

Agile Testing @DUS, 22.11.2018

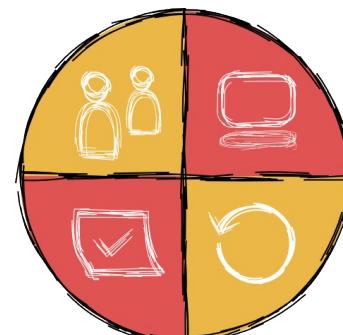
Testen von und mit Infrastruktur
“Integration Testing Done Right” ☺

Sandra Parsick

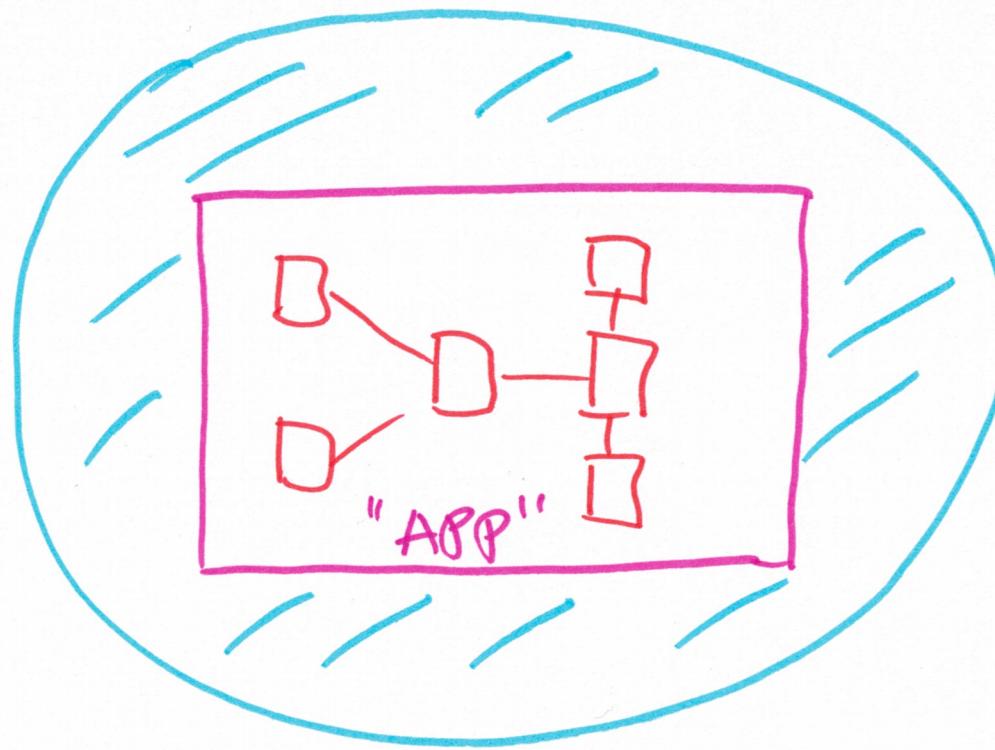
mail@sandra-parsick.de
@SandraParsick

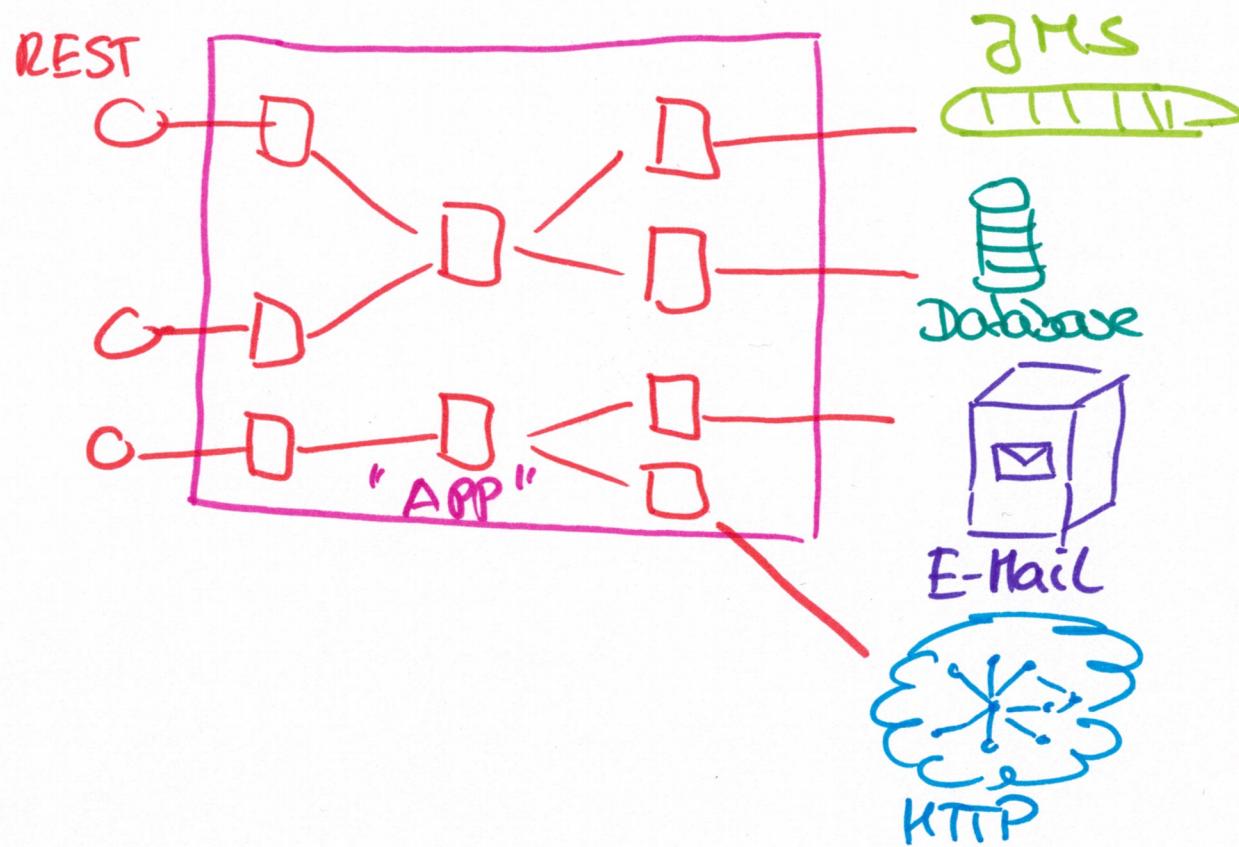
Zu meiner Person

- Sandra Parsick
- Freiberuflicher Softwareentwickler und Consultant im Java-Umfeld
- Schwerpunkte:
 - Java Enterprise Anwendungen
 - Agile Methoden
 - Software Craftmanship
 - Automatisierung von Entwicklungsprozessen
- Trainings
- Workshops
- Softwerkskammer Ruhrgebiet
- Twitter: @SandraParsick
- Blog:
<http://blog.sandra-parsick.de>
- E-Mail: mail@sandra-parsick.de

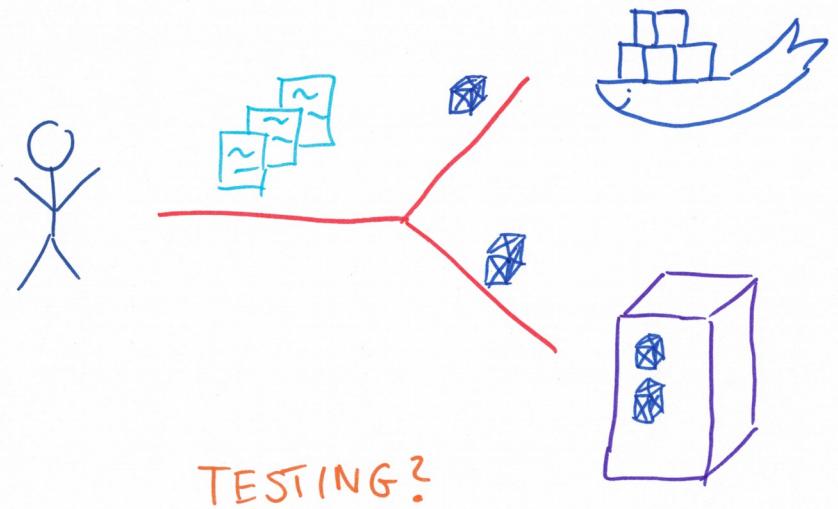
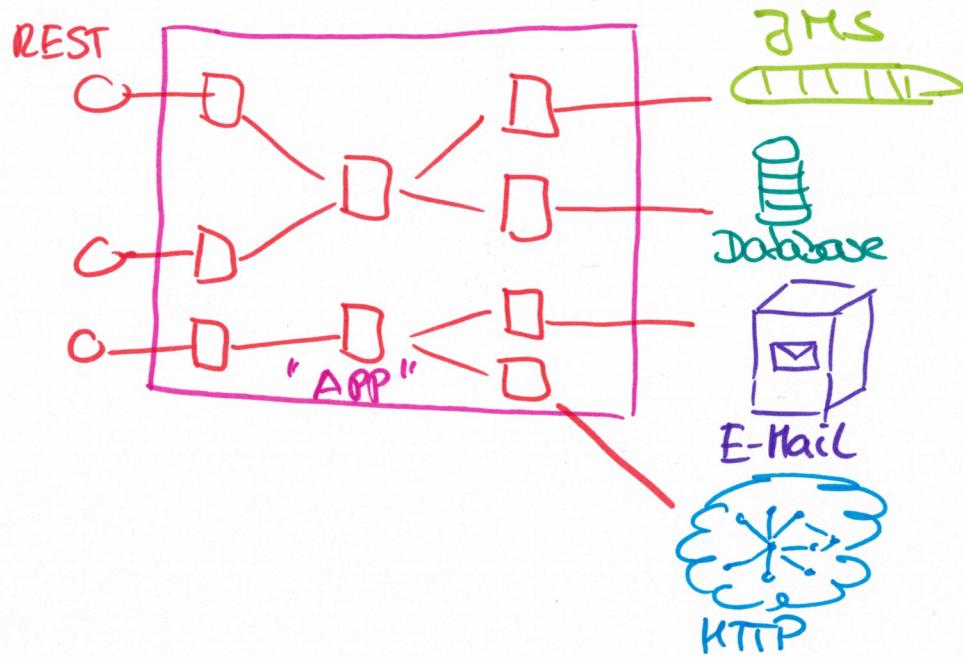


Motivation

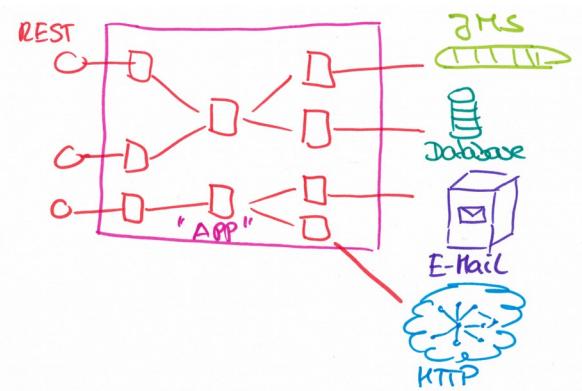








Testen mit Infrastruktur



Status Quo

- Interaktion mit Infrastruktur wird recht spät getestet
 - Feedback bei Fehlern recht spät
- Abhängig von einer bestimmten Infrastruktur
 - Manchmal auch schon beim Build (Bad smell)
 - False negative Fehlerquote recht hoch
 - Aufwändiges Setup
 - Testausführung langsam

