



tarent solutions GmbH

# Testcontainers: einfach Docker Container in Tests einbinden





Services migrieren - Tücken, Tricks und 3 Fallbeispiele

## Kurz zu mir ...

- Tim Steffens, **Softwareentwickler** und **Trainer** bei der  **tarent**
- Interessen
  - **Funktionale Programmierung**
  - **Qualität** von Software
  - **Wandern** & Klettern
- die **tarent**
  - **hilft Kunden** gute Software zu bauen
    - in-house oder vor Ort
    - Schulungen
  - baut **eigene Software** (z.B. snabble.io)
  - hat Standorte in **Bonn, Köln, Berlin & Bukarest**



Wer seid Ihr?

A photograph of a man sitting on a large, light-colored rock, facing away from the camera towards a horizon filled with dense, white clouds. The sky above the clouds is a warm, golden-yellow color, suggesting either sunrise or sunset. The foreground consists of dark, silhouetted vegetation and rocks.

# Agenda

- Docker
- Testcontainers
- Live Coding Session #1
- Selenium
- Live Coding Session #2

An aerial photograph showing a narrow, winding body of water, likely a river or a coastal inlet, curving through a landscape of dark green vegetation and patches of brown, sandy soil. The water has a textured appearance with white foam or spray along its edges.

*Unterbrecht mich!*

*Docker*





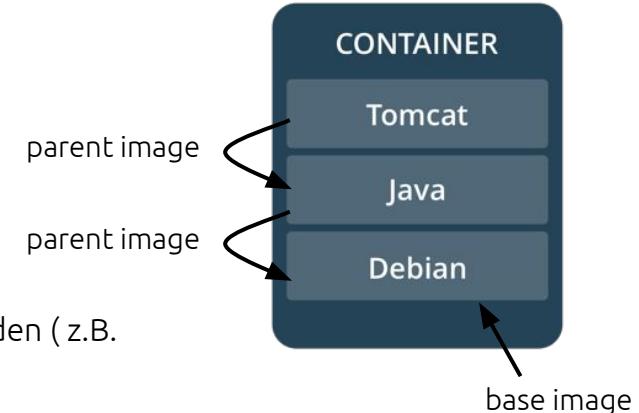
# Was ist ein Docker Container?

- ein **Linux Prozess** in einer isolierten Umgebung ausgeführt wird (kernel namespaces und cgroups)
- **Sandbox** mit definierten Ressourcen (**Dateisystem**, Netzwerk, ...)
  - "Sandbox ist die englischsprachige Bezeichnung für Sandkiste oder Sandkasten und bezeichnet allgemein einen isolierten Bereich, innerhalb dessen jede Maßnahme keine Auswirkung auf die äußere Umgebung hat." - Wikipedia
- benutzt den **Kernel** vom zugrunde liegenden OS
  - ist also **keine virtuelle Maschine**
- basiert auf einem **Docker Image**



# Was ist ein Docker Image?

- **Template** für Container
- schreibgeschütztes **Speicherabbild**
- **FS-Diff** zu dem **Parent Image**
- kann aus einer **Registry** geladen werden ( z.B. <https://hub.docker.com/> )
- **Identifikation** über **<repository>:<Tag>** , z.B.: postgres:11.5
  - Default für "Tag": latest
  - bei abweichender Registry:  
my-registry.com:4711/my-image:0.1

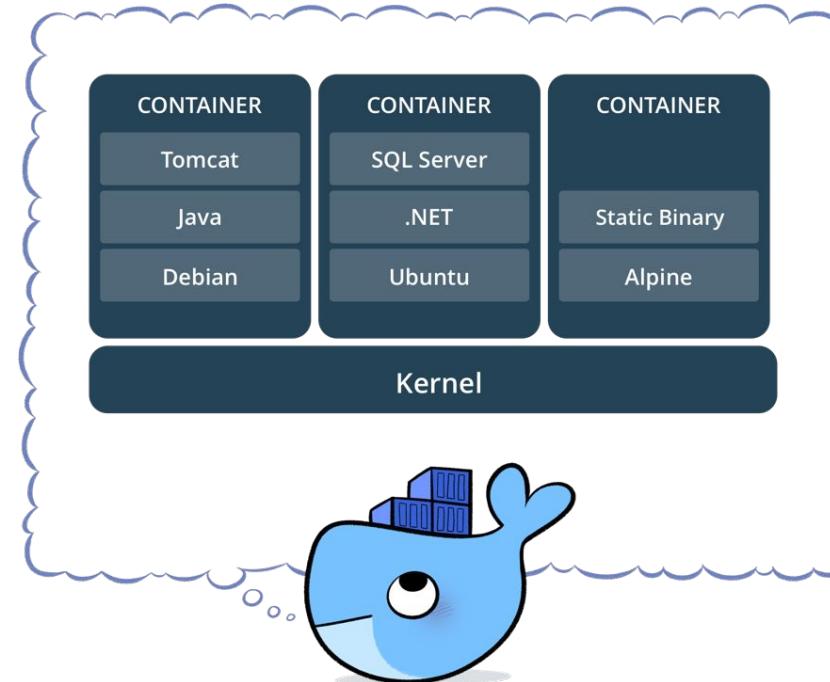




Docker



# Container & Images





# Docker Engine

- Sammlung von **Tools & Bibliotheken**
- u.a. **docker**-Kommando
- **Container verwalten**
  - starten / stoppen
  - auflisten
  - löschen
  - ...
- **Images verwalten**
  - downloaden
  - erstellen, auflisten, löschen
  - ...
- ...



