Integrating TDD effortlessly in Go projects

Joana Roque Durão

Our Situation

Majority of tests are service tests orchestrated with docker compose

- Container running the service to test + Container running the tests + dependencies (databases, message queues, etc)
- Code to test is triggered using external APIs
- Test assertions done inspecting directly dependencies

Problems

- Tests took ages to finish
- Difficult to run
  - Required scripts for additional configuration
  - Running from IDE only if the dependencies were spin up separately
- Difficult to maintain and understand

We wanted

- Reduce the complexity
- Reduce maintenance and development time
- Run and debug tests easily from the IDE
- Write tests before writing the code to be tested

Our Path to TDD friendly tests

Step 1 - Tests calling the code directly

- Refactored tests to instantiate dependencies and call their methods directly
- Removed the service image from the docker compose

Our Path to TDD friendly tests

Step 2 - Running tests in parallel

- Remove the need for docker compose and start dependencies per test
- Refactor the tests to remove the JUnit suite style tests and use a go idiomatic style

Testcontainers-go

- https://golang.testcontainers.org/
- Manages the creation and the cleanup of containerized dependencies

Tips

- Don’t start by using it with your docker-compose file
- Setup container for dependencies in the beginning of each test
    - Explicit cleanup is not needed, ryuk takes care of it
- Ports for the container map to a random port on the machine
    - Setup drivers/connectors with the port (or endpoint) from the package API

After testcontainers

- Removed the scripts to setup the dependencies
- We could finally run and debug tests via the IDE
- For many dependencies, the tests could still take a bit to run

Mocks are still our best friends 😁

- Reduce the usage of containerized dependencies to save build time
- Don’t use containers for dependencies that don’t have a complex behaviour if you don’t need to test it
  - Message queues, communication mechanisms or simply behaviours that don’t need to be tested

Thank you