Integration vs Integrated Tests

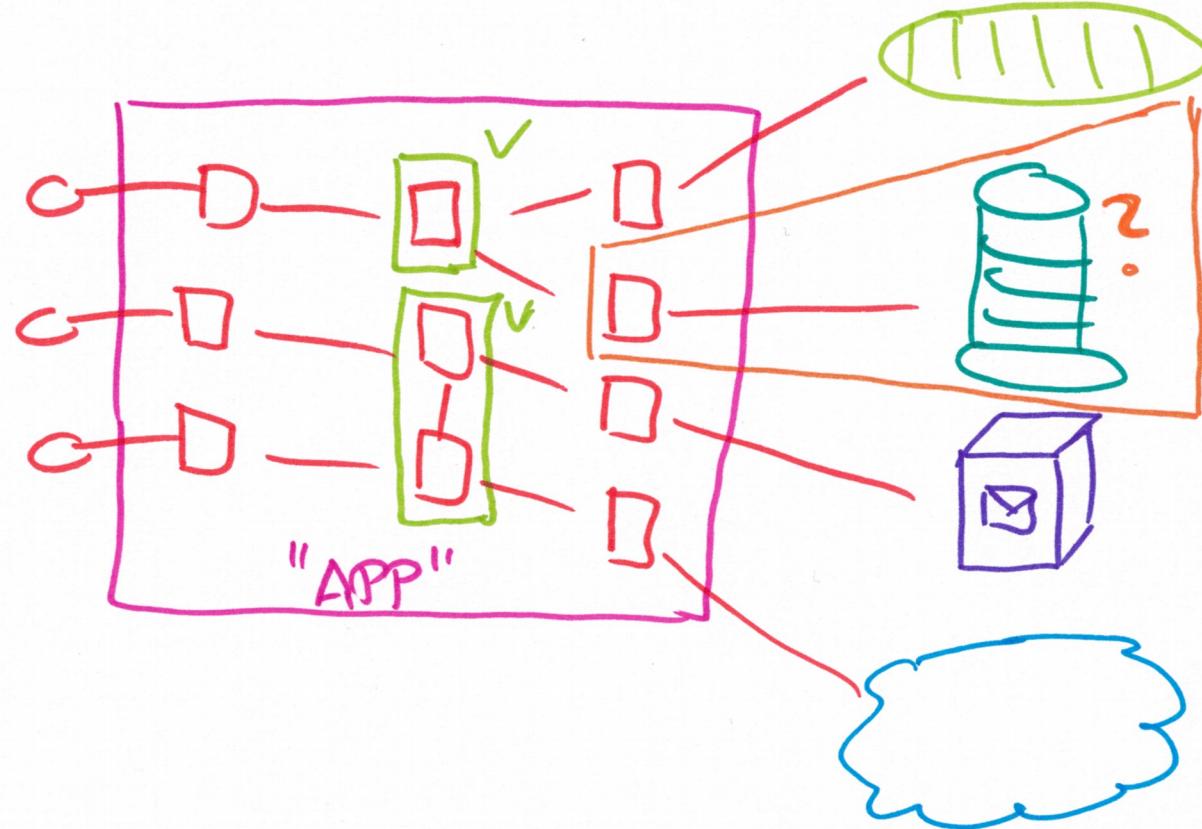
Integrated Tests

A test that will pass or fail based on the correctness of another system.

J.B.Rainsberger

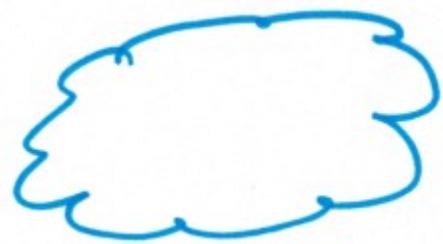
Signs for having Integrated Tests:

- We spin up other services in a local testing environment
- We test against other services in a shared testing environment
- Changes to your system breaks tests for other systems
from Spotify Blog Post



Anforderungen

- Build muss unabhängig von anderen Systemen sein
 - Minimierung der false negative Fehler
 - Entkopplung zwischen den Entwicklern
- Feedback bei Fehlern so schnell wie möglich
- Testausführung beschleunigen



HTTP



E-Mail



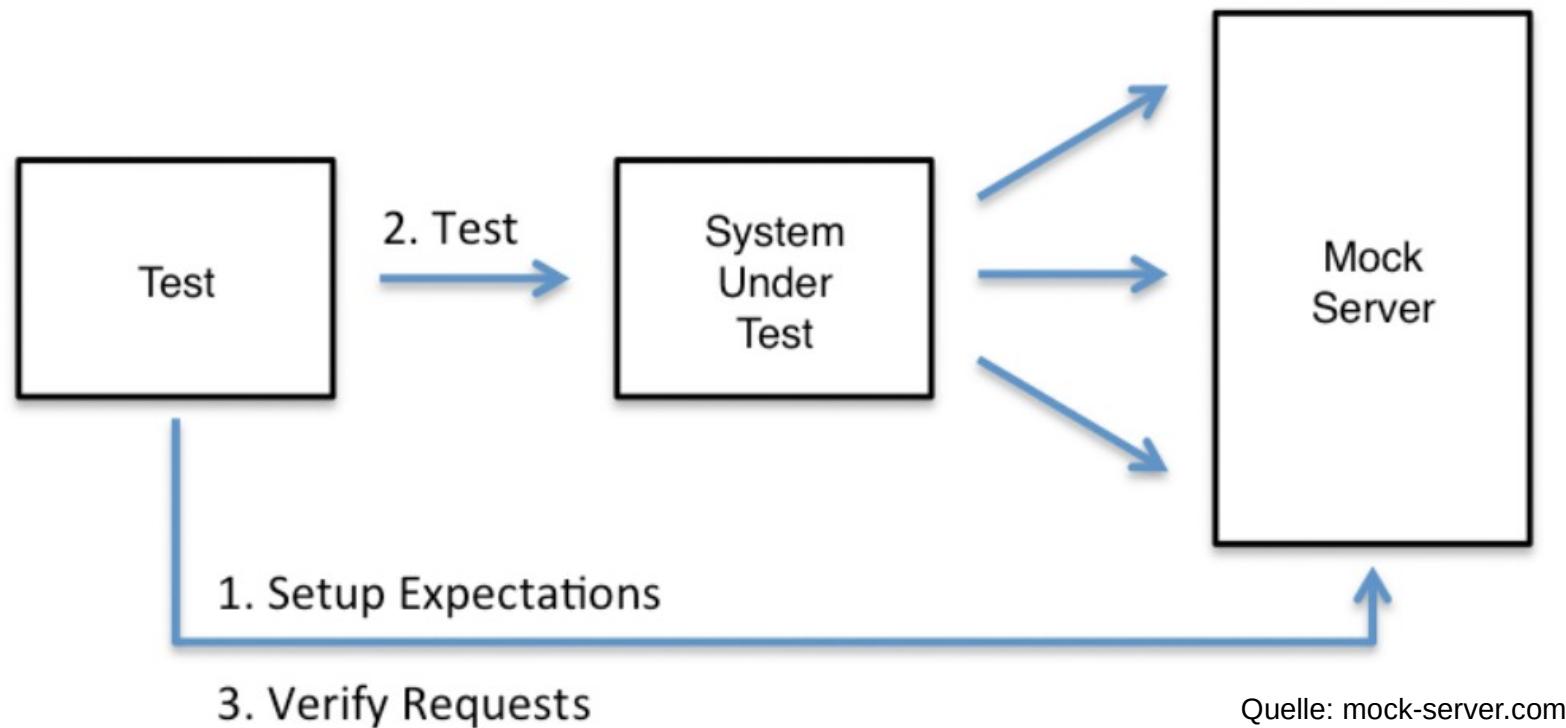
Datenbank

HTTP



Mocking HTTP Calls

- MockServer (<http://mock-server.com/>)
- WireMock (<http://wiremock.org/>)



MockServer

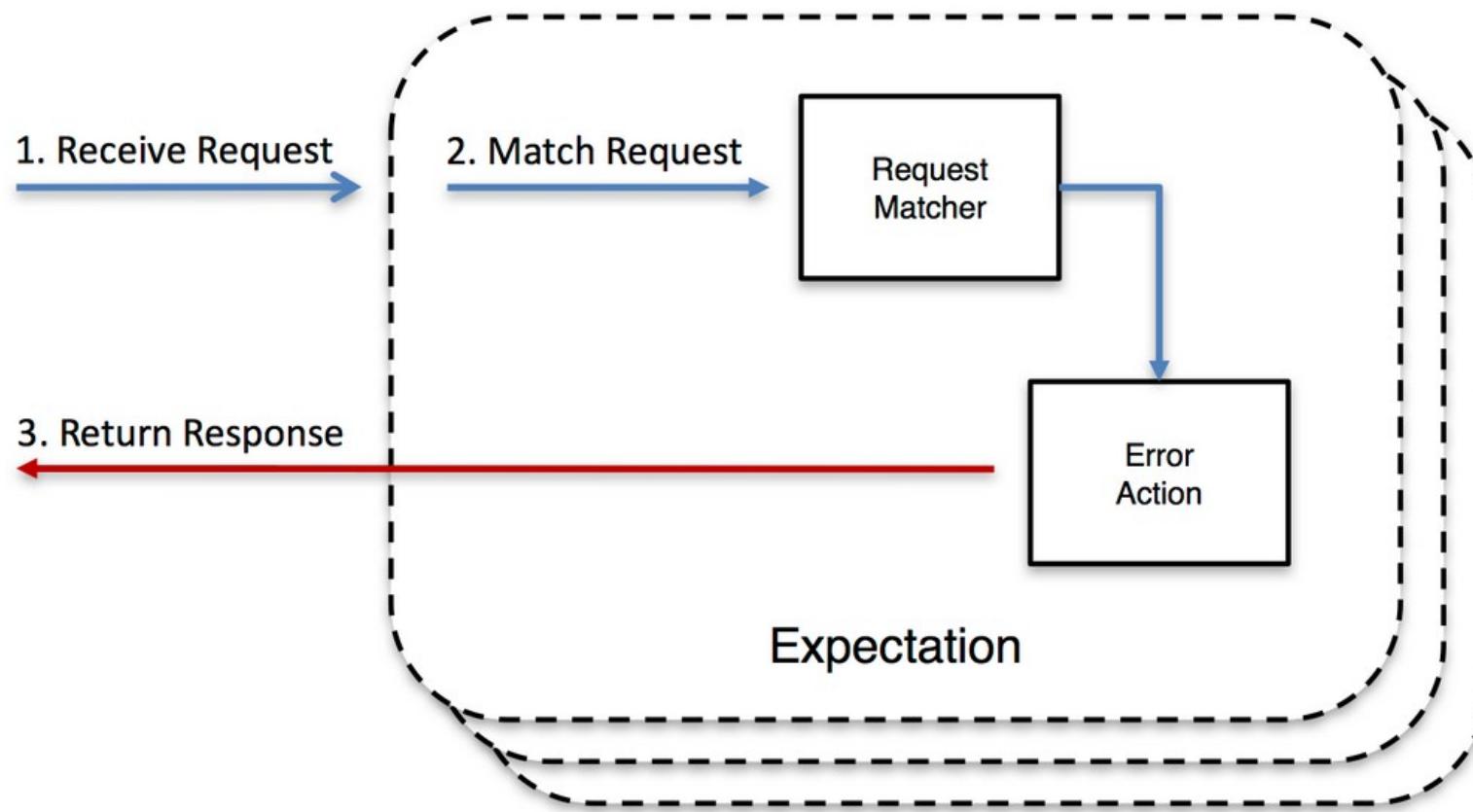
- Java und JavaScript API
- Java Rule
- Maven Plugin
- Npm Plugin
- Grunt Plugin
- Standalone (CLI)
- Docker Container