Docker

# Container Starten



```
docker run -p 8080:80 httpd:2.4
```



Docker

# Container Starten



Port-Forwarding

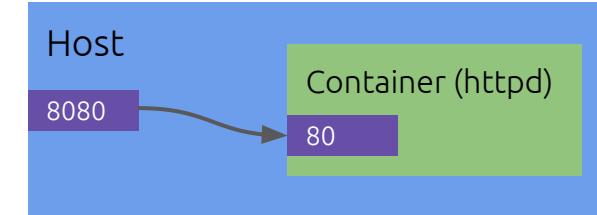
```
docker run -p 8080:80 httpd:2.4
```

Image



Docker

# Container Starten



Port-Forwarding

```
docker run -p 8080:80 httpd:2.4
```

Image



Docker



# Images bauen

- **Dockerfile** anlegen

```
FROM nginx
COPY index.html /usr/share/nginx/html/
```

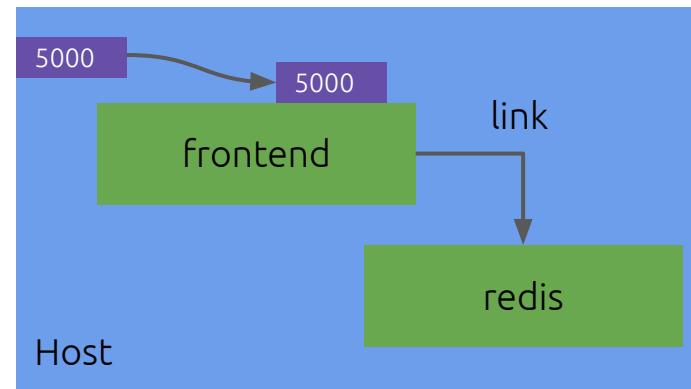
- **Image bauen** und Namen geben

```
docker build --pull -t my-nginx .
```



# Docker Compose

- Tool um eine **Multi-Container-Applikationen zu definieren und auszuführen**
- **YAML Datei** um Container zu **konfigurieren**
- **einzelnes Kommando** um **alle Container** der Konfiguration zu erstellen und zu **starten**



```
version: '3'
services:
  frontend:
    build: .
    ports:
      - "5000:5000"
    links:
      - redis
  redis:
    image: redis
```



Docker



# Installation

- docker engine: <https://docs.docker.com/install/>
- docker compose: <https://docs.docker.com/compose/install/>



# Warum Container?

- **einfaches Handling**
  - einzige Installations-Abhängigkeit zum Start eines Containers:  
**docker** - das System bleibt ansonsten unverändert!
  - für Entwicklung: leichtgewichtig z.B. unterschiedliche Datenbanken ausprobieren
- **portabel**
  - Java App: Java Version + ggf. JCE Policies
  - C++ App: libs & header files in richtiger Version
- Es ist einfach die **exakte Version** aus **PROD** zu bekommen
- **einfaches Deployment & einfache Skalierung** in der **Cloud**



# Warum Container für Tests?

- Direkt mit der **richtigen Version** testen
  - z.B. **DB**-Version aus **Prod** anstelle einer **In-Memory-Datenbank** testen (z.B. H2)
  - Verhalten kann sich stark unterscheiden!
- **Kompatibilität** z.B. mit **verschiedenen Datenbanken** lässt sich einfach prüfen
- **Selenium Grid** funktioniert **out of the box** - auf jedem Rechner!
- mit **docker-compose** lässt sich **lokal** einfach eine **komplett integrierte Umgebung** simulieren



An aerial photograph of a busy shipping port terminal. The scene is filled with numerous shipping containers stacked in tall, organized piles. The containers come in various colors, including white, blue, red, and orange. They are arranged in a grid-like pattern across the frame. In the center, there's a large, dark area that appears to be a loading or unloading zone. Several yellow and black industrial vehicles, likely cranes or trucks, are visible, some with their arms extended over the containers. The ground is a mix of dark asphalt and lighter-colored gravel or dirt paths. On the far left, there's a small building with a sign that reads "Q8". The overall impression is one of a well-organized and active industrial port environment.

Testcontainers



# Testcontainers

- Sammlung von Bibliotheken um **Docker Container einfach** aus einem Test **starten** zu können
- Verfügbar für verschiedene Sprachen / Plattformen:
  - **java**, go, js, python, scala etc.
  - <https://github.com/testcontainers/>
- **Open Source** (MIT License)
- am meisten verwendet: **testcontainers-java**
  - <https://github.com/testcontainers/testcontainers-java>
  - Dokumentation: <https://www.testcontainers.org/>
  - Einfache Einbindung in **JUnit** und **Spock**

An aerial photograph of a coastal area. The left side features a large expanse of green fields with small white specks, likely snow or flowers. A narrow, winding river or path cuts through the center-left. The right side shows a sandy beach and a rocky cliff face. The water is a light blue-green color.

*Spock*

*weder noch!*

*JUnit*



Testcontainers

# Testcontainers

- Unterstützte Container-Typen / Module
  - Docker Compose Module
  - Elasticsearch container
  - Kafka Containers
  - Mockserver Module
  - Nginx Module
  - RabbitMQ Module
  - **Webdriver Containers**
  - ...
  - **Generic Container**



## Testcontainers



# Testcontainers Einbinden

- Gradle:

```
testCompile "org.testcontainers:testcontainers:1.12.1"
```

- Maven

```
<dependency>
  <groupId>org.testcontainers</groupId>
  <artifactId>testcontainers</artifactId>
  <version>1.12.1</version>
  <scope>test</scope>
</dependency>
```

- neuste Version: <https://search.maven.org/search?q=a:testcontainers>

An aerial photograph showing a river or coastal area from above. The water is a vibrant green color, with numerous small white bubbles or foam scattered across its surface. A narrow, brown, sandy strip of land or a sandbar runs diagonally across the frame, separating the green water from a darker, more turbulent area. The overall texture is somewhat abstract due to the high angle and lighting.

