Java Forum Stuttgart, 04.07.2019

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

Sandra Parsick

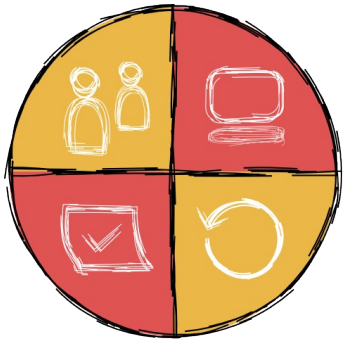
mail@sandra-parsick.de
@SandraParsick

Zu meiner Person

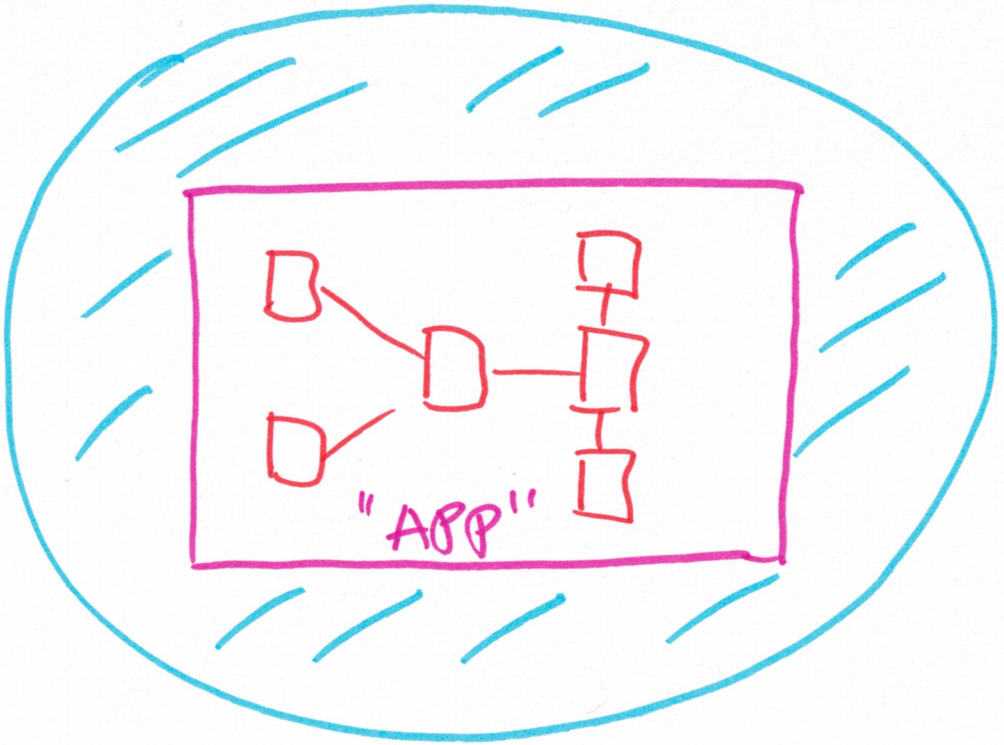
- Sandra Parsick
- Freiberuflicher Softwareentwickler und Consultant im Java-Umfeld
- Schwerpunkte:
  - Java Enterprise Anwendungen
  - Agile Methoden
  - Software Craftmanship
  - Automatisierung von Entwicklungsprozessen
- Trainings
- Workshops
- Softwerkskammer Ruhrgebiet
- Oracle Groundbreaker Ambassador

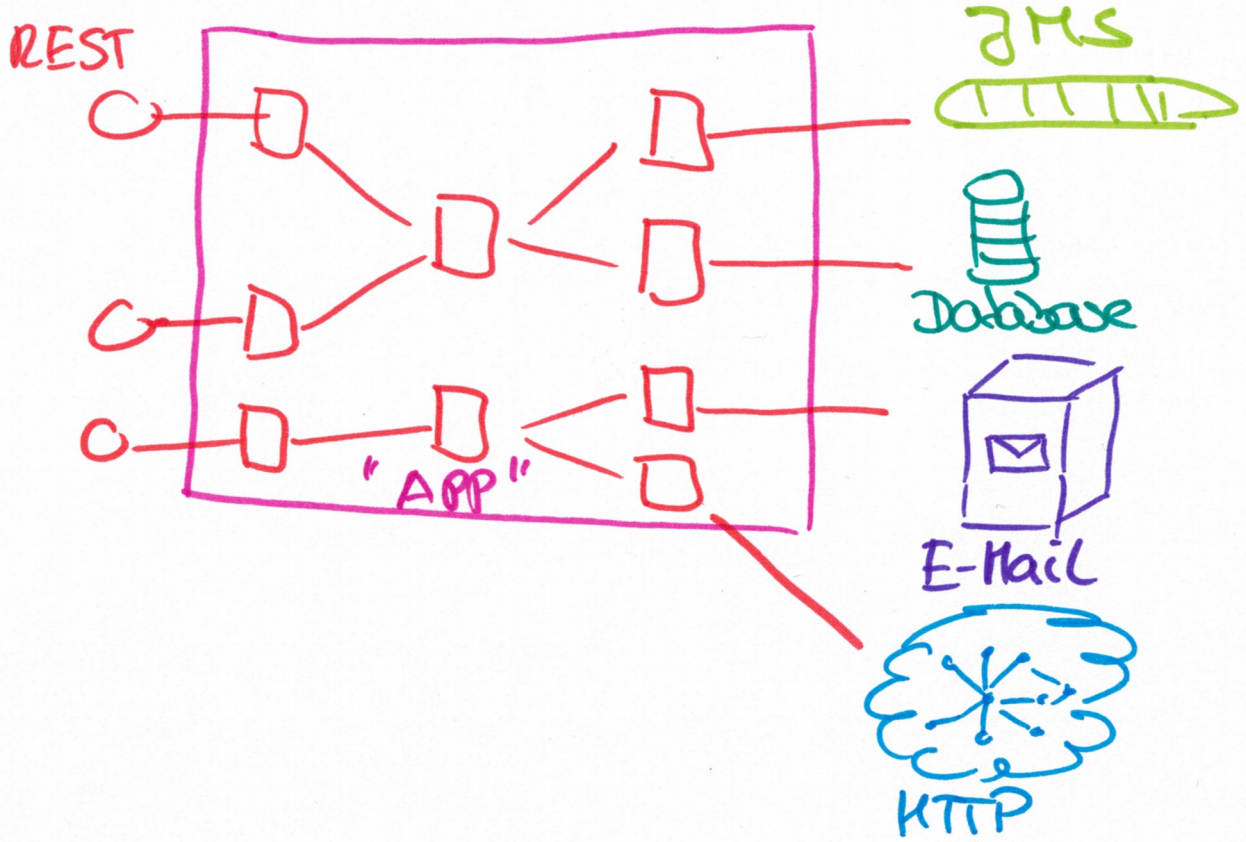
- Twitter: @SandraParsick
- Homepage: https://www.sandra-parsick.de
- Blog: http://blog.sandra-parsick.de
- E-Mail: mail@sandra-parsick.de

