AARHUS UNIVERSITET

Microservices and DevOps

DevOps and Container Technology

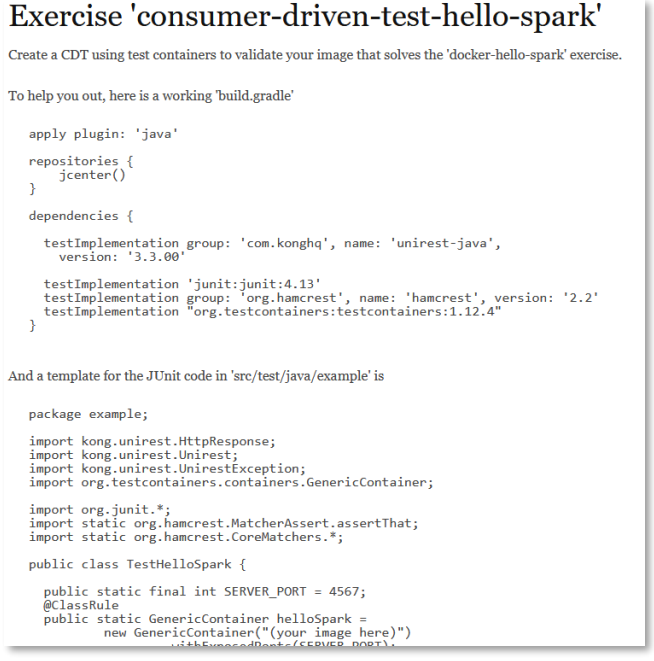
TestContainers Exercise

Henrik Bærbak Christensen

AARHUS UNIVERSITET

Exercise 1

- Create a CDT for your ‘hello spark’ web server using test containers
  - See ‘consumer-driven-test-hello-spark’ on Mandatory iteration 3



Exercise 'consumer-driven-test-hello-spark'

Create a CDT using test containers to validate your image that solves the 'docker-hello-spark' exercise.

To help you out, here is a working 'build.gradle'

```
apply plugin: 'java'

repositories {
    jcenter()
}

dependencies {

  testImplementation group: 'com.konghq', name: 'unirest-java',
    version: '3.3.00'

  testImplementation 'junit:junit:4.13'
  testImplementation group: 'org.hamcrest', name: 'hamcrest', version: '2.2'
  testImplementation "org.testcontainers:testcontainers:1.12.4"
}
```

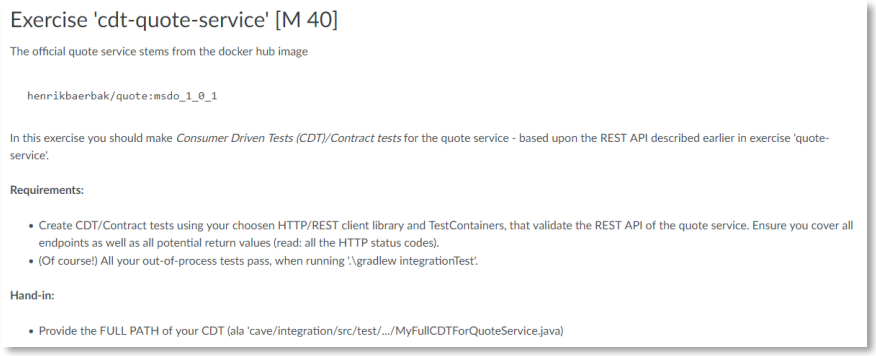
And a template for the JUnit code in 'src/test/java/example' is

```
package example;

import kong.unirest.HttpResponse;
import kong.unirest.Unirest;
import kong.unirest.UnirestException;
import org.testcontainers.containers.GenericContainer;

import org.junit.*;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.CoreMatchers.*;

public class TestHelloSpark {

  public static final int SERVER_PORT = 4567;
  @ClassRule
  public static GenericContainer helloSpark =
          new GenericContainer("(your image here)")
```

AARHUS UNIVERSITET

Exercise 2

- Solve the mandatory exercises
  - ‘cdt-quote-service’



Exercise 'cdt-quote-service' [M 40]

The official quote service stems from the docker hub image

```
henrikbaerbak/quote:msdo_1_0_1
```

In this exercise you should make *Consumer Driven Tests (CDT)/Contract tests* for the quote service - based upon the REST API described earlier in exercise 'quote-service'.