Maven

weder noch!

Gradle



Testcontainers

# Mit Containern arbeiten

```
import org.testcontainers.containers.GenericContainer

GenericContainer apache =
    new GenericContainer("httpd:2.4")
        .withExposedPorts(80)

apache.start()

println("Accessible via
        http://localhost: ${apache.getMappedPort(80)} ")

System.in.newReader().readLine()

apache.stop()
```



Testcontainers

# Mit Containern arbeiten

```
import org.testcontainers.containers.GenericContainer

GenericContainer apache =
    new GenericContainer("httpd:2.4")
        .withExposedPorts(80)

apache.start()

println("Accessible via
        http://localhost: ${apache.getMappedPort(80)} ")

System.in.newReader().readLine()

apache.stop()
```

Ports werden automatisch auf  
freien Random-Port gemapped



Testcontainers

# Postgres Container

```
import org.testcontainers.containers.PostgreSQLContainer

PostgreSQLContainer postgres =
    new PostgreSQLContainer()

postgres.start()

println("Accessible via ${postgres.getJdbcUrl()}")

System.in.newReader().readLine()

postgres.stop()
```



Testcontainers

# Integration in JUnit 4

```
@ClassRule  
public static GenericContainer apache =  
    new GenericContainer("httpd:2.4")  
        .withExposedPorts(80);
```

- Start und Stop wird von Testcontainers mittels @ClassRule vorgenommen
- Der Ryuk Sidecar-Container stellt sicher, dass alle networks, volumes und evtl. nicht gestoppte Container entfernt werden:  
<https://github.com/testcontainers/moby-ryuk>



Testcontainers

# Integration in Spock

```
testCompile "org.testcontainers:spock:1.12.1"
```

```
--
```

```
@Testcontainers
class DatabaseTest extends Specification {

    @Shared
    GenericContainer apache =
        new GenericContainer("httpd:2.4")
            .withExposedPorts(80)
```

- `@Shared` gibt an, dass Container zwischen den einzelnen Tests nicht neu gestartet werden sollen

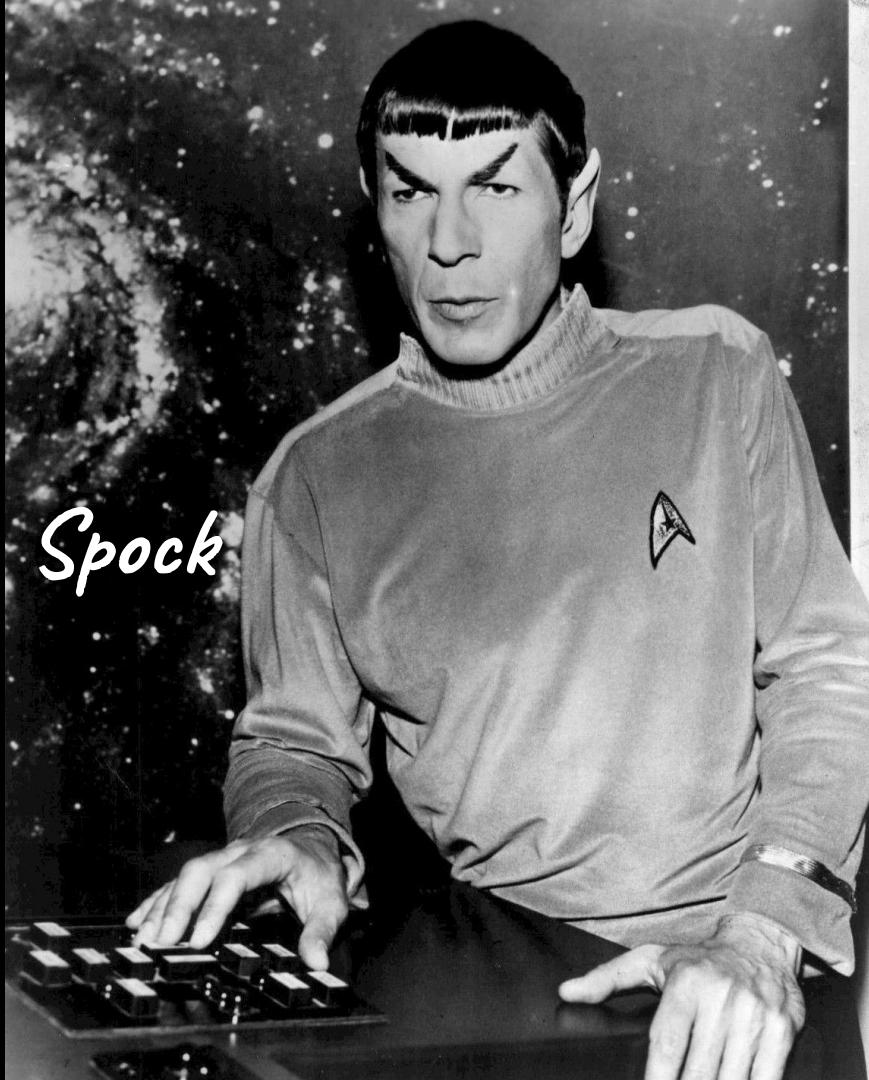


Testcontainers

# Integration in Spring Boot

in application.yml:

```
spring:  
  jpa.hibernate.ddl-auto : none  
  datasource:  
    driver-class-name :  
      org.testcontainers.jdbc.ContainerDatabaseDriver  
    url: jdbc:tc:mysql:5.7.22://localhost/ta_yourtrain  
    username: tarent  
    password: tarent
```



