

Herbstcampus, 07.09.17

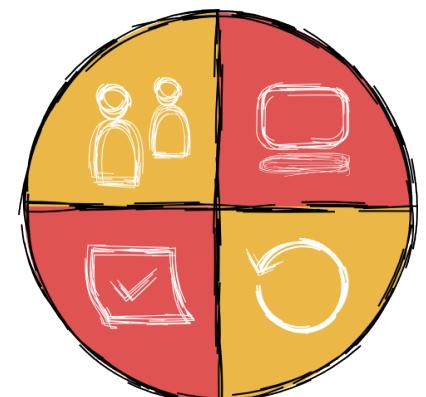
Es muss nicht immer gleich Docker sein
IT Automation, die zu einem passt

Sandra Parsick

mail@sandra-parsick.de
@SandraParsick

Zur 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



Agenda

1. Motivation
2. Container vs. Provisionierungswerzeuge
3. IT Automation, die zu einem passt
4. Wann Container einsetzen?

Motivation



Warum Ansible,
wenn Docker auch
diese Probleme löst?

Ansible for Developers

Aber dafür haben wir
doch Docker



Lass unsere Deployment
mit Ansible automatisieren

Wir wollen doch nächstes
Jahr oder später
Docker einsetzen

Zu aufwendig
zwischendurch
Ansible
einzuführen



Wieso wird Ansible
mit Docker
gleich gesetzt?

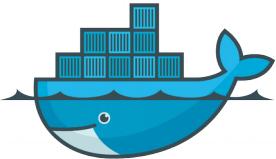
Ist der Unterschied
Container und
Provisionierungswerzeug
nicht klar?

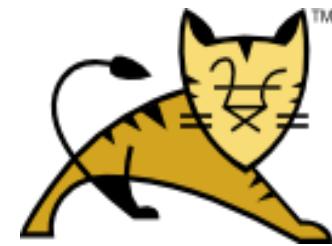
Container vs. Provisionierungswerzeuge

Container - Allgemein

Container

- verpacken Anwendungen und ihre Abhängigkeiten zu einer Einheit
- isolieren diese von anderen Anwendungen
- standardisieren die Art und Weise der Auslieferung von Anwendungen

Container \in {  docker } ?



Provisionierungswerkzeug - Allgemein

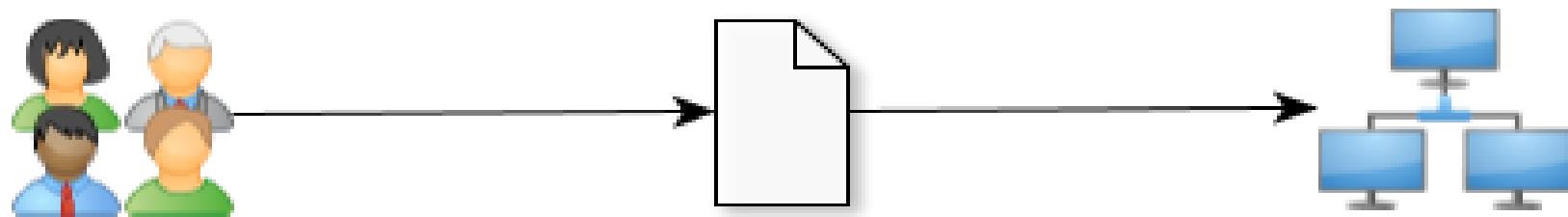
Provisionierungswerkzeug

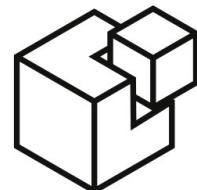
- automatisiert die Provisionierung eines Servers

Server-Provisionierung

- eine Menge an Schritten, um einen Server mit Daten und Software vorzubereiten
 - Resourcen zuweisen und konfigurieren
 - Middleware installieren und konfigurieren
 - Anwendungen installieren und konfigurieren