MockServer



Quelle: mock-server.com

MockServer



Quelle: mock-server.com

```
1 ▼ {
2   "count": 37,
3   "next": "https://swapi.co/api/starships/?page=2",
4   "previous": null,
5   "results": [
6     {
7       "name": "Executor",
8       "model": "Executor-class star dreadnought",
9       "manufacturer": "Kuat Drive Yards, Fondor Shipyards",
10      "cost_in_credits": "1143350000",
11      "length": "19000",
12      "max_atmosphering_speed": "n/a",
13      "crew": "279144",
14      "passengers": "38000",
15      "cargo_capacity": "2500000000",
16      "consumables": "6 years",
17      "hyperdrive_rating": "2.0",
18      "MGLT": "40",
19      "starship_class": "Star dreadnought",
20      "pilots": [],
21      "films": [
22        "https://swapi.co/api/films/2/",
23        "https://swapi.co/api/films/3/"
24      ],
25      "created": "2014-12-15T12:31:42.547000Z",
26      "edited": "2017-04-19T10:56:06.685592Z",
27      "url": "https://swapi.co/api/starships/15/"
28    },

```

<http://swapi.co/api/starships>

```
public class StarWarsClient {

    private final RestTemplate restTemplate;
    private final String baseUrl;

    public StarWarsClient(String protocol, String hostName, int port) {
        this.restTemplate = new RestTemplateBuilder().build();
        try {
            this.baseUrl = new URL(protocol, hostName, port, file: "").toString();
        } catch (MalformedURLException e) {
            throw new RuntimeException(e);
        }
    }

    public List<Starship> findAllStarships() {
        List<Starship> starships = new ArrayList<>();
        String nextPageUrl = baseUrl + "/api/starships";
        do {
            String forObject = restTemplate.getForObject(nextPageUrl, String.class);
            Map<String, Object> jsonMap = new GsonJsonParser().parseMap(forObject);
            nextPageUrl = (String) jsonMap.get("next");
            for (Map result : (List<Map>) jsonMap.get("results")) {
                starships.add(Starship.from(result));
            }
        }
        while (nextPageUrl != null);
        return starships;
    }
}
```

```
@Rule
public MockServerRule mockServerRule = new MockServerRule( target: this);

private MockServerClient mockServerClient = mockServerRule.getClient();
private StarWarsClient clientUnderTest = new StarWarsClient( protocol: "http", hostName: "localhost", mockServerRule.getPort());

@Test
public void findAllStarships() {
    mockServerClient
        .when(request()
            .withMethod("GET")
            .withPath("/api/starships"))
        .respond(response()
            .withBody(testData))
    );
    mockServerClient
        .when(request()
            .withMethod("GET")
            .withPath("/api/starships2"))
        .respond(response()
            .withBody(testData2))
    );

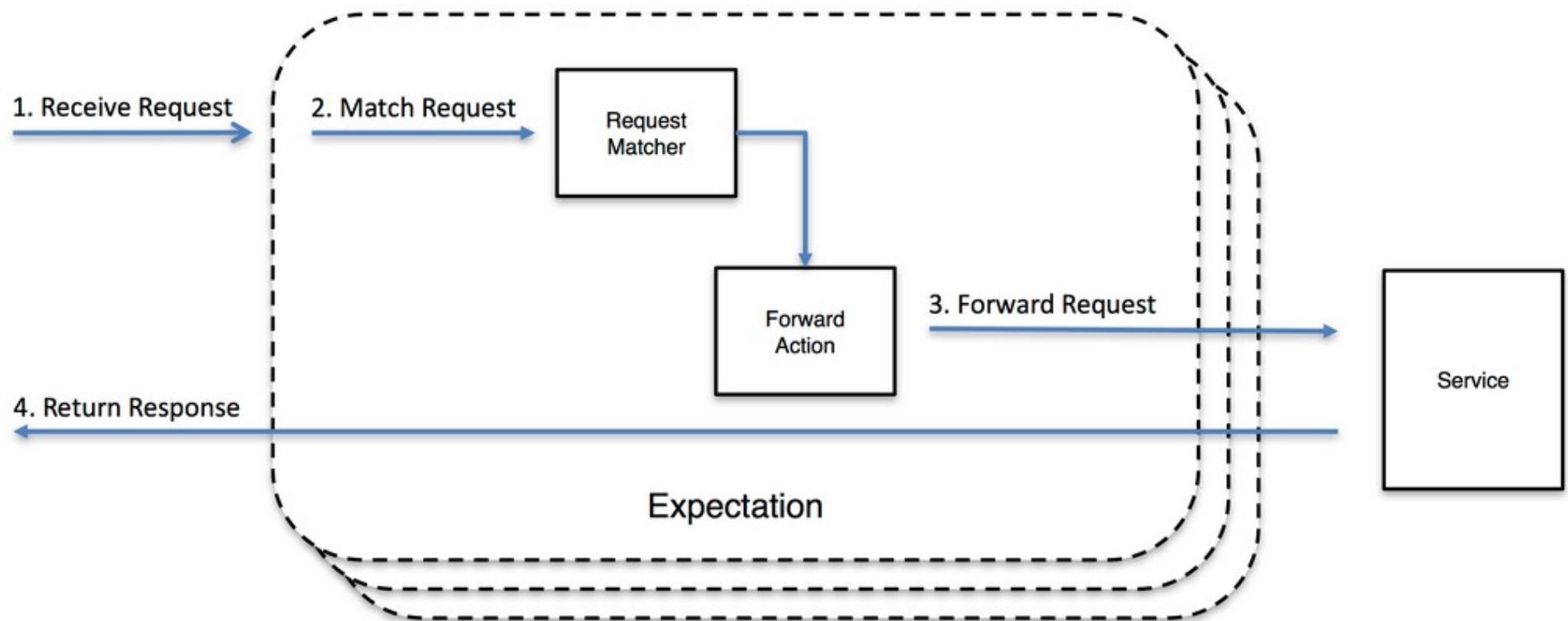
    List<Starship> allStarships = clientUnderTest.findAllStarships();
    assertThat(allStarships).hasSize(11);
}
```

Don't mock API you don't own

Verified Fakes

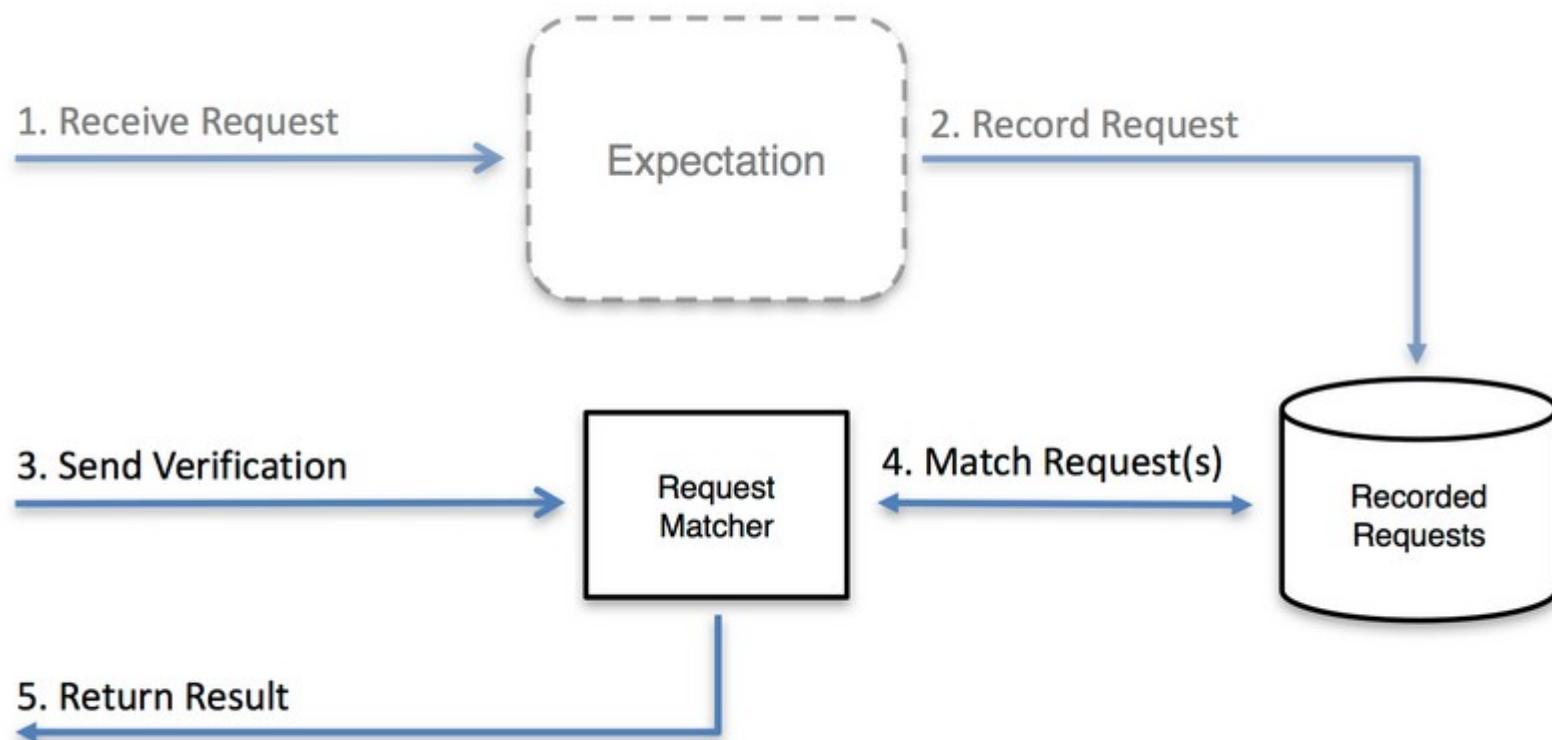
Tests, die die richtige API aufrufen und verifizieren ob die Test Doubles richtig sind

