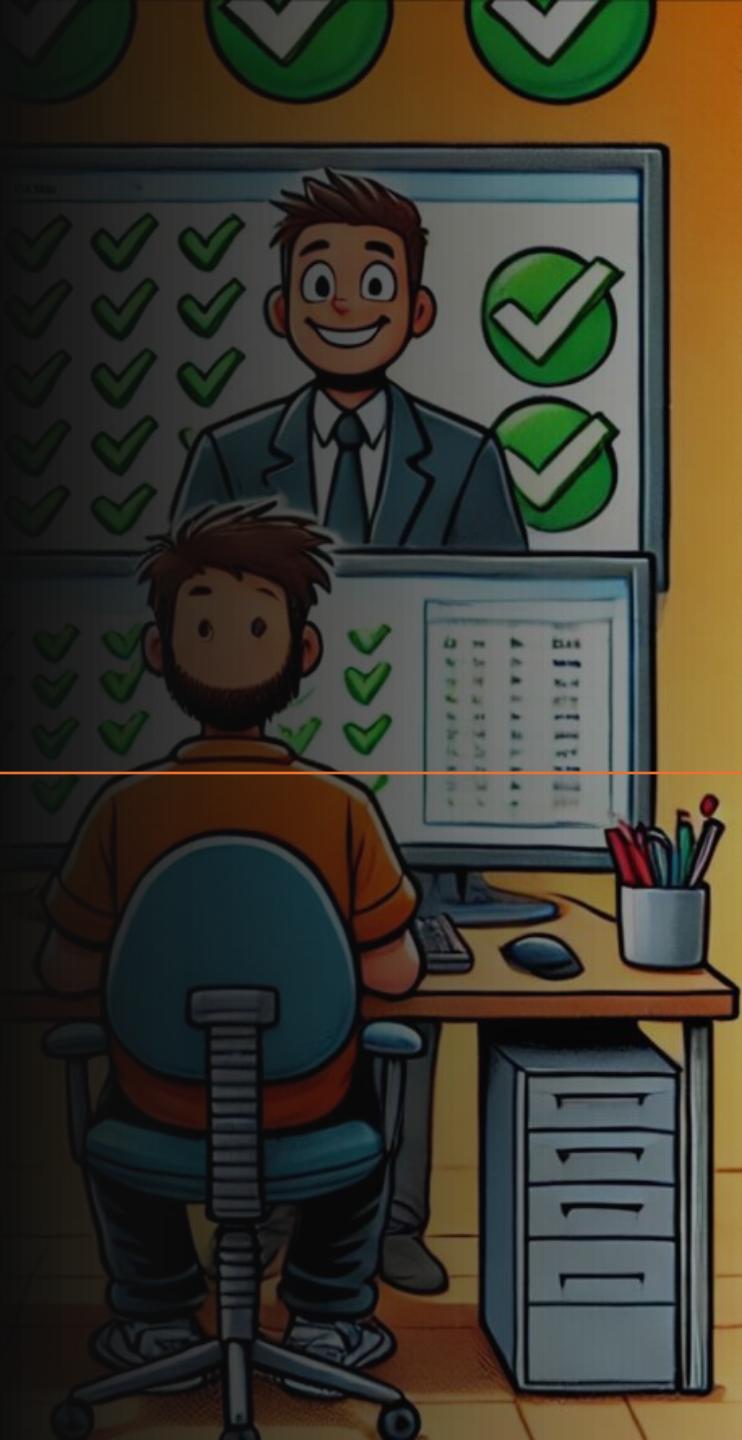
Targeted

Scenario: combine Scenarios A and B
Given I have a default customer
When I do something
Then I should see 'XXX'
When I do something else
Then I should see 'YYY'





