

Dokumentation Zur Projektarbeit

Bereitstellen von Docker-Container mit Ansible

Verfasser: Melek Öztürk

Modul: Cloud Computing

Prüfer: Prof. Dr. Sebastian Speiser

Abgabetermin: 22.06.2021



ANSIBLE



Inhaltsverzeichnis

1. Motivation	3
2. Problemstellung	4
3. Lösungsansatz	6
4. Workflow	7
5. Implementierung	8
6. Architekturschaubild	21
7. Erkenntnisse	23
8. Fazit	24

1. Motivation

Die zunehmende Verbreitung der Virtualisierung in Verbindung mit der zunehmenden Leistung von Servern haben dazu geführt, Infrastructure as Code einzuführen und zu verwenden. Infrastructure as Code dient zur Erleichterung und Automatisierung des Konfigurationsprozesses. Statt die Konfiguration von Betriebssystemen manuell durchzuführen, werden Konfigurationstools eingesetzt.

Der vorliegende Projektbericht beschäftigt sich mit einem Konfigurationstool „Ansible“ und einer Container Technologie „Docker“. Beide dienen zur Automatisierung von individuellen Konfigurationswünschen der Server. Kurz zu den Begrifflichkeiten: Docker ist eine freie Software, dass das Erstellen und Ausführen von Anwendungen mithilfe von Containern erleichtert. Container ermöglichen einem Entwickler, eine Anwendung mit Bibliotheken und anderen Abhängigkeiten, zu verpacken und als Paket bereitzustellen.

Ansible ist ein leistungsfähiges Konfigurationsmanagement-Tool. Das Tool ist ein Open Source-Tool für Softwarebereitstellung, Konfigurationsmanagement und Anwendungsbereitstellung, das Infrastructure as Code ermöglicht. Die Besonderheit und auch die Einzigartigkeit von Ansible im Vergleich zu anderen Management-Tools ist, dass es auch für die Bereitstellung und Orchestrierung verwendet wird.

Ziel dieser Projektarbeit ist es, mehrere Docker Container auf einer virtuellen Maschine, die das Betriebssystem Ubuntu enthält, laufen zu lassen. Dabei sollen die Docker Container auf den verteilten Anwendungen miteinander kommunizieren. Das ist mein individueller Konfigurationswunsch gewesen, dass ich mit Ansible konfiguriert, automatisiert und bereitgestellt habe. Die Motivation dieses Projektes ist es, dass Fungieren und die Individualität der beiden Bereitstellungstools darzustellen. Die Problematiken und die dazugehörigen Lösungsansätze werden im weiteren Verlauf des Projektberichtes thematisiert.

2. Problemstellung

Auf der Entwicklungsebene ist es wichtig die Nachhaltigkeit als Zielvorgabe zu sehen. Ebenso von großer Bedeutung sind die Zeit, Transparenz und Fehlervermeidung bei einer IT-Automatisierung. Um diese grundlegenden Vorteile in das Projekt hineinfließen zu lassen, benötigt es zum einen Fragen die auf eine Problemstellung zurückführen und zum anderen den Lösungsansatz. In diesem Abschnitt werde ich zunächst die Problemstellungen näher erläutern und die Fragen beantworten, die sich am Anfang angehäuft haben.

Wie oben bereits erwähnt verwende ich Ansible um Docker Container bereitzustellen und zu automatisieren. Um eine Automatisierung in Wege zu leiten, benötigt es vorherige Informationen. Die Informationen sind zum Beispiel, welche Pakete müssen vorinstalliert werden, um die Docker Technologie auf einer virtuellen Maschine mit dem Betriebssystem Ubuntu laufen zu lassen. Ist die Kommunikation von Docker-Containern mit Ansible realisierbar?