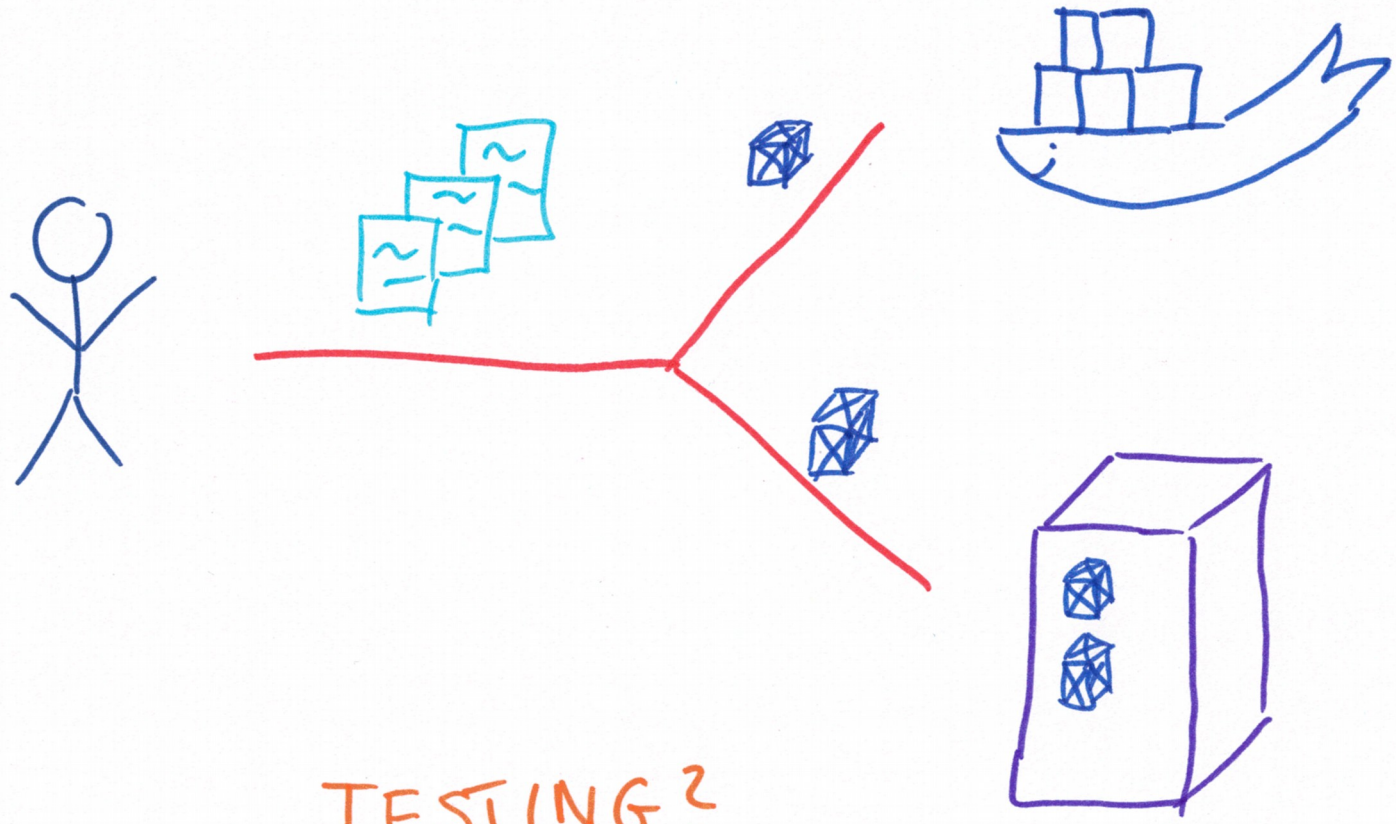
Oracle Groundbreakers

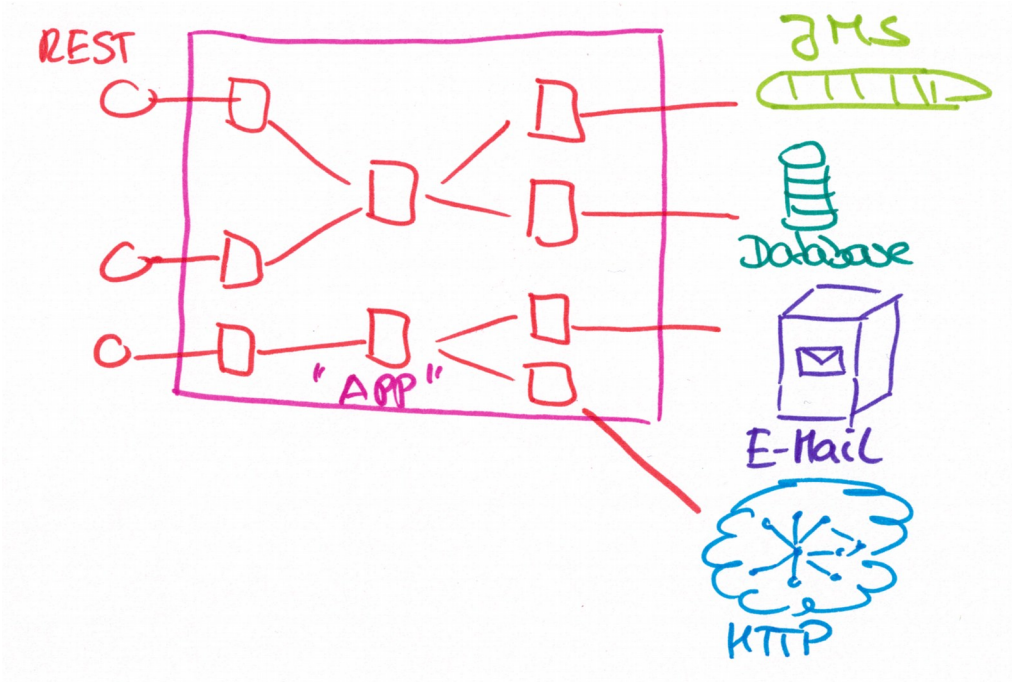


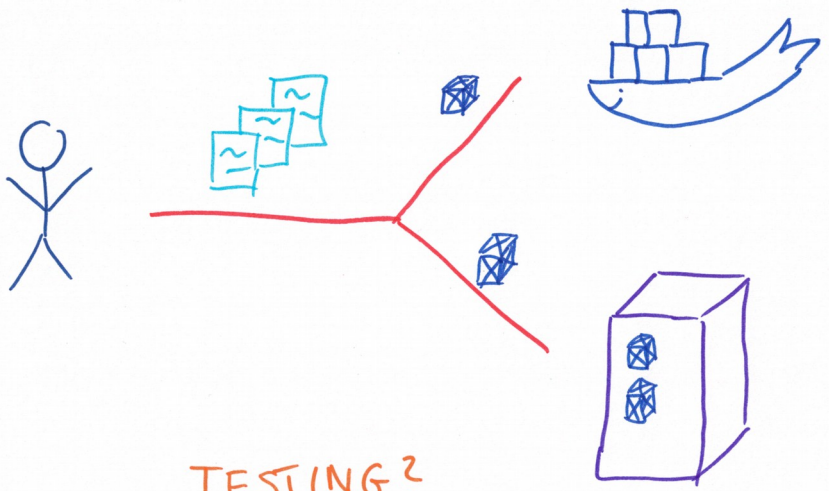
Motivation


"APP"

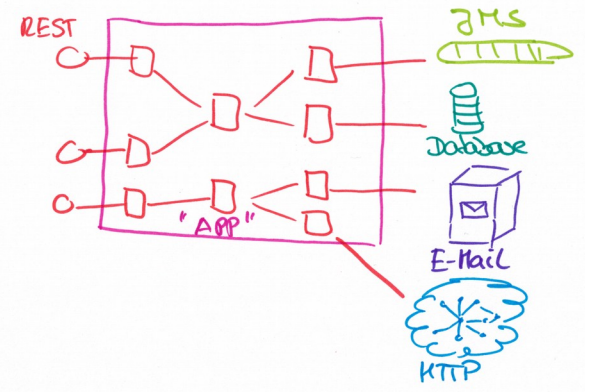

REST
JMS
Database
"APP"
E-Mail
HTTP


TESTING?


