

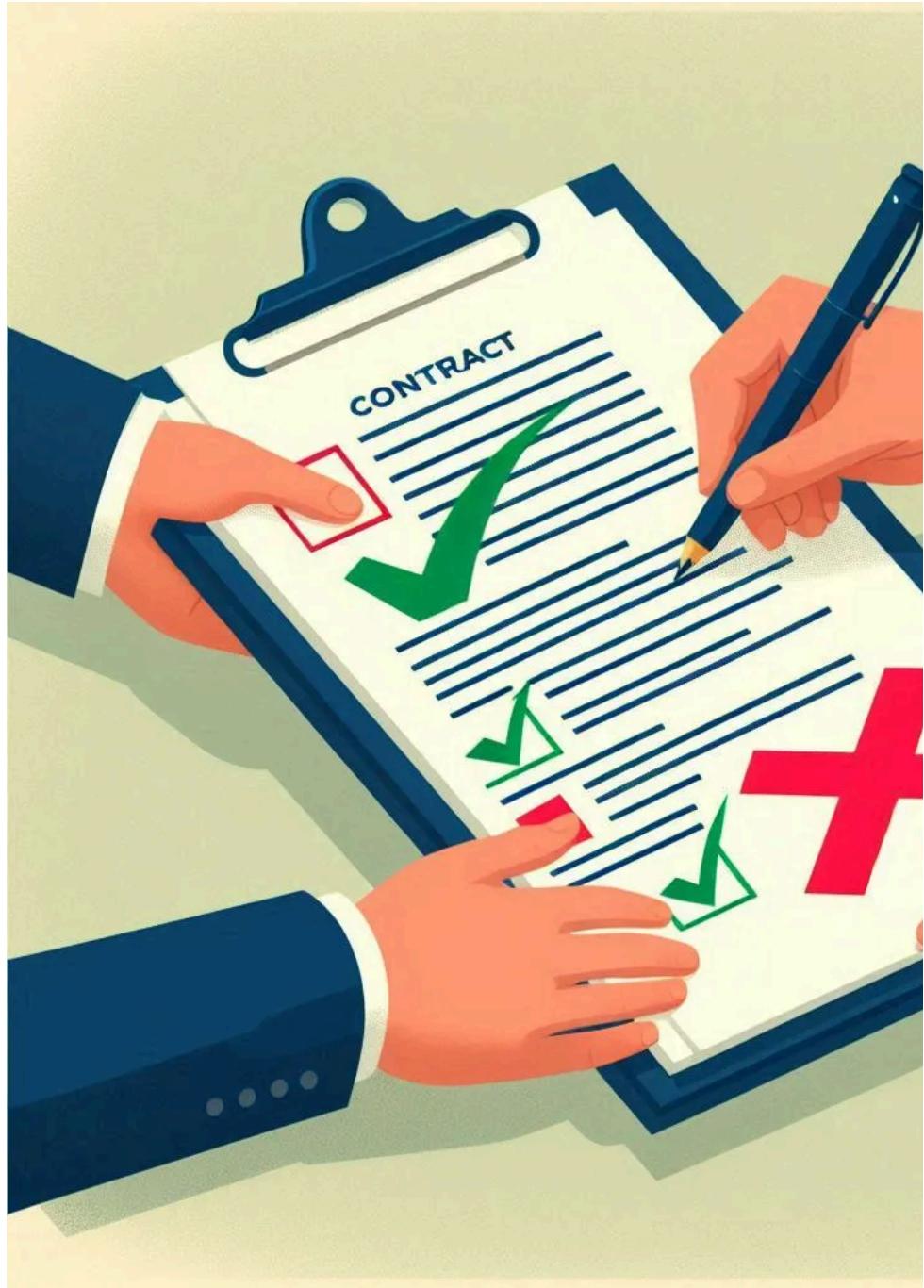
Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)

## Intro to PACT for .NET Core: Integration with PactFlow



Ajay kumar

6 min read · Jul 1, 2024

[Listen](#)[Share](#)[More](#)

This is in continuation of PACT for .NET series where we will be looking into one of the ways to integrate our APIs with PactFlow.

For a short introduction, PactFlow is a platform that can act as a broker between consumers and providers services/systems. Consumers can publish their contracts and providers can consume the same contracts and try to verify them.

