Testcontainers - Integration Testing Done Right

https://github.com/christophwille

There is no silver bullet

It depends

because...

"it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."

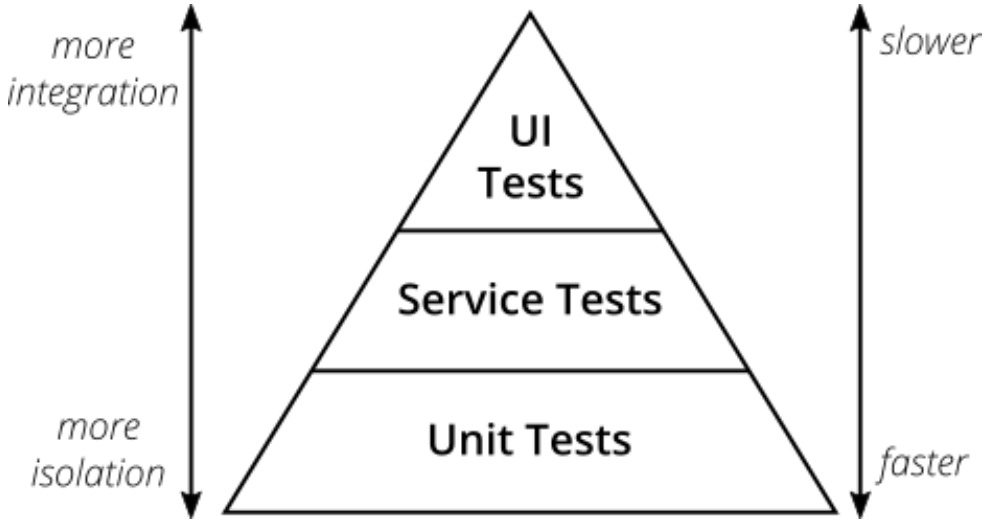
https://en.wikipedia.org/wiki/Law_of_the_instrument

About Me

- Way too long in this business
- Wrote a couple of books way back when
- Still proud IC (Individual Contributor)
- Red tape guy (mostly) for ILSpy
- “The grass is always greener on the other side” guy

https://github.com/christophwille/demosatconferences/tree/master/UpdateConferenceKrakow25

The Test Pyramid



https://martinfowler.com/articles/practical-test-pyramid.html#IntegrationTests

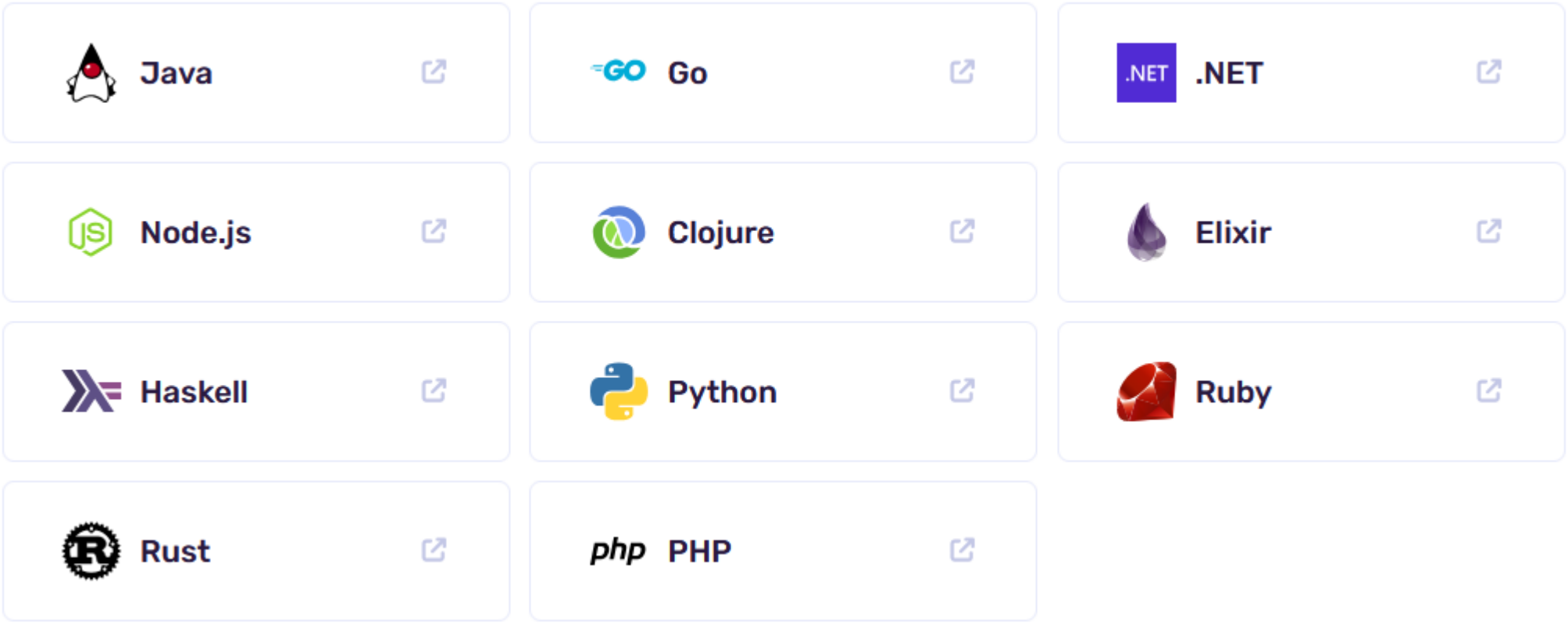
The (Most Common) Selling Point

- Integration Tests for Databases
  - Local Install?
  - Docker (Podman)?
  - … and then concurrency hits
  - … and weird test ordering bugs appear
  - … and cleanup (after test crashes)