REST
JMS
Database
"APP"
E-Mail
HTTP


TESTING?

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

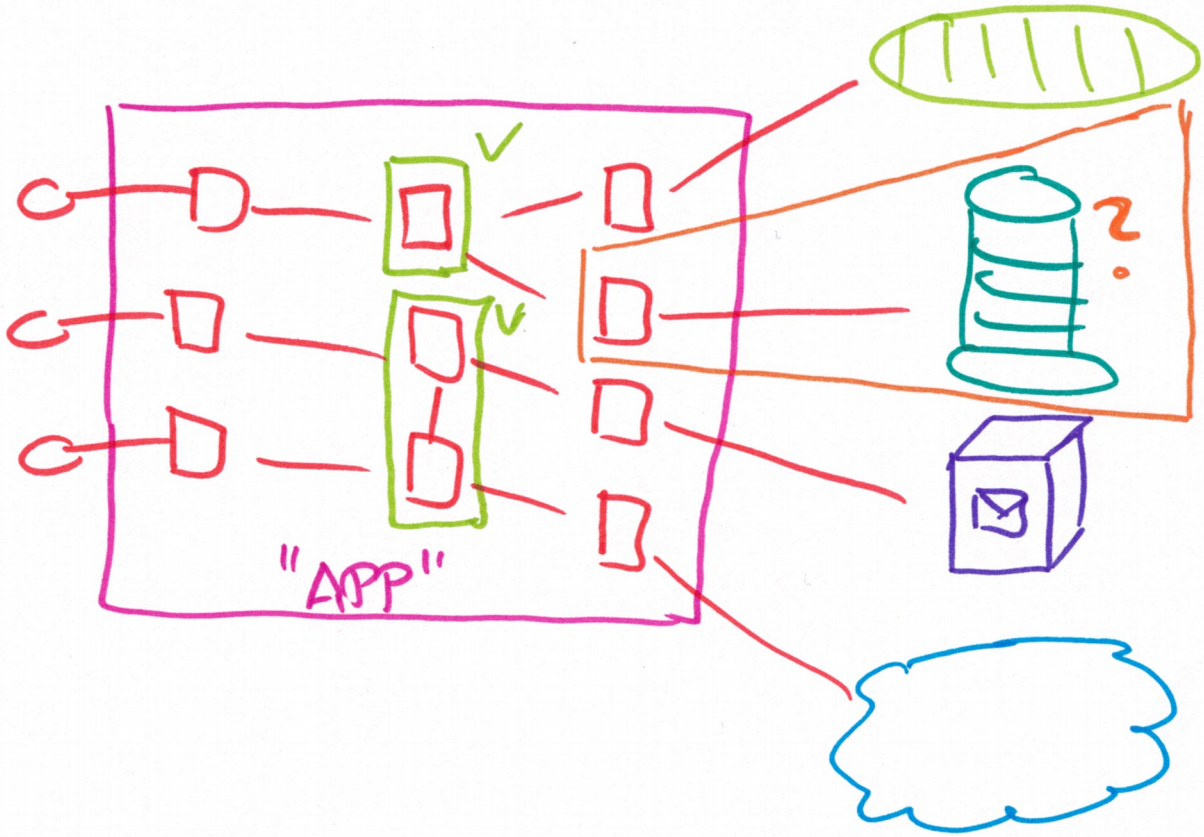
A test that will pass or fail based on the correctness of another system.

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


V
V
"APP"