*Spock*



Spock

# Spock



*"Spock is a **testing** and specification **framework** for **Java** and **Groovy** applications. What makes it stand out from the crowd is its **beautiful** and **highly expressive** specification **language**. Thanks to its **JUnit runner**, Spock is compatible with most IDEs, build tools, and continuous integration servers."*

- <http://spockframework.org/spock/docs/1.3/introduction.html>

- basiert auf der JVM Sprache Groovy: <http://www.groovy-lang.org/>



Spock



## Groovy

- **Script-Sprache**, die dynamische und statische Typisierung zulässt
- läuft Java Virtual Machine (**JVM**)

```
def list = [1,2,3,4]
[1,2,3,4] == [1,2,3,4] // = true!!

def map = [name: "Arnold", role: "Agile Tester"]
map.name // = "Arnold"

class Employee { String name; String role }
def employee = new Employee(name: "Arnold", role: "Agile
Tester")
employee.name // = "Arnold"
```

An aerial photograph showing a river or coastal area from above. The water is a vibrant green color, with numerous small white bubbles or foam scattered across its surface. A narrow, light-colored strip of land or a sandbar runs diagonally across the frame, creating a winding path through the water. The land appears to be a mix of dark green vegetation and lighter brown soil or sand. The overall texture is somewhat grainy and organic.

Java

weder noch!

Groovy



Spock

# Beispiel



```
class ListSpec extends Specification {  
  
    def "Should be able to remove from list"() {  
        given:  
        def list = [1, 2, 3, 4]  
  
        when:  
        list.remove(0)  
  
        then:  
        list == [2, 3, 4]  
    }  
}
```



Spock



# Einbinden

- gradle

```
apply plugin: "groovy"  
implementation "org.codehaus.groovy:groovy-all:2.5.7"  
testCompile  
"org.spockframework:spock-core:1.3-groovy-2.5"
```