Reliable



```
4
5   ✓ Scenario: Test A
6     Given I have a default customer
7     When I do something
8     Then I should see a customer with property 'XXX'
9
10  ✓ Scenario: Test B
11    Given I have a default customer
12    When I do something else
13    Then I should see a customer with property 'YYY'
```

log_202501241525156920250124.log ✘ ×

```
1 [15:25:03.751+01:00 INF] Test A | customerId= '7777'
2 [15:26:05.754+01:00 INF] Test A | customerId= '7777' has done something
3 [15:27:07.765+01:00 INF] Test B | customerId= '8888'
4 [15:28:10.766+01:00 INF] Test B | customerId= '8888' has done something else
5 [15:29:12.757+01:00 INF] Test A | customerId= '7777' has 'XXX', while expected 'XXX'
6 [15:30:19.766+01:00 INF] Test B | customerId= '7777' has 'XXX', while expected 'YYY'
7
```



Context leakage

```
private int _executedTestScenarios = 0;

[BeforXScenario] → [BeforeTestRun]
public void BeforeScenario()
{
    // Only execute this block on the very first test scenario
    if (_executedTestScenarios == 0)
    {
        Task.Run(() => CleanupOldData()).Wait();
    }
    else
    {
        Log.Information($"Skipped testdata clean-up for this individual scenario");
    }
    _executedTestScenarios++
}
```



Deadlocks



Informative



Test run finished: 9 Tests (8 Passed, 1 Failed, 0 Skipped) run in 12,7 sec

Test

- ▲ X ReqNRollerCoasterProject (9)
- ▲ X ReqNRollerCoasterProject.Fe... .
- ▲ X AddNumbersFeature (3)
 - ✓ AddANegativeNumbers
 - X AddTwoNumbers
 - ✓ AddTwoOtherNumbers
- ▷ ✓ DivideNumbersFeature (3)
- ▷ ✓ MultiplyNumbersFeature (3)

▶ Run | ▶ Debug | Ask Copilot

Test Detail Summary

X AddTwoNumbersSource: [AddNumbers.feature](#) line 9

Duration: 4,1 sec

Message:

Expected `_result` to be 120, but found 130 (difference of 10).

Stack Trace:

Standard Output:

[19:54:49.318+01:00 INF] BeforeScenario - Starting scenario: Add two numbers

Given the first number is 50

[19:54:49.318+01:00 INF] _firstNumber = 50

-> done: CalculatorStepDefinitions.GivenTheFirstNumberIs(50) (0,0s)

And the second number is 70

[19:54:50.332+01:00 INF] _secondNumber = 70

-> done: CalculatorStepDefinitions.GivenTheSecondNumberIs(70) (0,0s)

When the two numbers are added

[19:54:51.344+01:00 INF] _result = 130

-> done: CalculatorStepDefinitions.WhenTheTwoNumbersAreAdded() (0,0s)

Then the result should be 120

-> error: Expected `_result` to be 120, but found 130 (difference of 10). (0,1s)

Clear and organized logging

```
log_202501281954451920250128.log ✘ x
13 [19:54:47.283+01:00 INF] Add a negative numbers | _result = 46
14 [19:54:49.318+01:00 INF] Divide two numbers | BeforeScenario - Starting scenario: Divide two numbers
15 [19:54:49.318+01:00 IIF] Add two numbers | BeforeScenario - Starting scenario: Add two numbers
16 [19:54:49.318+01:00 IIF] Multiply two numbers | BeforeScenario - Starting scenario: Multiply two numbers
17 [19:54:49.318+01:00 IIF] Divide two numbers | firstNumber = 100
18 [19:54:49.318+01:00 IIF] Add two numbers | firstNumber = 50
19 [19:54:49.318+01:00 IIF] Multiply two numbers | firstNumber = 60
20 [19:54:50.332+01:00 IIF] Divide two numbers | secondNumber = 20
21 [19:54:50.332+01:00 IIF] Add two numbers | secondNumber = 70
22 [19:54:50.332+01:00 IIF] Multiply two numbers | _secondNumber = 70
23 [19:54:51.344+01:00 IIF] Multiply two numbers | _result = 4200
24 [19:54:51.344+01:00 IIF] Divide two numbers | result = 5
25 [19:54:51.344+01:00 IIF] Add two numbers | result = 130
26 [19:54:53.365+01:00 INF] Divide two other numbers | BeforeScenario - Starting scenario: Divide two other
27 [19:54:53.365+01:00 INF] Divide two other numbers | _firstNumber = 60
```



Clear and organized logging

All pipelines > Demo Pipeline Run in Parallel

Save Create release ...

