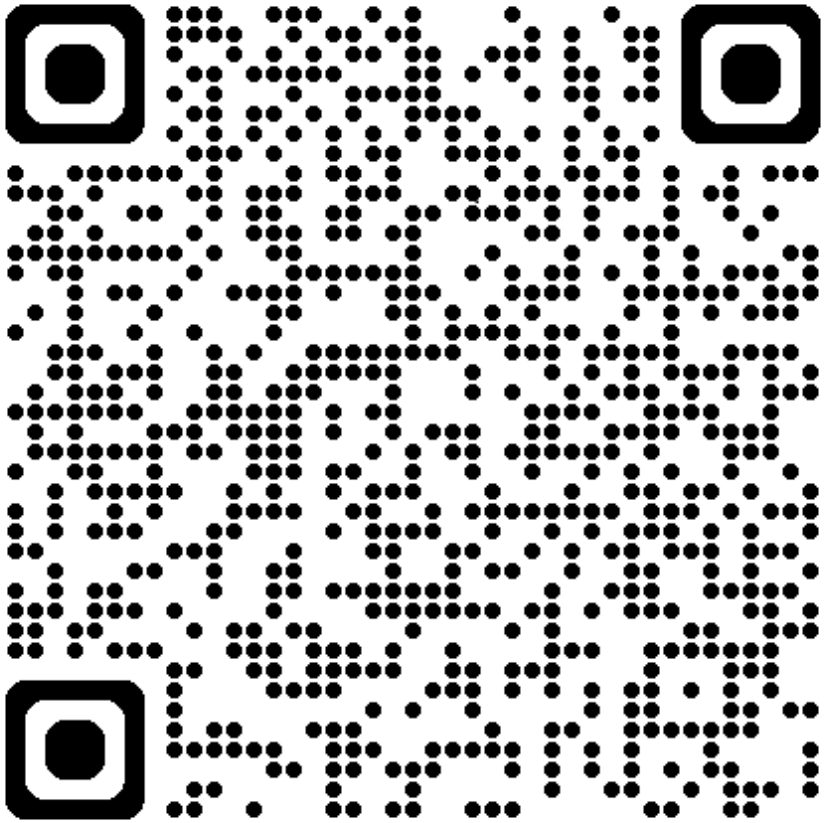
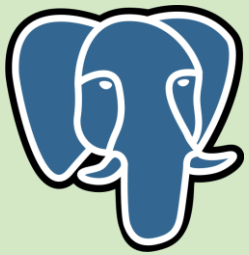


No More SQLite

How to Write Tests With EF Core Using Testcontainers

Daniel Ward

Works on my machine

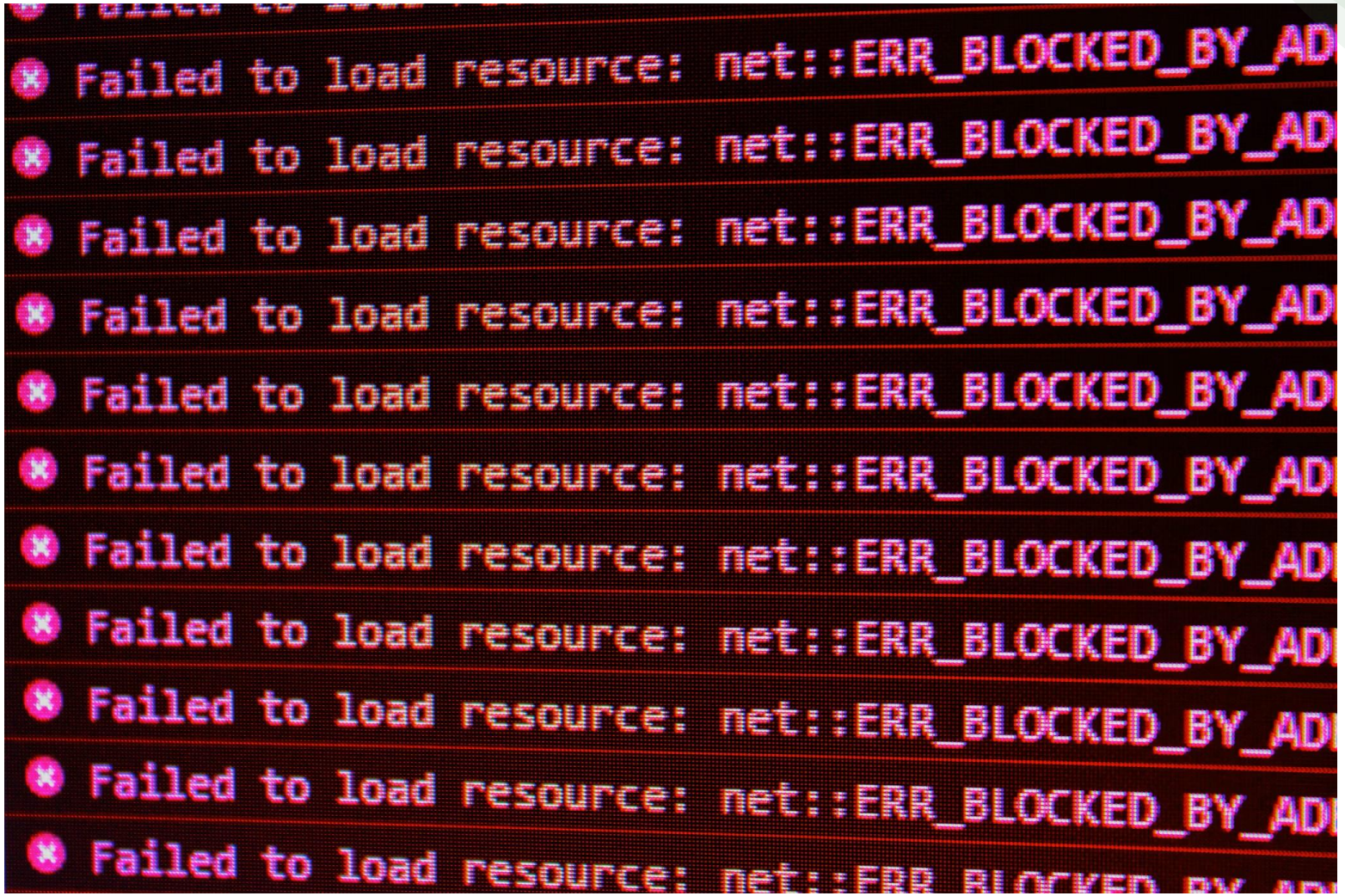


You’re good!

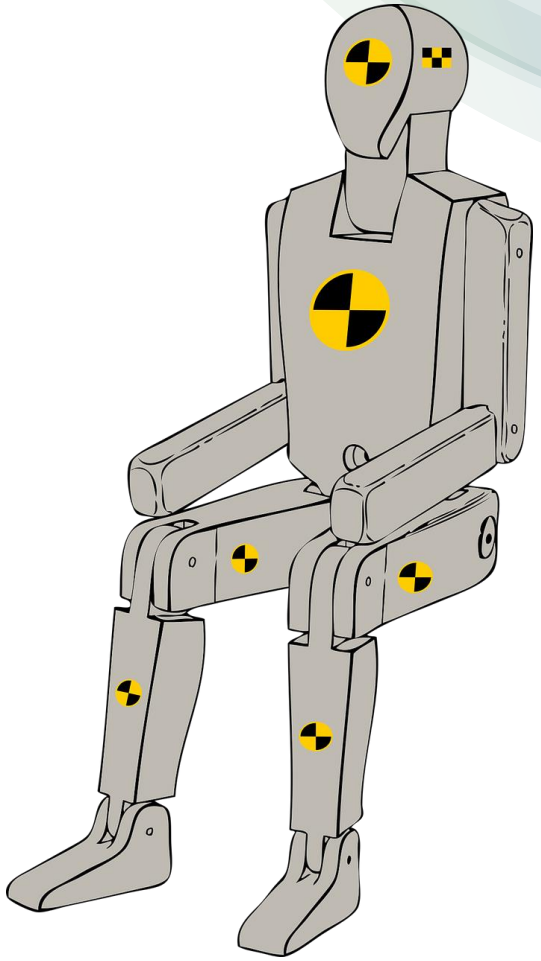
No man’s land

No man’s land

False positives, False Negatives



If you use another SQL engine, you are testing another system!



Why does that matter?

Me

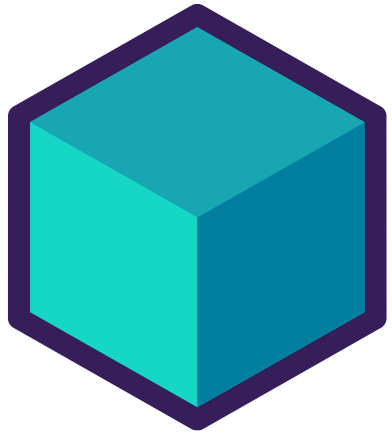
- Software developer, consultant
- MVP .NET
- Co-organizer of the San Antonio/Austin .NET User Group
- daninacan.com
- @danielwarddev
- daniel-ward-dev

LEAN TECHNIQUES

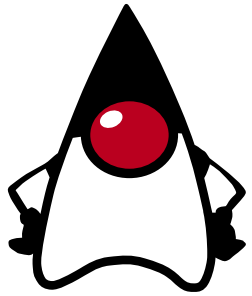
Traditional Options

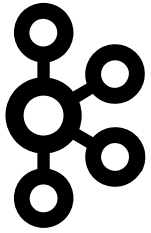
	Pros	Cons
EF in-memory provider	Easy to set up	Very unlike real database
SQLite provider	Easy to set up	Unlike real database
Real database	Same engine as production	Lots more work to maintain
Docker container	Same engine as production	More work to maintain

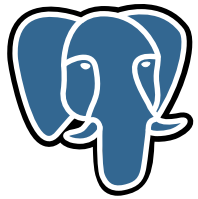
Take the existing Docker workflow, but automate it and add tooling

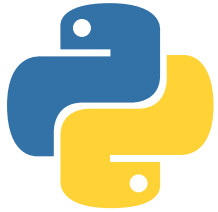


Testcontainers

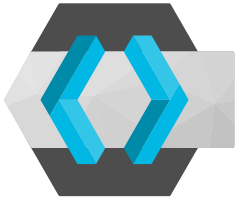


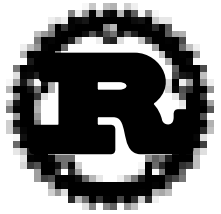


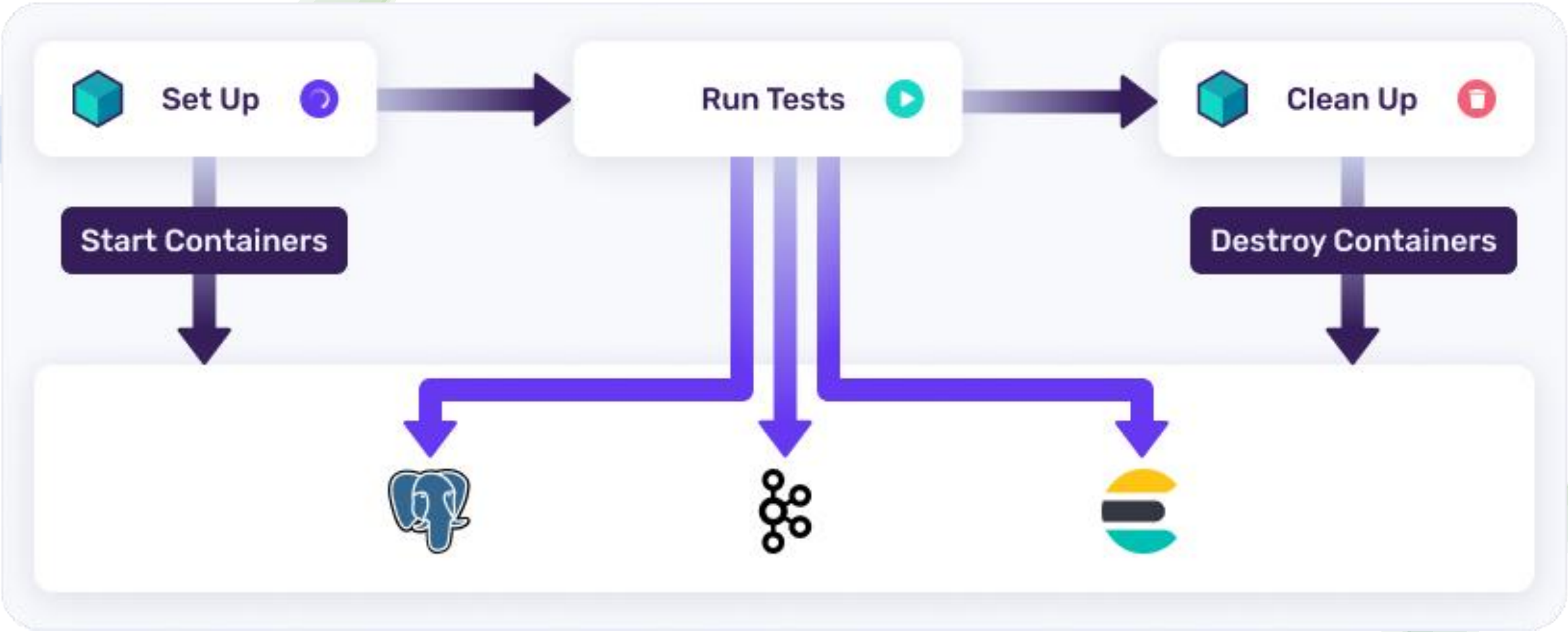




Lightweight, throwaway containers



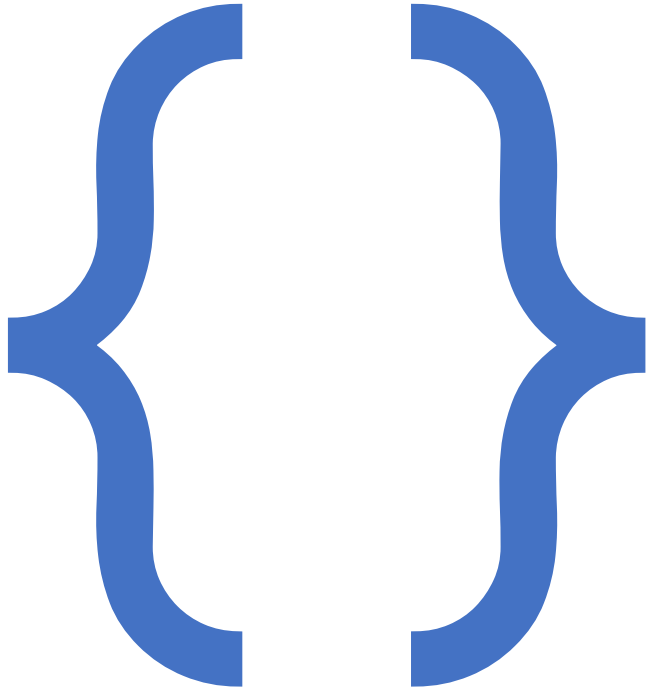



Set Up
Run Tests
Clean Up
Start Containers
Destroy Containers

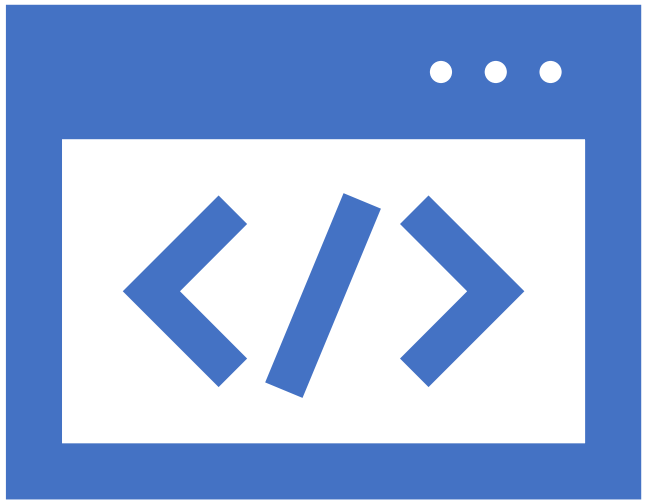
```
private readonly PostgreSqlContainer _container = new PostgreSqlBuilder().Build();
```

```
await _container.StartAsync();
```

```
await _container.DisposeAsync();
```