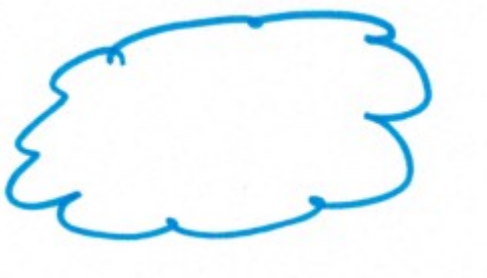
Anforderungen

- 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

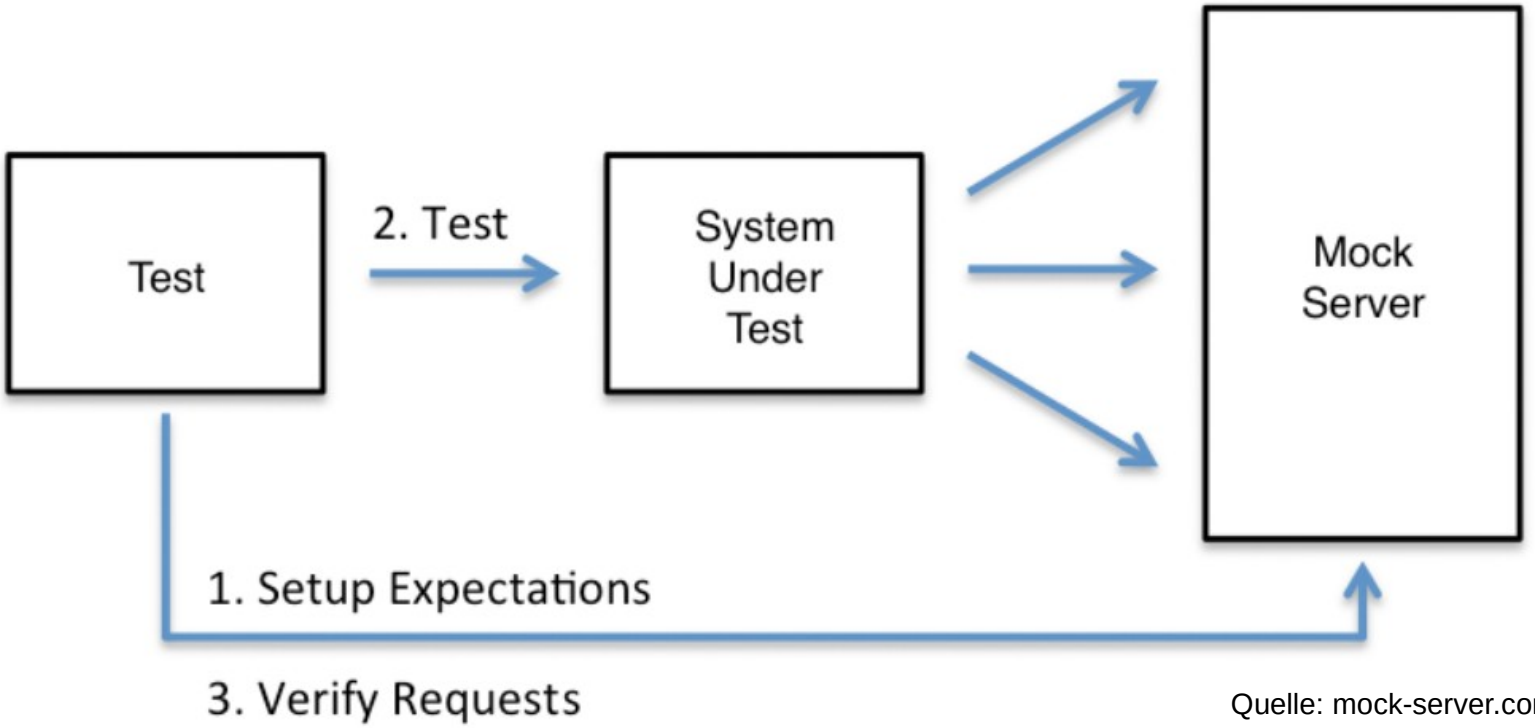
Datenbank

HTTP



Mocking HTTP Calls

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

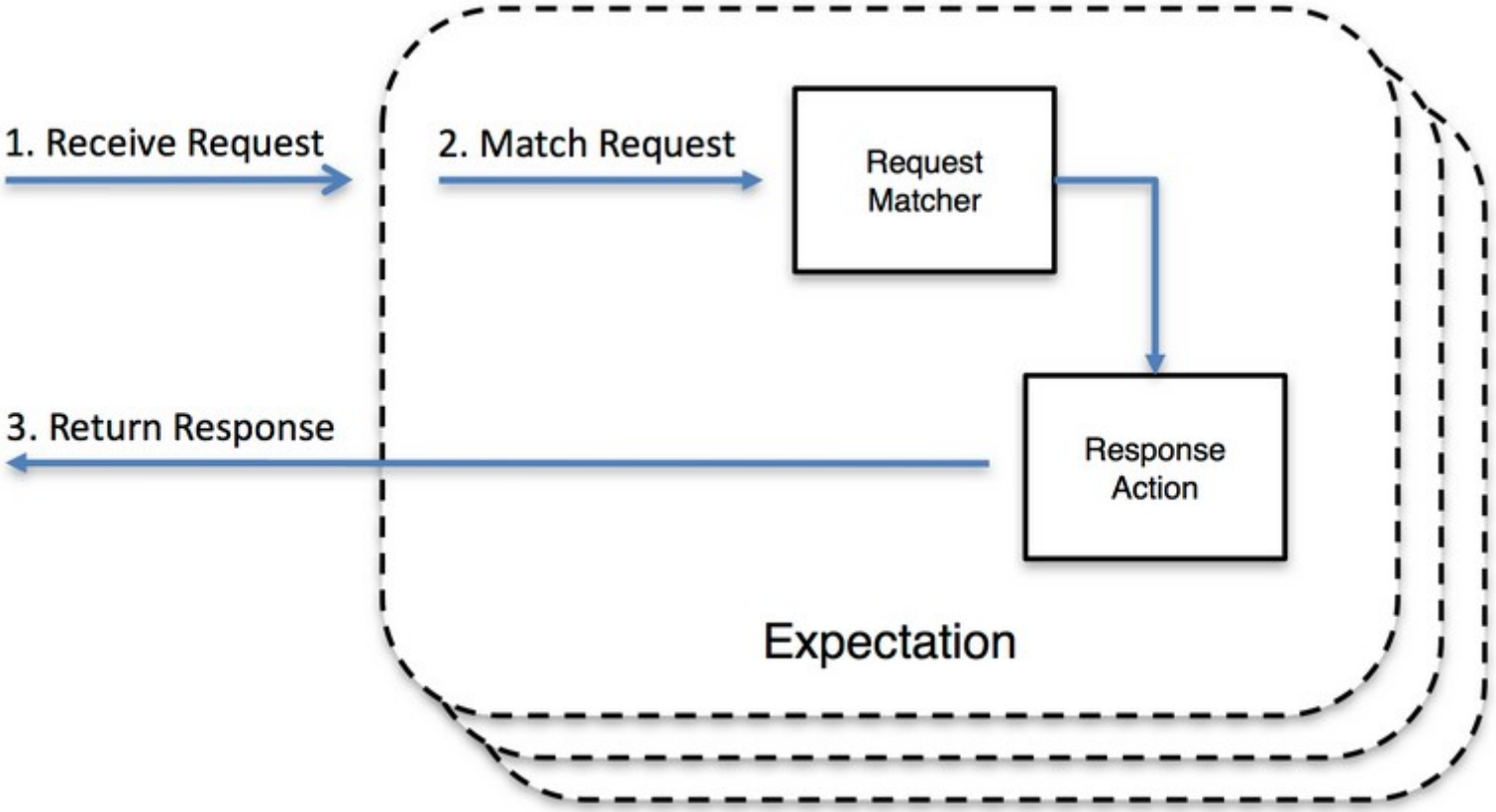


Quelle: mock-server.com

MockServer

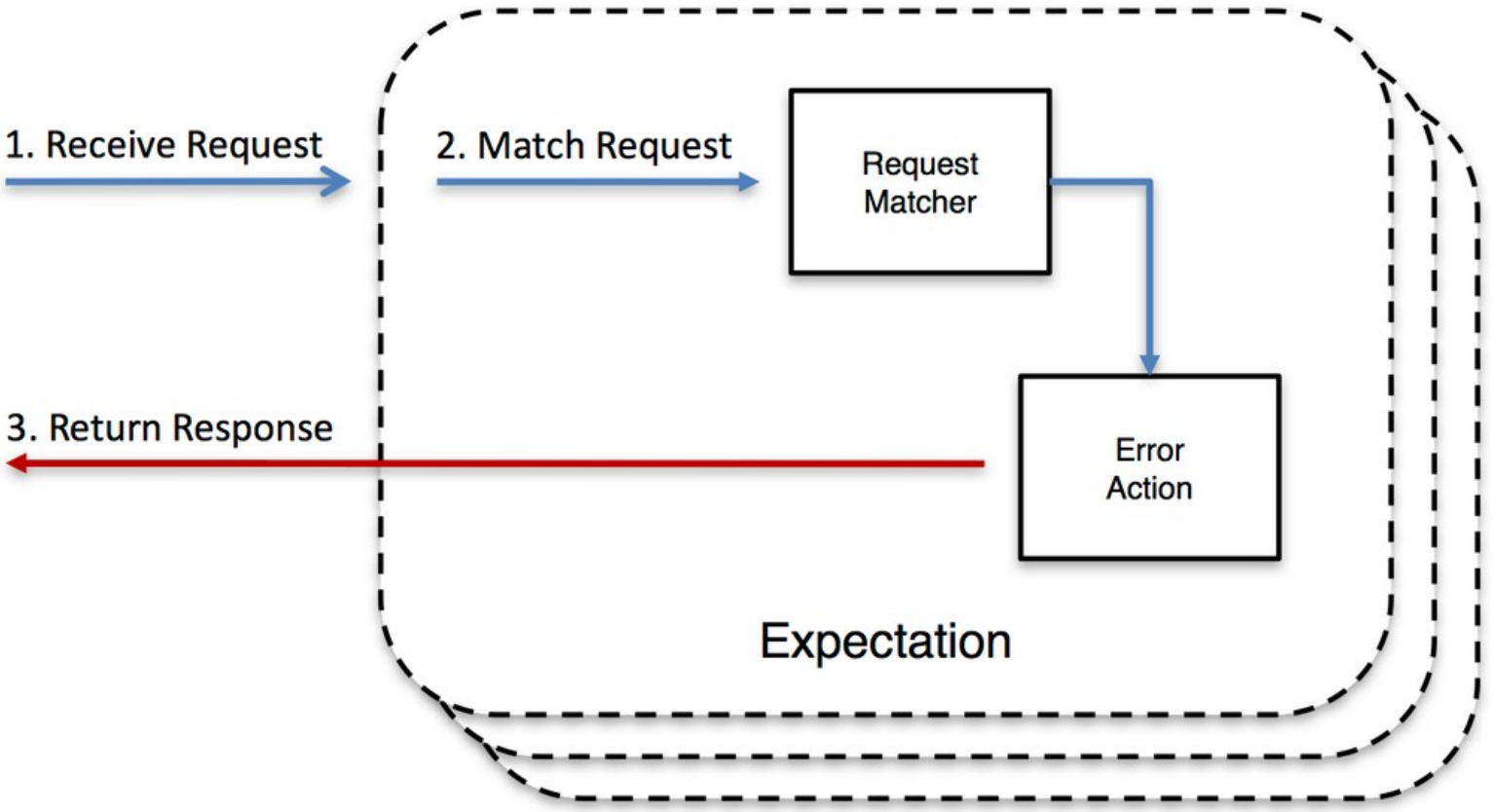
- Java und JavaScript API
- Java Rule
- Maven Plugin
- Npm Plugin
- Grunt Plugin
- Standalone (CLI)
- Docker Container