+

Pipeline Tasks Variables Retention Options History

Part 1 - Sub Selection Of Test Scenario's Deployment process

Agent job Run on agent

Run Tests .NET Core

Upload Release Artifact - ... Publish Release Artifact

Publish Release Artifact

Task version 1.*

Display name *
Upload Release Artifact - Log

Path to file or folder that should be uploaded (folders will be zipped) *
\$(System.DefaultWorkingDirectory)\PathToProject\bin\Release\net8.0\serilogs

Control Options

Output Variables



logging in the pipeline

The screenshot shows a pipeline run interface. At the top, the pipeline path is "Demo Pipeline Run in Parallel > Release-6 > Part 1 - Sub Selection Of Test Scenario's". A red box highlights the "Logs" tab in the navigation bar. Another red box highlights the "Download all logs" button. The pipeline has two stages: "Deployment process" (Failed) and "Agent job" (Failed). The "Agent job" stage details show it started at 30-12-2024, 17:48:30 and took 32s. It includes two tasks: "Initialize job" (succeeded, 3s) and "Run Tests" (Failed, 28s, with 2 errors and 1 warning).

D Demo Pipeline Run in Parallel > Release-6 > Part 1 - Sub Selection Of Test Scenario's Failed

+ Pipeline Tasks Variables Logs Tests Deploy Cancel Refresh Download all logs Edit ...