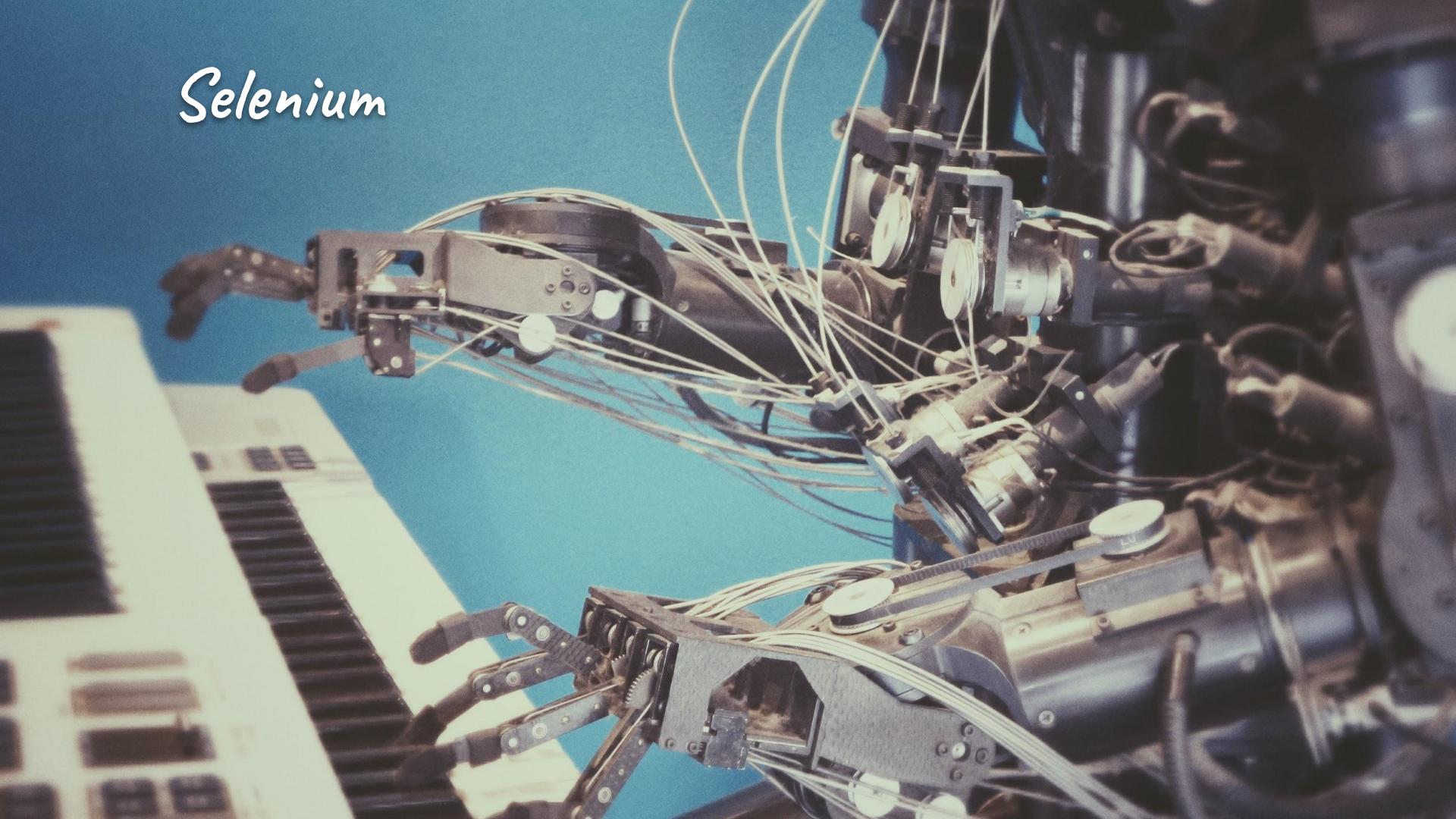
- maven

see <https://github.com/spockframework/spock-example/blob/master/pom.xml>



# *Live Coding Session #1*

Selenium



An aerial photograph of a coastal area. The left side shows a large expanse of green fields with small white specks, likely sheep. A narrow, winding river or stream flows from the center-left towards the bottom right, its path marked by brownish water and a lighter-colored, sandy bank. The right side of the image is dominated by a steep, rocky cliff face, with patches of green vegetation growing on its upper slopes.

Selenium

weder noch!

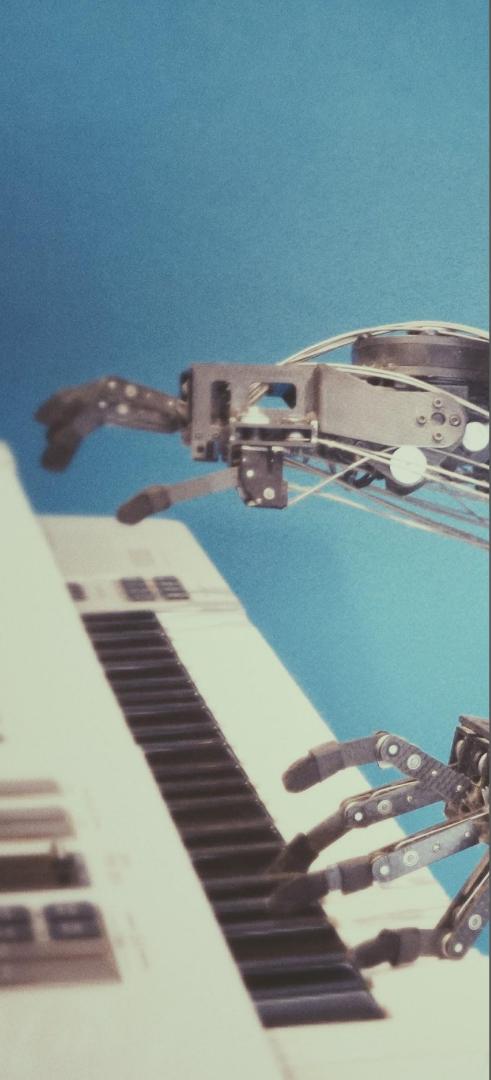
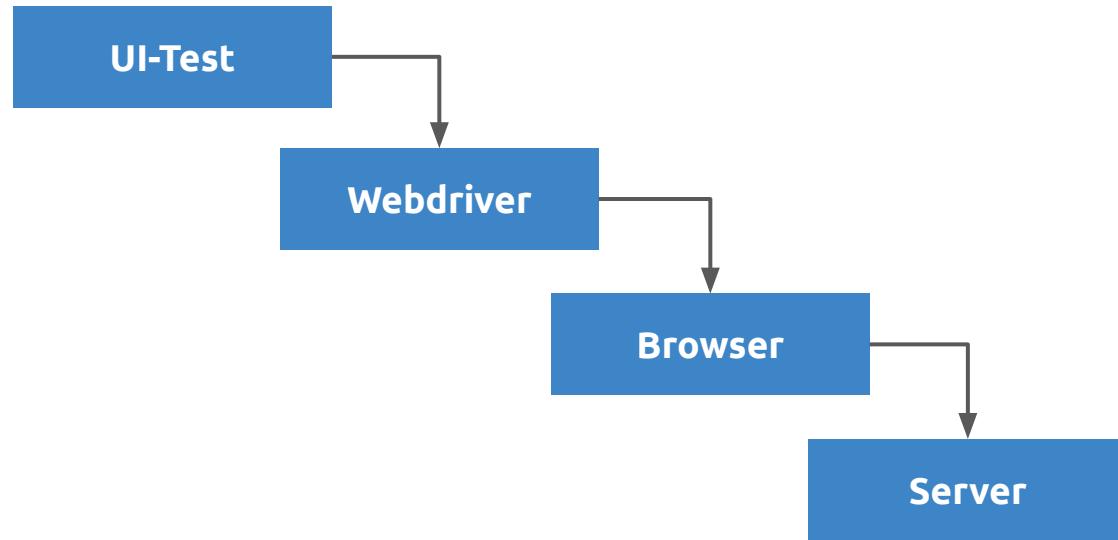
Geb