Das sind die Fragen, die mich am Anfang meines Projektes beschäftigt haben. Zunächst möchte ich näher auf die Fragen eingehen und die Antworten einführen. Nachdem ich die Antworten zusammengebracht habe, grenze ich die Problemstellung ein. Die Antwort auf die Frage welche Pakete müssen installiert werden, um die Docker Technologie auf einer virtuellen Maschine mit dem Betriebssystem **Ubuntu** laufen zu lassen werde ich wie folgt auflisten:

-APT-Transport zum Herunterladen über das http Secure Protocol (HTTPS)

Dieser API-Transport ermöglicht die Verwendung von Repositorys, auf die über das http Secure-Protokoll zugegriffen wird.

-Certification Authority" (CA/Zertifizierungsstelle)

Eine CA (Certificate Authority oder Certification Authority) ist eine vertrauenswürdige Instanz, eine Zertifizierungsstelle, die digitale Zertifikate herausgibt.

-Client for URLs (curl)

Curl ist ein Programm, das ermöglicht, ohne Benutzerinteraktion Dateien von oder zu einem Server zu übertragen.

-GNU-Privatsphärenschutz (abk. GnuPG)

GNU-Privatsphärenschutz, ist ein freies Kryptographiesystem. Es dient zum Ver- und Entschlüsseln von Daten sowie zum Erzeugen und Prüfen.

-Apt-Software allgemeinen Eigenschaften

Diese Software bietet eine Abstraktion der verwendeten apt – Repositorys. Es ermöglicht die einfache Verwaltung von Distribution und der Softwarequellen unabhängiger Softwareanbieter.

-Hinzufügen der Schlüsselverwaltung apt-key

Apt-key wird benötigt, um die Liste der Schlüssel zu verwalten die von apt zum Authentifizieren von Paketen verwendet werden. Pakete die mit diesen Schlüsseln authentifiziert wurden, gelten als vertrauenswürdig

-Hinzufügen eines Docker Repositorys

Das Docker Repository wird benötigt, um später die Software Docker aus dem Docker Repository zu installieren.

Die zweite Frage lautet, ob die Kommunikation zwischen den Containern mit Ansible realisierbar ist. Nach intensiver Recherche hat sich herausgestellt, dass sich die Netzwerkkommunikation mit Ansible konfigurieren lässt. In dem vierten Abschnitt der Implementierung, werde ich die Verwirklichung der Netzwerkkommunikation darstellen und erläutern.

Die konkrete Problemstellung bestand darin, einen Weg zu finden, wie Ansible die Container Prozesse ausführen kann und Dateien in einen Container bekommt. Üblicherweise verwenden Konfigurationssysteme SSH als das Protokoll der Wahl, welches wiederum voraussetzen würde, dass ein „sshd“ auf dem jeweiligen Container laufen müsste. Laut meiner Recherche ist es aus verschiedenen Gründen unerwünscht, einen „sshd“ auf einem Container laufen zu lassen. Zum einen wird das Debuggen schwieriger und die Konfiguration der SSH müsste schon im finalen Image eingebaut werden, das wiederum nicht sicher ist. Doch wie funktioniert der Zugriff in einen Docker Container im Remote Server. Darauf folgt noch eine Frage und zwar woher weiß ich das eine Docker Kommunikation stattfindet? Die Frage und auch den Lösungsansatz werde ich in dem darauffolgenden Abschnitt näher erläutern.