„Infrastructure As Code“





SALTSTACK



CHEF™



Puppet

CFEngine



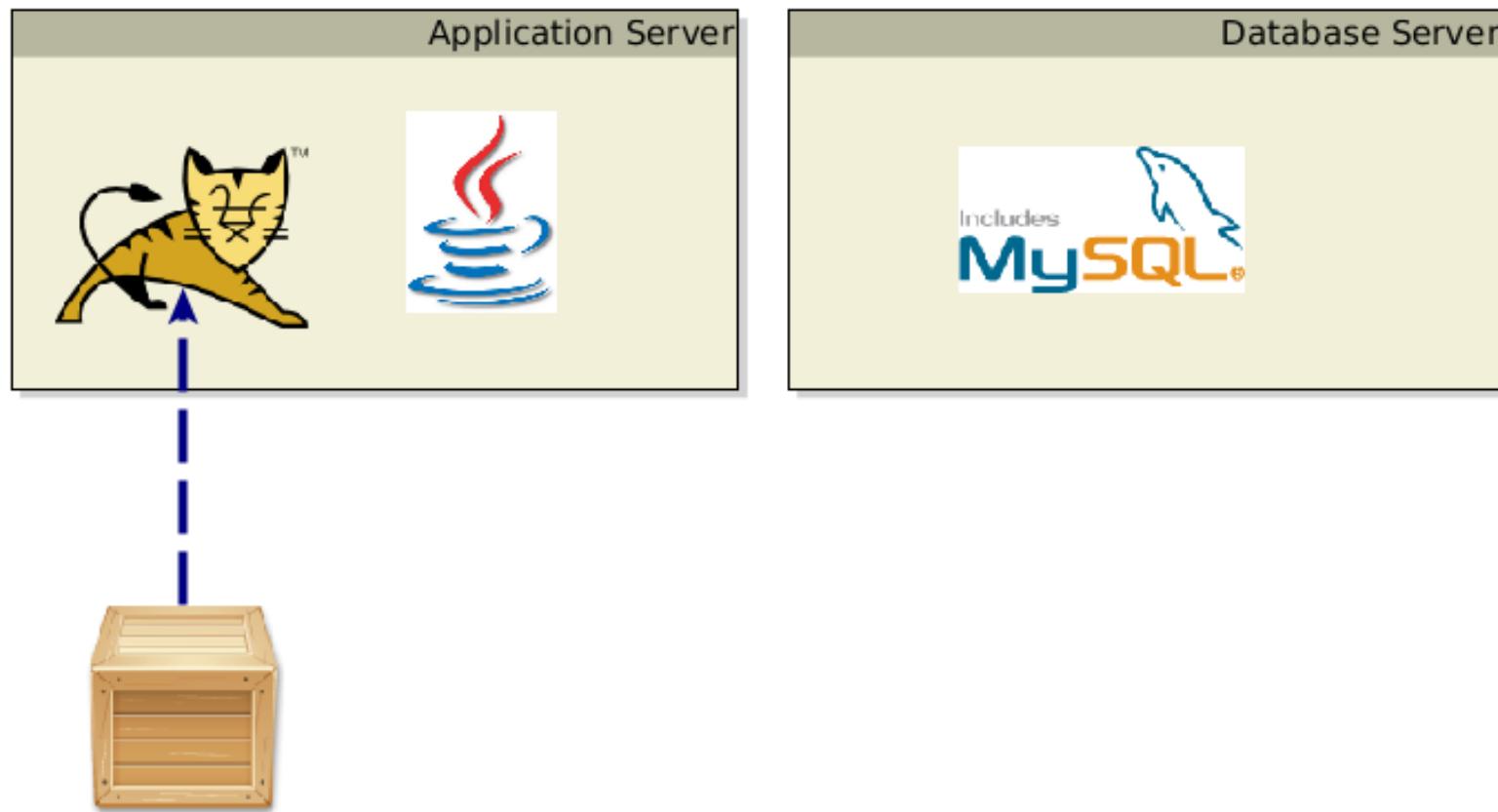
Ansible

IT Automation, die zu einem passt

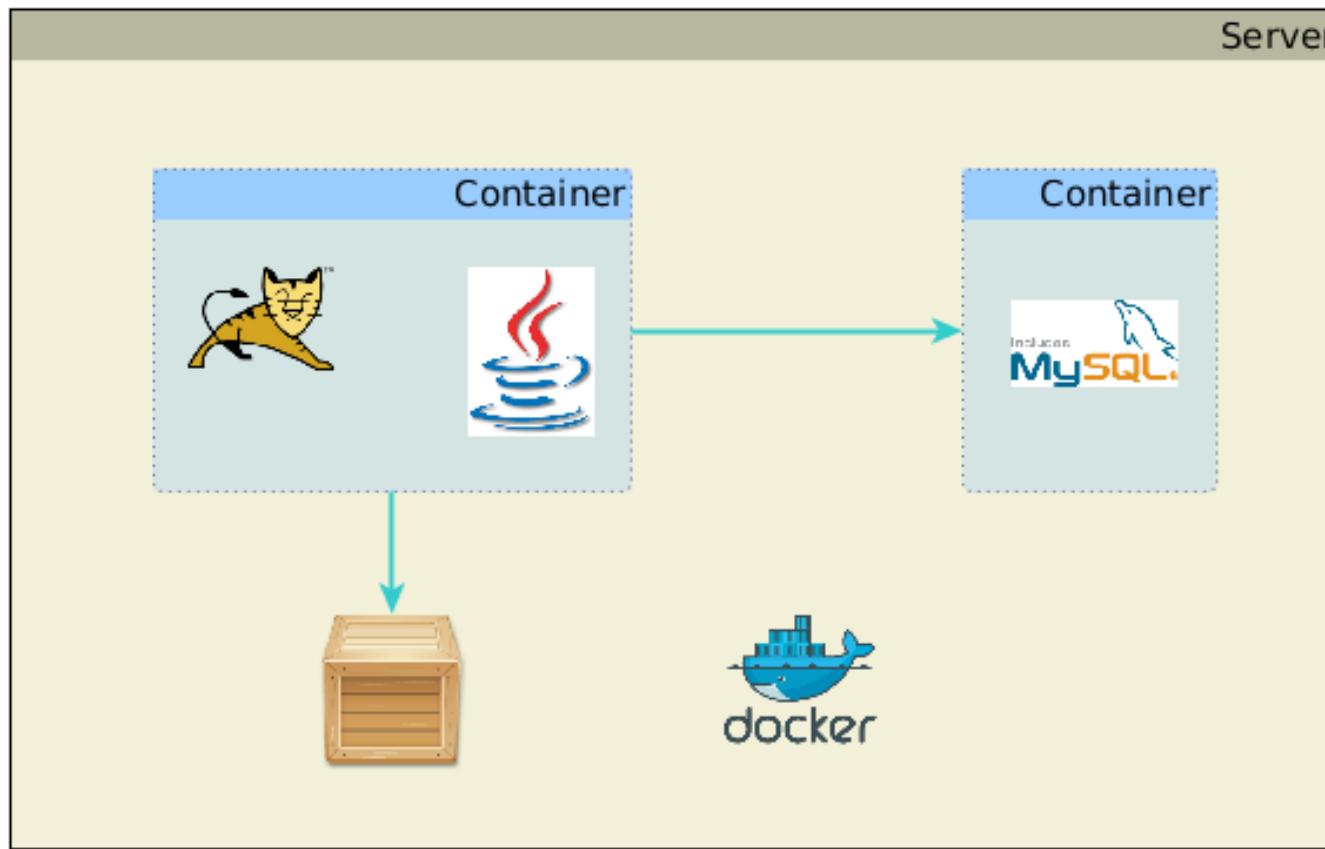
Reiseroute

- ▶ Automatisierung der bestehenden Infrastruktur mit Ansible
- ▶ Wiederverwendung der Ansible Playbooks für die Docker-Image-Erstellung
- ▶ Automatisierung des Docker-Image-Lifecycles mit Ansible
- ▶ Verteilung der Docker-Container auf die Server mit Ansible

Ausgangslage



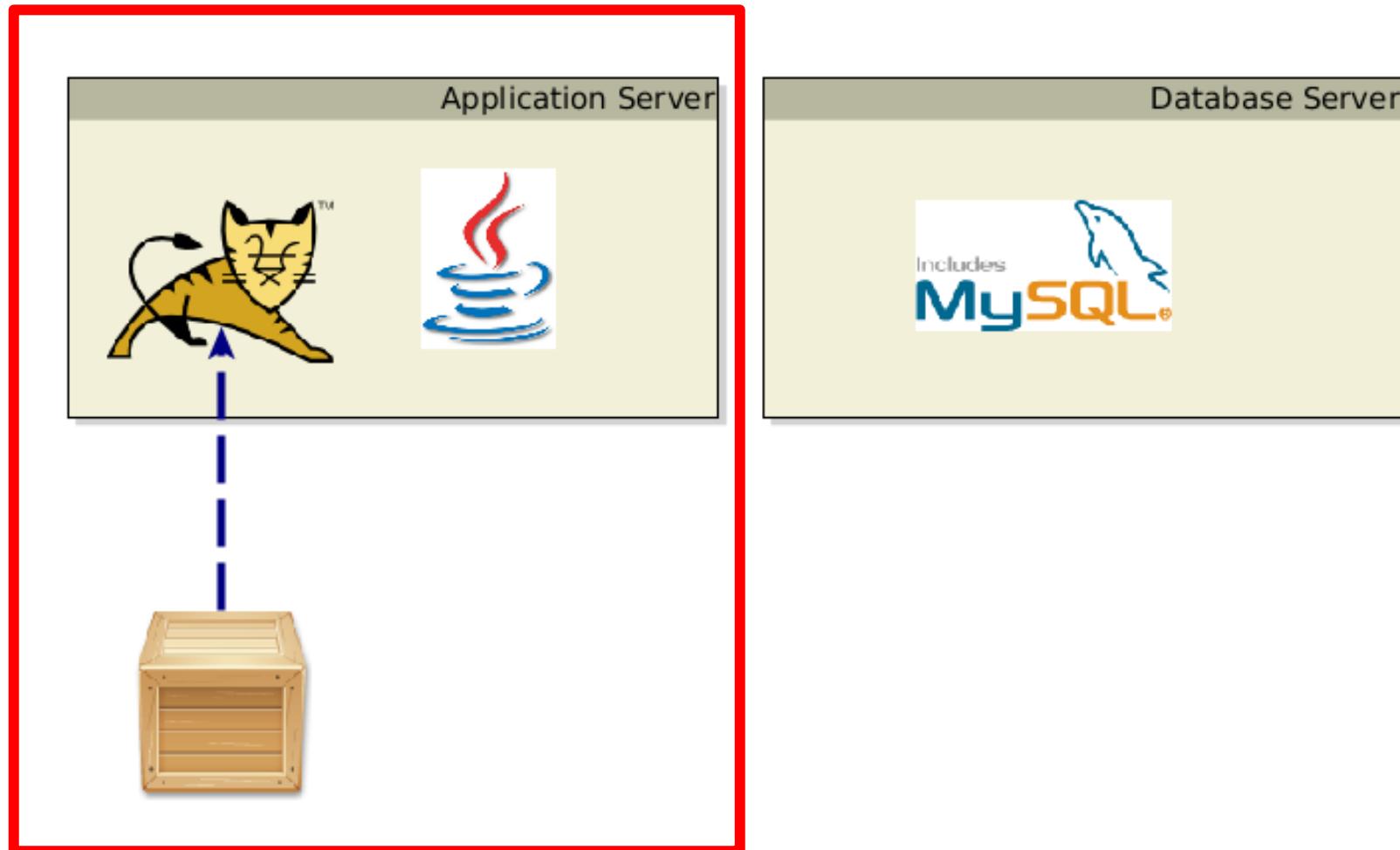
Zielinfrastruktur



Erste Reiseetappe

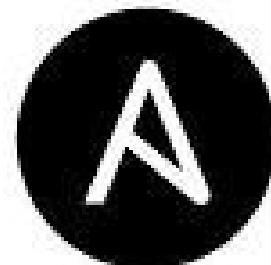
Automatisierung der bestehenden Infrastruktur mit Ansible

Ausgangslage

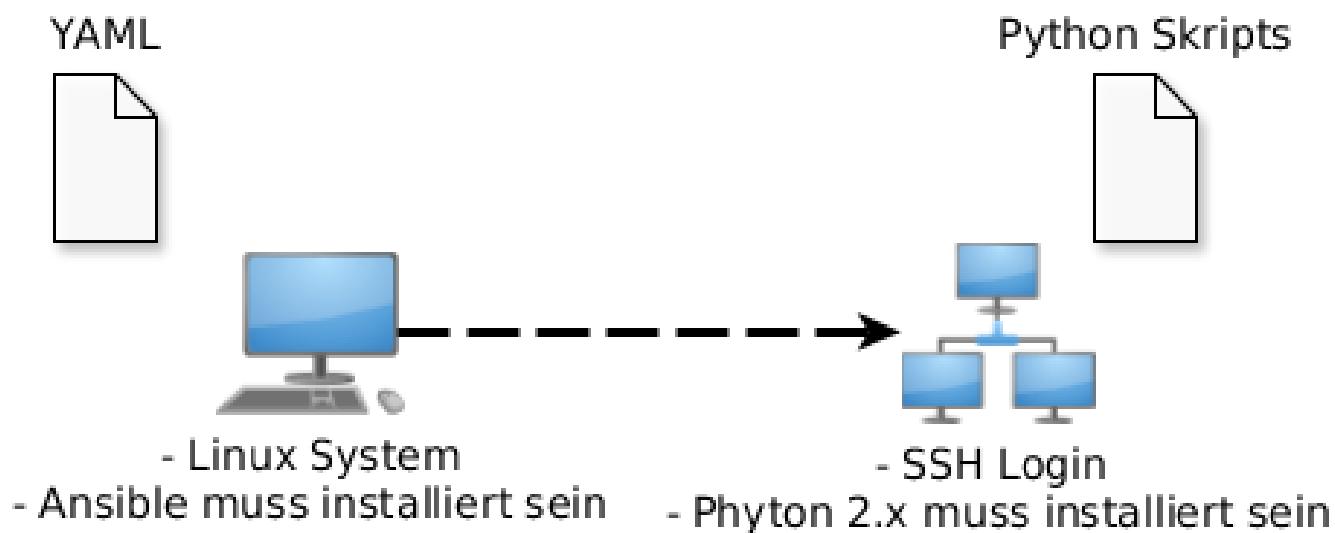


Ansible

- Provisionierungswerkzeug
- Sprache: Python
- Ansible Skripte (genannt *Playbooks*): YAML



Funktionsweise



Exkurs: YAML

YAML

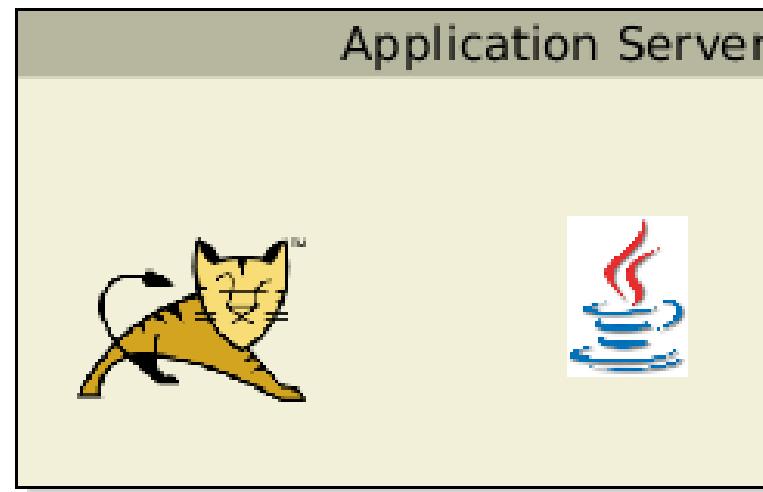
```
---
```

```
foo: "bar"
baz:
  - "qux"
  - "quxx"
corge: null
grault: 1
garply: true
waldo: "false"
fred: "undefined"
emptyArray: []
emptyObject: {}
emptyString: ""
```

JSON

```
{
  "foo": "bar",
  "baz": [
    "qux",
    "quxx"
  ],
  "corge": null,
  "grault": 1,
  "garply": true,
  "waldo": "false",
  "fred": "undefined",
  "emptyArray": [],
  "emptyObject": {},
  "emptyString": ""
}
```