MockServer



Quelle: mock-server.com

MockServer



Quelle: mock-server.com

```
{
  "count": 37,
  "next": "https://swapi.co/api/starships/?page=2",
  "previous": null,
  "results": [
    {
      "name": "Executor",
      "model": "Executor-class star dreadnought",
      "manufacturer": "Kuat Drive Yards, Fondor Shipyards",
      "cost_in_credits": "1143350000",
      "length": "19000",
      "max_atmosphering_speed": "n/a",
      "crew": "279144",
      "passengers": "38000",
      "cargo_capacity": "250000000",
      "consumables": "6 years",
      "hyperdrive_rating": "2.0",
      "MGLT": "40",
      "starship_class": "Star dreadnought",
      "pilots": [],
      "films": [
        "https://swapi.co/api/films/2/",
        "https://swapi.co/api/films/3/"
      ],
      "created": "2014-12-15T12:31:42.547000Z",
      "edited": "2017-04-19T10:56:06.685592Z",
      "url": "https://swapi.co/api/starships/15/"
    },
```


http://swapi.co/api/starships

```java
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;
    }
}
```

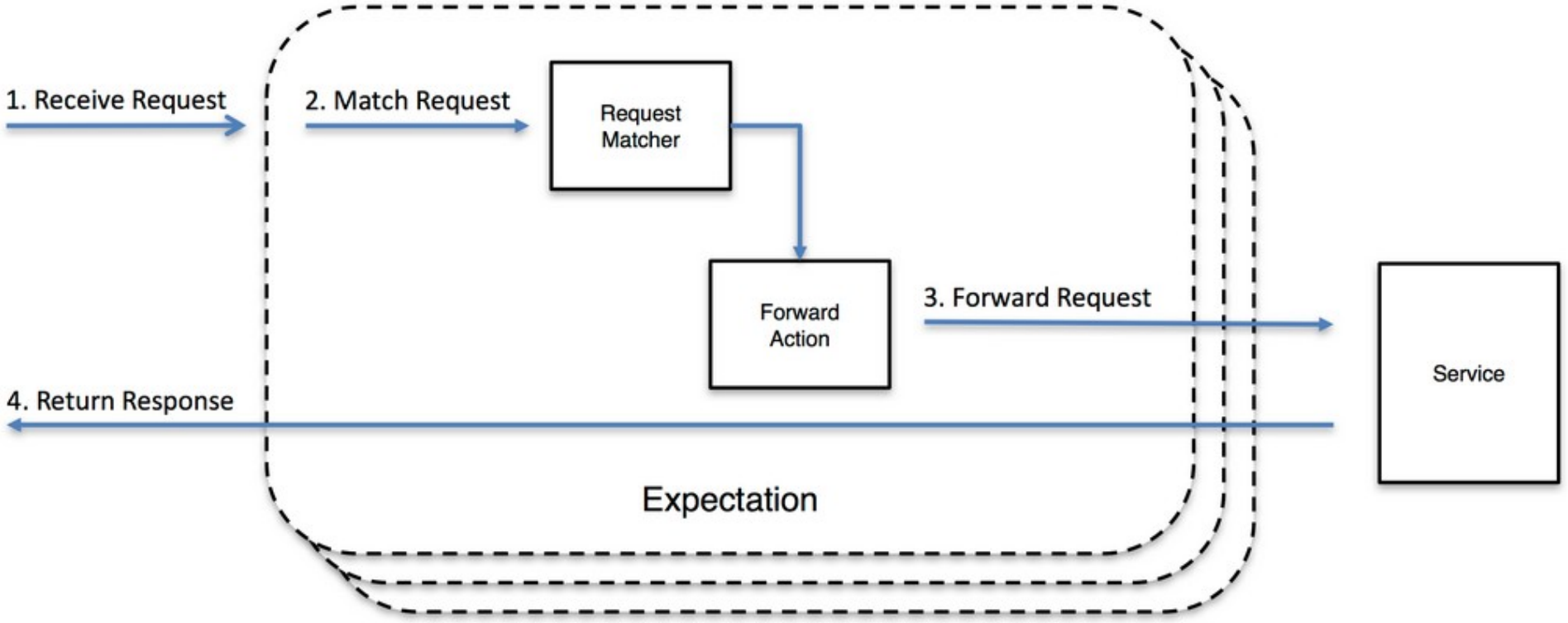
```java
@Rule
public MockServerRule mockServerRule = new MockServerRule( target: this);

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

```
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     }
```

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
    - Given-when-then Struktur

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

```
@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"));
    }
}
```

Rest Assured – Spring MVC

```java
@RunWith(SpringRunner.class)
@WebMvcTest(StarWarsMovieController.class)
public class StarWarsMovieControllerITest {

    @Autowired
    private MockMvc mockMvc;

    @Test
    public void findAllMovies(){
        given().mockMvc(mockMvc).
        when().get( s: "/starwars/movies").
        then().statusCode(200)
                .body( s: "title",  hasItems("A New Hope", "The Force Awakens"))
                .body( s: "director", hasItems("George Lucas", "J. J. Abrams"));
    }

    @Test
    public void findAllMovies_plainMockMvc() throws Exception {
        mockMvc.perform(get( urlTemplate: "/starwars/movies"))
                .andExpect(status().isOk())
                .andExpect(jsonPath( expression: "[0].title").value( expectedValue: "A New Hope"))
                .andExpect(jsonPath( expression: "[0].director").value( expectedValue: "George Lucas"))
                .andExpect(jsonPath( expression: "[1].title").value( expectedValue: "The Force Awakens"))
                .andExpect(jsonPath( expression: "[1].director").value( expectedValue: "J. J. Abrams"));

    }

}
```

Unit Tests für Endpoints

```java
@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"));
    }
}
```

```java
class StarWarsMovieControllerTest {

    @Test
    void findAllMovies(){
        StarWarsMovieController controllerUnderTest = new StarWarsMovieController();
        List<StarWarsMovie> movies = controllerUnderTest.findAllMovies();

        assertThat(movies)
                .hasSize(2)
                .extracting( propertyOrField: "director")
                .containsExactlyInAnyOrder( "George Lucas","J. J. Abrams" );
    }
}
```

Datenbanken

Datenbanken

- Embedded Datenbanken
  - H2, Derby
  - Nicht Produktionsnah
  - Nicht alles testbar
- Standalone Datenbanken
  - Abhängigkeit zur bestimmten Infrastruktur
- Shared Datenbanken
  - Abhängigkeiten zwischen Entwickler
  - Hohe false negative Fehlerrate


TESTCONTAINERS