MockServer – Weitere Features



Quelle: mock-server.com

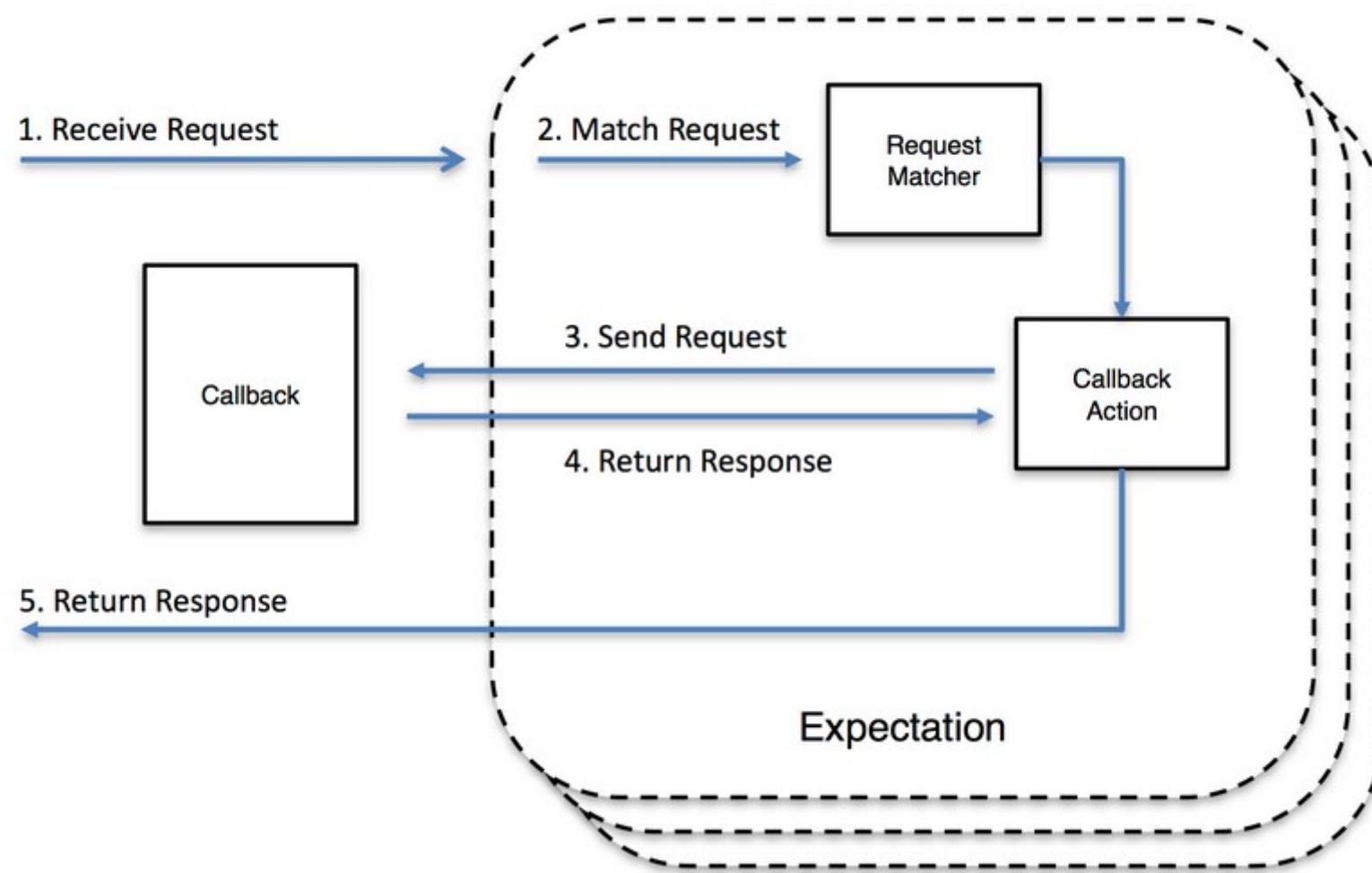
MockServer – Weitere Features



Quelle: mock-server.com

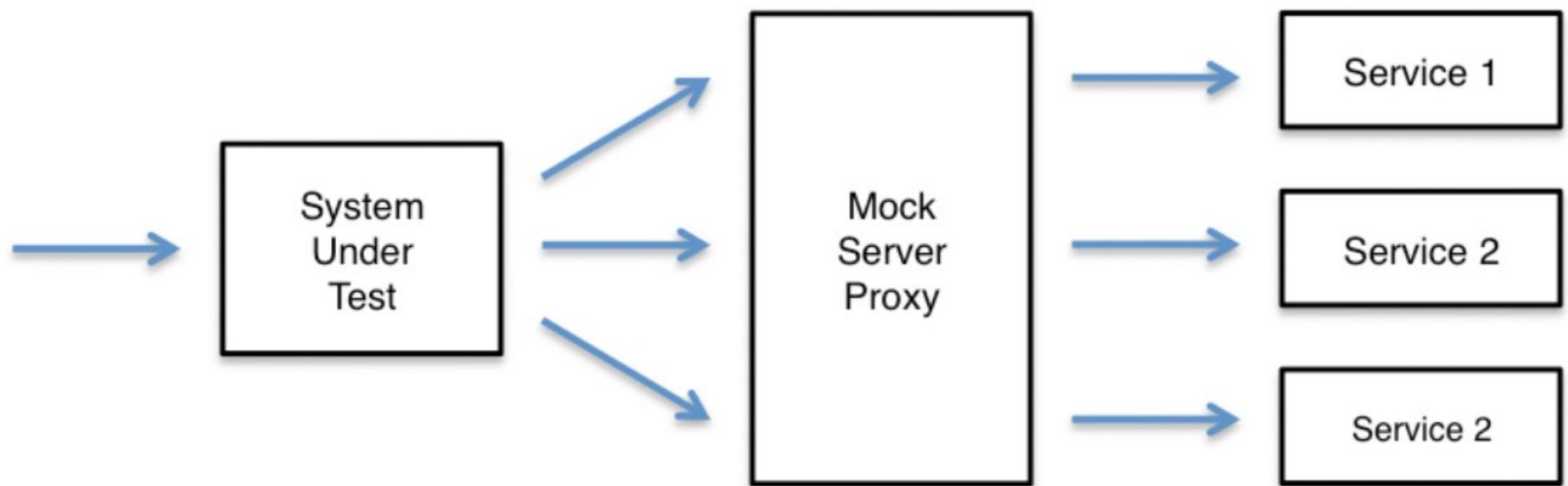
```
85 @Test
86 public void verifyFindAllStarshipsRequest(){
87     mockServerClient
88         .when(request())
89             .withMethod("GET")
90             .withPath("/api/starships")
91         )
92         .respond(response()
93             .withBody(testData)
94         );
95     mockServerClient
96         .when(request())
97             .withMethod("GET")
98             .withPath("/api/starships2")
99         )
100        .respond(response()
101            .withBody(testData2)
102        );
103
104     List<Starship> allStarships = clientUnderTest.findAllStarships();
105
106     assertThat(allStarships).hasSize(11);
107
108     mockServerClient
109         .verify(request()
110             .withMethod("GET")
111             .withPath("/api/starships"),
112             VerificationTimes.once());
113     mockServerClient
114         .verify(request()
115             .withMethod("GET")
116             .withPath("/api/starships2"),
117             VerificationTimes.once());
118 }
```

MockServer – Weitere Features



Quelle: mock-server.com

MockServer Proxy



Quelle: mock-server.com

MockServer Proxy

- proxy all requests using any of the following proxying methods:
 - Port Forwarding
 - Web Proxying (i.e. HTTP proxy)
 - HTTPS Tunneling Proxying (using HTTP CONNECT)
 - SOCKS Proxying (i.e. dynamic port forwarding)
- verify requests have been sent (i.e. in a test assertion)
- record requests and responses to analyse how a system behaves

Eigene HTTP Endpoints

- Jersey → JerseyTest
- Spring MVC → MockMvc
- Framework-unabhängig
 - REST Assured (<http://rest-assured.io/>)
 - „Killer“-Feature: JsonPath, XmlPath
 - Scala Support
 - Spring MVC Support

Rest Assured - JsonPath

```
{  
  "lotto":{  
    "lottoId":5,  
    "winning-numbers":[2,45,34,23,7,5,3],  
    "winners":[{  
      "winnerId":23,  
      "numbers":[2,45,34,23,3,5]  
    }, {  
      "winnerId":54,  
      "numbers":[52,3,12,11,18,22]  
    }]  
  }  
}
```

```
get("/lotto").then().body("lotto.lottoId", equalTo(5));  
  
get("/lotto").then().body("lotto.winners.winnerId", hasItems(23, 54));
```

<http://localhost:8080/lotto>

Quelle: Rest Assured Doc

Rest Assured - XMLPath

```
<greeting>
  <firstName>{params("firstName")}</firstName>
  <lastName>{params("lastName")}</lastName>
</greeting>
```

```
given().
    parameters("firstName", "John", "lastName", "Doe").
when().
    post("/greetXML").
then().
    body("greeting.firstName", equalTo("John")).
    body("greeting.lastName", equalTo("Doe"));
```

<http://localhost:8080/greetXML>

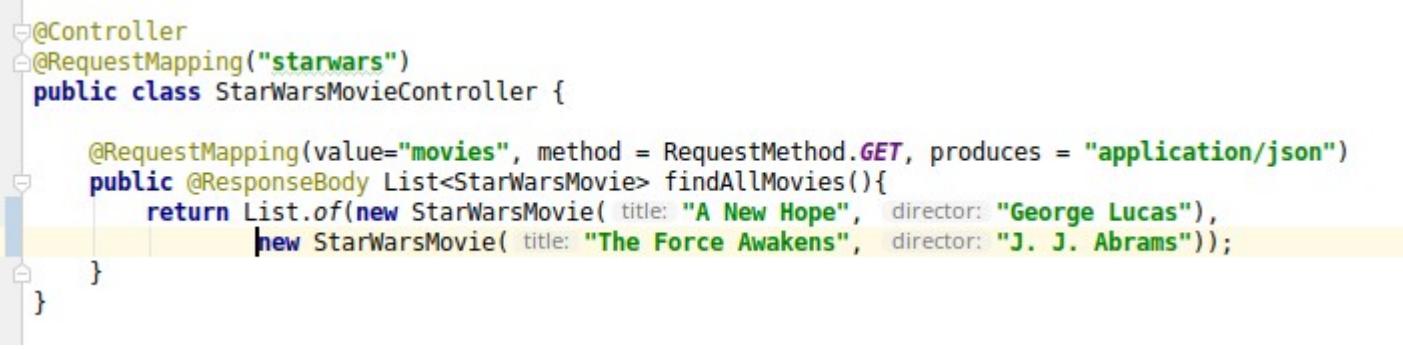
Quelle: Rest Assured Doc

Rest Assured – Spring MVC

```
1  @Controller
2  @RequestMapping("starwars")
3  public class StarWarsMovieController {
4
5      @RequestMapping(value="movies", method = RequestMethod.GET, produces = "application/json")
6      public @ResponseBody List<StarWarsMovie> findAllMovies(){
7          return List.of(new StarWarsMovie( title: "A New Hope", director: "George Lucas"),
8                      new StarWarsMovie( title: "The Force Awakens", director: "J. J. Abrams"));
9      }
10 }
```