Setup Application Server Playbook



```
1 - hosts: application-server
2   vars:
3     tomcat_version: 8.5.12
4     tomcat_base_name: apache-tomcat-{{ tomcat_version }}
5
6   tasks:
7     - name: ensure group vagrant exists
8       group: name=vagrant state=present
9
10    - name: ensure user vagrant exists
11      user: name=vagrant group=vagrant state=present
12
13    - name: install java
14      apt: name=openjdk-8-jdk state=present
15      become: yes
16      become_method: sudo
17
18    - name: Download current Tomcat 8 version
19      local_action: get_url url="http://archive.apache.org/dist/tomcat/tomcat-8/v{{ tomcat_version }}/bin/{{ tomcat_base_name }}.tar.gz" dest=/tmp
20
21    - name:
22      file: name=/opt mode=777
23      become: yes
24      become_method: sudo
25
26    - name: Install Tomcat 8
27      unarchive: src=/tmp/{{ tomcat_base_name }}.tar.gz dest=/opt creates=/opt/{{ tomcat_base_name }} owner=vagrant group=vagrant
28
29    - name: Set link to tomcat 8
30      file: src=/opt/{{ tomcat_base_name }} dest=/opt/tomcat state=link force=yes
31
32    - find: paths="/opt/{{ tomcat_base_name }}/bin" patterns="*.sh"
33      register: result
34
35    - name: ensure tomcat scripts are executable
36      file: name={{item.path}} mode=755
37      with_items: '{{ result.files }}'
38
39    - name: install tomcat as service
40      copy: src=tomcat.service dest=/etc/systemd/system/
41      become: yes
42      become_method: sudo
43
```

Inventories

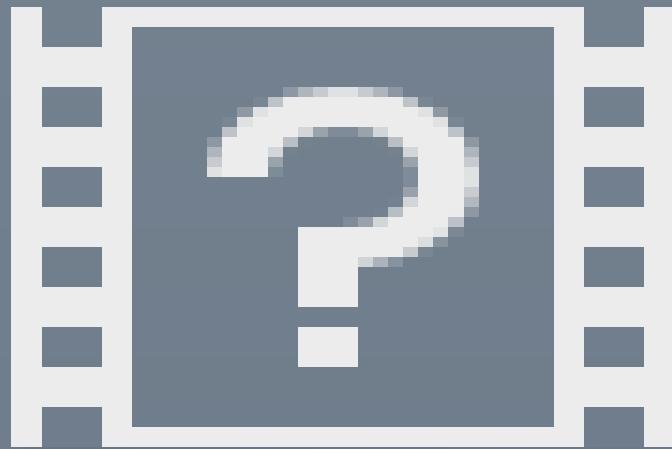
Production

```
1 [application-server]
2 192.168.33.10
3 ubuntu-server db_host=mysql01
4
5 [mysql-db-server]
6 mysql[01:10]
7
8 [oracle-db-server]
9 db-[a:f].oracle.company.com
10
11 [database-server:children]
12 mysql-db-server
13 oracle-db-server
14 |
15 [application-server:vars]
16 message="Welcome"
17
18 [database-server:vars]
19 message="Hello World!"
```

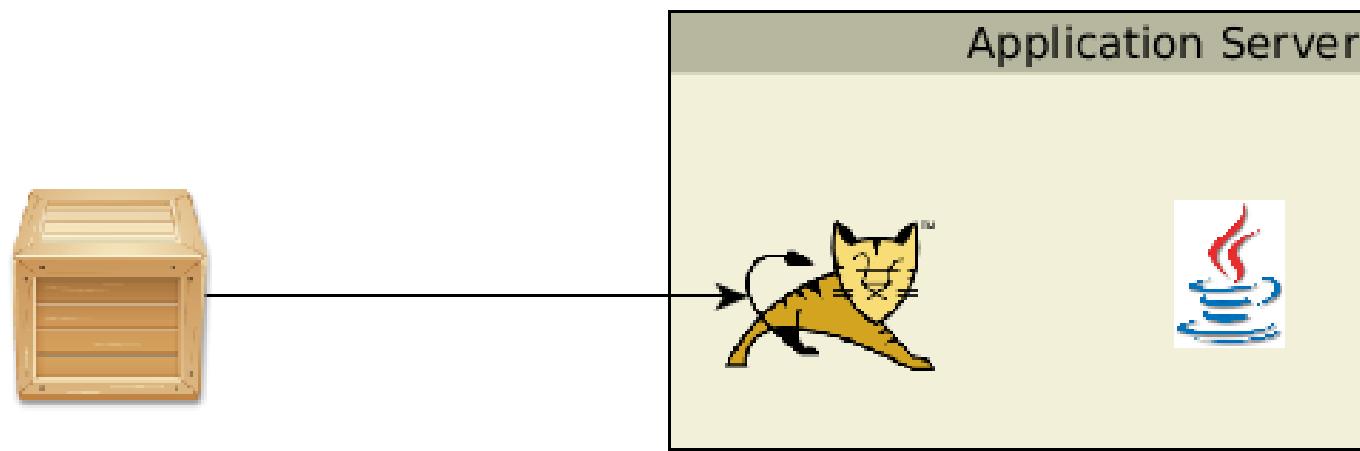
Test

```
1 [application-server]
2 192.168.33.10|
3
4 [database-server]
5 192.168.33.10
6
```

Setup Application Server Playbook



Java Webapplikation Deployment



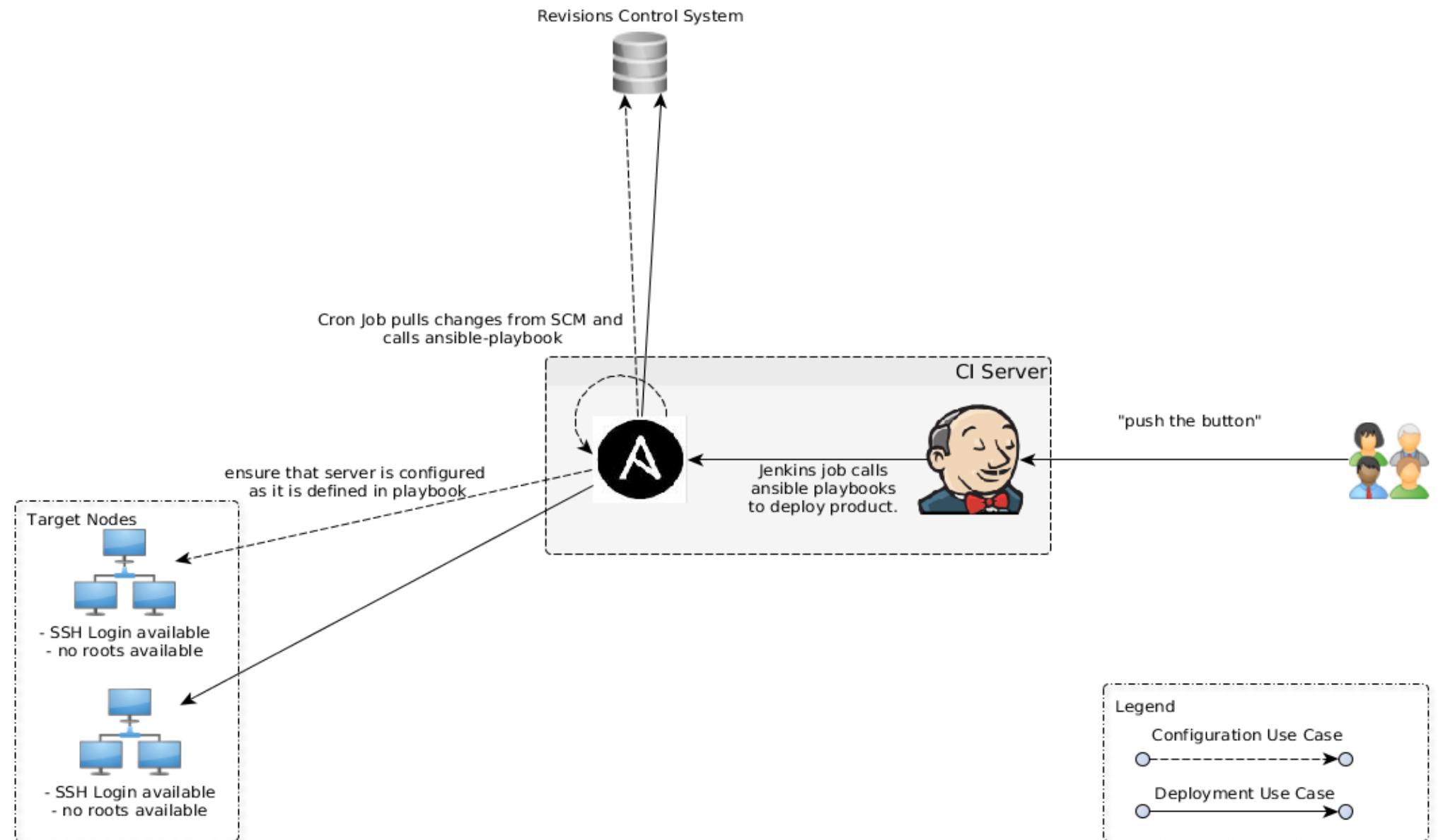
Deploy Application Playbook

```
1 - hosts: application-server
2 vars:
3   webapp_source_path: ./demo-web-application-1.war
4   webapp_target_name: demo
5   tomcat_app_base: /opt/tomcat/webapps
6 tasks:
7   - name: stop tomcat
8     service: name=tomcat state=stopped
9     become: true
10
11  - name: wait tomcat shutdown
12    wait_for: port=8080 state=stopped timeout=60
13
14  - name: cleanup {{ webapp_target_name }}
15    file: name={{tomcat_app_base}}/{{ webapp_target_name }} state=absent
16
17  - name: delete previous backup
18    file: path={{ tomcat_app_base }}/{{ webapp_target_name }}.war.previous state=absent
19
20  - name: create new backup
21    command: mv {{ tomcat_app_base }}/{{ webapp_target_name }}.war {{ tomcat_app_base }}/{{ webapp_target_name }}.war.previous
22    ignore_errors: yes
23
24  - name: copy webapp {{ webapp_source_path }} to {{ webapp_target_name }}
25    copy: src={{ webapp_source_path }} dest={{ tomcat_app_base }}/{{ webapp_target_name }}.war mode=660
26
27  - name: start tomcat
28    service: name=tomcat enabled=yes state=started
29    become: true
30
31  - name: wait for tomcat to start
32    wait_for: port=8080 timeout=60
33
```

Deploy Application Playbook



Ansible Infrastruktur



Wie werden Ansible Skripte getestet?