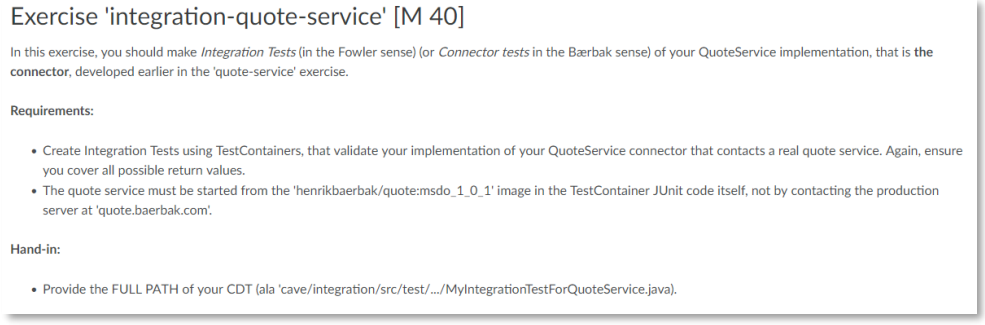
Requirements:

- Create CDT/Contract tests using your choosen HTTP/REST client library and TestContainers, that validate the REST API of the quote service. Ensure you cover all endpoints as well as all potential return values (read: all the HTTP status codes).
- (Of course!) All your out-of-process tests pass, when running '.\gradlew integrationTest'.

Hand-in:

- Provide the FULL PATH of your CDT (ala 'cave/integration/src/test/.../MyFullCDTForQuoteService.java)

  - ‘integration-test-quote-service’



Exercise 'integration-quote-service' [M 40]

In this exercise, you should make *Integration Tests* (in the Fowler sense) (or *Connector tests* in the Bærbak sense) of your QuoteService implementation, that is **the connector**, developed earlier in the 'quote-service' exercise.

Requirements:

- Create Integration Tests using TestContainers, that validate your implementation of your QuoteService connector that contacts a real quote service. Again, ensure you cover all possible return values.
- The quote service must be started from the 'henrikbaerbak/quote:msdo_1_0_1' image in the TestContainer JUnit code itself, not by contacting the production server at 'quote.baerbak.com'.

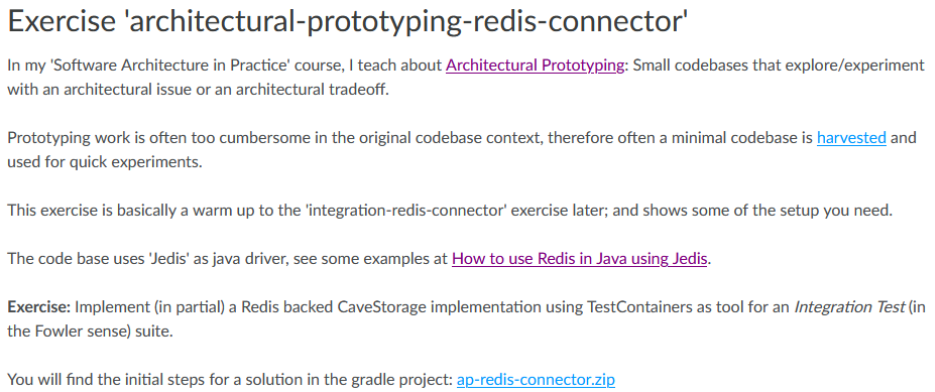
Hand-in:

- Provide the FULL PATH of your CDT (ala 'cave/integration/src/test/.../MyIntegrationTestForQuoteService.java).

AARHUS UNIVERSITET

Exercise 3

- *Begin* building an *Architectural Prototype* that explores implementing the CaveStorage interface using the NoSQL database ‘Redis’.
  - *Hum hum, actually we will talk Redis later today…*
- *Find a zip with starter code on mandatory iteration 3.*



Exercise 'architectural-prototyping-redis-connector'

In my 'Software Architecture in Practice' course, I teach about Architectural Prototyping: Small codebases that explore/experiment with an architectural issue or an architectural tradeoff.

Prototyping work is often too cumbersome in the original codebase context, therefore often a minimal codebase is harvested and used for quick experiments.

This exercise is basically a warm up to the 'integration-redis-connector' exercise later; and shows some of the setup you need.

The code base uses 'Jedis' as java driver, see some examples at How to use Redis in Java using Jedis.

Exercise: Implement (in partial) a Redis backed CaveStorage implementation using TestContainers as tool for an *Integration Test* (in the Fowler sense) suite.

You will find the initial steps for a solution in the gradle project: ap-redis-connector.zip