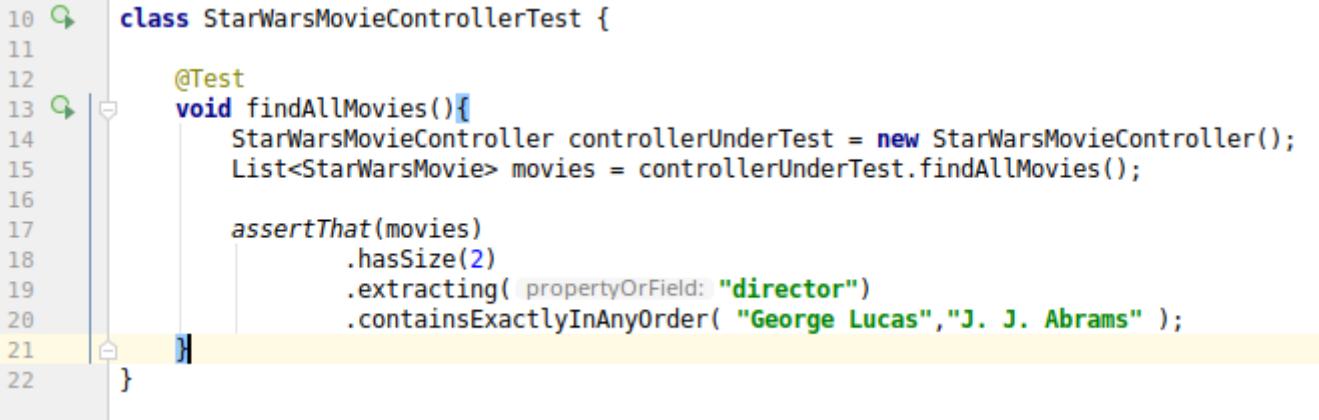
```
13  @RunWith(SpringRunner.class)
14  @WebMvcTest(StarWarsMovieController.class)
15  public class StarWarsMovieControllerIT {
16
17      @Autowired
18      private MockMvc mockMvc;
19
20      @Test
21      public void findAllMovies(){
22          given().mockMvc(mockMvc).
23          when().get( s: "/starwars/movies").
24          then().statusCode(200)
25              .body( s: "title", hasItems("A New Hope", "The Force Awakens"))
26              .body( s: "director", hasItems("George Lucas", "J. J. Abrams"));
27      }
28
29 }
```

Unit Tests für Endpoints



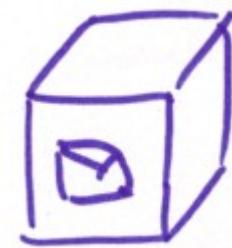
```
@Controller
@RequestMapping("starwars")
public class StarWarsMovieController {

    @RequestMapping(value="movies", method = RequestMethod.GET, produces = "application/json")
    public @ResponseBody List<StarWarsMovie> findAllMovies(){
        return List.of(new StarWarsMovie( title: "A New Hope", director: "George Lucas"),
                      new StarWarsMovie( title: "The Force Awakens", director: "J. J. Abrams"));
    }
}
```



```
10 Q  class StarWarsMovieControllerTest {
11
12     @Test
13     Q void findAllMovies(){
14         StarWarsMovieController controllerUnderTest = new StarWarsMovieController();
15         List<StarWarsMovie> movies = controllerUnderTest.findAllMovies();
16
17         assertThat(movies)
18             .hasSize(2)
19             .extracting( propertyOrField: "director")
20             .containsExactlyInAnyOrder( "George Lucas","J. J. Abrams" );
21
22     }
}
```

E-Mail



E-Mail

- GreenMail (<http://www.icegreen.com/greenmail/#>)
 - Lightweight sand boxed mail server
 - Sending and Retrieving
 - SMTP, POP3, IMAP incl. SSL
 - JUnit integration (JUnit4)
 - Usage model:
 - Java API
 - Standalone
 - WAR
 - Docker Container

Greenmail

```
public class MailClient {

    private final JavaMailSender javaMailSender;

    public MailClient(JavaMailSender javaMailSender) { this.javaMailSender = javaMailSender; }

    public void sendMessage(String from, String to, String body) throws MessagingException {
        MimeMessage message = javaMailSender.createMimeMessage();
        message.setFrom(new InternetAddress(from));
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
        message.setText(body);
        javaMailSender.send(message);
    }

    private GreenMail greenMail = new GreenMail(ServerSetupTest.SMTP);
    private MailClient clientUnderTest;

    @BeforeEach
    void setUp(){
        JavaMailSender javaMailSender = createJavaMailSender();
        clientUnderTest = new MailClient(javaMailSender);
        greenMail.start();
    }

    @Test
    void sendMessage() throws MessagingException, IOException {
        String message = "text";
        String from = "from@example.com";
        String to = "to@example.com";
        clientUnderTest.sendMessage(from, to, message);

        MimeMessage[] receivedMessages = greenMail.getReceivedMessages();
        assertThat(receivedMessages).hasSize(1);
        String content = (String) receivedMessages[0].getContent();
        assertThat(content.trim()).isEqualTo(message);
    }

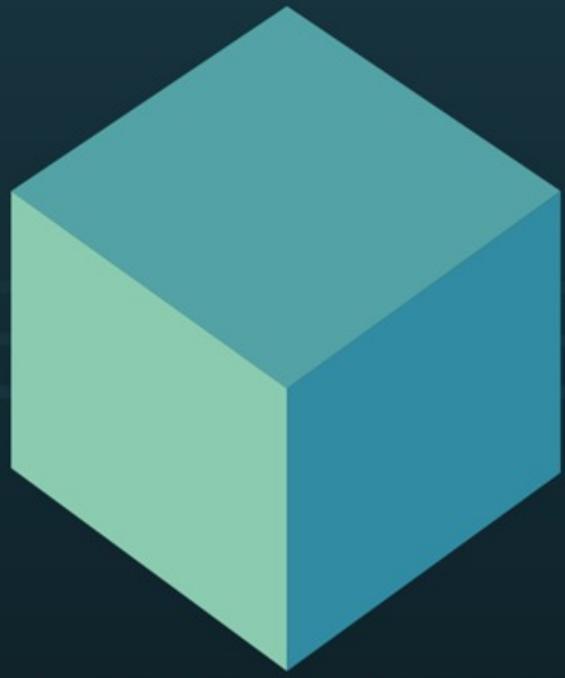
    @AfterEach
    void cleanUp() { greenMail.stop(); }
}
```

Datenbanken



Datenbanken

- Embedded Datenbanken
 - H2, Derby
 - Nicht Produktionsnah
 - Nicht alles testbar
- Standalone Datenbanken
 - Abhangigkeit zur bestimmten Infrastruktur
- Shared Datenbanken
 - Abhangigkeiten zwischen Entwickler
 - Hohe false negative Fehlerrate



TESTCONTAINERS

```
@Testcontainers
class PersonRepositoryJUnit5Test {

    @Container
    private PostgreSQLContainer postgres = new PostgreSQLContainer();

    private PersonRepository repositoryUnderTest;

    @BeforeEach
    public void setup(){
        HikariConfig hikariConfig = new HikariConfig();
        hikariConfig.setJdbcUrl(postgres.getJdbcUrl());
        hikariConfig.setUsername(postgres.getUsername());
        hikariConfig.setPassword(postgres.getPassword());

        HikariDataSource ds = new HikariDataSource(hikariConfig);
        Flyway flyway = new Flyway();
        flyway.setDataSource(ds);
        flyway.migrate();

        repositoryUnderTest = new PersonRepository(ds);
    }

    @Test
    void saveAndFindAllPerson() {
        Person person = new Person();
        person.setFirstName("firstName");
        person.setLastName("lastName");
        person.setJobTitle("jobTitle");

        repositoryUnderTest.save(person);

        List<Person> persons = repositoryUnderTest.findAllPersons();
        assertThat(persons).hasSize(1).contains(person);
    }
}
```

```
@Testcontainers
class DbMigrationJUnit5Test {

    @Container
    private MySQLContainer mysqlDb = new MySQLContainer();

    @Test
    void testDbMigrationFromTheScratch(){
        Flyway flyway = new Flyway();
        flyway.setDataSource(mysqlDb.getJdbcUrl(), mysqlDb.getUsername(), mysqlDb.getPassword());

        flyway.migrate();
    }
}
```

```
public class PersonRepositoryJUnit4Test {

    @Rule
    public PostgreSQLContainer postgres = new PostgreSQLContainer();

    private PersonRepository repositoryUnderTest;

    @Before
    public void setup(){
        HikariConfig hikariConfig = new HikariConfig();
        hikariConfig.setJdbcUrl(postgres.getJdbcUrl());
        hikariConfig.setUsername(postgres.getUsername());
        hikariConfig.setPassword(postgres.getPassword());

        HikariDataSource ds = new HikariDataSource(hikariConfig);
        Flyway flyway = new Flyway();
        flyway.setDataSource(ds);
        flyway.migrate();

        repositoryUnderTest = new PersonRepository(ds);
    }

    @Test
    public void saveAndFindAllPerson() {
        Person person = new Person();
        person.setFirstName("firstName");
        person.setLastName("lastName");
        person.setJobTitle("jobTitle");

        repositoryUnderTest.save(person);

        List<Person> persons = repositoryUnderTest.findAllPersons();
        assertThat(persons).hasSize(1).contains(person);
    }
}
```

```
public class DbMigrationJUnit4Test {  
  
    @Rule  
    public MySQLContainer mysqlDb = new MySQLContainer();  
  
    @Test  
    public void testDbMigrationFromTheScratch(){  
        Flyway flyway = new Flyway();  
        flyway.setDataSource(mysqlDb.getJdbcUrl(), mysqlDb.getUsername(), mysqlDb.getPassword());  
  
        flyway.migrate();  
    }  
}
```