- ansible-playbook --check
- ansible-playbook --syntax-check
- Jenkins + Vagrant
- Rspec tests



ServerSpec Tests

```
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/systemd/system/tomcat.service') 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
14
```

ServerSpec Tests

```
plain-ansible : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
sparsick@sparsick-ThinkPad-T460s ~/dev/NetBeansProjects/ansible-docker-talk/plain-ansible $ rake spec
/usr/bin/ruby2.3 -I/var/lib/gems/2.3.0/gems/rspec-support-3.6.0/lib:/var/lib/gems/2.3.0/gems/rspec-core-3.6.0/lib /var/lib/gems/2.3.0/gems/rspec-core-3.6.0/exe/rspec --pattern spec/ansible_demo/\*_spec.rb
...
Finished in 1.5 seconds (files took 6.24 seconds to load)
3 examples, 0 failures

sparsick@sparsick-ThinkPad-T460s ~/dev/NetBeansProjects/ansible-docker-talk/plain-ansible $ rake spec
/usr/bin/ruby2.3 -I/var/lib/gems/2.3.0/gems/rspec-support-3.6.0/lib:/var/lib/gems/2.3.0/gems/rspec-core-3.6.0/lib /var/lib/gems/2.3.0/gems/rspec-core-3.6.0/exe/rspec --pattern spec/ansible_demo/\*_spec.rb
..F

Failures:

1) Command "ls /opt/tomcat1" exit_status should eq 0
On host `ansible_demo'
Failure/Error: its(:exit_status) { should eq 0 }

  expected: 0
  got: 2

  (compared using ==)
  sudo -p 'Password: ' /bin/sh -c ls\ /opt/tomcat1
# ./spec/ansible_demo/app_spec.rb:12:in `block (2 levels) in <top (required)>'

Finished in 1.54 seconds (files took 6.73 seconds to load)
3 examples, 1 failure

Failed examples:

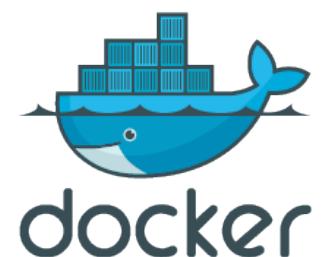
rspec ./spec/ansible_demo/app_spec.rb:12 # Command "ls /opt/tomcat1" exit_status should eq 0
/usr/bin/ruby2.3 -I/var/lib/gems/2.3.0/gems/rspec-support-3.6.0/lib:/var/lib/gems/2.3.0/gems/rspec-core-3.6.0/lib /var/lib/gems/2.3.0/gems/rspec-core-3.6.0/exe/rspec --pattern spec/ansible_demo/\*_spec.rb failed
sparsick@sparsick-ThinkPad-T460s ~/dev/NetBeansProjects/ansible-docker-talk/plain-ansible $ █
```

Zweite Reiseroute

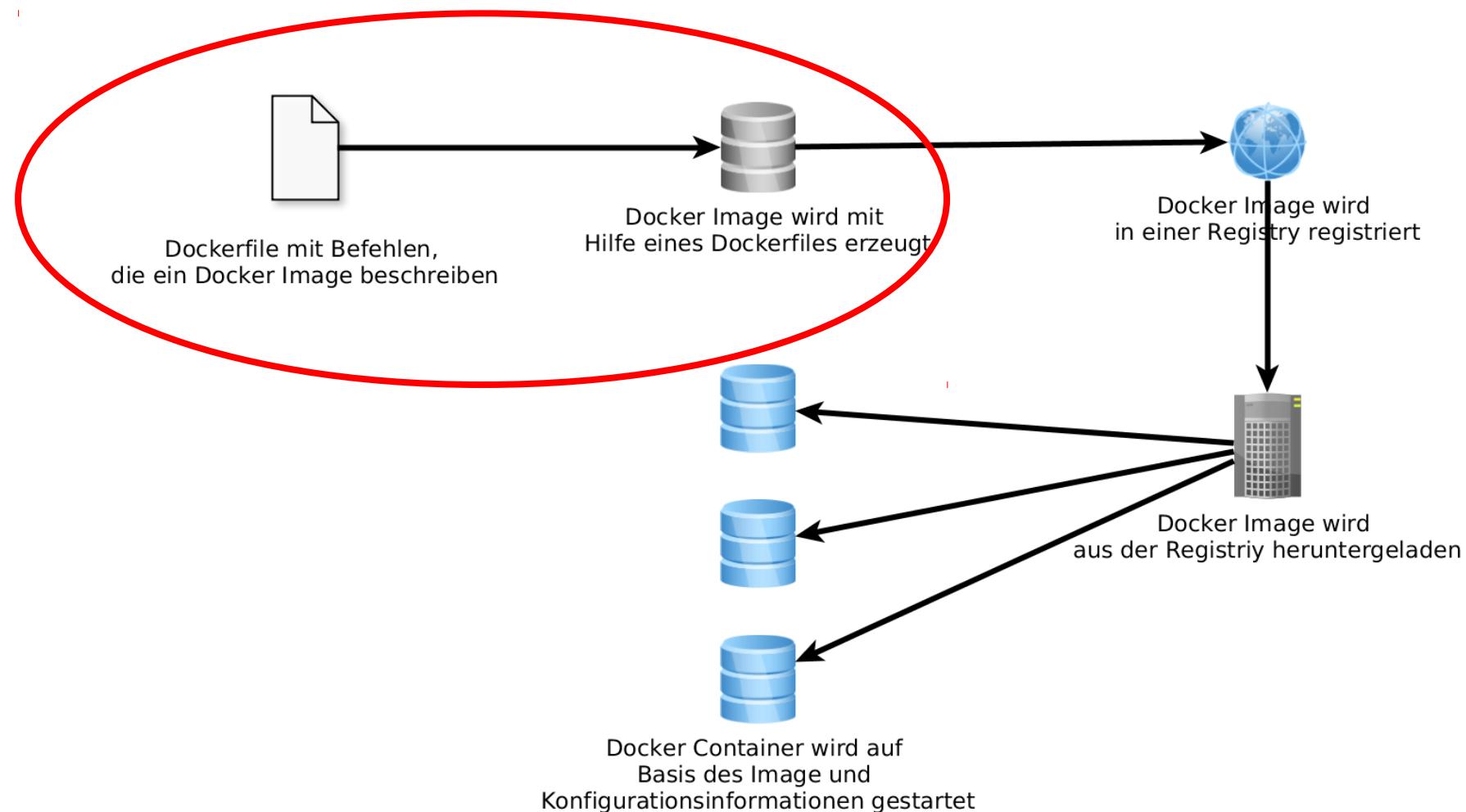
Wiederverwendung der Ansible Playbooks für die Docker-Image- Erstellung

Docker

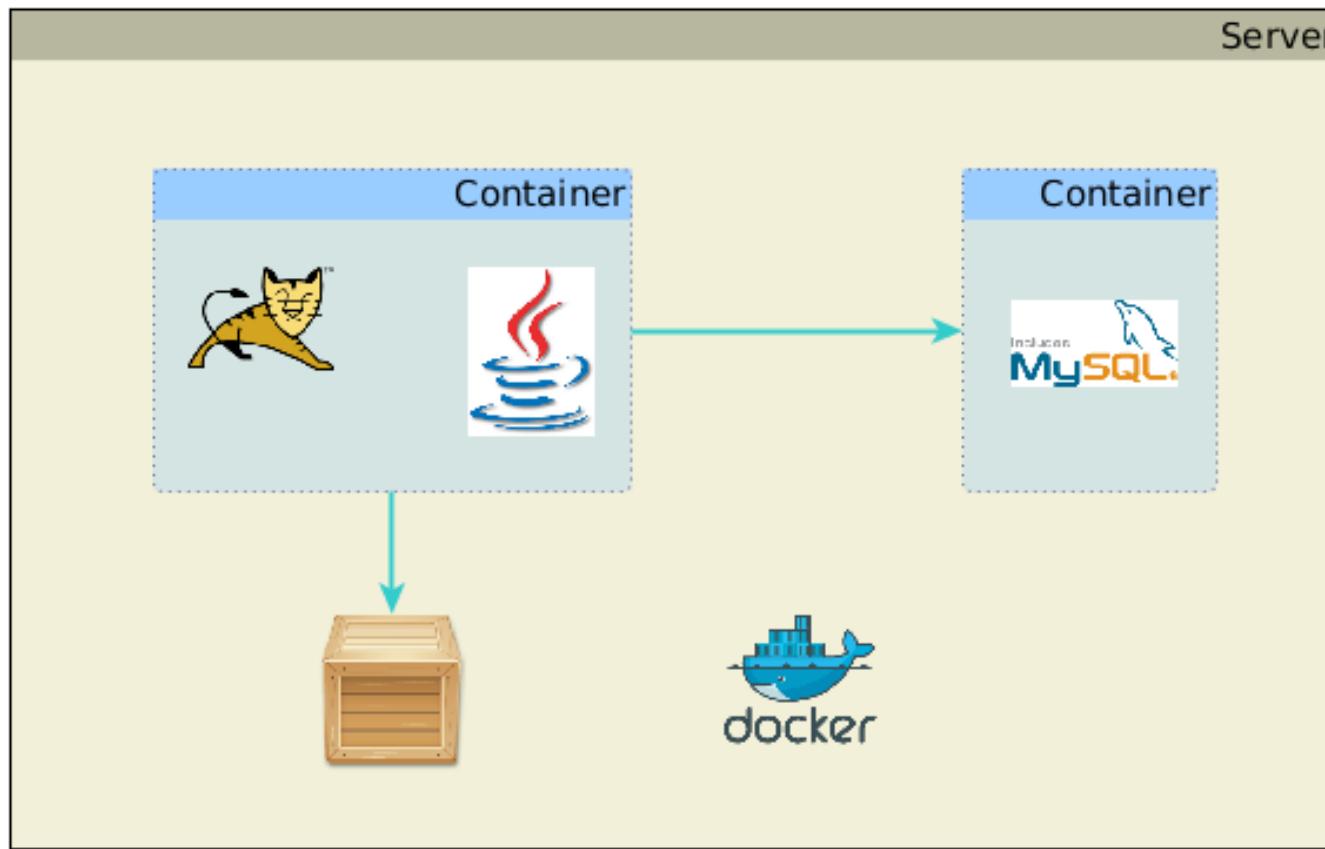
- Verwaltungswerkzeug für Container
- Weitere Werkzeuge aus dem Docker Universum (Auszug):
 - Docker Compose: Hilft beim Definieren und beim Laufen von Multi-Container Anwendungen
 - Docker Registry: Repository Manager für Docker Images



Docker Lifecycle



Reminder - Zielinfrastruktur



Was brauchen wir?