Selenium

# Was ist Selenium?

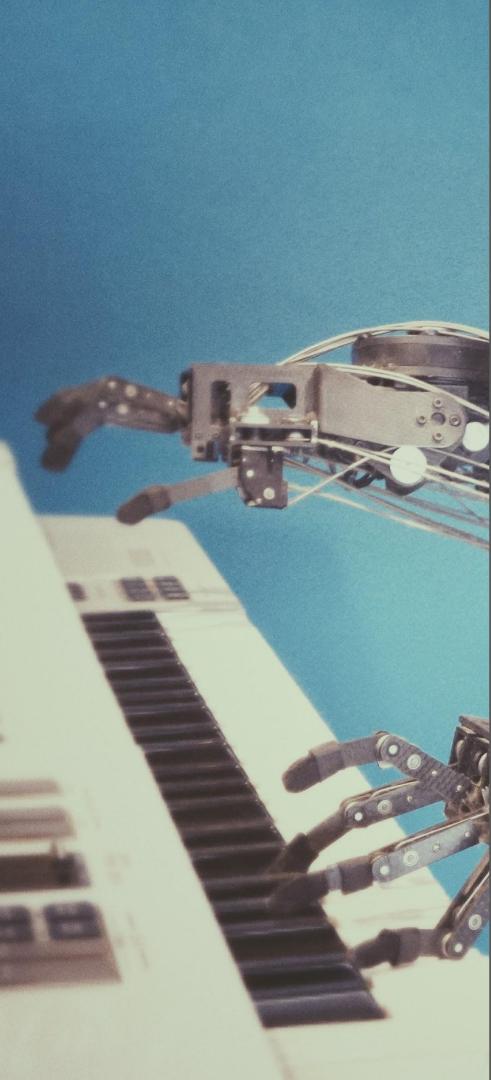
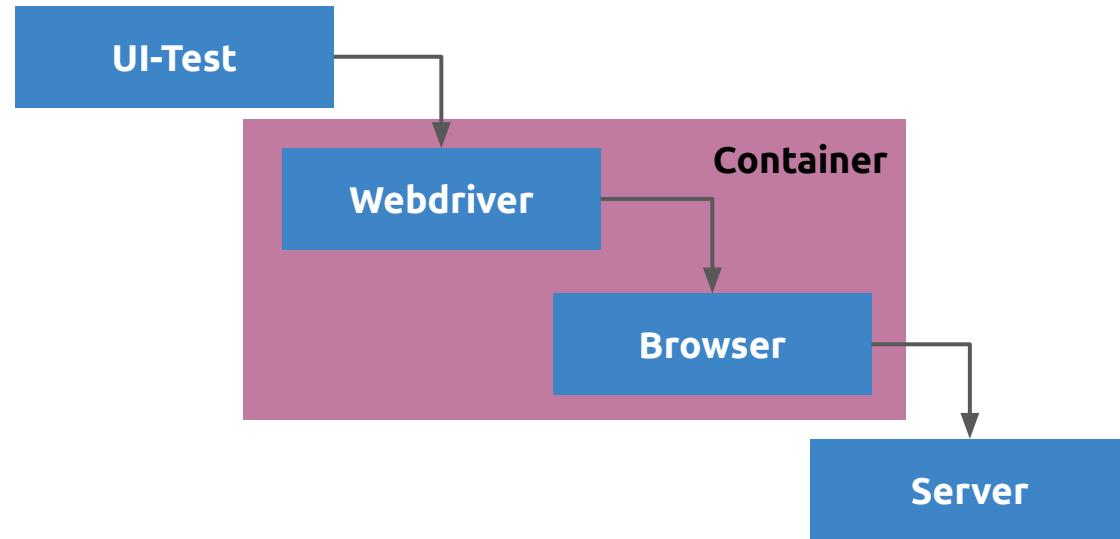
- *Selenium automates browsers. That's it! - <https://www.seleniumhq.org/>*





Selenium

# Selenium im Container





Selenium

# Selenium Webdriver Container

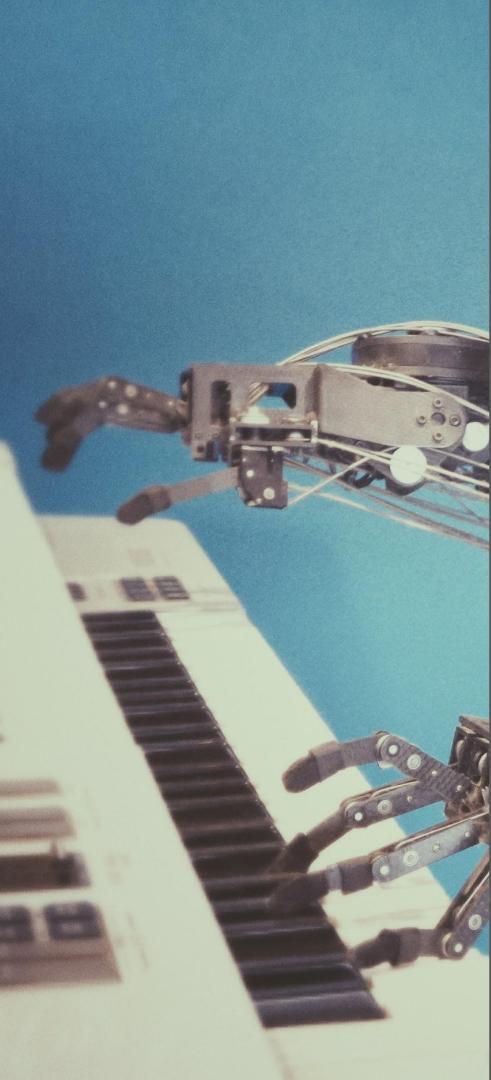
```
import org.openqa.selenium.remote.RemoteWebDriver
import org.testcontainers.containers.BrowserWebDriverContainer
import static org.testcontainers.containers.

    BrowserWebDriverContainer.VncRecordingMode. RECORD_ALL

BrowserWebDriverContainer chrome = new
    BrowserWebDriverContainer()
        .withCapabilities( new ChromeOptions())
        .withRecordingMode( RECORD_ALL, new File("./"))

chrome.start()

RemoteWebDriver driver = chrome.getWebDriver()
```