TESTS

```
Running db.migration.DbMigrationITest
INFO - DockerClientProviderStrategy - Found docker client settings from environment
INFO - DockerClientProviderStrategy - Found Docker environment with Environment variables, system properties and defaults. Resolved:
  dockerHost=unix:///var/run/docker.sock
  apiVersion='{UNKNOWN_VERSION}'
  registryUrl='https://index.docker.io/v1/'
  registryUsername='sparsick'
  registryPassword='null'
  registryEmail='null'
  dockerConfig='DefaultDockerClientConfig[dockerHost=unix:///var/run/docker.sock, registryUsername=sparsick, registryPassword=<null>, registryEmail=<null>]'

INFO - DockerClientFactory      - Docker host IP address is localhost
INFO - DockerClientFactory      - Connected to docker:
  Server Version: 17.05.0-ce
  API Version: 1.29
  Operating System: Linux Mint 18.2
  Total Memory: 19511 MB
    i Checking the system...
      ✓ Docker version is newer than 1.6.0
      ✓ Docker environment has more than 2GB free
      ✓ File should be mountable
      ✓ Exposed port is accessible
INFO - [mysql:latest]           - Creating container for image: mysql:latest
INFO - [mysql:latest]           - Starting container with ID: 2668be66c2631e49b5bcb4e180665d223525ec896ea78034326076d5f9063d53
INFO - [mysql:latest]           - Container mysql:latest is starting: 2668be66c2631e49b5bcb4e180665d223525ec896ea78034326076d5f9063d53
INFO - [mysql:latest]           - Waiting for database connection to become available at jdbc:mysql://localhost:32769/test using query 'SELECT 1'
INFO - [mysql:latest]           - Obtained a connection to container (jdbc:mysql://localhost:32769/test)
INFO - [mysql:latest]           - Container mysql:latest started
INFO - VersionPrinter          - Flyway 4.0.3 by Boxfuse
INFO - DbSupportFactory         - Database: jdbc:mysql://localhost:32769/test (MySQL 5.7)
INFO - DbValidate               - Successfully validated 2 migrations (execution time 00:00.011s)
INFO - MetaDataTableImpl        - Creating Metadata table: `test`.`schema_version`
INFO - DbMigrate                - Current version of schema `test`: <> Empty Schema <>
INFO - DbMigrate                - Migrating schema `test` to version 1.0.0 - create person table
INFO - DbMigrate                - Migrating schema `test` to version 2.0.0 - add column job title
INFO - DbMigrate                - Successfully applied 2 migrations to schema `test` (execution time 00:00.133s).
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 13.9 sec
```

Testcontainers

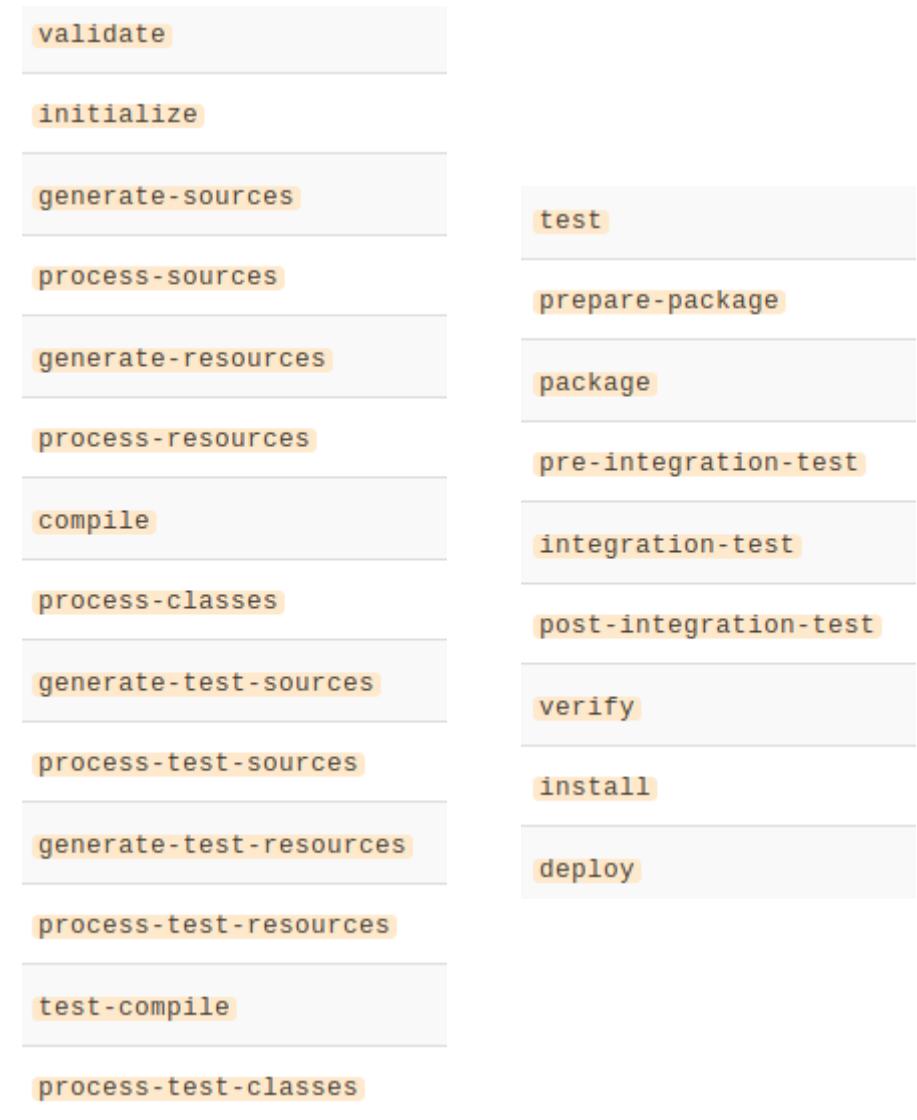
- Temporary database containers - spezielle MySQL, PostgreSQL, Oracle XE und Virtuoso container
- Webdriver containers - Dockerized Chrome oder Firefox browser für Selenium/Webdriver Operationen mit automatischer Videoaufnahme
- Kafka container – Dockerized Kafka (einzelner Knoten)
- Generic containers – irgendein Docker Container
- Docker compose – Wiederverwendung von Docker Compose YAML Datei
- Dockerfile containers – Container direkt von einem Dockerfile

Docker Maven Plugin (DMP)

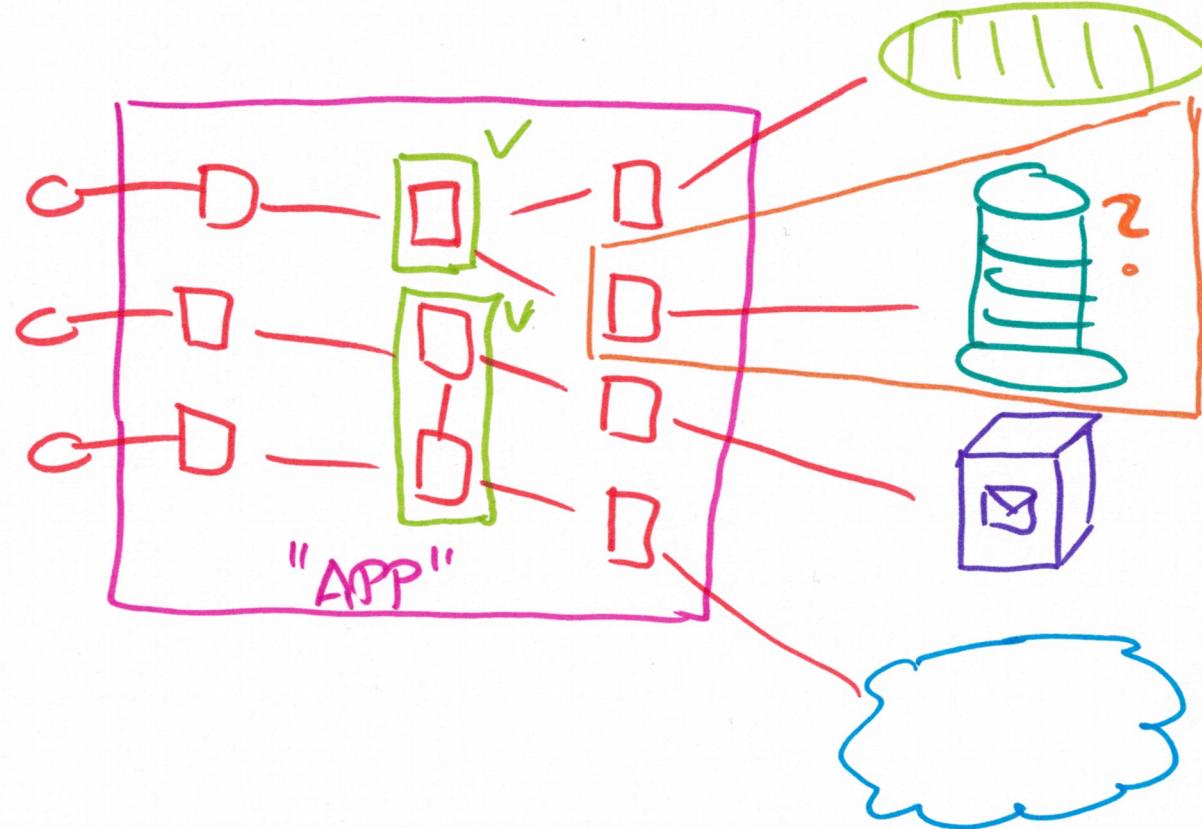
- fabric8io/docker-maven-plugin (<http://dmp.fabric8.io>)
- Kann Docker Image bauen
- Aber auch Container starten und stoppen

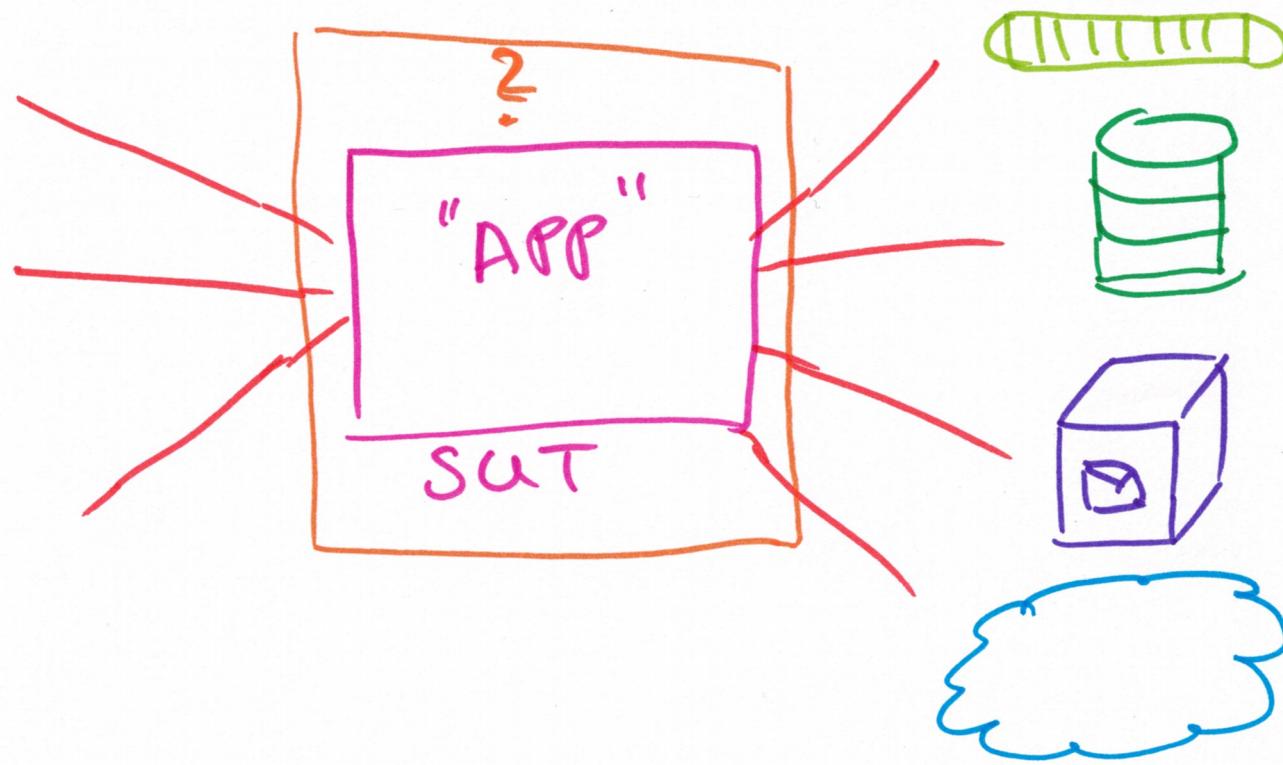
Maven Phase: Integration-Test

- Ausgangspunkt:
 - Es gibt Integrationstests gegen embedded oder standalone Datenbank
- Improvement:
 - DMP startet Container vor den Tests (*pre-integration-test*)
 - DMP stoppt Contianer nach den Tests (*post-integration-test*)