3. Lösungsansatz

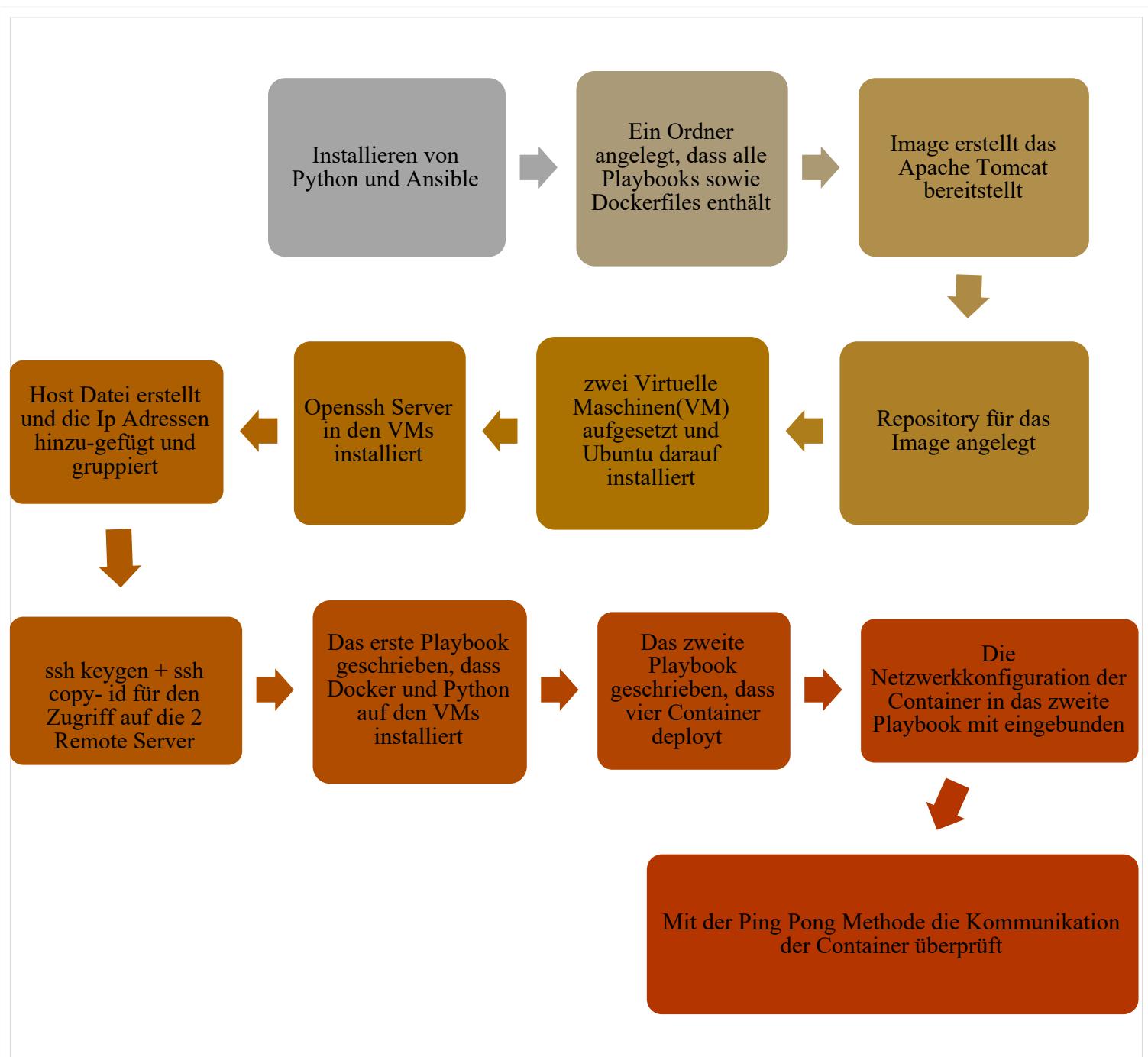
Zunächst hat sich bei der Suche nach der Lösung herauskristallisiert, dass es einen simplen Lösungsansatz gibt, um meine Problemstellung zu lösen. Hier möchte ich kurz meine Fragestellung beziehungsweise meine Problemstellung nochmals aufgreifen: Zum einen möchte ich wissen, ob die Implementierung der Docker Netzwerkkommunikation funktioniert und der Informationsaustausch der Container stattfindet und zum anderen wie greife ich auf einen Docker Container zu. Die Lösung lautet „exec“. Der Befehl „exec“ wird verwendet, um mit bereits ausgeführten Container auf dem Docker-Host zu interagieren. Zudem ermöglicht es eine Sitzung innerhalb des Standardverzeichnisses des Containers zu starten. Des Weiteren ist der Befehl auch mit anderen Optionen kombinierbar.