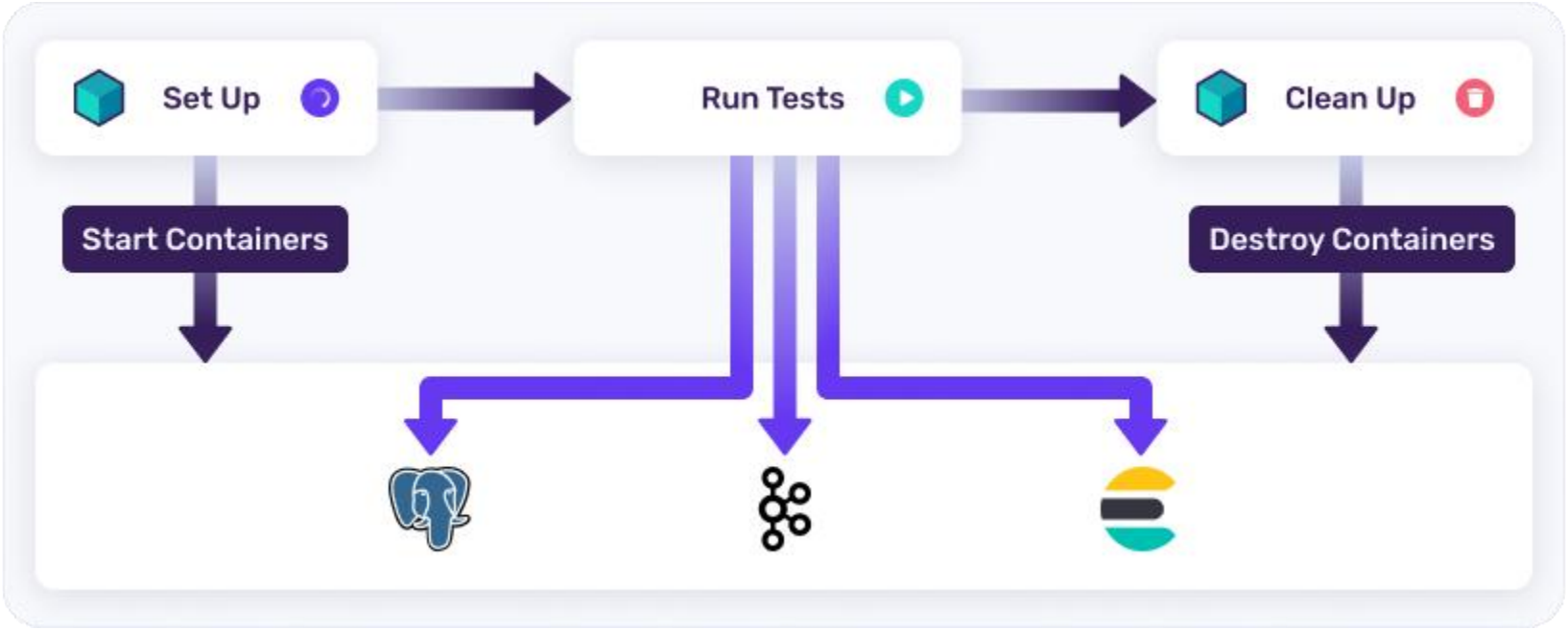
Use Cases

- Data access layer integration tests
  - https://testcontainers.com/guides/getting-started-with-testcontainers-for-dotnet/
- UI/Acceptance tests
- Application integration tests

Not a .NET Thing



Testcontainers workflow



https://testcontainers.com/getting-started/#testcontainers-workflow

Modules

- https://dotnet.testcontainers.org/modules/
  - official ones https://www.nuget.org/profiles/Testcontainers
    - PostgreSql
    - MsSql
    - Redis
    - Azurite
    - CosmosDb
    - Keycloak
    - ServiceBus
    - ...

Detour – Entity Framework

- Why not… ?
  - SQLite Provider
  - In-Memory Provider

Jiri and Shay talk about EF Core testing and Maurycy corrects them

"We hate the in-memory provider" quote, Shay Rojansky
https://www.youtube.com/watch?v=FV5e3-5IOuw (Apr 24, 2025)

- Why unit testing doesn't cut it for the EF team
- The secret why it takes only 5 minutes
- Cram in as much as possible in a suite
- Getting resilience in tests via retries, ordering across providers

Testing.Platform, the new way to run .NET tests

- https://youtu.be/9Jz47Ze9LOI (your tests in containers)

Demo time

- Getting started

But… Wall Clock Time Sucks

- A new container / test is actually a bit much
- What are your options?
  - Make use of collections
  - Respawn https://github.com/jbogard/Respawn
  - Build your own container image

Demo time

- Homework
  - https://github.com/danielwarddev/TestingWithDb

A Few Gems

- “Works on my machine”
- “Latest is not a version”
- “Size matters”

Testcontainers Can Do More

- Private registries
- Remote daemon
- Reuseable containers
- Your own dockerfiles, …

- But… not everything (in all SDKs)
  - https://github.com/testcontainers/testcontainers-dotnet/issues/122

It Depends – Azure Functions

- Yes, you can stick the Function app in a container
- Or only the dependencies

- Consider your options
  - Copy the Azure Functions' team approach
  - Aspire
    - https://learn.microsoft.com/en-us/dotnet/aspire/serverless/functions

- Other tools also might be better "outside"
  - Dev Proxy, WireMock.Net, …

https://github.com/christophwille/poc-oh/tree/main/src/TestcontainersPlayground/TestcontainersPlayground.AzFunc

Summary

- Testcontainers are a tool you should have in your toolbelt
- Testing the real “thing” gives confidence
- Test stability is a good thing
- And yes, wall clock time for CI matters!