In PactFlow one can visualise failing/passing contracts, total contract integrations, dependency graph and much more. For a better re

You need to register into pact for for a limited free or paid account. For this demo i am using free account that can support up to 2 co

Now, for this article we will be following the same Student and Result API scenario, where student is the Provider service and Result s

To learn more about demo repo that we will be using in this article, please follow previous articles in this series:

### **Contract testing for APIs: Intro to PACT for .NET Core**

Recently i came a across a situation where i had to explore contract testing using PACT framework.

medium.com

### **Intro to PACT for .NET Core: Events based systems**

This is continuation of PACT for .NET series where I am going to cover usage of pact in event-based systems.

medium.com

## **Consumer Side**

Once the PACT contract file is generated by the consumer there are several ways we can push or upload the contract to pact flow. You [https://docs.pact.io/getting\\_started/sharing\\_pacts](https://docs.pact.io/getting_started/sharing_pacts).

For this post i have used a combination of their REST APIs and PactNet library.

The below utility code uploads the contract file to the pact broker. Among all the parameters, it requires pact broker base URI and rea information by logging into your pact flow account and reaching out to settings section.

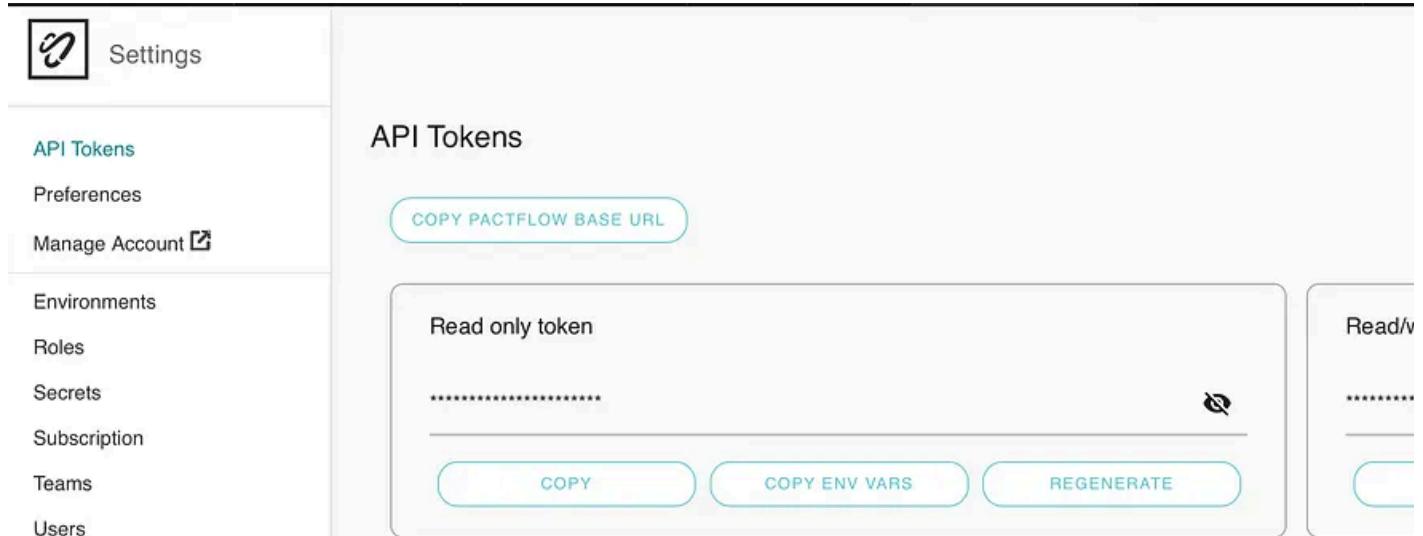


Image 1: PactFlow settings section

```
// Code to push pact contract file to pact flow
using System.Net;
using System.Net.Http.Headers;
using System.Text;

namespace PactNet.ConsumerOne.UnitTesting;

public abstract class PactBrokerUtiliy
{
    private const string PactSubUrl = "{0}/pacts/provider/{1}/consumer/{2}/version/{3}";

    /// <summary>