Was es noch so gibt

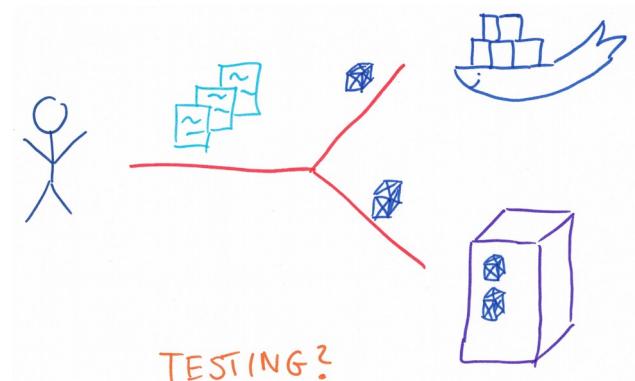


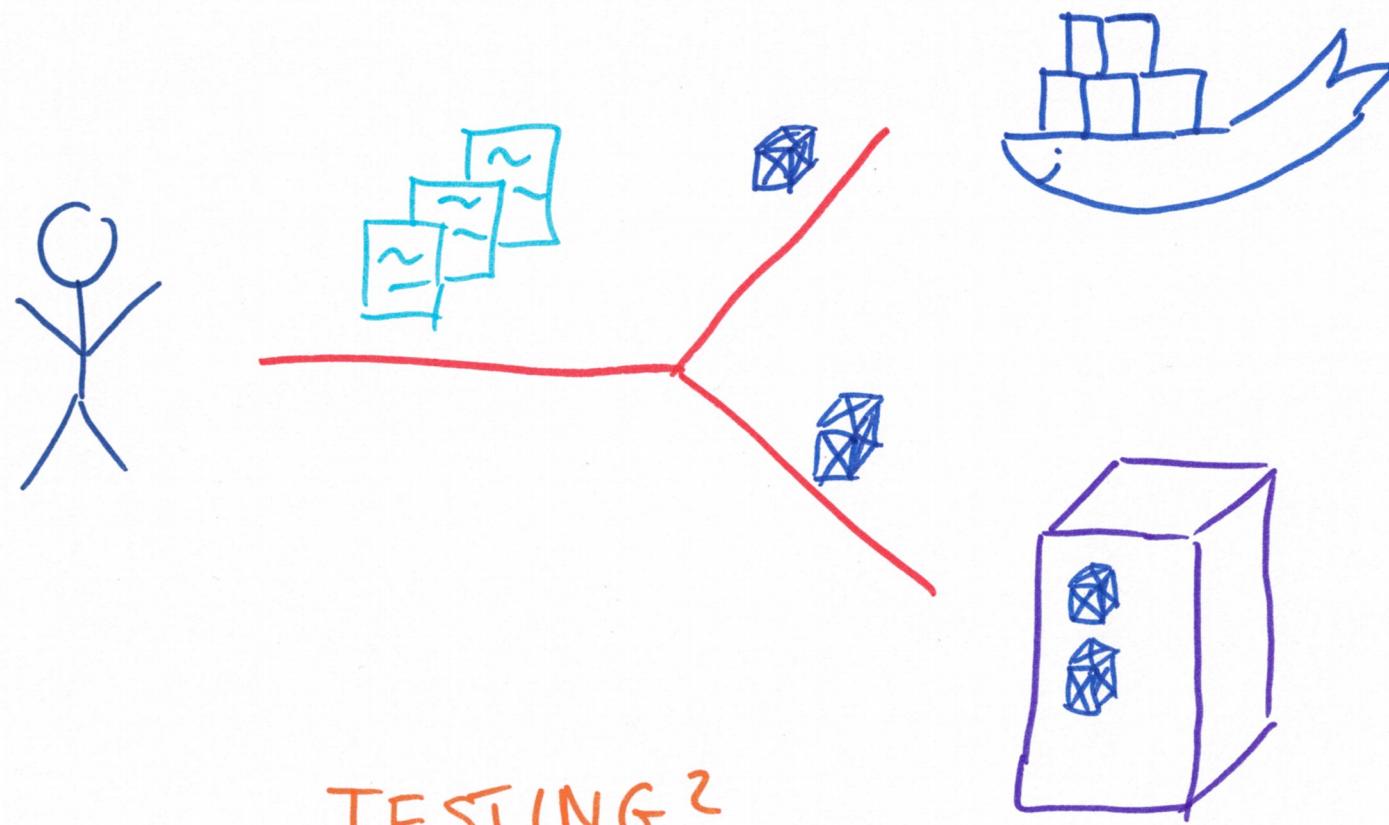


Citrus Integration Testing



Testen von Infrastruktur





Infrastructure as Code

- Shell Skripte
 - Bash, ksh, etc
- Provisionierungswerkzeuge
 - Puppet, Chef, Salt und Ansible
- Container
 - Dockerfile

Shell-Skripte

- Codequalität
 - Shellcheck (<https://www.shellcheck.net/>)
- Funktionale Qualität
 - Shunit2 (<https://github.com/kward/shunit2>)

ShellCheck

finds bugs in your shell scripts.

You can cabal, apt, dnf or brew install it locally right now.

Paste a script to try it out:

Your Editor (Ace)

Load an example Report bug Mobile paste:

```
9 chown -R www-data:www-data /var/local svn
10 mv /etc/apache2/mods-available/dav_svn.conf /etc/apache2/mods-available/dav_svn.conf.bak
11 touch /etc/apache2/mods-available/dav_svn.conf
12
13 echo '<Location /svn>
14     DAV svn
15     SVNParentPath /var/local svn
16     SVNListParentPath on
17     </Location>' >> /etc/apache2/mods-available/dav_svn.conf
18
19 service apache2 restart
20
21 if [-f /vagrant/conditus-test-repo.dump ]
22 then
23     svnadmin load /var/local svn/conditus < /vagrant/conditus-test-repo.dump
24 fi
25
26 chmod -R a+rwx /var/local svn
```

ShellCheck Output

```
$ shellcheck myscript
Line 23:
if [-f /vagrant/conditus-test-repo.dump ]
^-- SC1035: You need a space after the [ and before the ].
```

Shellcheck

```
shellcheck setup-svn.sh
```

```
In setup-svn.sh line 23:
```

```
if [-f /vagrant/conditus-test-repo.dump ]  
  ^-- SC1035: You need a space after the [ and before the ].
```

Shunit2

```
1  #! /bin/sh
2  # file: examples/math_test.sh
3
4  testAdding() {
5      result=`add_generic 1 2`
6      assertEquals \
7          "the result of '${result}' was wrong" \
8          3 "${result}"
9
10     # Disable non-generic tests.
11     [ -z "${BASH_VERSION:-}" ] && startSkipping
12
13     result=`add_bash 1 2`
14     assertEquals \
15         "the result of '${result}' was wrong" \
16         3 "${result}"
17 }
18
19 oneTimeSetUp() {
20     # Load include to test.
21     . ./math.inc
22 }
23
24 # Load and run shUnit2.
25 . ./.shunit2
```

Provisionierungswerkzeuge (Beispiel Ansible)

- Codequalität
 - ansible-lint
(<https://github.com/willthames/ansible-lint>)
- Funktionale Qualität
 - Serverspec (Tool unabh.)
(<https://serverspec.org/>)
 - Testinfra (Tool unabh.)
(<https://testinfra.readthedocs.io/en/latest/>)

```
1   hosts: application-server
2   vars:
3     tomcat_version: 8.5.8
4     tomcat_base_name: apache-tomcat-{{ tomcat_version }}
5     #catalina_opts: "-Dkey=value"
6
7   tasks:
8     - name: install java
9       apt: name=openjdk-8-jdk state=present
10      become: yes
11      become_method: sudo
12
13     - name: Download current Tomcat 8 version
14       local_action: get_url url="http://archive.apache.org/dist/tomcat/tomcat-8/v{{ tomcat_version }}}/bin/{{ tomcat_base_name }}.tar.gz" dest=/tmp
15
16     - name:
17       file: name=/opt mode=777
18       become: yes
19       become_method: sudo
20
21     - name: Install Tomcat 8
22       unarchive: src=/tmp/{{ tomcat_base_name }}.tar.gz dest=/opt creates=/opt/{{ tomcat_base_name }} owner=vagrant group=vagrant
23
24     - name: Set link to tomcat 8
25       file: src=/opt/{{ tomcat_base_name }} dest=/opt/tomcat state=link force=yes
26
27     - name: setup setenv.sh
28       template: dest="/opt/{{ tomcat_base_name }}/bin/setenv.sh" src="roles/tomcat8/templates/setenv.sh.j2" mode=755
29       when: catalina_opts is defined
30
31     - find: paths="/opt/{{ tomcat_base_name }}/bin" patterns="*.sh"
32       register: result
33
34     - name: ensure tomcat scripts are executable
35       file: name={{item.path}} mode=755
36       with_items: '{{ result.files }}'
37
38     - name: install tomcat as service
39       copy: src=roles/tomcat8/files/tomcat.service dest=/etc/systemd/system/
40       become: yes
41       become_method: sudo
42
```