```java
@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);
    }
}
```

```java
@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();
    }

}
```

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running db.migration.DbMigrationITest
INFO  - ertyClientProviderStrategy - Found docker client settings from environment
INFO  - ckerClientProviderStrategy - 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>,registryEma

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
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
- Weitere spezifische Container – Elasticsearch, Kafka, Apache Pulsar, Mockserver, Toxiproxy, Nginx, Hashicorp Vault
- Generic containers – irgendein Docker Container
- Docker compose – Wiederverwendung von Docker Compose YAML Datei
- Dockerfile containers – Container direkt von einem Dockerfile

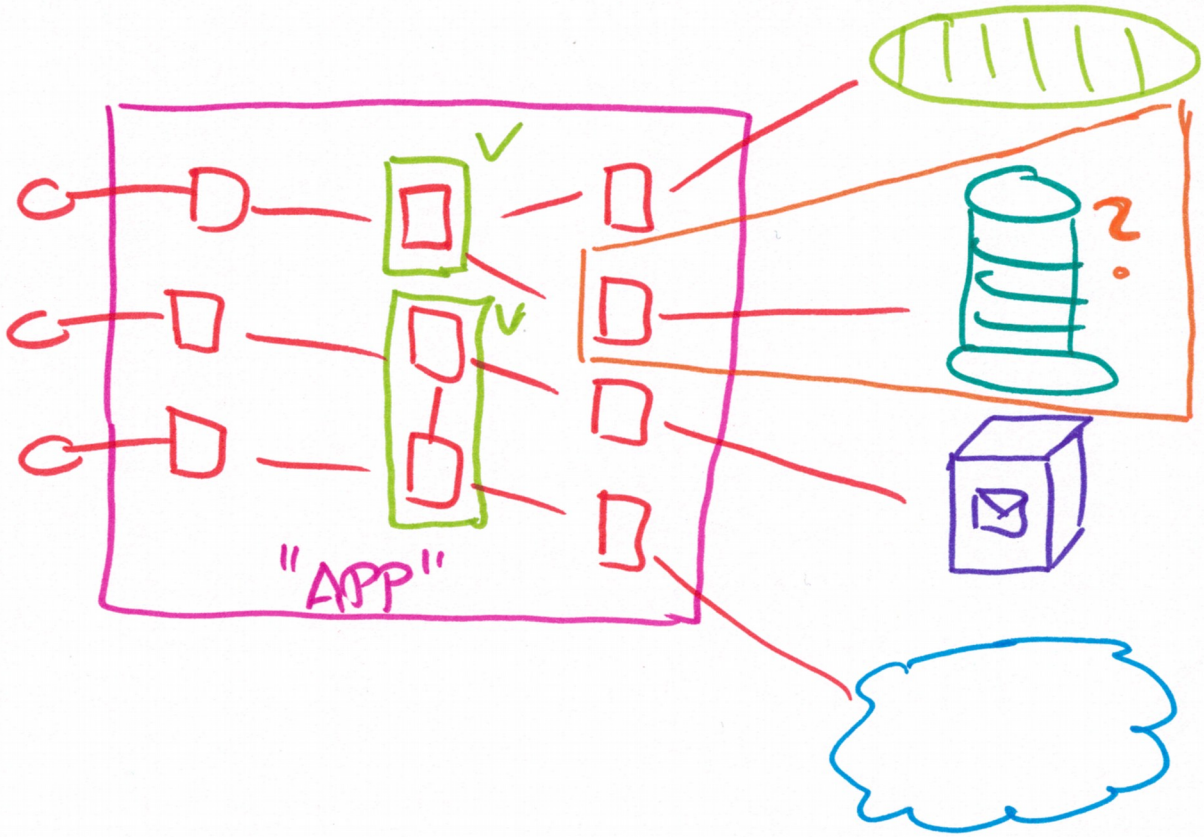
Docker Maven Plugin (DMP)

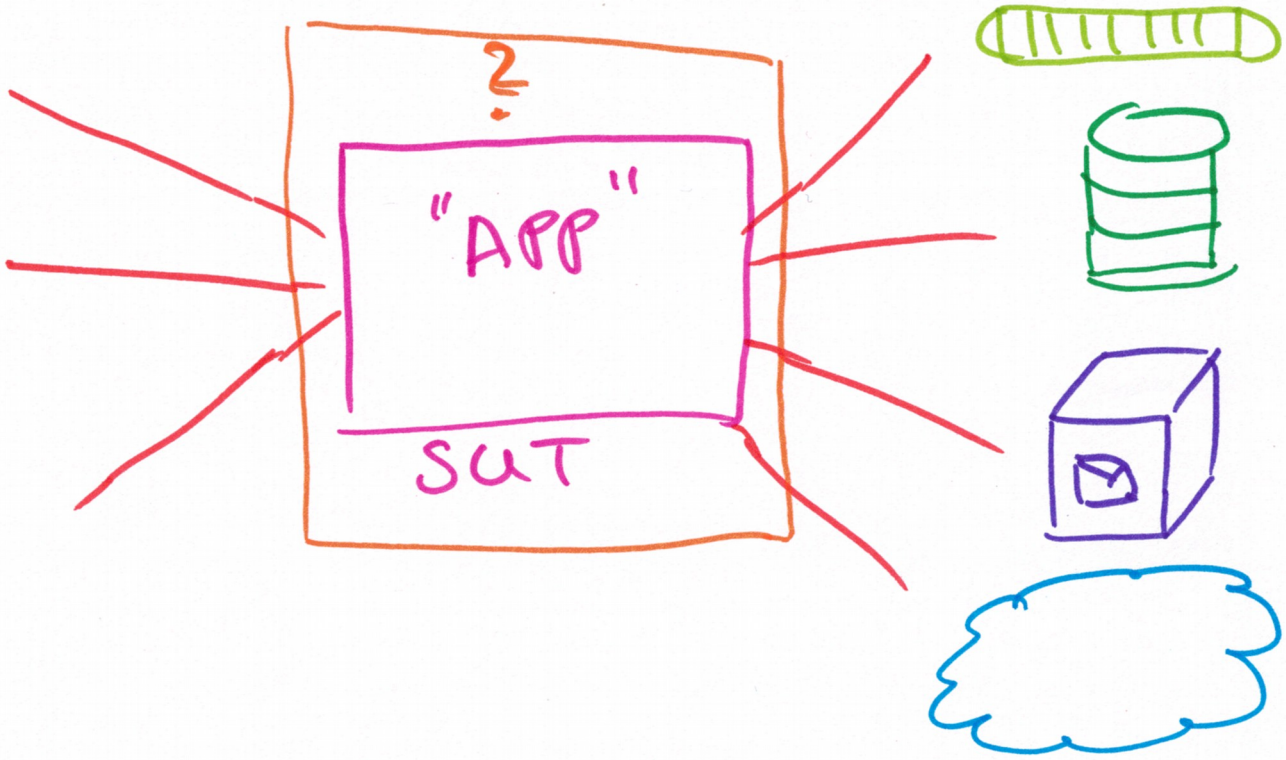
- fabric8io/docker-maven-plugin (http://dmp.fabric8.io)
- Kann Docker Image bauen
- Aber auch Container starten und stoppen

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


V
V
"APP"

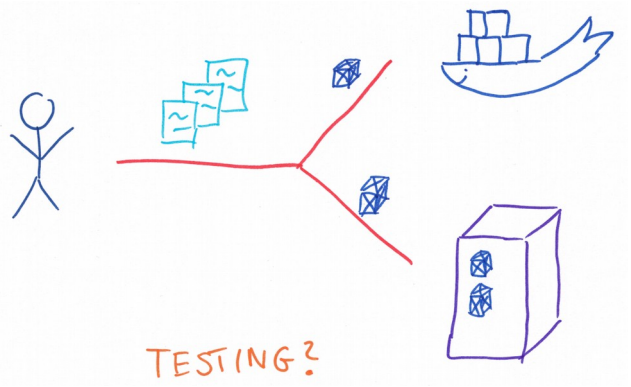

?
"APP"
SUT

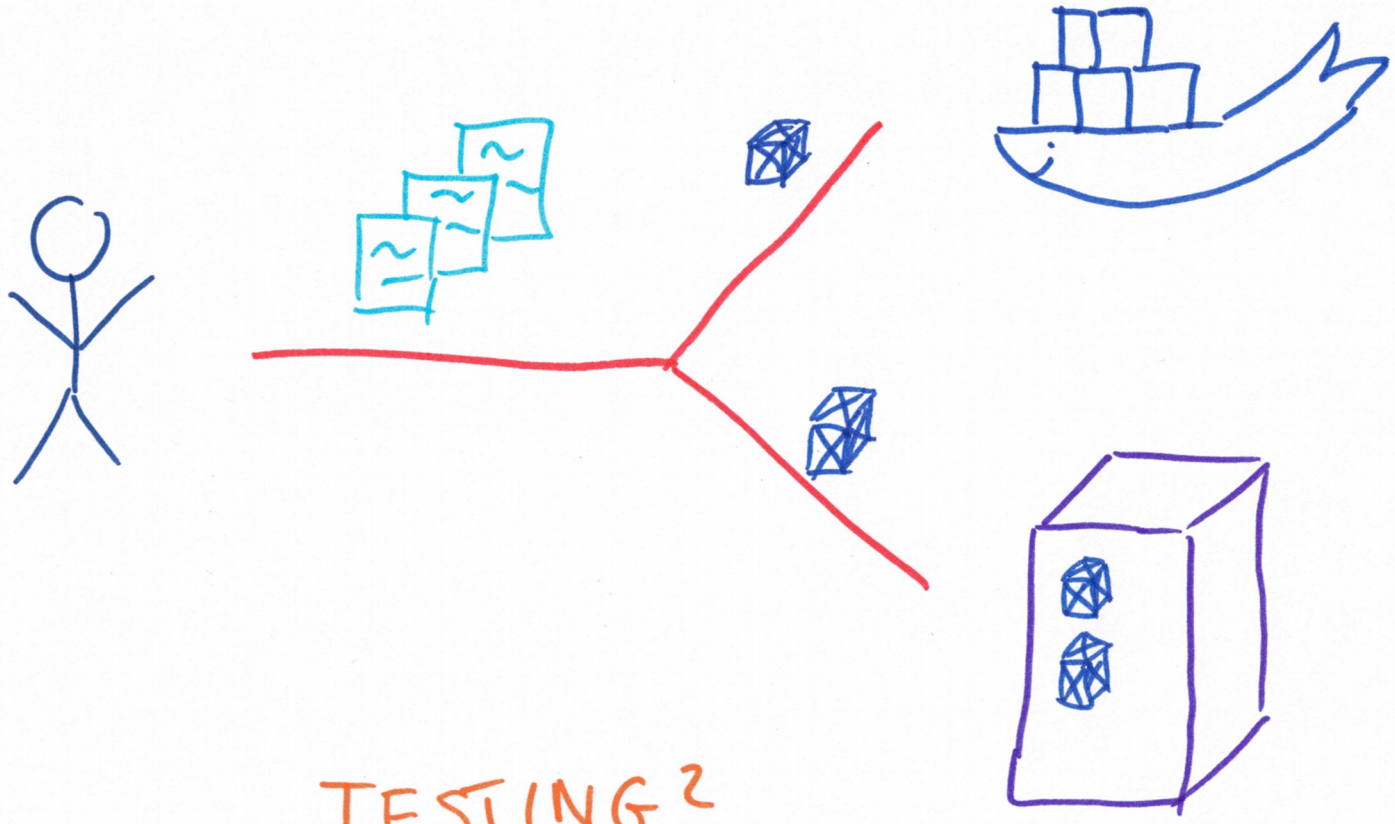
Citrus Integration Testing

Resilience Tests

- ToxyProxy - framework for simulating network conditions (http://toxiproxy.io/)
  - NEW: Testcontainers Integration
- Chaos Monkey for Spring Boot – Chaos Engineering for Spring Boot Apps (https://codecentric.github.io/chaos-monkey-spring-boot)

Testen von Infrastruktur




TESTING?

Infrastructure as Code

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

Provisionierungswerkzeuge (Beispiel Ansible)

- Codequalität
  - Ansible-lint (https://github.com/ansible/ansible-lint)
  - yamllint (Yaml allgemein) (https://github.com/adrienverge/yamllint)
- Funktionale Qualtität
  - Serverspec (Tool unabh.) (https://serverspec.org/)
  - Testinfra (Tool unabh.) (https://testinfra.readthedocs.io/en/latest/)
  - Goss (Tool unabh.) (https://goss.rocks/)