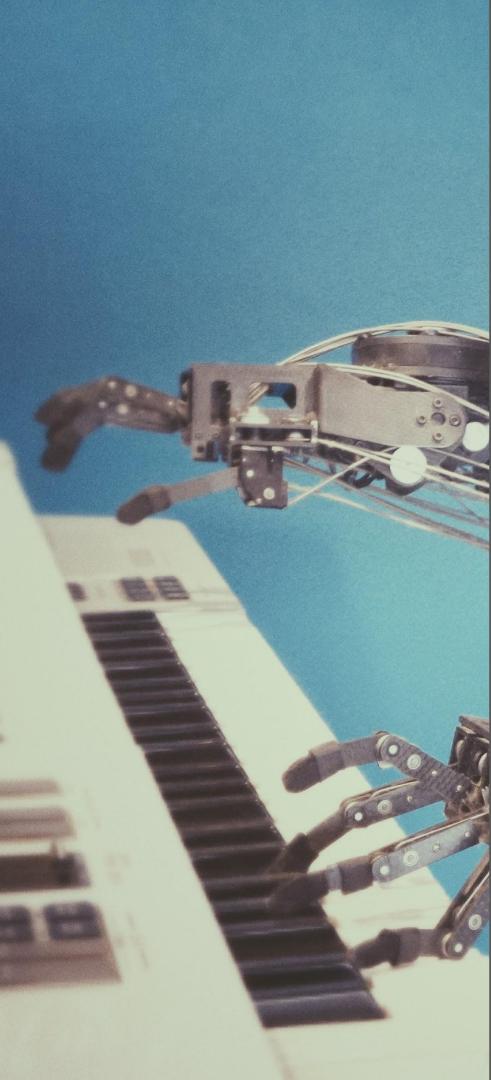


Selenium

# Geb

- <https://gebish.org>
- **DSL für UI-Tests**
- Setzt auf **Selenium** auf
- einfache und **schöne Syntax**

```
Browser.drive {  
    go "http://myapp.com/login"  
    assert $("h1").text() == "Please Login"  
    $("form.login").with {  
        username = "admin"  
        password = "password"  
        login().click()  
    }  
    assert $("h1").text() == "Admin Section"  
}
```





# *Live Coding Session #2*



# Wenn Ihr Hilfe braucht

- beim **automatisierten Testen**  
(Docker, Junit, Groovy/Spock, UI-Tests mit Selenium/Geb, Gatling, ...),
- einem **Software Projekt** im allgemeinen  
(Webdevelopment, Microservices, AI, Machine Learning, AR / VR)
- oder beim Aufbauen von **Wissen und Skills** in diesen Gebieten  
**(Training)**

**Sprecht mich gerne an!**



# Kotlin Schulung von Christian Schmidt @ JavaLand 2020

- JavaLand 2020
  - Konferenz rund um Java
  - 17. bis 19. März 2020
  - Phantasialand Brühl (bei Bonn)
  - <https://www.javaland.eu/>
- Kotlin – Eine praktische Einführung
  - Syntax & Konzepte
  - Übung: Webservice mit REST-API

# Vielen Dank!

**Tim Steffens**  
Softwareentwickler &  
Trainer  
[t.steffens@tarent.de](mailto:t.steffens@tarent.de)  
[@tmstff](https://github.com/tmstff)

Rochusstraße 2-4  
53123 Bonn  
  
Telefon: 0228 54881-0  
Telefax: 0228 54881-235  
  
[academy@tarent.de](mailto:academy@tarent.de)  
[www.tarent.de/academy](http://www.tarent.de/academy)



A black dog with a dark, shaggy coat is sitting on a weathered wooden fence. Its mouth is open, showing its tongue and teeth, giving it a happy or panting expression. The background is a soft-focus outdoor setting with green trees and foliage.

*Backup*



Docker

# Tip: regelmäßig aufräumen

```
$ docker system prune
```

WARNING! This will remove:

- all stopped containers
- all networks not used by at least one container
- all dangling images
- all build cache

Are you sure you want to **continue?** [y/N] **y**



Docker

# Tip: regelmäßig aufräumen



```
$ docker system prune -a --volumes
```

WARNING! This will remove:

- all stopped containers
- all networks not used by at least one container
- all volumes not used by at least one container
- all images without at least one container associated to them
- all build cache

Are you sure you want to **continue**? [y/N] y



Testcontainers



# Zugriff Container Logs

- `final String logs = container.getLogs();`

- Oder zu einem SLF4J logger schicken:

```
container.followOutput(new  
    Slf4jLogConsumer(LOGGER));
```

A photograph of a large pyramid, likely the Great Pyramid of Giza, under a blue sky with scattered white clouds. The pyramid's surface is made of many stone blocks. In the foreground, there is a sandy ground with some people and animals, including camels and horses. The word "Testpyramide" is overlaid in the center of the image in a white, cursive font.

*Testpyramide*



e2e

A large pyramid divided into four horizontal sections. The top section is orange and labeled "e2e". The second section from the top is purple and labeled "Integration". The third section is blue and labeled "Component". The bottom section is green and labeled "Unit". The pyramid is set against a background of a cloudy sky and a few people and animals at the base.