```

```

/// Utility method to upload pact contract file to pact flow
/// </summary>
/// <param name="pactFlowBaseUrl">Can be obtained from pact flow settings</param>
/// <param name="consumerName">Consumer service name</param>
/// <param name="providerName">provider service name</param>
/// <param name="contractJson">Contract JSON in string format</param>
/// <param name="accessToken">Access token obtained from pact flow settings</param>
/// <param name="consumerVersion">This needs to be something unique version number or string. Should not be repeated.</param>
/// <returns>Status code to the http response</returns>
public static async Task<HttpStatusCode> PublishPactContract(string pactFlowBaseUrl, string consumerName, string providerName, string contractJson, string accessToken)
{
    var httpClient = new HttpClient();
    httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("bearer", accessToken);
    var fullUrl = string.Format(PactSubUrl, pactFlowBaseUrl, providerName, consumerName, string.IsNullOrEmpty(consumerVersion) ? consumerName : consumerVersion);
    var response =
        await httpClient.PutAsync(fullUrl, new StringContent(contractJson, Encoding.UTF8, "application/json"));
    var responseContent = await response.Content.ReadAsStringAsync();
    return response.StatusCode;
}
}

```

Now we can use this utility method after the pact contract file is generated by the consumer. Some thing like below

```

// Code excerpt from ReportCardApiPactFlowTest.cs class
[Fact]
public async Task Get_Student_When_The_StudentId_Is_Invalid()
{
    // Arrange
    _pactBuilder
        .UponReceiving("A GET request to retrieve the student with invalid student id")
        .Given("There is student is at least one valid student present")
        .WithRequest(HttpMethod.Get, "/students/some-invalid-id")
        .WithHeader("Accept", "application/json")
        .WillRespond()
        .WithStatus(HttpStatusCode.NoContent);

    _pactBuilder.Verify(ctx =>
    {
        // Act
        var client = new StudentClient(ctx.MockServerUri);
        Assert.ThrowsAsync<Exception>(async () => await client.GetStudentById("some-invalid-id"));
    });
}

await UploadPactContract();
}

private static async Task UploadPactContract()
{
    var pactPath = Path.Combine("..",
        "..",
        "..",
        "pacts",
        "ConsumerOne-Student API.json");

    var contractJson = File.ReadAllText(pactPath);
    var pactFlowBaseUrl = Environment.GetEnvironmentVariable("PACT_FLOW_BASE_URL"); // For testing purposes, you may even hardcode this
    var pactFlowToken = Environment.GetEnvironmentVariable("PACT_FLOW_TOKEN"); // For testing purposes, you may even hardcode this
    var statusCode = await PactBrokerUtiliy.PublishPactContract(
        pactFlowBaseUrl,
        "ConsumerOne",
        "Student API",
        contractJson,
        pactFlowToken,
        Guid.NewGuid().ToString());
}

```

You may notice that i have used environment variables at few places. This is to avoid mentioning account details that i have used.

```
Environment.GetEnvironmentVariable("PACT_FLOW_BASE_URL");
// It will translate to something like https://your_account_name.pactflow.io

Environment.GetEnvironmentVariable("PACT_FLOW_TOKEN");
// It will translate to some random string
```

If you are using it for demo purpose or learning you may also choose to hardcode these to the ones you obtained from pact flow settings. You can mention these environment variables before you run tests.

After all these code changes and settings are in place, we can run the test and should expect a success. Following things will happen when we run the test:

- Our request verification and assertion will happen
- Pact contract file will be generated at specified location
- The same pact file content will be read and passed on to utility method to be sent as payload to HTTP request to pact flow

Once the test is green, we can head to pact flow home page to see if any integrations (consumer-provider)/contracts have been uploaded. Below screenshot shows the pact flow interface:

The screenshot shows the SMARTBEAR PactFlow web application. On the left sidebar, there is a dropdown for 'Select team' and a search bar for 'Filter your pacts'. Below these, there are buttons for 'Integration' and 'ConsumerOne ☞ Student API'. The main content area is titled 'ConsumerOne ☞ Student API'. It displays a summary card for the integration, which is labeled 'Unverified' and has a minus sign icon. The card contains the following information:

CONSUMER VERSION	18b9095a-18da-480b-addc-e9190aa840b2
BRANCH	N/A
ENVIRONMENTS	N/A
TAGS	N/A
PUBLISHED AT	a few seconds ago

Image 2: Student API and its consumer integration in pact flow

If you observe the above image, we can see the contract is there, but it's in unverified mode. It is because this pact has not been verified yet.