Deployment process Failed

Agent job Failed · 2 errors 1 warning

Agent job Started: 30-12-2024, 17:48:30 ... 32s

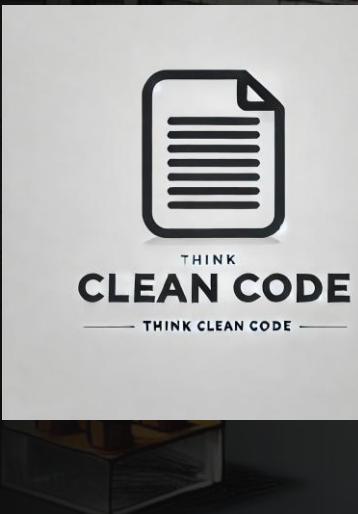
Initialize job · succeeded 3s

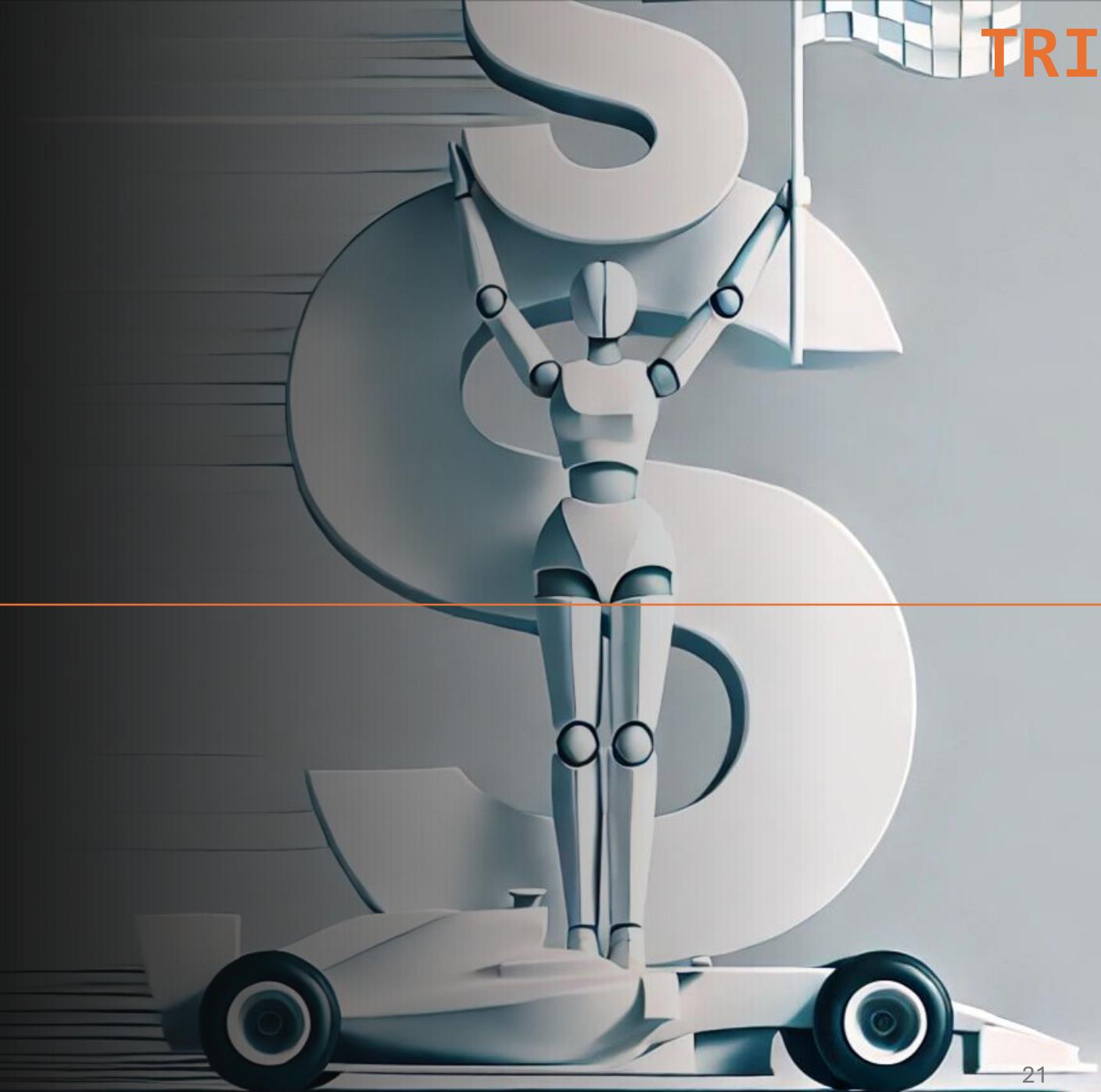
Run Tests · 2 errors 1 warning 28s



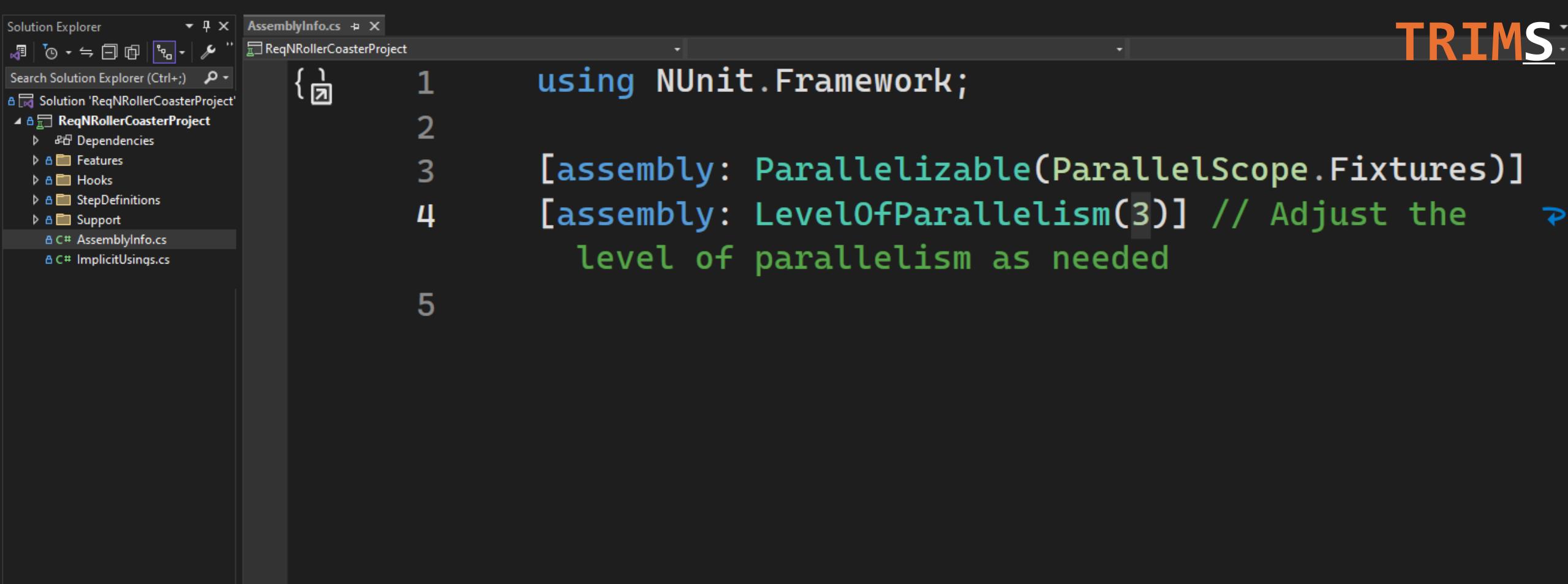
logging in the pipeline

🔧 Maintainable



A large, stylized illustration of a white and blue race car. The car has a prominent front wing and a large rear wing. It is positioned on the right side of the slide, angled slightly towards the viewer. A checkered racing flag is