Integration

Component

Unit

→ [\[fowler-testpyramid\]](#)



- Test vom **Gesamtsystem**, inkl. aller externen Systeme
- häufig nur manuell möglich



## Beispiele

- Navigationssoftware in echtem Auto
- Online-Bestellung inkl. Lieferung



- Test vom **Gesamtsystem**, inkl. aller externen Systeme
  - häufig nur manuell möglich
- 
- Test von **Teilsystem**
  - Zusammenspiel der Services wird getestet



## Beispiele

- Navigationssoftware in echtem Auto
- Online-Bestellung inkl. Lieferung
- automatisierter UI-Test für Webshop mit Happy-Path für Einkauf



- Test vom **Gesamtsystem**, inkl. aller externen Systeme
- häufig nur manuell möglich
- Test von **Teilsystem**
- Zusammenspiel der Services wird getestet
- Test von **Service**
- Blackbox Test
- Alle Umsysteme gemockt



## Beispiele

- Navigationssoftware in echtem Auto
- Online-Bestellung inkl. Lieferung
- automatisierter UI-Test für Webshop mit Happy-Path für Einkauf
- Im User-Service werden per Rest neue Benutzer angelegt und wieder gelöscht



- Test vom **Gesamtsystem**, inkl. aller externen Systeme
- häufig nur manuell möglich
- Test von **Teilsystem**
- Zusammenspiel der Services wird getestet
- Test von **Service**
- Blackbox Test
- Alle Umsysteme gemockt
- Isolierter Test einer **Klasse**
- Abhängigkeiten werden gemockt

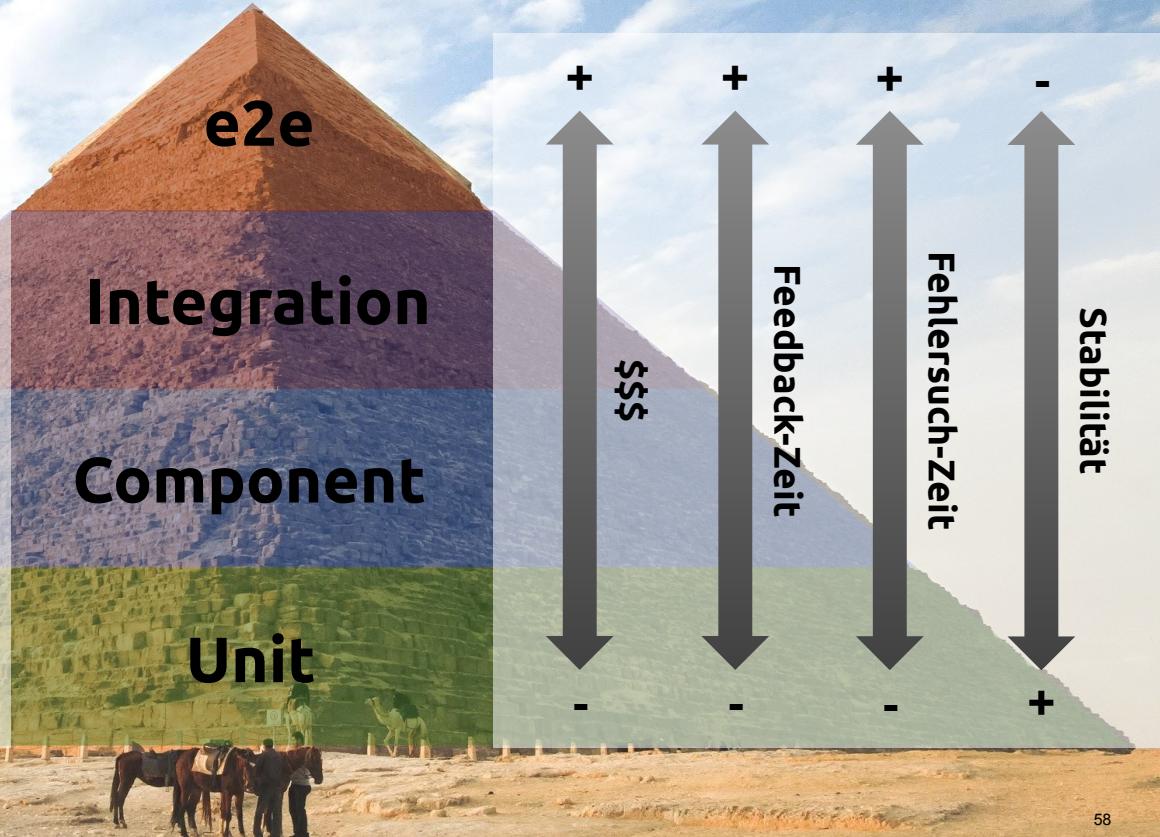


## Beispiele

- Navigationssoftware in echtem Auto
- Online-Bestellung inkl. Lieferung
- automatisierter UI-Test für Webshop mit Happy-Path für Einkauf
- Im User-Service werden per Rest neue Benutzer angelegt und wieder gelöscht
- Funktionieren alle Berechnungen des CurrencyConverters?
- funktioniert `asList()` ?



- Test vom **Gesamtsystem**, inkl. aller externen Systeme
- häufig nur manuell möglich
- Test von **Teilsystem**
- Zusammenspiel der Services wird getestet
- Test von **Service**
- Blackbox Test
- Alle Umsysteme gemockt
- Isolierter Test einer **Klasse**
- Abhängigkeiten werden gemockt





# Für diesen Workshop besonders relevant

