🧪 Hands off deployments in Kotlin

Mat Johnson

You have raised a pull request

How many people do you need to deploy a change to production?

You have raised a pull request

How many people do you need to deploy a change to production?

2

Someone to approve your pull request and yourself 🤯

How?

The pipeline has the controls and ensures quality

How?

The pipeline has the controls and ensures quality **

** 🚁 Escape clause: Incubating features should be feature flagged off in production

How?

The technique to be shared is to utilise the acceptance criteria as automated tests to test the artefact, as a control to ensure artefact is the highest quality

Why?

Acceptance criteria is living requirements

Why?

Acceptance tests can be shared with Product, Sales and Managers!

Why?

Acceptance tests help to build the right thing!

Why?

Acceptance tests allow for rapid change by ensuring that key user journeys are tested

Acceptance Test Anatomy

- Microservice under test
- Supporting services like databases & kafka represented as docker containers
- Cumbersome services mocked using wiremock docker container
- Set of tests written in BDD style

Acceptance Test Overview

- Send requests as if consumer of service under test
- Assert that the output meet expected customer outcomes
- Supporting services are containers

Why Behaviour Driven Design (BDD)?

Problem

- You need to make a change
- You open a test
- It is over 1,000 lines long
- Test methods have no naming standard
- Technical tests

Why Behaviour Driven Design (BDD)?

- Can be written in JUnit5 or Cucumber
- Abstracts nuance and complexity
- Business focused

How big is a Acceptance Test?

- Bigger than an Unit test
- Bigger than an Integration test

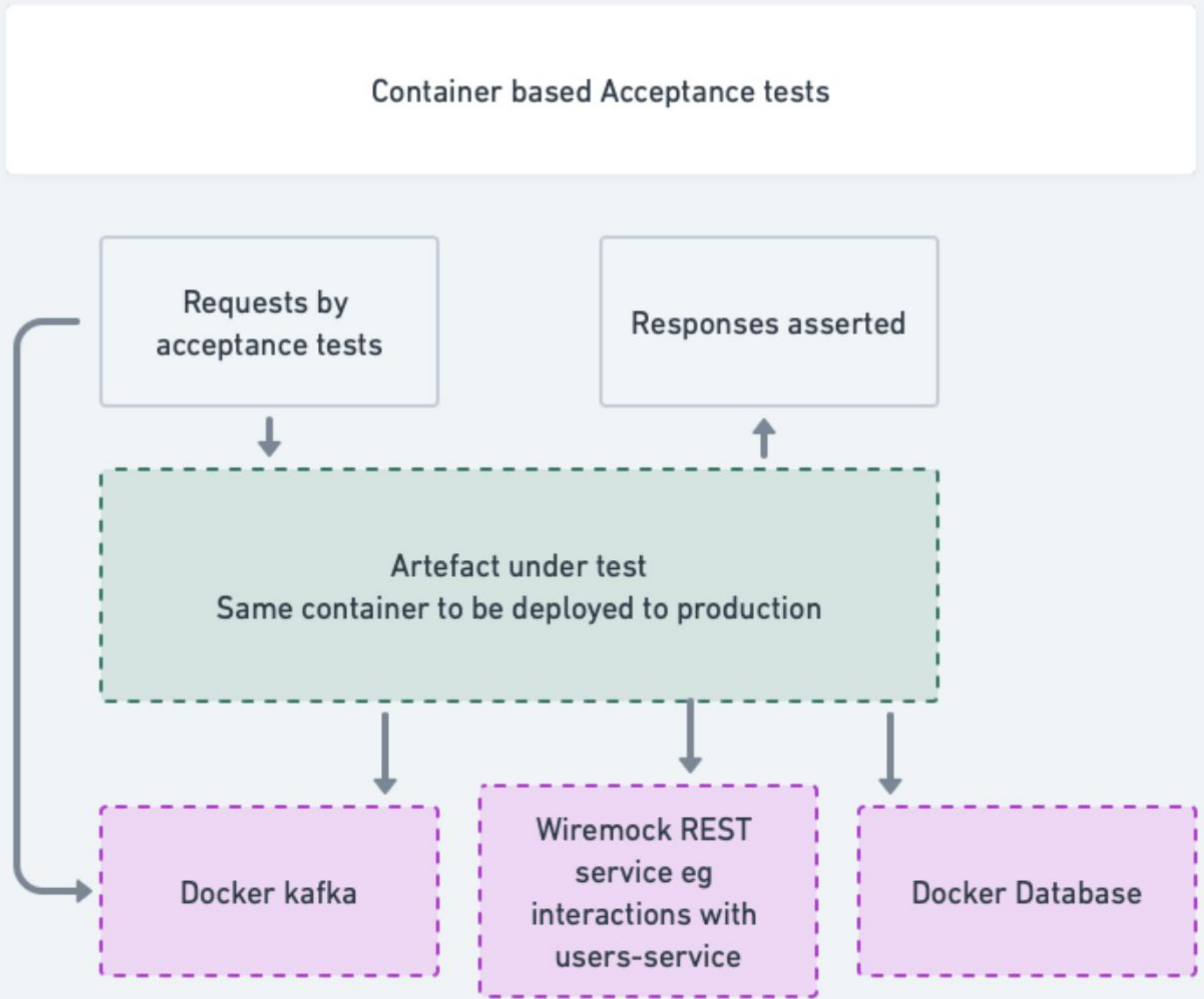
How big is a Acceptance Test?