- Zwei Dockerfiles
 - Mysql.df
 - Tomcat.df
- WAR Datei wird über Volume eingebunden

Dockerfile ohne Ansible (tomcat.df)

```
1 FROM ubuntu:16.04
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-8/v8.5.13/bin/apache-tomcat-8.5.13.tar.gz -o /opt/tomcat.tar.gz && \
6     tar -xf /opt/tomcat.tar.gz -C /opt && \
7     rm -f /opt/tomcat.tar.gz && \
8     ln -s /opt/apache-tomcat-8.5.13 /opt/tomcat && \
9     apt-get remove curl -y
10
11 ENV CATALINA_HOME /opt/tomcat
12 ENV PATH $CATALINA_HOME/bin:$PATH
13 WORKDIR $CATALINA_HOME
14
15 ENTRYPOINT [ "catalina.sh"]
16 CMD ["run"]
17 EXPOSE 8080
18
```

Dockerfile mit Ansible (tomcat.df)

```
1 FROM williamyeh/ansible:ubuntu16.04
2
3 WORKDIR /tmp
4 COPY plain-ansible/* /tmp/
5
6 RUN echo application-server > inventory
7 RUN ansible-playbook -i inventory setup-app.yml --connection=local
8
9 ENV CATALINA_HOME /opt/tomcat
10 ENV PATH $CATALINA_HOME/bin:$PATH
11 WORKDIR $CATALINA_HOME
12
13 ENTRYPOINT [ "catalina.sh"]
14 CMD [ "run"]
15 EXPOSE 8080
16
```

Dockerfile mit Ansible (mysql.df)

```
1 FROM williamyeh/ansible:ubuntu16.04
2
3 WORKDIR /tmp
4 COPY plain-ansible/* /tmp/
5
6 RUN echo database-server > inventory
7 RUN ansible-playbook -i inventory setup-db.yml --connection=local
8
9 USER mysql
10 ENTRYPOINT ["/usr/bin/mysqld_safe"]
11 EXPOSE 3306
12
```

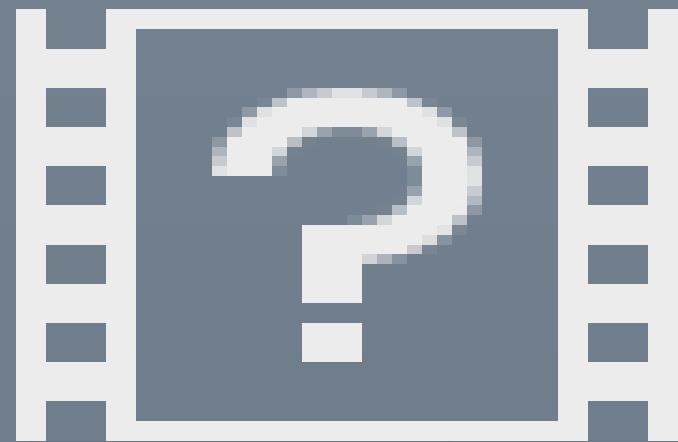
Docker Lifecycle Demo



Es geht auch kürzer – Docker Compose

```
1 version: '3'  
2 services:  
3   database:  
4     image: "sparsick/mysql:ansible"  
5   tomcat:  
6     image: sparsick/tomcat:ansible  
7     ports:  
8       - "8080:8080"  
9     links:  
10    - database  
11  volumes:  
12    - ./plain-ansible/demo-web-application-1.war:/opt/tomcat/webapps/demo.war  
13 |
```

Docker Compose Demo



Dockerfile mit Ansible - Pros und Cons

- ✓ Vorhandene Skripte können wiederverwendet werden
 - ✗ Es werden Abhängigkeiten in das Image mit gepackt, die die Anwendung nicht braucht
 - Image wird unnötig groß
 - ✗ Alternative: Ansible + Python (De)-Installation als RUN Schritt
 - Image-Build-Dauer erhöht sich
- ✓ Reicht aus um generell seine Anwendung im Docker Container zu testen

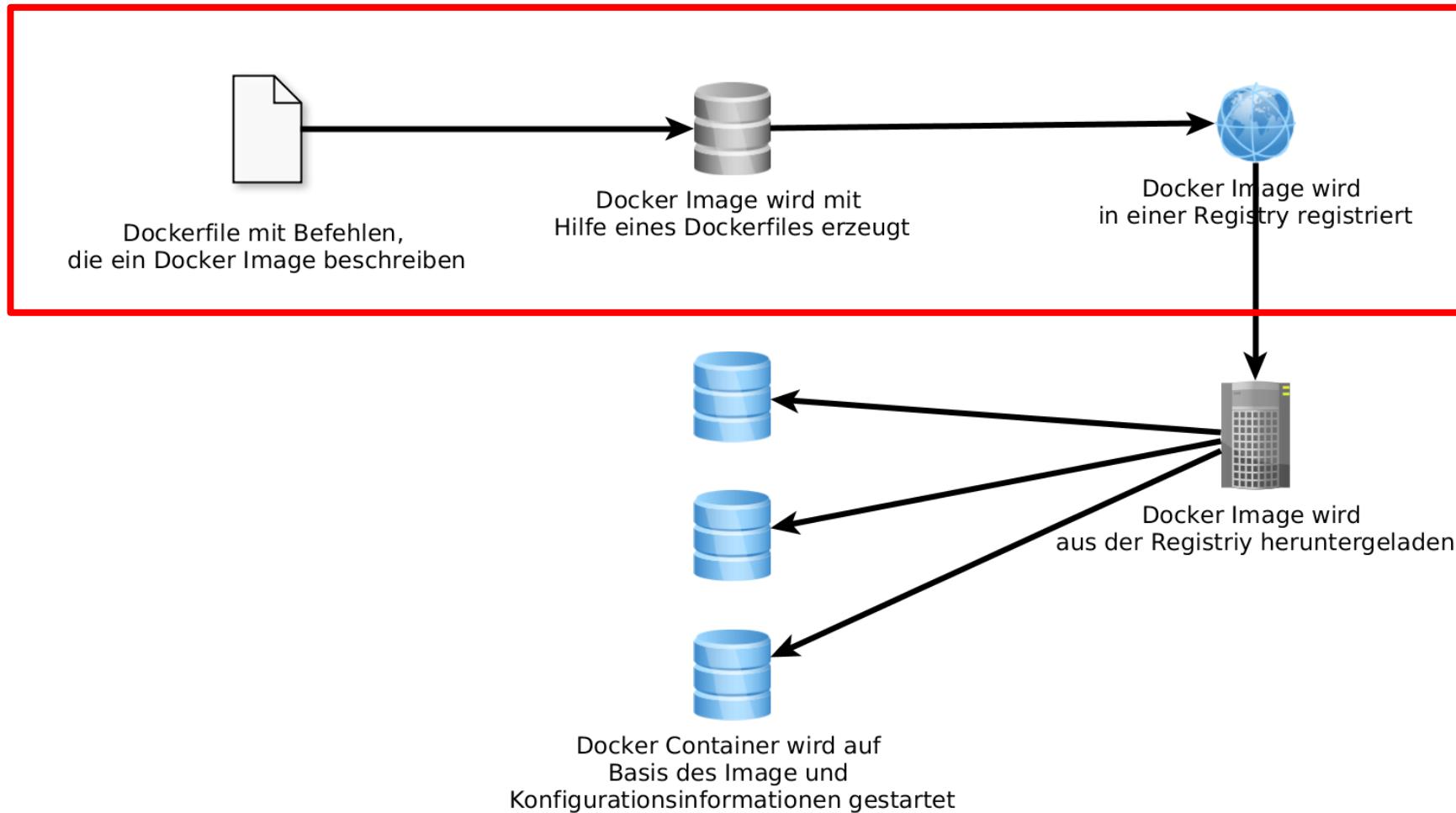
Alternativen

- Template
- Ansible-Container
- Packer
- Rocker
- Etc.
- Überblick verschafft Talk „Docker Container Loading“ von Roland Huß