## Provider side pact verification

Provider side is relatively straight forward. I have written another unit test in the same unit test class, which will take the pact file sou-

```
[Fact]
public void Ensure_StudentApi_Honours_Pact_With_ConsumerOne_Using_PactFlow()
{
    // Arrange
    var config = new PactVerifierConfig
    {
        Outputters = new List<IOutput>
        {
            new XunitOutput(_output),
        },
        LogLevel = PactLogLevel.Information
    };
}
```

```

var pactFlowBaseUri = Environment.GetEnvironmentVariable("PACT_FLOW_BASE_URL"); // For testing purposes, you may even hardcode
var pactFlowToken = Environment.GetEnvironmentVariable("PACT_FLOW_TOKEN"); // For testing purposes, you may even hardcode

// Act // Assert
IPactVerifier pactVerifier = new PactVerifier(config);
pactVerifier
    .ServiceProvider("Student API", _fixture.ServerUri)
    .WithPactBrokerSource(new Uri(pactFlowBaseUri), configure =>
{
    configure.TokenAuthentication(pactFlowToken);
    configure.PublishResults(true, "1.0.0"); // Any version
})
    .WithProviderStateUrl(new Uri(_fixture.ServerUri, "/provider-states"))
    .Verify();
}

// The below piece of code is the key in verification with pact flow as source
.WithPactBrokerSource(new Uri(pactFlowBaseUri), configure =>
{
    configure.TokenAuthentication(pactFlowToken);
    configure.PublishResults(true, "1.0.0"); // Any version
})

```

Now finally after all this setup in place. If we run the pact provider test we should see the test being success, if we run it locally. Some

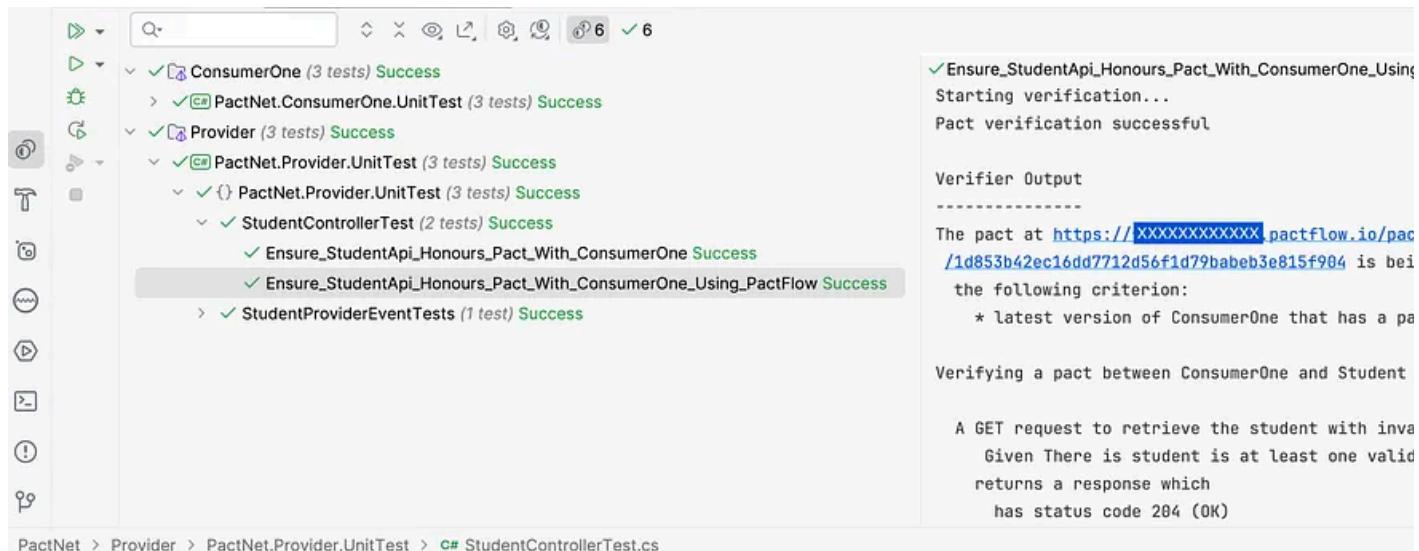


Image 3: Pact flow success console logs

As a result of this we should also be able to see that the pact integration that was showing as unverified earlier has changed to verified.

CONSUMER VERSION	18b9095a-18da-480b-addc-e9190aa840b2
BRANCH	N/A
ENVIRONMENTS	N/A
TAGS	N/A
PUBLISHED AT	41 minutes ago

The screenshot shows the PactFlow interface. On the left, there's a sidebar with 'Select team', 'Filter your pacts', and a search bar. Below that are buttons for 'Integration' and 'ConsumerOne ☞ Student API'. The main content area has a green checkmark icon and the title 'A pact between ConsumerOne and Student'. It's divided into sections: 'Consumer Details' (Consumer Version: 18b9095a-18da-480b-addc-e9190aa840b2, with a 'More consumer details' link), 'Provider Details' (Provider Version: 1.0.0, with a 'More provider details' link), 'Pact' (Specification Version: 3.0.0), and 'Interactions'. Under 'Interactions', there are two green checkmarks: 'A GET request to retrieve the student with invalid student id given there is student with id 067a50e' and 'A GET request to retrieve the student given there is student with id 067a50e'.

Image 5: Image mentioning that the interactions have been passed

## Failure scenario!

Lets make some changes in our code to deliberately make the code fail. For demo we can change the student service to return `firstN` service. After making this change, when the unit test is run, we can see the below result in pact verification.

The screenshot shows the PactFlow interface. The sidebar is identical to the previous one. The main content area has a red X icon and the title 'ConsumerOne ☞ Student API'. It's divided into sections: 'Failed' (with a large red X icon), 'CONSUMER VERSION' (18b9095a-18da-480b-addc-e9190aa840b2), 'BRANCH' (N/A), 'ENVIRONMENTS' (N/A), 'TAGS' (N/A), and 'PUBLISHED AT' (an hour ago). To the right of each section header is a small letter (P, 1, B, N, E, T, V, a) which likely corresponds to a column in a table.

Image 6: Pact flow failure

The screenshot shows the PactFlow application interface. On the left, there's a sidebar with navigation links: 'Select team', 'Filter your pacts' (with a search icon), 'Integration', and 'ConsumerOne ↔ Student API'. The main content area has a title 'A pact between ConsumerOne and Student' with a red 'X' icon. It's divided into sections: 'Consumer Details' (Consumer Version: 18b9095a-18da-480b-addc-e9190aa840b2, 'More consumer details'), 'Provider Details' (Provider Version: 1.0.0, 'More provider details'), and 'Pact' (Specification Version: 3.0.0). Below these is the 'Interactions' section, which displays a failure message: 'A GET request to retrieve the student given there is student with id 067a50c5-0b23-485e-b018-17c66b2422ff' with a red 'X' icon. This is followed by 'Mismatches (1)' and a detailed error message: 'Body: \$: Actual map is missing the following keys: firstName'. At the bottom, it shows the 'Request' with Method: GET and Path: /students/067a50c5-0b23-485e-b018-17c66b2422ff.

Image 7: Pact flow failure details

For reference, the code repository being discussed is available on github link: <https://github.com/ajaysskumar/pact-net-example>

Thanks for reading through. Please share feedback, if any, in comments or on my email ajay.a338@gmail.com.

[Pact](#) [Pactnet](#) [Pactflow](#) [Dotnet](#) [Contract Testing](#)



## Written by Ajay kumar

5 Followers · 2 Following

Software enthusiast in architecture and testing. Movie and series lover. Welcome to my blog of professional and personal passions.

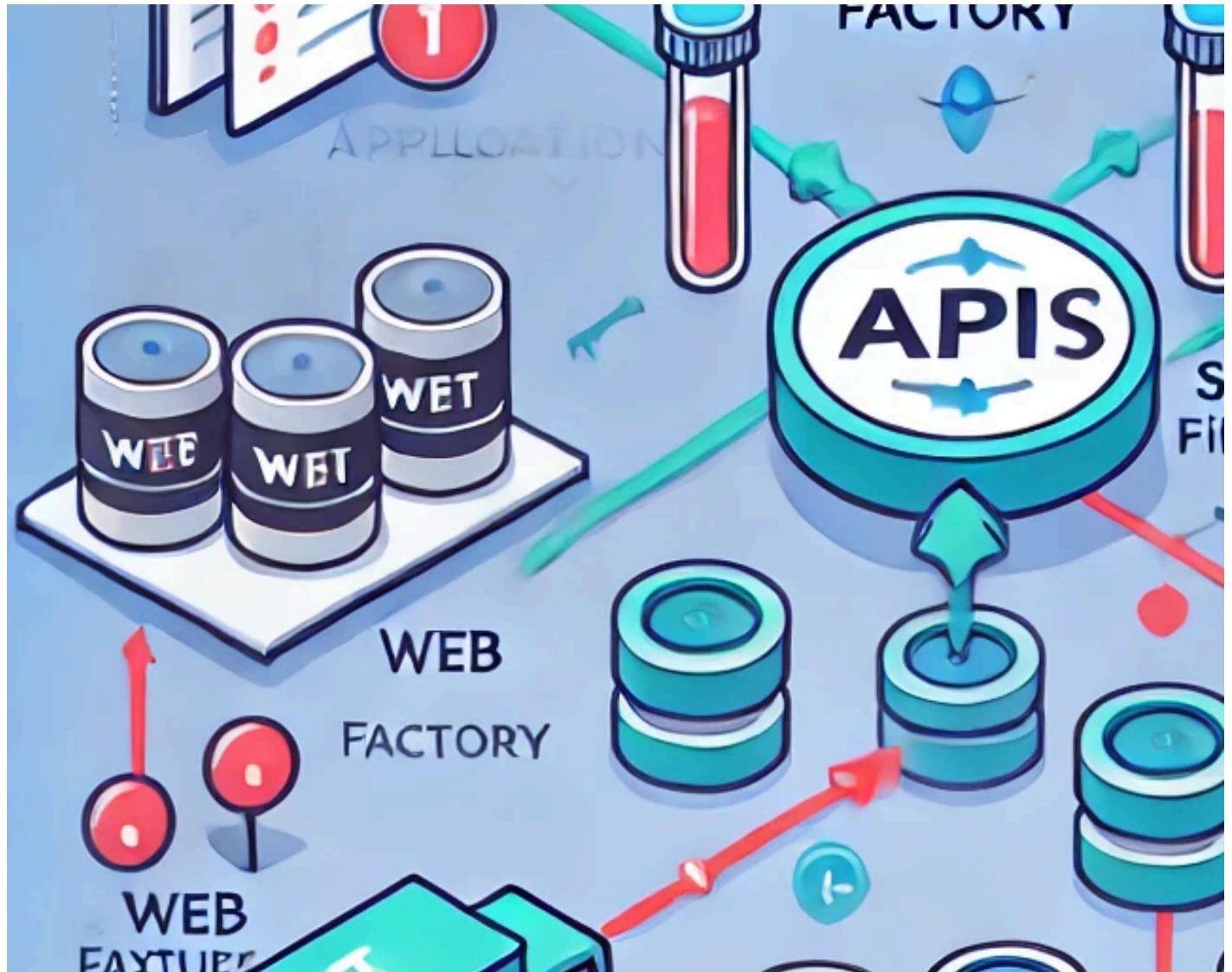
No responses yet



Ajay kumar

What are your thoughts?

More from Ajay kumar



 Ajay kumar

## Integration testing for dotnet core APIs: Handling database

Welcome to the 2nd post in our Integration testing series. You may check out the previous post that is about Introduces with the concept of...