```yaml
- hosts: application-server
  vars:
    tomcat_version: 8.5.8
    tomcat_base_name: apache-tomcat-{{ tomcat_version }}
    #catalina_opts: "-Dkey=value"

  tasks:
    - name: install java
      apt: name=openjdk-8-jdk state=present
      become: yes
      become_method: sudo

    - name: Download current Tomcat 8 version
      local_action: get_url url="http://archive.apache.org/dist/tomcat/tomcat-8/v{{ tomcat_version }}/bin/{{ tomcat_base_name }}.tar.gz" dest=/tmp

    - name:
      file: name=/opt mode=777
      become: yes
      become_method: sudo

    - name: Install Tomcat 8
      unarchive: src=/tmp/{{ tomcat_base_name }}.tar.gz dest=/opt creates=/opt/{{ tomcat_base_name }} owner=vagrant group=vagrant

    - name: Set link to tomcat 8
      file: src=/opt/{{ tomcat_base_name }} dest=/opt/tomcat  state=link force=yes

    - name: setup setenv.sh
      template: dest="/opt/{{ tomcat_base_name }}/bin/setenv.sh" src="roles/tomcat8/templates/setenv.sh.j2" mode=755
      when: catalina_opts is defined

    - find: paths="/opt/{{ tomcat_base_name }}/bin" patterns="*.sh"
      register: result

    - name: ensure tomcat scripts are executable
      file: name={{item.path}} mode=755
      with_items: '{{ result.files }}'

    - name: install tomcat as service
      copy: src=roles/tomcat8/files/tomcat.service dest=/etc/systemd/system/
      become: yes
      become_method: sudo
```

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
```