In dem zweiten Abschnitt der Problemstellung habe ich noch eine Frage offen, die ich beantworten möchte. Die Frage war: „Woher weiß ich das eine Docker Kommunikation stattfindet? Mit dem Befehl „docker inspect“ lassen sich alle Netzwerkkonfigurationen der einzelnen Docker Container sehen. Jedoch kann ich mit diesem Befehl nicht sehen, ob eine Kommunikation stattfindet. Die Docker-Kommunikation wird mithilfe der Ping-Pong Methode ersichtlich.

Die Implementierung und der Beweis, dass die Docker interagieren, werde ich in dem darauffolgenden Abschnitt thematisieren. Zusätzlich werde ich meine Thesen mit Screenshots belegen.

4. Workflow

Dieser Abschnitt widmet sich der Realisierung des Ansible-Docker Projekts über mehrere Schritte. Um alle nötigen Schritte im Blick zu behalten, bietet sich ein strukturierter Workflow an. Die untere Abbildung 1 stellt die Phasen des Projektes dar.



5. Implementierung

Der oben dargestellte Workflow besteht aus zwölf Phasen. In diesem Abschnitt werden die Ergebnisse und Erläuterungen der einzelnen Phasen aufgezeigt.

Um das Automatisierungstool mit seinen Automatisierungsaufgaben und die Bereitstellungsdetails in YAML-Dateien zu nutzen, benötigt es auf dem Lokalen Rechner, die Installation von Ansible und Python.

```
melekozturk@Meleks-MacBook-Pro ~ % ansible --version
ansible 2.10.7
  config file = None
  configured module search path = ['~/Users/melekozturk/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/Cellar/ansible/3.1.0/libexec/lib/python3.9/site-packages/ansible
  executable location = /usr/local/bin/ansible
  python version = 3.9.2 (default, Mar 26 2021, 23:27:12) [Clang 12.0.0 (clang-1200.0.32.29)]
melekozturk@Meleks-MacBook-Pro ~ %
```

Screenshot 1: Fertige Installation von Ansible

```
[melekozturk@Meleks-MacBook-Pro ~ % python
Python 3.9.2 (default, Mar 26 2021, 23:27:12)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> ]
```

Screenshot 2: Fertige Installation von Python

Der zweite Arbeitsschritt ist das Anlegen eines Ordners. Dieser Ordner dient zur Ordnung und ist der Ausführungsort der Definitionsdateien von Docker und Ansible.

```
melekozturk@Meleks-MacBook-Pro ~ % cd ansible_docker_project
melekozturk@Meleks-MacBook-Pro ansible_docker_project % pwd
~/Users/melekozturk/ansible_docker_project
melekozturk@Meleks-MacBook-Pro ansible_docker_project %
```

Screenshot 3: Zeigt das ein Ordner namens ansible_docker_project angelegt wurde

In der dritten Phase geht es um die Erstellung eines Dockerfiles. Dieses Dockerfile das auch als Image gilt, beinhaltet die Schritte zur Installation und Bereitstellung eines Apache Tomcat Webserver.

```
↳ dockerfile ×
↳ dockerfile > ...
1  #Dockerfile
2
3  #base image
4  FROM alpine
5
6  #jdk11
7  RUN apk add openjdk11
8
9  #tomcat
10 RUN wget https://www-eu.apache.org/dist/tomcat/tomcat-9/v9.0.46/bin/apache-tomcat-9.0.46.tar.gz
11 RUN tar xvfz apache*.tar.gz -C /opt
12 RUN rm apache-tomcat-9.0.46.tar.gz
13 RUN rm -rf /opt/apache-tomcat-9.0.46/webapps/*
14
15 WORKDIR /opt/apache-tomcat-9.0.46/webapps
16
17 ENV PATH $PATH:$CATALINA_HOME/opt/apache-tomcat-9.0.46/bin/
18
19 CMD ["catalina.sh","run"]
```

Screenshot 4: Ein Docker Image für die Installation und Bereitstellung des Apache Tomcat Webserver