Ansible-Lint

```
~/dev/workspace/infra-testing-talk/infrastructure-as-code-testing/ansible(master ✓) ansible-lint *.yml
[ANSIBLE0009] Octal file permissions must contain leading zero
setup-app.yml:16
Task/Handler: file name=/opt mode=777

[ANSIBLE0011] All tasks should be named
setup-app.yml:16
Task/Handler: file name=/opt mode=777

[ANSIBLE0009] Octal file permissions must contain leading zero
setup-app.yml:27
Task/Handler: setup setenv.sh

[ANSIBLE0011] All tasks should be named
setup-app.yml:31
Task/Handler: find patterns=*.sh paths=/opt/{{ tomcat_base_name }}/bin

[ANSIBLE0009] Octal file permissions must contain leading zero
setup-app.yml:34
Task/Handler: ensure tomcat scripts are executable

[ANSIBLE0002] Trailing whitespace
setup-db.yml:4
```

ServerSpec

```
1 require 'spec_helper'  
2  
3 ▼ describe package('mysql-server') do  
4   it { should be_installed }  
5 end  
6  
7 ▼ describe service('mysql') do  
8   it { should be_enabled }  
9   it { should be_running }  
10 end  
11  
12 ▼ describe 'MySQL config parameters' do  
13   ▼ context mysql_config('bind-address') do  
14     its(:value) { should eq '0.0.0.0' }  
15   end  
16 end  
17
```

```
1 require 'spec_helper'  
2  
3 ▼ describe package('openjdk-8-jdk') do  
4   it { should be_installed }  
5 end  
6  
7 ▼ describe command('ls /etc/init.d/tomcat') do  
8   its(:exit_status) { should eq 0 }  
9 end  
10  
11 ▼ describe command('ls /opt/tomcat') do  
12   its(:exit_status) { should eq 0 }  
13 end
```



ServerSpec

```
~/dev/workspace/infra-testing-talk/infrastructure-as-code-testing/ansible(master ✓) rake spec
/usr/bin/ruby2.5 -I/var/lib/gems/2.5.0/gems/rspec-support-3.7.1/lib:/var/lib/gems/2.5.0/gems/rspec-core-3.7.1/lib /
var/lib/gems/2.5.0/gems/rspec-core-3.7.1/exe/rspec --pattern spec/test/\*_spec.rb

Package "openjdk-8-jdk"
  should be installed

Command "ls /etc/systemd/system/tomcat.service"
  exit_status
  should eq 0

Command "ls /opt/tomcat"
  exit_status
  should eq 0

Package "mysql-server"
  should be installed

Service "mysql"
  should be enabled
  should be running

MySQL config parameters
  Mysql config "bind-address"
    value
    should eq "0.0.0.0"

Finished in 1.9 seconds (files took 8.24 seconds to load)
7 examples, 0 failures
```

testinfra

```
def test_mysql_is_installed(host):
    mysql = host.package("mysql-server")
    assert mysql.is_installed

def test_mysql_service_is_running(host):
    mysql = host.service("mysql")
    assert mysql.is_enabled
    assert mysql.is_running

def test_mysql_config_parameter_exists(host):
    mysql_conf = host.file("/etc/mysql/mysql.conf.d/mysqld.cnf")
    assert mysql_conf.contains("bind-address = 0.0.0.0")
```



testinfra

```
1 def test_openjdk_is_installed(host):
2     openjdk = host.package("openjdk-8-jdk")
3     assert openjdk.is_installed
4
5 def test_tomcat_service_exists(host):
6     assert host.file("/etc/systemd/system/tomcat.service").exists
7
8 def test_tomcat_folder_exists(host):
9     assert host.file("/opt/tomcat").exists
```

testinfra

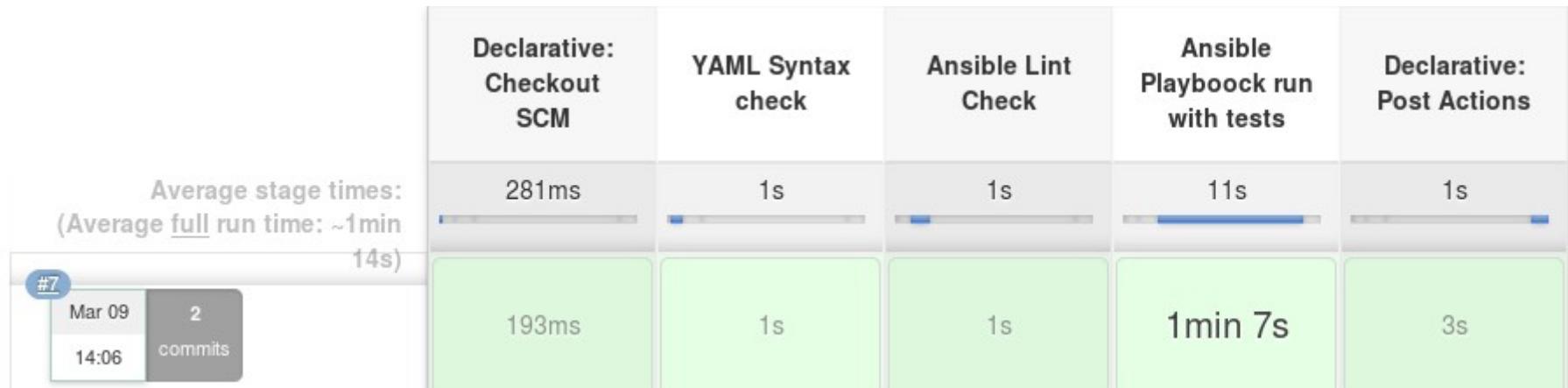


```
~/dev/workspace/infra-testing-talk/infrastructure-as-code-testing/ansible(master ✓) py.test --connection=ansible --  
ansible-inventory inventories/test -v tests/*.py  
===== test session starts =====  
platform linux2 -- Python 2.7.15rc1, pytest-3.6.3, py-1.5.4, pluggy-0.6.0 -- /usr/bin/python  
cachedir: .pytest_cache  
rootdir: /home/sparsick/dev/workspace/infra-testing-talk/infrastructure-as-code-testing/ansible, inifile:  
plugins: testinfra-1.14.1  
collected 6 items  
  
tests/test_app.py::test_openjdk_is_installed[ansible://192.168.33.10] PASSED [ 16%]  
tests/test_app.py::test_tomcat_service_exists[ansible://192.168.33.10] PASSED [ 33%]  
tests/test_app.py::test_tomcat_folder_exists[ansible://192.168.33.10] PASSED [ 50%]  
tests/test_db.py::test_mysql_is_installed[ansible://192.168.33.10] PASSED [ 66%]  
tests/test_db.py::test_mysql_service_is_running[ansible://192.168.33.10] PASSED [ 83%]  
tests/test_db.py::test_mysql_config_parameter_exists[ansible://192.168.33.10] PASSED [100%]  
  
===== 6 passed in 4.97 seconds =====
```

Jenkins Pipeline

```
1 pipeline {
2     agent any
3     stages {
4         stage('YAML Syntax check') {
5             steps {
6                 ansiblePlaybook inventory: 'inventories/test', extras: '--syntax-check', playbook: 'setup-app.yml'
7                 ansiblePlaybook inventory: 'inventories/test', extras: '--syntax-check', playbook: 'setup-db.yml'
8             }
9         }
10        stage('Ansible Lint Check') {
11            steps {
12                sh 'ansible-lint *.yml'
13            }
14        }
15        stage('Ansible Playbook run with tests') {
16            steps {
17                sh 'cd ..; vagrant up'
18                ansiblePlaybook inventory: 'inventories/test', playbook: 'setup-app.yml'
19                ansiblePlaybook inventory: 'inventories/test', playbook: 'setup-db.yml'
20                sh 'py.test --connection=ansible --ansible-inventory inventories/test -v tests/*.py'
21            }
22        }
23    }
24    post {
25        always {
26            sh 'cd ..; vagrant group destroy -f'
27        }
28    }
29 }
```

Jenkins Pipeline



Container (Beispiel Docker)

- Codequalität
 - Dockerlint (<https://github.com/RedCoolBeans/dockerlint>)
 - Haskell Dockerfile Linter – hadolint (<https://github.com/hadolint/hadolint>)
- Funktionale Qualität
 - Testinfra
 - Serverspec
 - Container Structure Tests
<https://github.com/GoogleContainerTools/container-structure-test>

```
1 ➤ FROM ubuntu:18.04
2
3 RUN apt-get update -y && \
4     apt-get install openjdk-8-jre curl -y && \
5     curl http://archive.apache.org/dist/tomcat/tomcat-9/v9.0.10/bin/apache-tomcat-9.0.10.tar.gz -o
6     /opt/tomcat.tar.gz && \
7     tar -xf /opt/tomcat.tar.gz -C /opt && \
8     rm -f /opt/tomcat.tar.gz && \
9     ln -s /opt/apache-tomcat-9.0.10 /opt/tomcat && \
10    apt-get remove curl -y
11
12 ENV CATALINA_HOME /opt/tomcat
13 ENV PATH $CATALINA_HOME/bin:$PATH
14 WORKDIR $CATALINA_HOME
15
16 ENTRYPOINT [ "catalina.sh"]
17 CMD ["run"]
18 EXPOSE 8080
```

Hadolint

```
hadolint tomcat.df
tomcat.df:3 DL3008 Pin versions in apt get install. Instead of `apt-get install <package>` use `apt-get install <package>=<version>`
tomcat.df:3 DL3009 Delete the apt-get lists after installing something
tomcat.df:3 DL3015 Avoid additional packages by specifying `--no-install-recommends`
```

Container Structure Tests

```
1  # container structure tests
2  schemaVersion: "2.0.0"
3
4  metadataTest:
5    env:
6      - key: CATALINA_HOME
7        value: /opt/tomcat
8    exposedPorts: ["8080"]
9    entrypoint: ["catalina.sh"]
10   cmd: ["run"]
11   workdir: "/opt/tomcat"
12
13
14  fileExistenceTests:
15
16    - name: 'catalina.sh'
17      path: '/opt/tomcat/bin/catalina.sh'
18      shouldExist: true
19      permissions: '-rwxr-x---'
20
```

Container Structure Tests

```
container-structure-test test --image sparsick/tomcat9:latest --config tomcat-test.yaml

=====
===== Test file: tomcat-test.yaml =====
=====

INFO: File Existence Test: catalina.sh
--- RUN: File Existence Test: catalina.sh
--- PASS
--- RUN: Metadata Test
--- PASS

=====
===== RESULTS =====
=====

Passes:    2
Failures:  0
Total tests: 2

PASS
```

Vorher: docker build -t sparsick/tomcat9 -f tomcat.df .

Fazit

Fragen?

mail@sandra-parsick.de
@SandraParsick

<https://github.com/sparsick/infra-testing-talk>

Literatur

- <http://coding-is-like-cooking.info/2018/04/pre-tested-integration-back-to-the-basis-of-ci/>
- <http://blog.thecodewhisperer.com/permalink/integrated-tests-are-a-scam>
- <http://blog.thecodewhisperer.com/permalink/clearing-up-the-integrated-tests-scam>
- https://bee42.com/de/blog/The_dark_age_of_container_testing/
- <https://labs.spotify.com/2018/01/11/testing-of-microservices/>
- https://martinfowler.com/bliki/IntegrationTest.html?utm_content=buffer2b62b&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer