Creating Realistic Unit Tests with Testcontainers

Paul Balogh, @javaducky

Testcontainers

Testcontainers

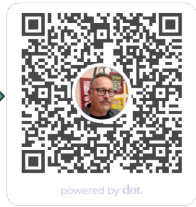
Disclaimer

Paul is **not** a part of Testcontainers nor Docker.

This is a "***first look***" presentation.

He is simply interested in improving reliability with **Continuous Delivery**.

Forward complaints to



Testcontainers 

Act I

1. **What's with unit tests?**
2. **@BeforeAll** - Overview of Testcontainers
3. **@Test** - Real-life applications, i.e. "the demo"
4. **@AfterAll** - What did we learn?

Testcontainers 

Difficulties

Unit testing can be difficult and controversial

- What coverage % is appropriate?
- Tests should be idempotent; requires that dependencies should not be modified
- Mocks can be brittle and unrealistic
- Can provide false sense of security

Testcontainers

Sorry...I don't have **the** answer

But, there is help!

Act II

Testcontainers 

1. What's with unit tests?
2. **@BeforeAll** - Overview of Testcontainers
3. **@Test** - Real-life applications, i.e. "the demo"
4. **@AfterAll** - What did we learn?

Testcontainers

“”

an **open source** framework for providing **throwaway**, lightweight instances of **databases**, [...], or just about anything that can run in a Docker container

Testcontainers

Overview

- Test **dependencies as code**

  No surprises, configured within your source code!

- Support for **many languages**

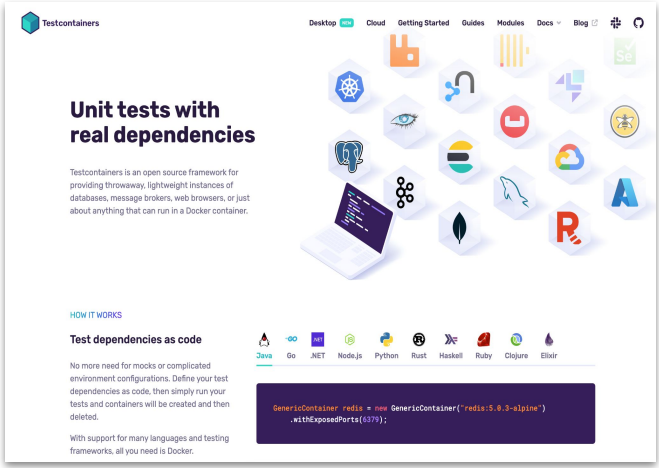
  Besides Go, can be used with Java, .NET, Python, Rust, Ruby, and more.

- **Test anything** you can containerize

  Over 50 modules available for databases, message brokers, and more.

- **“Dev-first”** integration testing

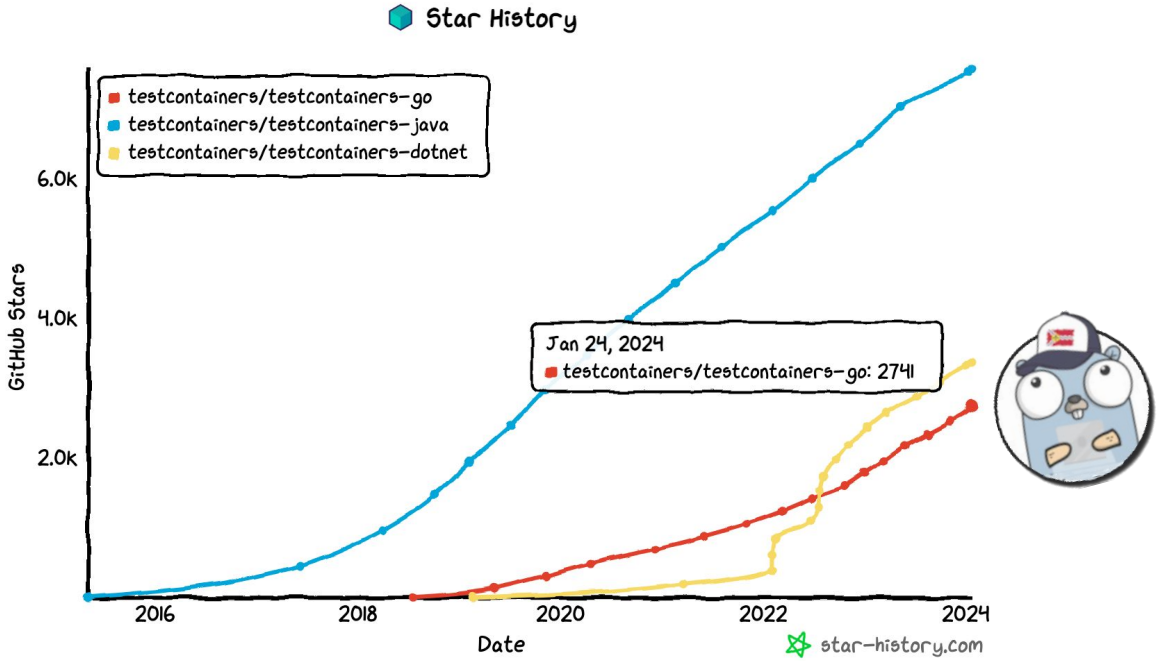
  Shift testing to the left and find issues earlier.



https://testcontainers.com/

Testcontainers

Overview



Supporting Go since 2018!

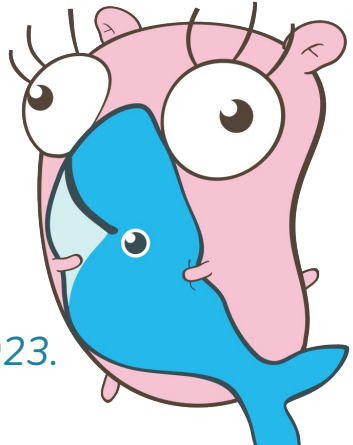
Testcontainers

Overview

Over 50 modules available...

Overview

ATOMICJAR >> docker



AtomicJar becomes part of Docker in December, 2023.

Testcontainers

Usage

1. **Specify** your target in code; no surprises
2. **Define** what it means to be ready; flexible checks
3. **Configure** your client
4. **Test** your integration

```
// Create and start the container
container, err := testcontainers.GenericContainer(
    ctx,
    testcontainers.GenericContainerRequest{
        ContainerRequest: testcontainers.ContainerRequest{
            Image:        "redis:5.0.3-alpine",
            ExposedPorts: []string{"6379/tcp"},
            WaitingFor:   wait.ForLog("Ready to accept connections"),
        },
        Started:          true,
    }
)

// Get your connection string for your client
connStr, err := container.ConnectionString(ctx, "sslmode=disable")
if err != nil {
    t.Fatal(err)
}

// Create your client
rdb := redis.NewClient(&redis.Options{
    Addr:     connStr,
    Password: "",
    DB:       0,
})
```

Act III

Testcontainers 

1. What's with unit tests?
2. **@BeforeAll** - Overview of Testcontainers
3. **@Test - Real-life applications, "the demo"**
4. **@AfterAll** - What did we learn?

Testcontainers

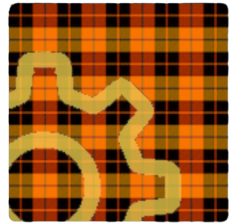
WeeSVC

the tiny microservice

Testcontainers

What is WeeSVC?

- An example **microservice** with RESTful APIs
- Backed by a relational **database**
- **Many** implementations; **same** API requirements
- Inspired by **TodoMVC** (https://todomvc.com/)
- **Compatibility** (*Contract*) testing with k6
- Started *way back* in **2019**!



Testcontainers

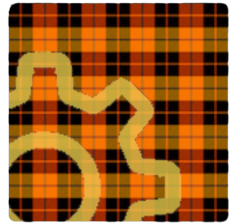
What is WeeSVC?

Highlights of the **Go implementation** include:

- **Gorilla Mux** for request routing
- **GORM** for database abstraction and migrations

  Admittedly, this is overkill.

- RDBMS support for **SQLite** and **Postgres**

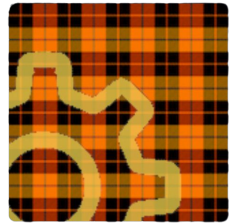


Testcontainers

What was added?

- Tests for **database** layer, `db/place.go`
  - Seed an ephemeral Postgres instance
  - Perform CRUD operations in isolation

  Multiple examples in example-testcontainer-usage branch!
- Test **compliance** of API, `api/place.go`
  - Multiple containers: Service and k6



Testcontainers

Code Review

https://github.com/weesvc/weesvc-gorilla

Testcontainers

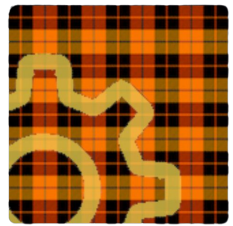
THIS SLIDE INTENTIONALLY LEFT BLANK

Testcontainers

What's next?

- Create a **profiling suite** for simulated load
- Automate creation of **comparative analyses**
- Add more **languages** and **frameworks**

- Contributors **welcome** and **appreciated**!



Act IV

Testcontainers 

1. What's with unit tests?
2. **@BeforeAll** - Overview of Testcontainers
3. **@Test** - Real-life applications, i.e. "the demo"
4. **@AfterAll** - What did we learn?

Testcontainers

What did we learn?

TDD principles influence system design to facilitate unit tests; this is **both** good and bad.

- **Good**: Encourages abstractions using patterns
- **Bad**: Excess abstraction can make code difficult to grok

*“Clear is **better** than clever.”*

Testcontainers

What did we learn?

Testcontainers **enable isolation** with real dependencies.

- Mocks can drift from reality, with issues creeping up in later stages of your development lifecycle
- Testing with real dependencies enables earlier detection of incompatibilities and conflicts

“Don’t panic.” *- Different context, I know, but you won’t break things!*

Testcontainers

What did we learn?

Testcontainers provide **consistent** experience.

- Executes the same locally as within CICD pipeline
- Test resources are automatically cleaned up
- Wait strategies ensure dependencies are available

Sweet.

Testcontainers

Evaluation

Based upon Paul's *very* subjective scoring...

Testcontainers

Evaluation

- Easy to get started
- Fast; no significant overhead*
- Flexible options for usage

- **YES...**you should absolutely use this!

Thank you!

Connect with Paul as
@javaducky or **linkedin/in/pabalogh**

https://github.com/weesvc/weesvc-gorilla

Testcontainers