Der darauffolgende Schritt ist das Hinzufügen eines öffentlichen Repositorys in Docker-Hub. In diesem Repository befindet sich das Image „dockerfile“. In den Screenshots wird außerdem ersichtlich sein, dass die Anwendung in einem Container läuft.

```
melekozturk@Meleks-MacBook-Pro ansible_docker_project % docker build -t dockerfile .
[+] Building 0.2s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 513B
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/alpine:latest
=> [1/7] FROM docker.io/library/alpine
=> CACHED [2/7] RUN apk add openjdk11
=> CACHED [3/7] RUN wget https://www-eu.apache.org/dist/tomcat/tomcat-9/v9.0.46/bin/apache-tomcat-9.0.46.tar.gz
=> CACHED [4/7] RUN tar xvfz apache*.tar.gz -C /opt
=> CACHED [5/7] RUN rm apache-tomcat-9.0.46.tar.gz
=> CACHED [6/7] RUN rm -rf /opt/apache-tomcat-9.0.46/webapps/*
=> CACHED [7/7] WORKDIR /opt/apache-tomcat-9.0.46/webapps
=> exporting to image
=> => exporting layers
=> => writing image sha256:c9c7d03bd83c7923b851b5f2eb182f063bf03ae5e017c27b816223922fb9d6c1
=> => naming to docker.io/library/dockerfile

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
melekozturk@Meleks-MacBook-Pro ansible_docker_project %
```

Screenshot 5: Image erstellt namens „dockerfile“

```

melekozturk@Meleks-MacBook-Pro ~ % docker run -p 8081 dockerfile
NOTE: Picked up JDK JAVA_OPTS: --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED
NOTE: Picked up JDK JAVA_OPTIONS: --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED
12-Jun-2021 07:43:29.228 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version name: Apache Tomcat/9.0.46
12-Jun-2021 07:43:29.228 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built: May 8 2021 17:38:52 UTC
12-Jun-2021 07:43:29.228 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version number: 9.0.46.0
12-Jun-2021 07:43:29.228 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name: Linux
12-Jun-2021 07:43:29.228 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Version: 4.19.25-linuxkit
12-Jun-2021 07:43:29.228 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Architecture: amd64
12-Jun-2021 07:43:29.231 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home: /usr/lib/jvm/java-11-openjdk
12-Jun-2021 07:43:29.233 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Version: 11.0.9+11-alpine-r1
12-Jun-2021 07:43:29.233 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Vendor: Alpine
12-Jun-2021 07:43:29.233 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java(TM) SE Runtime Environment: /opt/apache-tomcat-9.0.46
12-Jun-2021 07:43:29.234 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_HOME: /opt/apache-tomcat-9.0.46
12-Jun-2021 07:43:29.273 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.lang=ALL-UNNAMED
12-Jun-2021 07:43:29.276 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.io=ALL-UNNAMED
12-Jun-2021 07:43:29.276 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.util=ALL-UNNAMED
12-Jun-2021 07:43:29.277 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.util.concurrent=ALL-UNNAMED
12-Jun-2021 07:43:29.280 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.rmi=ALL-UNNAMED
12-Jun-2021 07:43:29.281 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.io=ALL-UNNAMED
12-Jun-2021 07:43:29.282 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.util=ALL-UNNAMED
12-Jun-2021 07:43:29.282 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.util.concurrent=ALL-UNNAMED
12-Jun-2021 07:43:29.283 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.rmi=sun.rmi.transport=ALL-UNNAMED
12-Jun-2021 07:43:29.288 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.util.logging.config=java.util.logging.config=ALL-UNNAMED
12-Jun-2021 07:43:29.288 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.util.logging.config=java.util.logging.config=org.apache.juli.ClassLoaderLogManager
12-Jun-2021 07:43:29.289 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.util.logging.config=java.util.logging.config=org.apache.juli.ClassLoaderLogManager
12-Jun-2021 07:43:29.289 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.level=FINEST
12-Jun-2021 07:43:29.289 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.FileHandler.level=FINEST
12-Jun-2021 07:43:29.289 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.FileHandler.size=2843
12-Jun-2021 07:43:29.293 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=org.apache.catalina.webresources
12-Jun-2021 07:43:29.293 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dorg.apache.catalina.security.SecurityListener.UMASK=0027
12-Jun-2021 07:43:29.294 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dignore.endorsed.dirs
12-Jun-2021 07:43:29.294 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dcatalina.base=/opt/apache-tomcat-9.0.46
12-Jun-2021 07:43:29.294 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dcatalina.home=/opt/apache-tomcat-9.0.46
12-Jun-2021 07:43:29.294 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.io.tmpdir=/opt/apache-tomcat-9.0.46/temp
12-Jun-2021 07:43:29.299 INFO [main] org.apache.catalina.core.AprLifecycleEvent.lifecycleEvent The Apache Tomcat Native library which allows using OpenSSL was not found on the java.library.path: [/usr/lib/jvm/java-11-openjdk/lib/server:/usr/lib/jvm/java-11-openjdk..:/lib:/usr/java/packages/lib:/usr/lib64:/lib64:/usr/lib]
12-Jun-2021 07:43:29.887 INFO [main] org.apache.coyote.AbstractProtocol.init Initializing ProtocolHandler ["http-nio-8088"]
12-Jun-2021 07:43:29.957 INFO [main] org.apache.catalina.startup.Catalina.load Server initialization in [126] milliseconds
12-Jun-2021 07:43:30.181 INFO [main] org.apache.catalina.startup.Catalina.start StandardEngine[localhost-startup] Starting Socket endpoint: [Apache Tomcat/9.0.46]
12-Jun-2021 07:43:30.128 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8088"]
12-Jun-2021 07:43:30.166 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [200] milliseconds