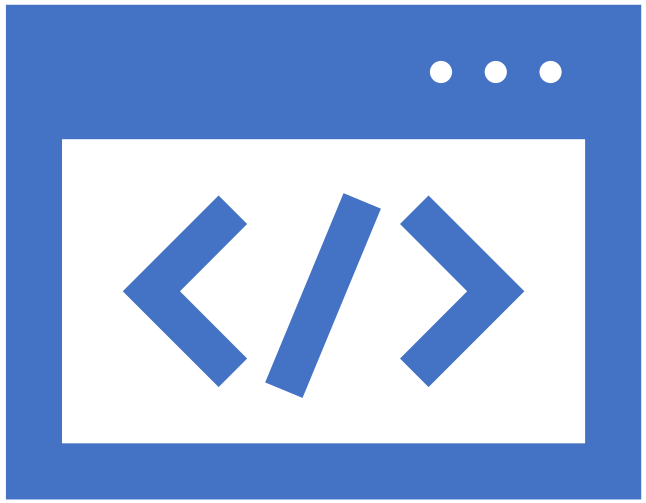


Coding Demo



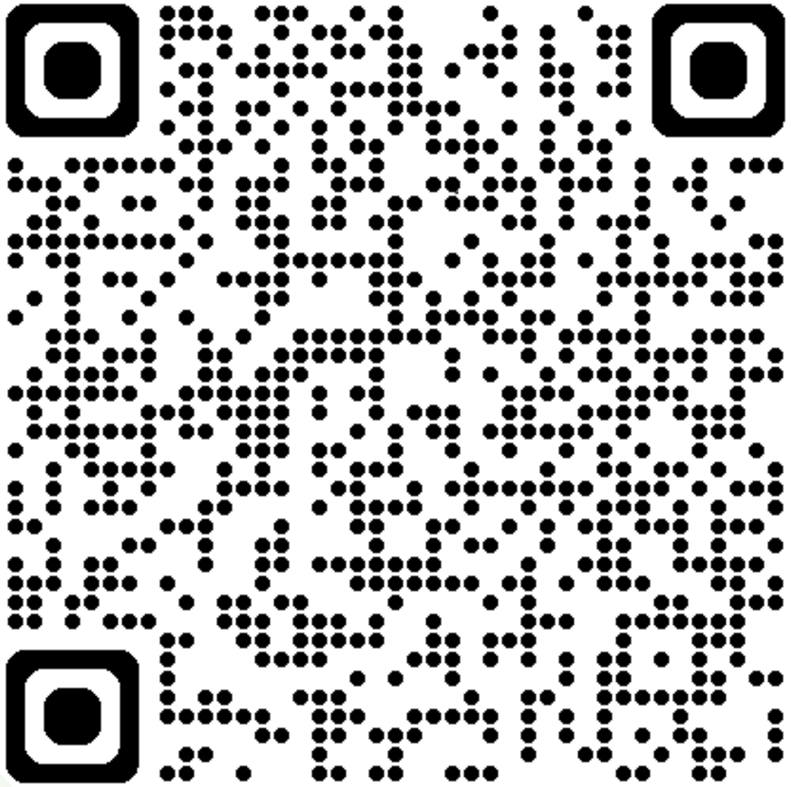
Honorable Mentions

- Reusable containers
- Podman



Recap

- Traditional options for integration testing
- Testcontainers
- Coding demo
- Respawn



Thank You

- 🌐 daninacan.com
- 𝕏 @danielwarddev
- daniel-ward-dev