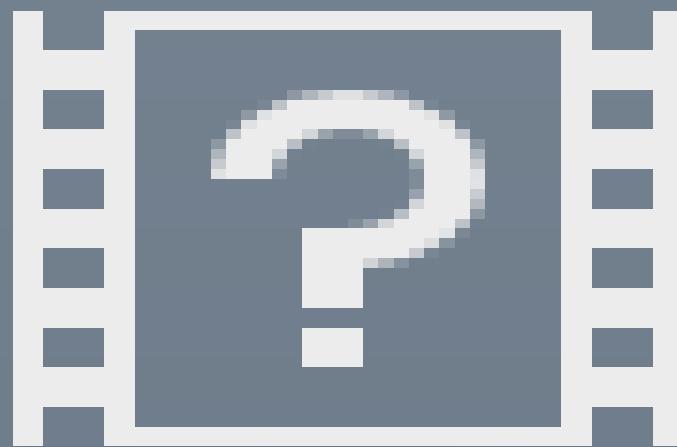
Dritte Reiseroute

Automatisierung des Docker-Image-Lifecycles mit Ansible

Docker Lifecycle



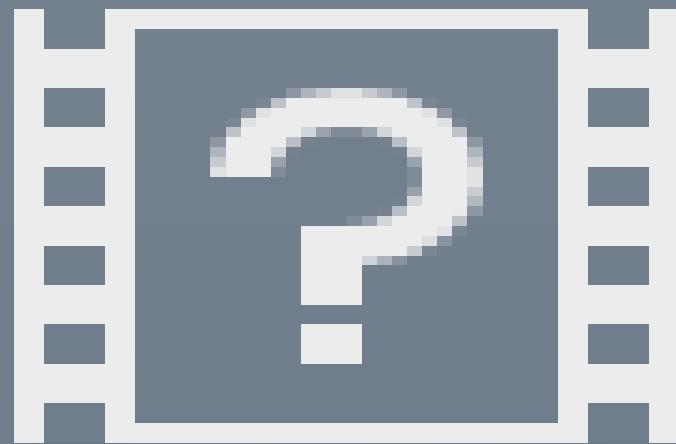
Docker Registry Lifecycle Demo



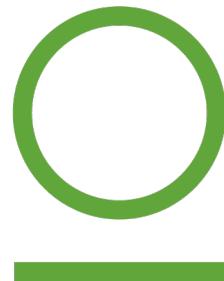
Docker Registry Lifecycle mit Ansible

```
1 v - hosts: localhost
2 v   vars:
3     registry_hostname: localhost
4 v   tasks:
5 v     - name: build and push tomcat image
6 v       docker_image:
7         name: sparsick/tomcat
8         dockerfile: tomcat.df
9         path: "{{ playbook_dir }}/../docker-image-ansible"
10        push: yes
11        repository: "{{ registry_hostname }}:5000/sparsick/tomcat:ansible"
12
13 v     - name: build and push mysql image
14 v       docker_image:
15         name: sparsick/mysql
16         dockerfile: mysql.df
17         path: "{{ playbook_dir }}/../docker-image-ansible"
18        push: yes
19        repository: "{{ registry_hostname }}:5000/sparsick/mysql:ansible"
```

Docker Registry Lifecycle mit Ansible Demo



Docker Registry Alternativen



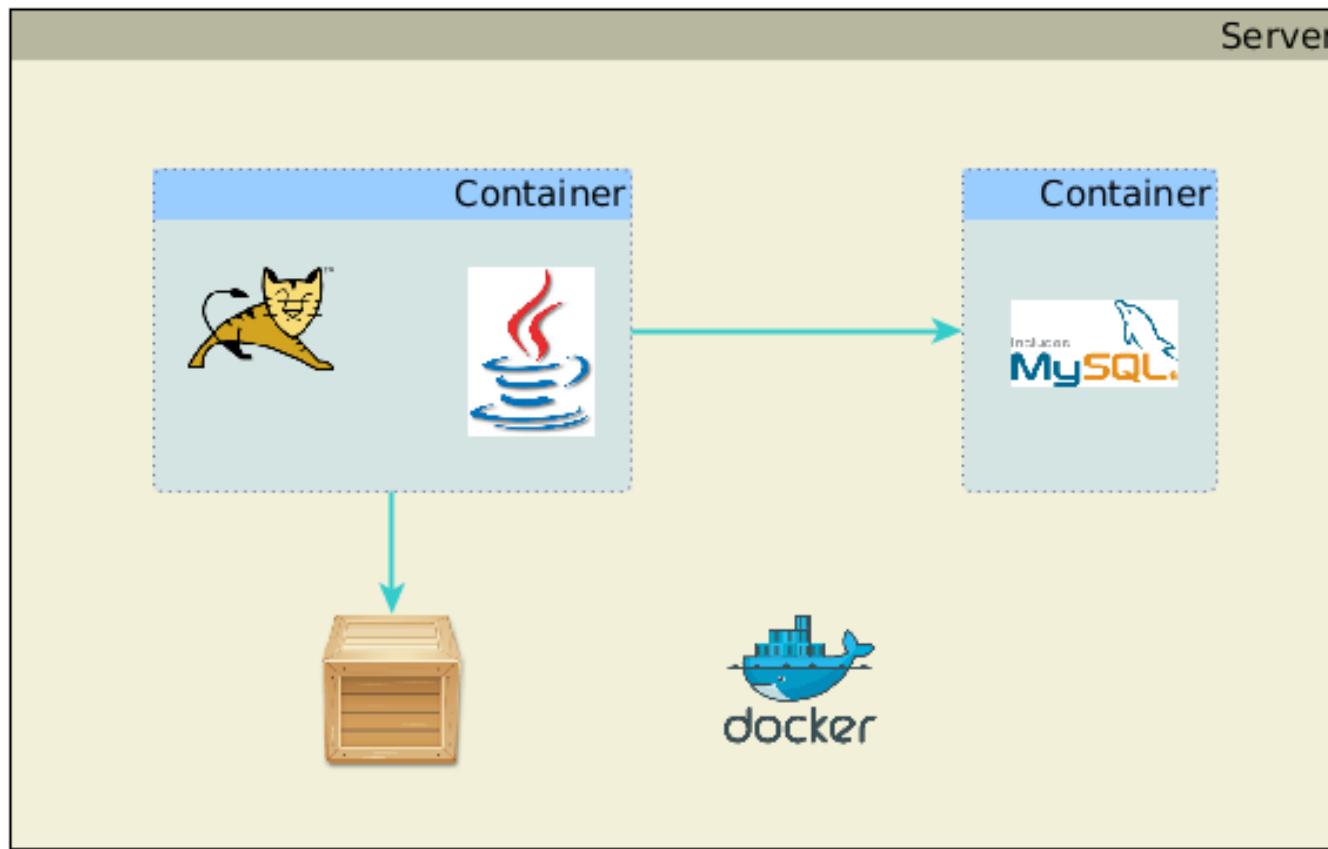
JFrog Artifactory



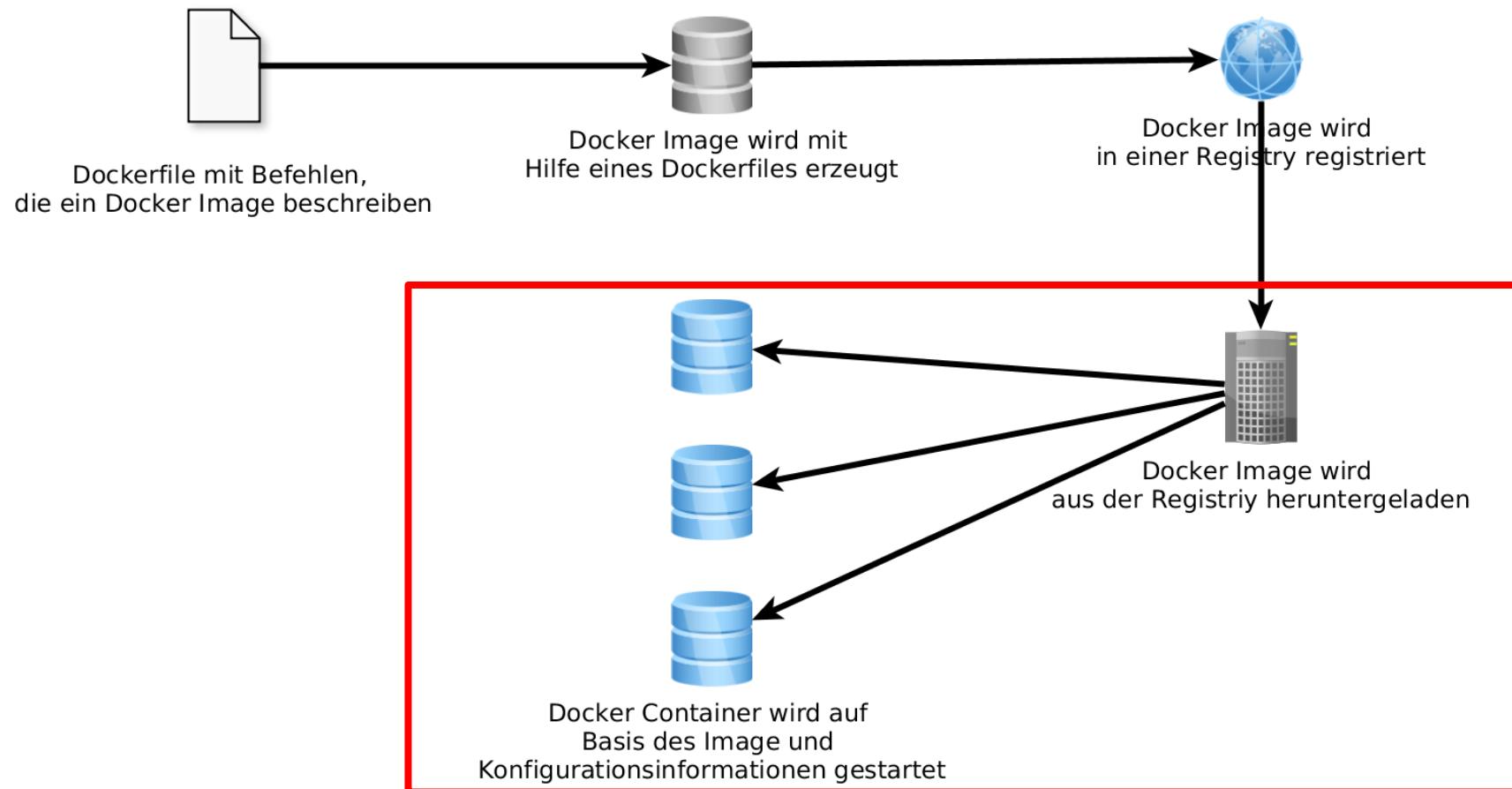
Letzte Reiseroute

Verteilung der Docker-Container auf
die Server mit Ansible

Reminder - Zielinfrastruktur



Docker Lifecycle



Was brauchen wir?