YAML is everywhere

- Ansible
- Saltstack
- Docker Compose
- Kubernetes
- Goss

yamllint

```
ansible (master*) » yamllint *.yml
setup-app.yml
  1:1       warning  missing document start "---"  (document-start)
  5:6       warning  missing starting space in comment  (comments)
  10:15     warning  truthy value is not quoted  (truthy)
  14:81     error    line too long (146 > 80 characters)  (line-length)
  18:15     warning  truthy value is not quoted  (truthy)
  22:81     error    line too long (129 > 80 characters)  (line-length)
  25:81     error    line too long (82 > 80 characters)  (line-length)
  28:81     error    line too long (102 > 80 characters)  (line-length)
  40:15     warning  truthy value is not quoted  (truthy)

setup-db.yml
  1:1       warning  missing document start "---"  (document-start)
  2:11      warning  truthy value is not quoted  (truthy)
  4:1       error    trailing spaces  (trailing-spaces)
  5:9       error    trailing spaces  (trailing-spaces)
  8:1       error    trailing spaces  (trailing-spaces)
  11:1      error    trailing spaces  (trailing-spaces)
  21:1      error    trailing spaces  (trailing-spaces)
  23:81     error    line too long (83 > 80 characters)  (line-length)
  23:83     error    trailing spaces  (trailing-spaces)
  24:1      error    trailing spaces  (trailing-spaces)
  26:6      error    wrong indentation: expected 4 but found 5  (indentation)
  27:43     error    trailing spaces  (trailing-spaces)
  28:1      error    trailing spaces  (trailing-spaces)
```

testinfra

```python
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")
```


```python
def test_openjdk_is_installed(host):
    openjdk = host.package("openjdk-8-jdk")
    assert openjdk.is_installed

def test_tomcat_service_exists(host):
    assert host.file("/etc/systemd/system/tomcat.service").exists

def test_tomcat_folder_exists(host):
    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

```
pipeline {
    agent any
        stages {
            stage('YAML Syntax check') {
                steps {
                    ansiblePlaybook inventory: 'inventories/test', extras: '--syntax-check', playbook: 'setup-app.yml'
                    ansiblePlaybook inventory: 'inventories/test', extras: '--syntax-check', playbook: 'setup-db.yml'
                }
            }
            stage('Ansible Lint Check') {
                steps {
                    sh 'ansible-lint *.yml'
                }
            }
            stage('Ansible Playbook run with tests') {
                steps {
                    sh 'cd ..; vagrant up'
                    ansiblePlaybook inventory: 'inventories/test', playbook: 'setup-app.yml'
                    ansiblePlaybook inventory: 'inventories/test', playbook: 'setup-db.yml'
                    sh 'py.test --connection=ansible --ansible-inventory inventories/test -v tests/*.py'
                }
            }
        }
    post {
        always {
            sh 'cd ..; vagrant group destroy -f'
        }
    }
}
```

Jenkins Pipeline

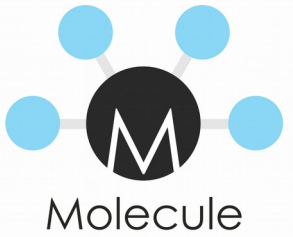
Ansible Playboock run with tests - 1m 8s

✓	vagrant up centos-docker — Shell Script	2s
✓	Invoke an ansible playbook	58s
✓	py.test --connection=ansible --ansible-inventory inventories/localhost_d... — Shell Script	8s
✓	vagrant destroy centos-docker -f — Shell Script	3s

Geht es nicht einfacher?

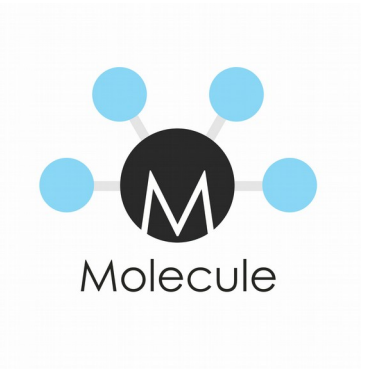
Für Ansible: Molecule

Molecule



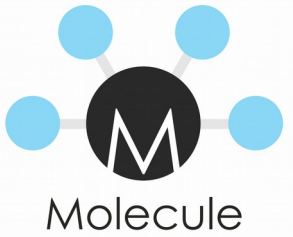
- Spezialisiert für Ansible Roles
- Wrapper um andere Werkzeuge, um komplette Test Szenarien aufzubauen
  - Driver Provider: Docker (default), Vagrant, Azure, EC2
  - Lint Provider: yamllint (default), ansible-lint, flake8 (for test code)
  - Verifier framework: TestInfra (default)

Molecule



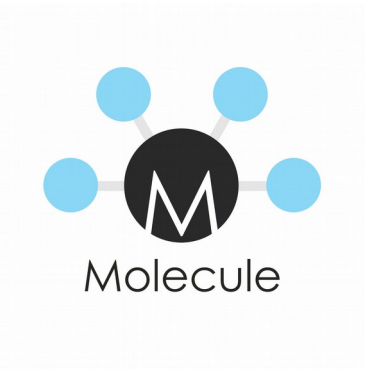

```
.
└── tomcat
    ├── defaults
    │   └── main.yml
    ├── handlers
    │   └── main.yml
    ├── meta
    │   └── main.yml
    ├── molecule
    │   └── default
    │       ├── Dockerfile.j2
    │       ├── molecule.yml
    │       ├── playbook.yml
    │       └── tests
    │           ├── test_default.py
    │           └── test_default.pyc
    ├── README.md
    ├── tasks
    │   └── main.yml
    └── vars
        └── main.yml
```

Molecule




```
roles (master*) » cat tomcat/molecule/default/molecule.yml
---
dependency:
  name: galaxy
driver:
  name: docker
lint:
  name: yamllint
platforms:
  - name: instance
    image: ubuntu:18.04
provisioner:
  name: ansible
  lint:
    name: ansible-lint
verifier:
  name: testinfra
  lint:
    name: flake8
```

Molecule




```
--> Test matrix

└── default
    ├── lint
    ├── cleanup
    ├── destroy
    ├── dependency
    ├── syntax
    ├── create
    ├── prepare
    ├── converge
    ├── idempotence
    ├── side_effect
    ├── verify
    ├── cleanup
    └── destroy
```

Container (Beispiel Docker)

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

```
FROM ubuntu:18.04

RUN apt-get update -y && \
    apt-get install openjdk-8-jre curl -y && \
    curl http://archive.apache.org/dist/tomcat/tomcat-9/v9.0.10/bin/apache-tomcat-9.0.10.tar.gz -o
 /opt/tomcat.tar.gz && \
    tar -xf /opt/tomcat.tar.gz -C /opt && \
    rm -f /opt/tomcat.tar.gz && \
    ln -s /opt/apache-tomcat-9.0.10 /opt/tomcat && \
    apt-get remove curl -y

ENV CATALINA_HOME /opt/tomcat
ENV PATH $CATALINA_HOME/bin:$PATH
WORKDIR $CATALINA_HOME

ENTRYPOINT [ "catalina.sh"]
CMD ["run"]
EXPOSE 8080
```

Hadolint

```
hadolint tomcat.df
tomcat.df:3 DL3008 Pin versions in apt get install. Instead of `apt-get install <pa
ckage>` 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

```yaml
# container structure tests
schemaVersion: "2.0.0"

metadataTest:
  env:
    - key: CATALINA_HOME
      value: /opt/tomcat
  exposedPorts: ["8080"]
  entrypoint: ["catalina.sh"]
  cmd: ["run"]
  workdir: "/opt/tomcat"


fileExistenceTests:

  - name: 'catalina.sh'
    path: '/opt/tomcat/bin/catalina.sh'
    shouldExist: true
    permissions: '-rwxr-x---'

```

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
- https://codewithoutrules.com/2016/07/31/verified-fakes/

Literatur

- Infrastructure as Code: Managing Servers in the Cloud von Kief Morris, O‘Reilly Media