attached to the top of the car's rear wing. The background behind the car is a dark grey.

⚡ Speedy

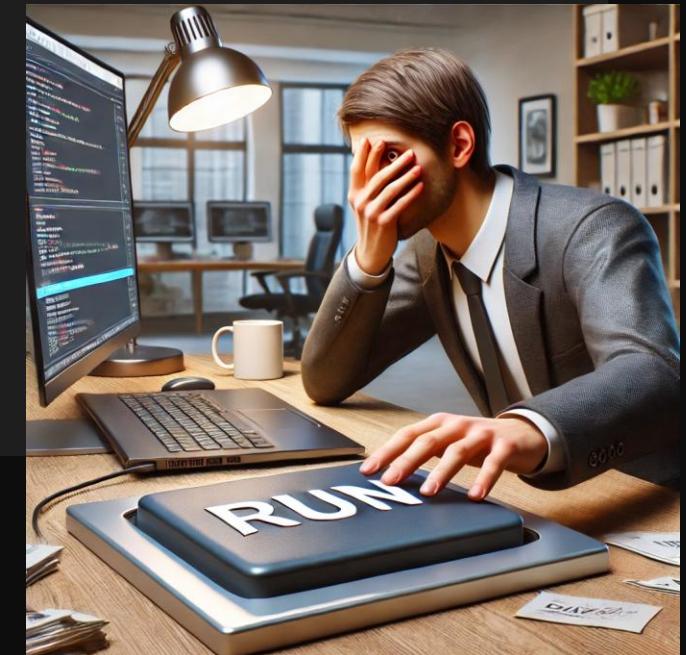


```
1  using NUnit.Framework;
2
3  [assembly: Parallelizable(ParallelScope.Fixtures)]
4  [assembly: LevelOfParallelism(3)] // Adjust the
5      level of parallelism as needed
```

 Greenfield setup



```
AssemblyInfo.cs  X
ReqNRollerCoasterProject
1   using NUnit.Framework;
2
3   [assembly: Parallelizable(ParallelScope.Fixtures)]
4   [assembly: LevelOfParallelism(3)] // Adjust the      ↵
5       level of parallelism as needed
```



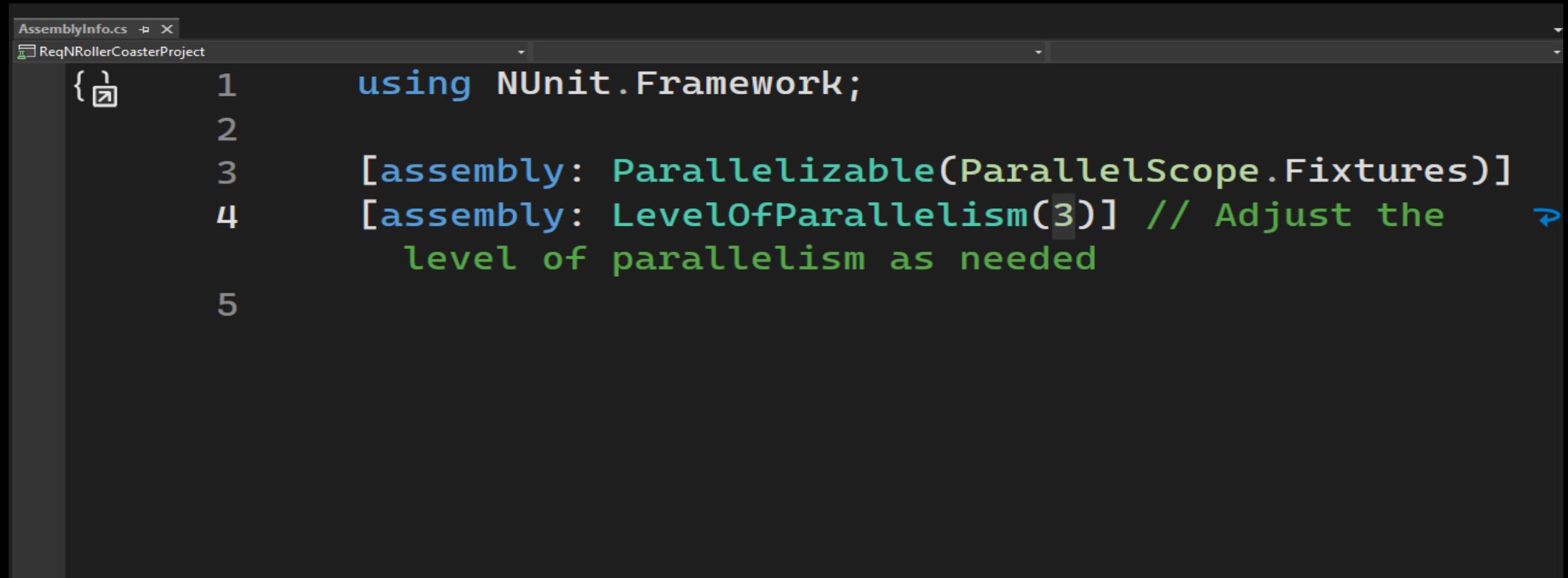


```
///<summary>
/// Cleanup statics from 'Context' class before each testscenario
///</summary>
[BeforeScenario]
public void BeforeScenarioCleanupStatics()
{
    // Add 'scenario' scope to seri log
    LogContext.PushProperty("Scenario", _scenarioContext.ScenarioInfo.Title);

    string[] assignedTags = _scenarioContext.ScenarioInfo.CombinedTags;
    if (assignedTags.Contains("RunInParallel"))
    {
        Log.Information("BeforeScenario - This test scenario should be able to run in parallel");
    }
    else
    {
        Log.Information("All shared 'Context' objects will be cleared");
        //todo: these contexts must be phased out, to run in parallel: It should not be necessary to reset
        //       contexts between multiple scenarios:
        Context.DummyA = null;
        Context.DummyB = null;
        Context.DummyC = null;
        Context.DummyD = null;
        Context.DummyE = null;
    }
}
```

```
log_202501241525156920250124.log ✘ ×
1 [15:25:03.751+01:00 INF] Test A | customerId= '7777'
2 [15:26:05.754+01:00 INF] Test A | customerId= '7777' has done something
3 [15:27:07.765+01:00 INF] Test B | customerId= '8888'
4 [15:28:10.766+01:00 INF] Test B | customerId= '8888' has done something else
5 [15:29:12.757+01:00 INF] Test A | customerId= '7777' has 'XXX', while expected 'XXX'
6 [15:30:19.766+01:00 INF] Test B | customerId= '7777' has 'XXX', while expected 'YYY'
7
```





A screenshot of the Visual Studio IDE showing the `AssemblyInfo.cs` file for a project named `ReqNRollerCoasterProject`. The code defines assembly-level attributes for parallel testing:

```
1  using NUnit.Framework;
2
3  [assembly: Parallelizable(ParallelScope.Fixtures)]
4  [assembly: LevelOfParallelism(3)] // Adjust the      ↵
5          level of parallelism as needed
```



Legacy setup

```
Solution Explorer DivideNumbers.feature X
Solution Explorer (Ctrl+.) 🔍
Solution 'ReqNRollerCoasterProject'
  Features
    AddNumbers.feature
    DivideNumbers.feature (highlighted)
    MultiplyNumbers.feature
  Hooks
  StepDefinitions
  Support
    AssemblyInfo.cs
    ImplicitUsings.cs

1 @demo @LegacyFunctionX
2 Feature: DivideNumbers
3   As a user
4     I want to divide two numbers
5     So that I can get the result
6
7   Simple calculator for divide two numbers
8
9   ✓ Scenario: Divide two numbers
10    Given the first number is 100
11    And the second number is 20
12    When the two numbers are divided
13    Then the result should be 5
14
15 @LegacyFunctionY
16   ✓ Scenario: Divide a negative numbers
17    Given the first number is 192
18    And the second number is 32
19    When the two numbers are divided
```



The screenshot shows the Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows a project named "ReqNRollerCoasterProject" containing files like "AddNumbers.feature", "DivideNumbers.feature", "MultiplyNumbers.feature", "AssemblyInfo.cs", and "ImplicitLinks.cs". The "reqnroll.json" file is highlighted with a green border.
- Code Editor:** Displays the "reqnroll.json" file content. The schema is defined as <https://schemas.reqnroll.net/reqnroll-config-latest.json>. The code includes comments and a generator section for parallel execution.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <configuration>
3      <comment1>Here's a place to provide an explanation of why the test scenarios below cannot yet be run in parallel.</comment1>
4      <comment2>Here too</comment2>
5      <generator>
6          <addNonParallelizableMarkerForTags>[
7              "NonParallel",
8              "LegacyFunctionX",
9              "AnotherTag"
10         ]</addNonParallelizableMarkerForTags>
11     </generator>
12 </configuration>
```



Legacy setup

The screenshot shows a Visual Studio interface with the following details:

- Solution Explorer:** Displays the project structure:
 - Solution 'ReqNRollerCoasterProject' (1 of 1 project)
 - ReqNRollerCoasterProject
 - Dependencies
 - Features
 - AddNumbers.feature
 - DivideNumbers.feature**
 - MultiplyNumbers.feature
 - Hooks
 - StepDefinitions
 - Support
 - DemoHelpers.cs
 - AssemblyInfo.cs
 - ImplicitUsings.cs
 - reqnroll.json
- DivideNumbers.feature:** The active file in the editor. It contains Gherkin steps and annotations. Annotations are present on the first two lines of the file header and around the "Scenario" keyword.

```
1 @demo @LegacyFunctionX
2 Feature: DivideNumbers
3   As a user
4     I want to divide two numbers
5     So that I can get the result
6
7   Simple calculator for divide two numbers
8
9   Scenario: Divide two numbers
10  Given the first number is 100
11  And the second number is 20
12  When the two numbers are divided
13  Then the result should be 5
14
```



```
/// <summary>
/// Cleanup statics from 'Context' class before each testscenario
/// </summary>
[BeforeScenario]
0 references
public void BeforeScenarioCleanupStatics()
{
    // Add 'scenario' scope to seri log
    LogContext.PushProperty("Scenario", _scenarioContext.ScenarioInfo.Title);

    string[] assignedTags = _scenarioContext.ScenarioInfo.CombinedTags;
    if (assignedTags.Contains("RunInParallel"))
    {
        Log.Information("BeforeScenario - This test scenario should be able to run in parallel");
    }
    else
    {
        Log.Information("All shared 'Context' objects will be cleared");
        //todo: these contexts must be phased out, to run in parallel: It should not be necessary to reset
        //       contexts between multiple scenarios:
        Context.DummyA = null;
        Context.DummyB = null;
        Context.DummyC = null;
        Context.DummyD = null;
        Context.DummyE = null;
    }
}
```



Legacy setup

 **T** argeted

 **R** eliable

 **I** nformative

 **M** aintainable

 **S** peedy

⚡ To summarize



Greenfield: Start parallel testing early—initial setup is simple.



Implementation: Use scenario tags, JSON config, and custom code if needed.

Legacy: Use a phased approach to introduce parallel runs.

Both: Clear custom logging is essential for quick issue tracing.





Give yourself
access to all the
content and url's
from this
presentation.
Start boosting
your test speed
tomorrow!