- Docker Installation automatisieren
- Docker Registry Installation automatisieren
- Container auf das Zielsystem verteilen
- WAR Datei wird über Volume eingebunden
→ muss auf den Zielsystem kopiert werden

Setup-dockerd.yml

```
1 - hosts: application-server
2   become: yes
3   become_method: sudo
4
5 tasks:
6   - name: ensure group vagrant exists
7     group: name=vagrant state=present
8
9   - name: ensure user vagrant exists
10    user: name=vagrant group=vagrant state=present
11
12  - name: install required package
13    apt: name="{{ item }}" state=present
14    with_items:
15      - apt-transport-https
16      - ca-certificates
17      - software-properties-common
18      - python-docker
19      - python-pip
20
21  - name: add Docker's official GPG key
22    apt_key: url="http://apt.dockerproject.org/gpg" state=present
23
24  - name: add Docker's repository
25    apt_repository: repo="deb https://apt.dockerproject.org/repo/ ubuntu-xenial main" update_cache=yes
26
27  - name: install latest Docker Engine
28    apt: name="docker-engine"
29
30  - name: add user vagrant to group docker
31    user: name=vagrant groups=docker state=present
```

Setup-docker-registry.yml

```
1  - hosts: application-server
2    become: yes
3    become_method: sudo
4
5    tasks:
6      - name: install latest Docker Engine
7        apt: name="docker-engine"
8
9      - name: add user vagrant to group docker
10        user: name=vagrant groups=docker state=present
11
12     - name: start docker registry
13       docker_container:
14         name: registry
15         image: registry:2
16         state: started
17       ports:
18         - "5000:5000"
```

Build-and-push-images.yml

```
1 v - hosts: localhost
2 v   vars:
3     registry_hostname: localhost
4 v   tasks:
5 v     - name: build and push tomcat image
6 v       docker_image:
7         name: sparsick/tomcat
8         dockerfile: tomcat.df
9         path: "{{ playbook_dir }}/../docker-image-ansible"
10        push: yes
11        repository: "{{ registry_hostname }}:5000/sparsick/tomcat:ansible"
12
13 v     - name: build and push mysql image
14 v       docker_image:
15         name: sparsick/mysql
16         dockerfile: mysql.df
17         path: "{{ playbook_dir }}/../docker-image-ansible"
18        push: yes
19        repository: "{{ registry_hostname }}:5000/sparsick/mysql:ansible"
```

Deploy-docker-container.yml

```
1 - hosts: application-server
2
3   tasks:
4     - name: start database container
5       docker_container:
6         name: database
7         image: localhost:5000/sparsick/mysql:ansible
8         state: started
9
10    - name:
11      file: name=/opt mode=777
12      become: yes
13      become_method: sudo
14
15    - name: prepare webapp location
16      file: path="/opt/webapps" state="directory" owner=vagrant group=vagrant
17
18    - name: copy java webapp archive
19      copy: src="../demo-web-application-1.war" dest="/opt/webapps/demo-web-application-1.war" owner=vagrant group=vagrant
20
21    - name: start tomcat container
22      docker_container:
23        name: tomcat
24        image: localhost:5000/sparsick/tomcat:ansible
25        state: started
26        volumes:
27          - "/opt/webapps/demo-web-application-1.war:/opt/tomcat/webapps/demo.war"
28        links:
29          - "database"
30        ports:
31          - "8080:8080"
```

Alternative Ansible mit Docker Compose Syntax

```
1 - hosts: application-server
2
3   tasks:
4     - name:
5       file: name=/opt mode=777
6       become: yes
7       become_method: sudo
8
9     - name: prepare webapp location
10    file: path="/opt/webapps" state="directory" owner=vagrant group=vagrant
11
12    - name: copy java webapp archive
13      copy: src="../demo-web-application-1.war" dest="/opt/webapps/demo-web-application-1.war" owner=vagrant group=vagrant
14
15    - docker_service:
16      project_name: java_web
17      definition:
18        version: '2'
19        services:
20          database:
21            image: "localhost:5000/sparsick/mysql:ansible"
22          tomcat:
23            image: localhost:5000/sparsick/tomcat:ansible
24            ports:
25              - "8080:8080"
26            links:
27              - database
28            volumes:
29              - /opt/webapps/demo-web-application-1.war:/opt/tomcat/webapps/demo.war
30
```

Docker-compose muss auf dem Zielmaschine installiert werden

Demo



Was haben wir gesehen?

- Provisionierungswerkzeuge (PW) können aktuelle Infrastrukturprobleme lösen
- Einsatz eines PW verbaut nicht den Weg hin zu einer Containerisierung der Infrastruktur
- PW kann bei der Umstellung helfen
- PW erleichtert das Container-Deployment

Wann Container einsetzen?

Mögliche Einsatzszenarien für Container

- Anwendung muss automatisch skaliert werden
- Unternehmen hat unterschiedliche Technologiestacks und will die Verteilung vereinheitlichen
- Bei Integrationstests innerhalb eines Builds (Testcontainer)