Oct 13, 2024  1

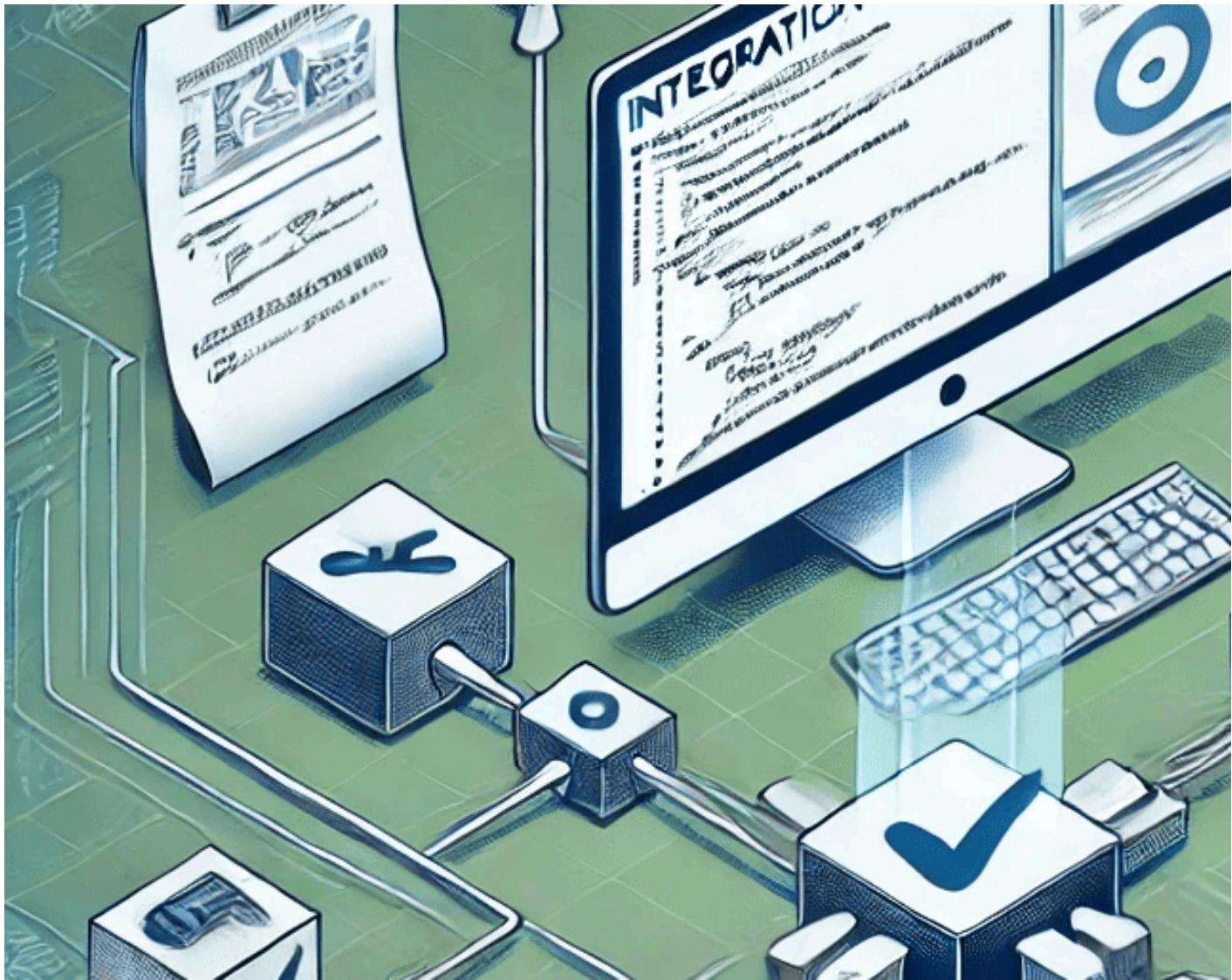


 Ajay kumar

## Contract testing for APIs: Intro to PACT for .NET Core

Recently i came across a situation where i had to explore contract testing using PACT framework.

Mar 27, 2024  1



 Ajay kumar

## Integration testing for dotnet core APIs: Introduction

This is going to be multi-part series on integration tests from introduction to advanced used cases.

Sep 29, 2024  1



 Ajay kumar

## Integration testing for dotnet core APIs: Handling 3rd party service calls using wiremock

This is continuation of the Integration testing in dotnet series. This time we will be covering the scenario where third party services...

Dec 15, 2024  2

---

[See all from Ajay kumar](#)

---

### Recommended from Medium



In .Net Programming by Sukhpinder Singh | C# .Net 

## 100 Expert C# Tips to Boost Your Coding Skills

I've compiled a list of 100 tips that have helped me write cleaner, more efficient, and maintainable code. Let me take you through these...

5d ago 277 8

The screenshot shows the new Google IDE interface. On the left is the Explorer panel with project structure: MYAPP (lib/main.dart, test, web, .idea, .idx, android, build, lib/main.dart). The main area is the Editor showing code for main.dart:

```

lib > main.dart > MyApp > build
57 _MyHomePageState extends State<MyHomePage> {
72   get build(BuildContext context) {
107     children: <Widget>[
108       const Text('You have pushed the button this man
109       Text(
110         '_counter',
111         style: Theme.of(context).textTheme.headlineM
112       ), // Text
113     ], // <Widget>[]
114   ), // Column
115 }, // Center
116 floatingActionButton: FloatingActionButton(
117   onPressed: _incrementCounter,
118   tooltip: 'Increment',
119   child: const Icon(Icons.add),
120 ), // This trailing comma makes auto-formatting nicer
121 ; // Scaffold
122
123
124

```

Below the editor are the PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL tabs. The OUTPUT tab shows logs:

```

2025-03-02T14:37:35Z [android] Running Gradle task 'assembleDebug'...
2025-03-02T14:37:39Z [android] ✓ Built build/app/outputs/flutter-apk/ap
2025-03-02T14:37:39Z [android] Lost connection to device.
2025-03-02T14:37:43Z [android] Syncing files to device sdk gphone64 x86
2025-03-02T14:37:43Z [android] <IDX> Preview running
2025-03-02T14:37:43Z [android] I/Choreographer( 7793): Skipped 124 fram
2025-03-02T14:37:43Z [android] I/Gralloc4( 7793): mapper 4.x is not sup
2025-03-02T14:37:43Z [android] W/OpenGLESRenderer( 7793): Failed to initi
2025-03-02T14:37:44Z [android] I/Choreographer( 7793): Skipped 50 frame
2025-03-02T14:37:48Z [android] D/ProfileInstaller( 7793): Installing pr

```

At the bottom are the OUTLINE and TIMELINE tabs.

In Coding Beauty by Tari Ibaba

## This new IDE from Google is an absolute game changer

This new IDE from Google is seriously revolutionary.

Mar 12 1.5K 95

# MASTER .NET CHANNELS NOW

Unlock Superior Background Task Performance



 AshokReddy

## Mastering .NET Channels: The Secret to High-Performance Background Processing

Introduction:

6d ago 63 4



 In ITNEXT by Zsolt Deak

## The Easiest Ways to Find Memory Leaks in Your Angular Application with Chrome (beginner's guide)

Just put the suspicious component in an @if

 4d ago  86

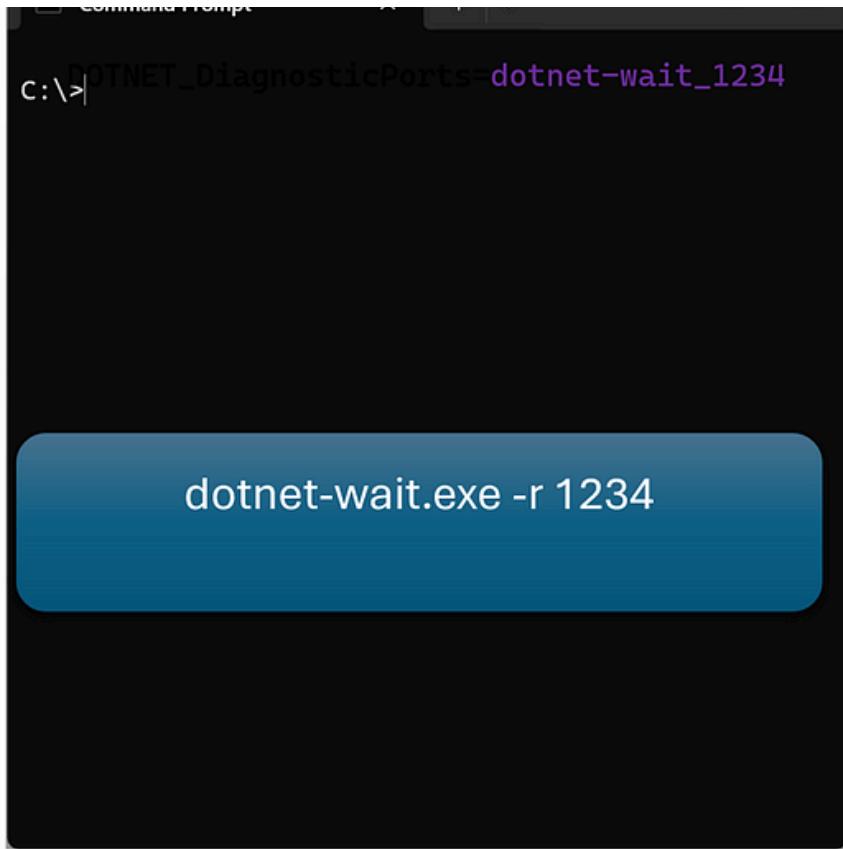


 In IT Dead Inside by Christopher Laine 

## The Mysterious Art of Estimating

And what you can do to get better at it

Nov 7, 2023 202 4



C:\> DOTNET\_DiagnosticPorts=dotnet-wait\_1234  
dotnet-wait.exe -r 1234

```
rds = new ReversedDiagnosticsServer(dotnet-wait_1234)
rds.Start();
var acceptTask = rds.AcceptAsync(...);

client = new DiagnosticsClient(acceptTask.Endpoint);
DiagnosticsClient::ResumeRuntime() ④
```



 Christophe Nasarre

## How to monitor .NET applications startup

This episode explains how to monitor the startup of a .NET application and get insights about its lock and wait contentions duration

6d ago 11

[See more recommendations](#)