```

```

melekozturk@Meleks-MacBook-Pro ~ % docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
8ba2495133cf        dockerfile          "catalina.sh run"   7 seconds ago      Up 5 seconds       0.0.0.0:55423->8081/tcp   nervous_meitner
melekozturk@Meleks-MacBook-Pro ~ %

```

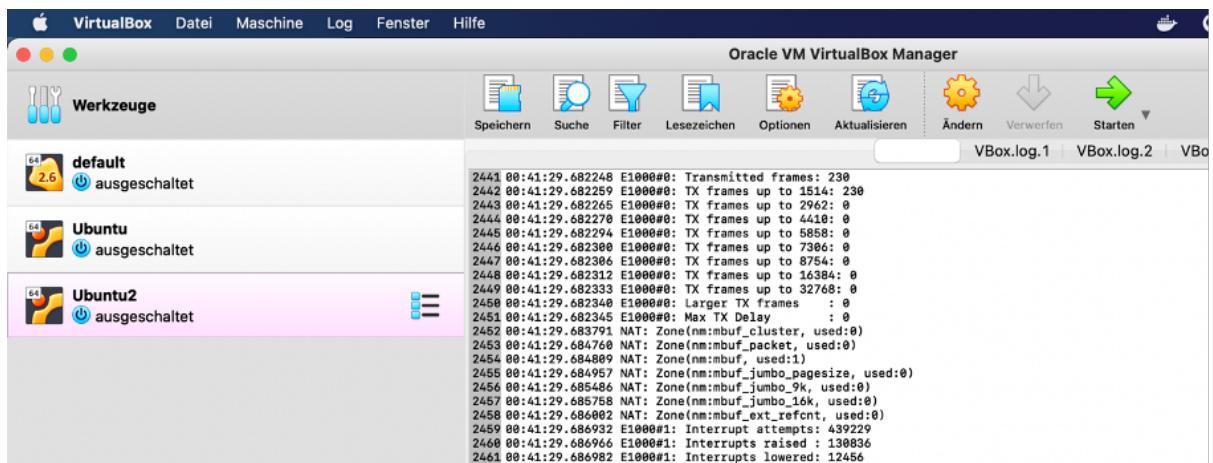
Screenshot 6: Das Image läuft in einem Container