- Smaller than an End to End test
- Less frail than an End to End test
- More focused on the artefact under test that E2E

Acceptance Test Variants

- Container based Acceptance Test
- Evolved integration test - Acceptance Test
- Just the acceptance criteria - Acceptance Test

1st Variant Container Based Acceptance Test


Container based Acceptance tests
Requests by acceptance tests
Responses asserted
Artefact under test
Same container to be deployed to production
Docker kafka
Wiremock REST service eg interactions with users-service
Docker Database

Container Based Acceptance Test

Advantages

- Exactly same artefact tested across all environments (killer feature).
- Makes library upgrades trivial. Including and especially library upgrades to deal mitigate security vulnerabilities (killer feature).

Container Based Acceptance Test

Disadvantages

- Mocking is more difficult and complex
- You will need to deal with security (JWT auth, encryption, etc)
- Can be difficult to run in CI due to docker-in-docker complexity
- Will need to pull or produce container locally for building the tests

Coding Demo Part 1a

Container based Acceptance test

Testcontainers

https://github.com/mathewdj/paper-scissors-ROCK-acceptance-tests

Coding Demo Part 1b

Container based Acceptance test

Cucumber and Kotlin

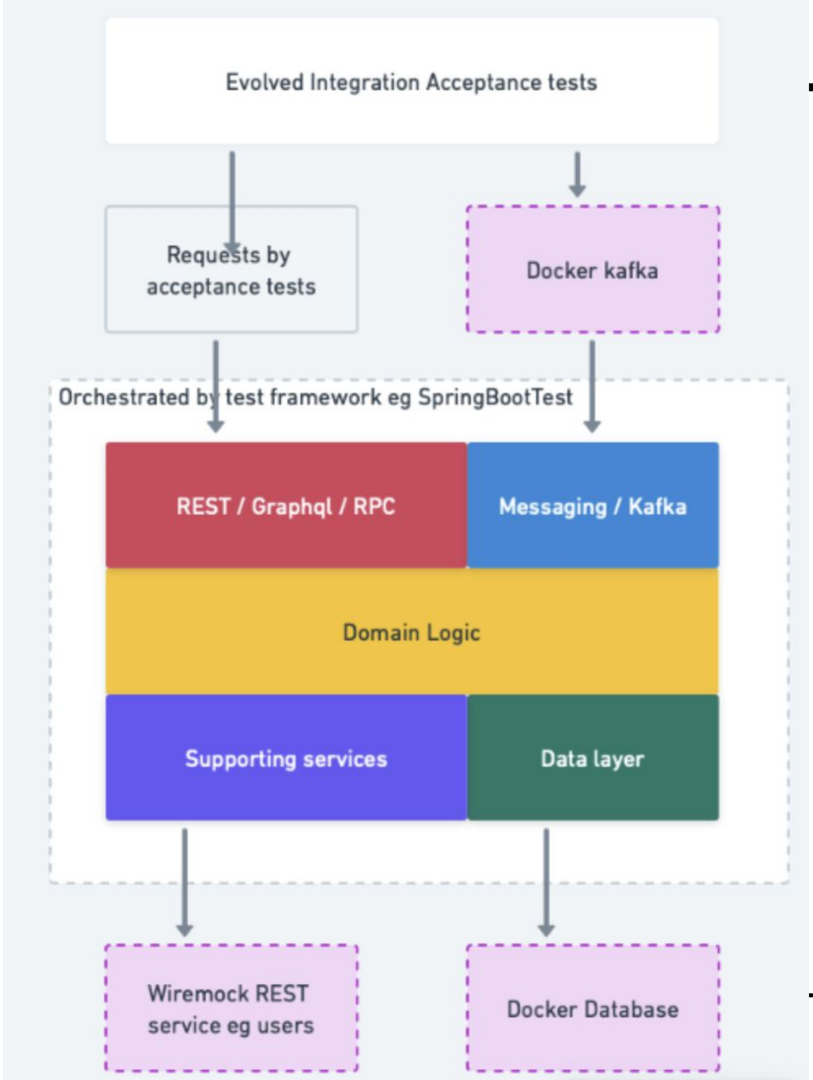
Use the same container that is going to be deployed to production

https://github.com/mathewdj/paper-scissors-ROCK-acceptance-tests

Segue: Why Test Containers?

- Faster to start than docker compose
- Better orchestration and service availability strategies

2nd Variant The Evolved Integration Test - Acceptance Test


Evolved Integration Acceptance tests
Requests by acceptance tests
Docker kafka
Orchestrated by test framework eg SpringBootTest
REST / Graphql / RPC
Messaging / Kafka
Domain Logic
Supporting services
Data layer
Wiremock REST service eg users
Docker Database

The Evolved Integration Test - Acceptance Test

Advantages

- Easy to mock external services
- Easy to run in CI
- Quick to write
- User stories are updated as the artefact changes 🤯
- Mocking & spying is easy

The Evolved Integration Test - Acceptance Test

Disadvantages

- Risk: artefact might start in integration tests but might not start in a real environment, due to test mocking
- Risk: Security related ingress might be skipped
- Not testing actual deployed artefact ie docker container

Coding Demo Part 2

The Evolved Integration Test - Acceptance Test

JUnit5 BDD Test written in Kotlin

https://github.com/mathewdj/paper-scissors-ROCK-acceptance-tests

Bonus tip: If you want to start using Kotlin, start writing your tests in Kotlin

3rd variant - Just the acceptance criteria

- Small step to automated acceptance tests
- Acceptance criteria lives with the code 🤯
- Mark acceptance tests @under-development to be ignored while feature is being developed

Horizon

- Run acceptance tests against a real environment
- Run acceptance tests against many environments (dev, staging and even production 🤯)

Summary

- Acceptance tests are a control to ensure the system does exactly what is meant to do
- Treat any acceptance criteria like gold
- Learned three different ways of doing acceptance tests
    - Evolved integration test
    - Container based
    - Just the acceptance criteria

Summary

- Write some automated acceptance tests
- Run in CI
- Change code aggressively
- Reduce toil

Questions