Integrationstests mit Container



Testcontainers

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

Testcontainers

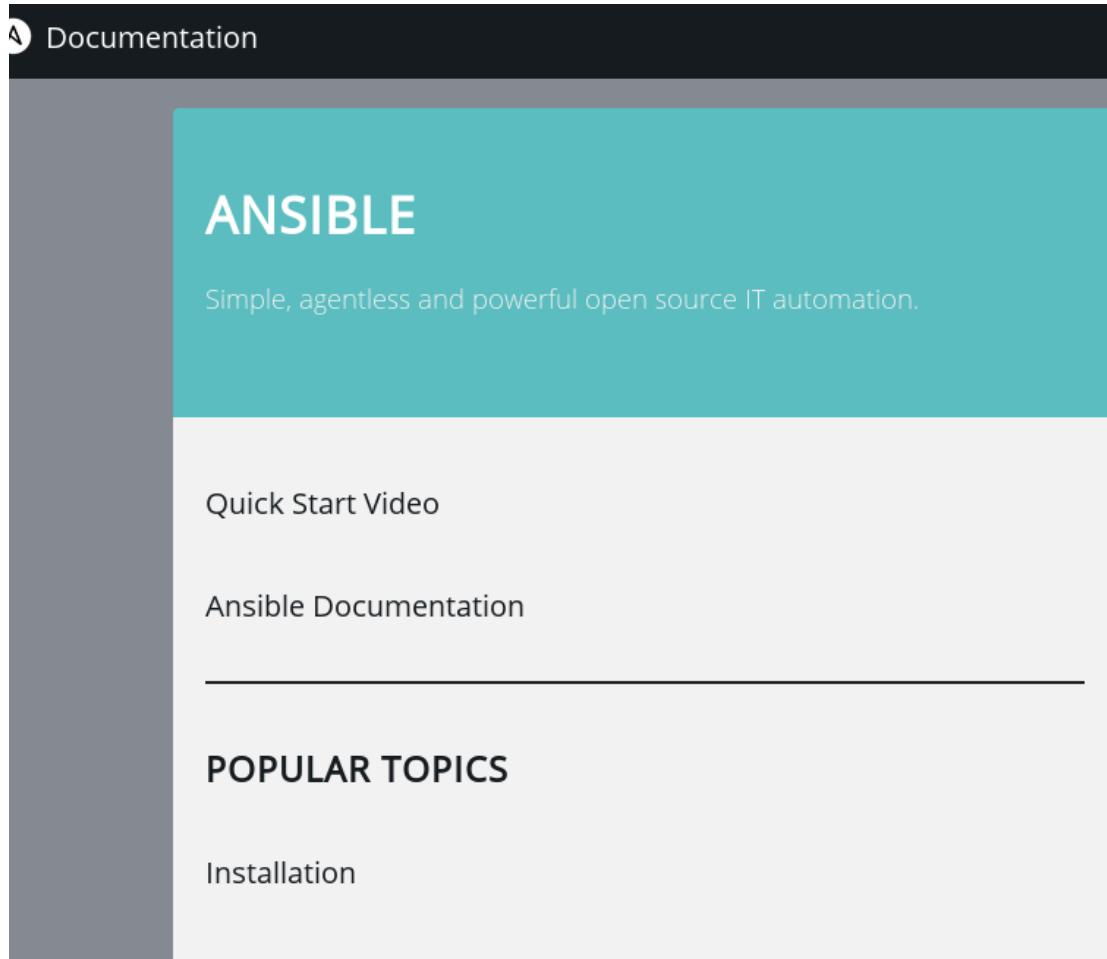
TESTS

```
Running db.migration.DbMigrationITest
INFO - eryClientProviderStrategy - 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>,registryEm...
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

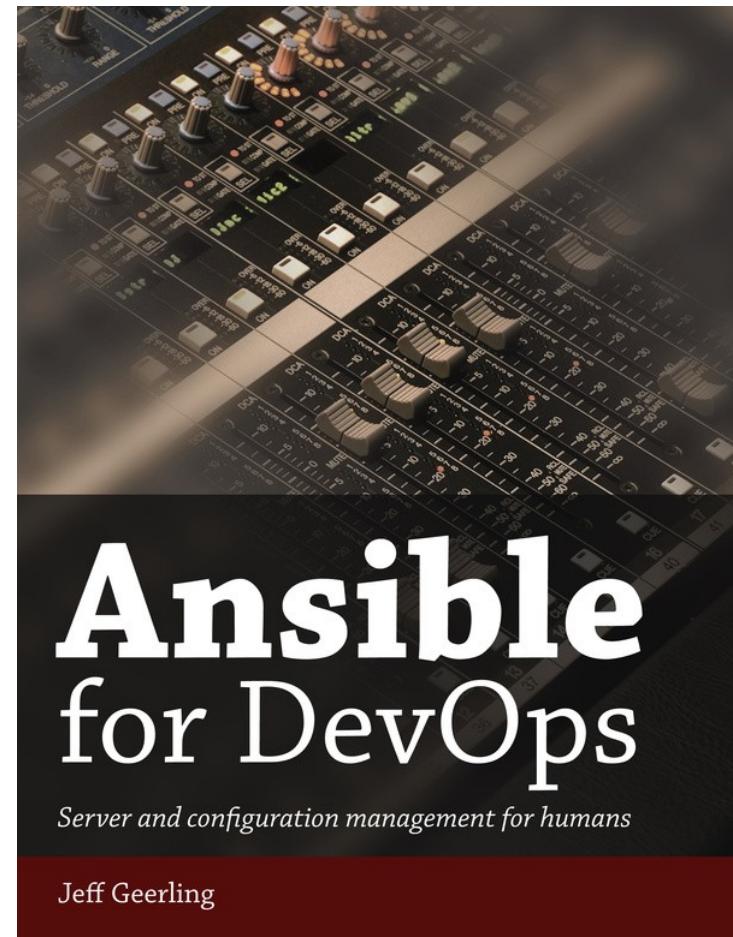
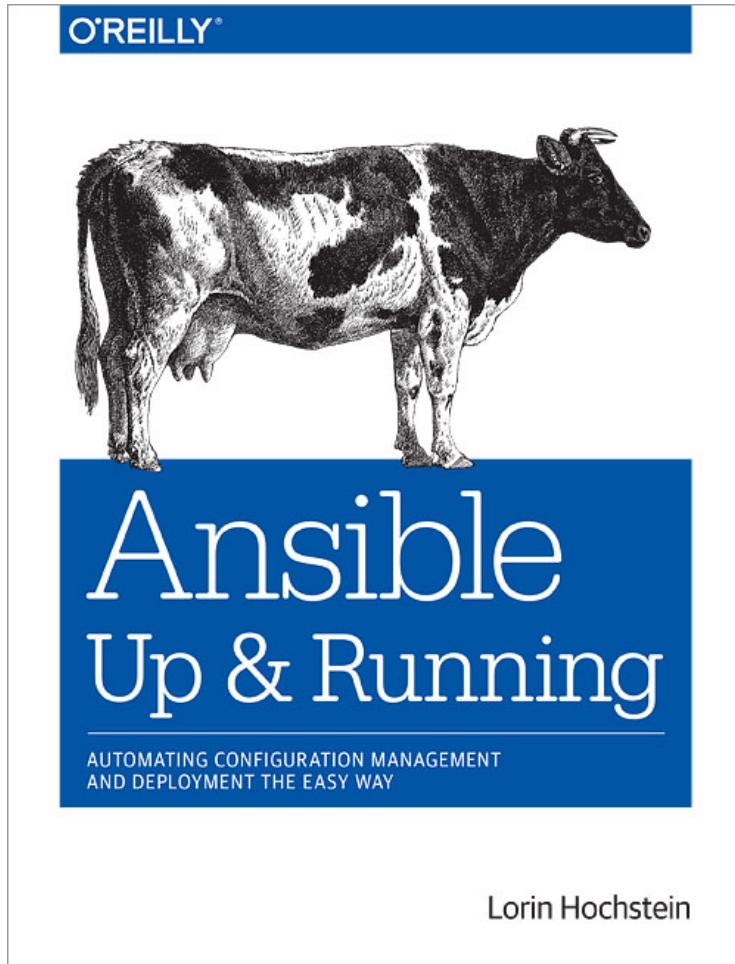
- Temporary database containers - spezielle MySQL, PostgreSQL, Oracle XE and Virtuoso container
- Webdriver containers - Dockerized Chrome oder Firefox browser für Selenium/Webdriver Operationen mit automatischer Videoaufnahme
- Generic containers – irgendein Docker Container
- Docker compose – Wiederverwendung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

Weitere Informationen



<http://docs.ansible.com/>

Weitere Informationen



Weitere Informationen



The image shows the front cover of the Java aktuell magazine, issue 04-2016. The title "Java aktuell" is prominently displayed at the top in large blue letters. Below it, the subtitle "Praxis. Wissen. Networking. Das Magazin für Entwickler Aus der Community – für die Community" is written. The date "04-2016 | Winter | www.ijug.eu" is at the top right. A central graphic features several overlapping arrows in red, yellow, green, and blue, pointing upwards and to the right. To the left of the arrows, there's a cluster of hexagonal icons representing various software development and networking concepts like email, locks, databases, and gear wheels. The word "JAVA" is written vertically in large blue letters. At the bottom left, there's a small barcode and some smaller text. The overall theme is "Software organisieren".



<http://bit.ly/2cZ0lrZ>



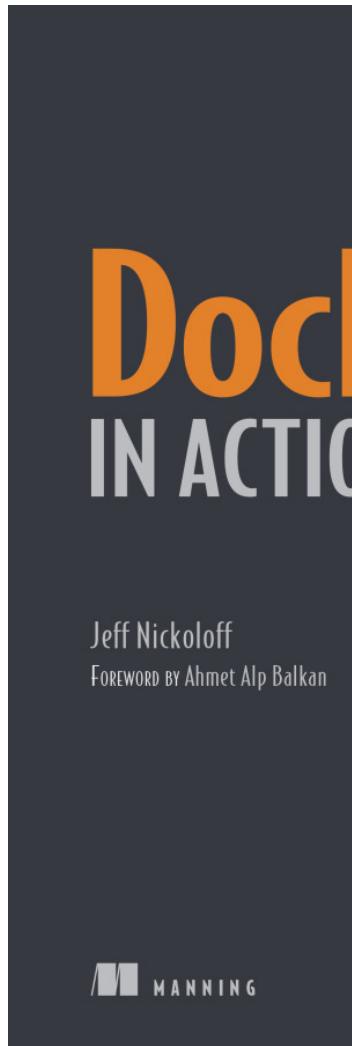
JUnit 5
Das nächste große
Release steht vor
der Tür

Ansible
Konfigurationsma-
nagement auch für
Entwickler

Spring Boot Starter
Komfortable Modula-
risierung und Konfi-
guration



Weitere Informationen



A screenshot of the Docker Documentation website. The header features the "docker docs" logo and navigation links for "What is Docker?", "Product", "Get Docker", "Docs", "Community", "Create Docker ID", and "Sign In". The main section has a red background with the heading "Docker Documentation" and a subtext about Docker's purpose. It includes two buttons: "Get Docker" and "Get Started". At the bottom, there are links for "Guides", "Product manuals", "Glossary", "Reference", and "Samples".

<https://docs.docker.com/>

Weitere Informationen

fachartikel



□ Sandra Parsick, geb. Kosmalla

(mailto:sandra-parsick.de)

Ist als freiberufliche Softwareentwicklerin und Beraterin im Java-Umfeld tätig. Seit 2008 beschäftigt sie sich mit agiler Softwareentwicklung in verschiedenen Rollen. Ihre Schwerpunkte liegen im Bereich der Java Enterprise-Anwendungen, agilen Methoden, Software Craftsmanship und in der Automatisierung von Softwareentwicklungsprozessen. In ihrer Freizeit engagiert sie sich in der Softwarekammer Ruhrgebiet.

Es muss nicht gleich Docker sein – IT-Automation, die zu einem passt

Docker ist in aller Munde und wird gerne als „Allheilmittel“ für Deployment-Probleme propagiert. Das führt zu der Annahme, automatisierte Deployments wären nur mit Docker möglich, obwohl Provisionierungswerzeuge wie Ansible Lösungen außerhalb der Container-Welt anbieten. Deren Einsatz wird oft gar nicht in Betracht gezogen, weil irgendwann – in ferner Zukunft – doch Docker im Unternehmen eingesetzt werden soll. Die Automatisierung wird immer weiter verschoben, weil der Aufwand in einem Schritt zu groß ist, obwohl Ansible mit wenig Mühe in der Gegenwart helfen könnte. Dieser Artikel zeigt, wie Ansible auf dem Weg zu einer Dockerisierung der Infrastruktur jetzt schon Probleme lösen kann und wie ein gemeinsamer Einsatz beider Technologien die Vorteile beider Welten kombiniert.

Wer kennt es nicht: Die Bereitstellung einer neuen Testinfrastruktur dauert ewig, weil sie manuell installiert werden muss; Die Produktionsnahme des neuen Releases hat wieder länger gedauert als geplant, weil die Konfiguration der Produktionsumgebung sich schon wieder von denen der Testumgebungen unterscheidet. Die Entwickler Alice und Bob kommen von einer Konferenz wieder, bei der sie gelernt haben, wenn sie ihre Deployments auf Docker umstellen, löst es ihre oben genannten Probleme. Doch der Betrieb wird die Umstellung auf Docker – eventuell, vielleicht – in einem oder zwei Jahren machen wollen.

Entwickler Carol schlägt vor, bis der Betrieb mit Docker so weit ist, die Serverkonfiguration und Deployments mit einem Provisionierungswerzeug wie Ansible zu automatisieren und davon ausgehend, die Dockerisierung der Infrastruktur voranzutreiben. Die Kollegen lehnen diesen Vorschlag ab, da sie sich für die Technologie Docker entschieden haben und aus ihrer

Sicht wäre es zu viel Aufwand, eine andere Technologie als Zwischenlösung zu nehmen. Somit quälen sie sich – eventuell, vielleicht – weitere ein bis zwei Jahre mit ihren Infrastrukturproblemen. Wie die Reise hätte aussehen können, wenn Carols Vorschlag angenommen worden wäre, beschreiben die nächsten Abschnitte.



Abb. 1: Ausgangsinfrastruktur

OBJEKTspektrum Online Themenspecial DevOps 2017

<http://bit.ly/2p7mxyB>

Weitere Informationen

- Talk „Docker Container Loading“ von Roland Huß
<https://github.com/ro14nd-talks/docker-container-loading/blob/master/docker-container-loading.pdf>
- Serverspecs <http://serverspec.org/>
- Testcontainers <https://www.testcontainers.org/>

A photograph of a large cargo ship docked at a port at night. The ship's hull is dark, and its superstructure is illuminated from within, showing multiple decks and levels. In the background, several large red industrial cranes are silhouetted against a dark sky. One crane has the brand name "EUROGATE" visible on its side. The overall scene is dimly lit, with the primary light source being the ship's own internal lights and the occasional bright light from the cranes.

Fragen?

@SandraParsick
mail@sandra-parsick.de
<https://github.com/sparsick/ansible-docker-talk.git>

Bildnachweise

<https://pixabay.com/de/vortrag-vorlesung-schule-2044619/>

<https://pixabay.com/de/spielsteine-figuren-holz-bunt-1743307/>

<https://pixabay.com/de/fragezeichen-birne-denken-idee-2010011/>

<https://pixabay.com/de/hamburg-hafen-kr%C3%A4ne-containerschiff-2103261/>