Screenshot 7: Der Apache Tomcat Webserver ist erreichbar

The screenshot shows a Docker Hub repository page for 'me1eq/webserverapache'. At the top, there's a search bar and navigation links for Explore, Repositories, Organizations, Get Help, and a user account. Below the header, it says 'Using 0 of 1 private repositories. [Get more](#)'. The main content area includes an 'Advanced Image Management' section with a link to 'View preview'. Below that, the repository name 'me1eq/webserverapache' is shown, along with a note that it 'does not have a description'. It was last pushed 20 days ago. To the right, there's a 'Docker commands' section with a button to 'Push a new tag to this repository' and a command line interface box containing 'docker push me1eq/webserverapache:tagname'. On the left, there's a 'Tags and Scans' section showing one tag ('latest') and vulnerability scanning status ('DISABLED'). On the right, there's a 'Recent builds' section with a note to 'Link a source provider and run a build to see build results here'. A 'Public View' button is also present.

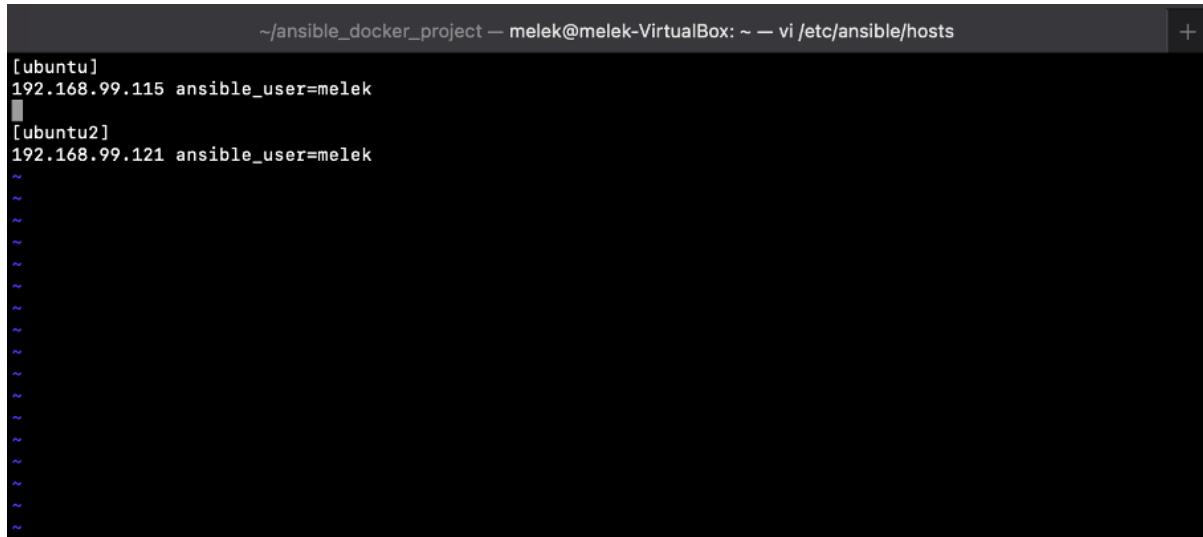
Screenshot 8: Repository in Docker Hub erstellt und Image gepusht

In dem nächsten Schritt des Workflows, geht es darum zwei virtuelle Maschinen aufzusetzen und darauf das Betriebssystem Ubuntu zu installieren.



Screenshot 9: 2 Virtuelle Maschine Ubuntu und Ubuntu2 mit den dazugehörigen Betriebssystemen

Ansible benötigt eine Host Datei oder auch Inventory bezeichnet, um mit den Remote Servern (die beiden VMs) zu kommunizieren. In der Datei wird festgelegt, welche Hosts orchestriert werden sollen, darüber hinaus lassen sie sich zu Gruppen zusammenfassen.



```
[ubuntu]
192.168.99.115 ansible_user=melek
[ubuntu2]
192.168.99.121 ansible_user=melek
```

Screenshot 10: Host Datei für die Orchestrierung

Jetzt benötigen die virtuellen Maschinen den openssh Server. Der SSH Server ermöglicht eine sichere, authentifizierte und verschlüsselte Verbindung zwischen zwei Rechnern über ein unsicheres Netzwerk.



```
melek@melek-VirtualBox:~$ sudo apt install openssh-server
[sudo] Passwort für melek:
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
openssh-server ist schon die neueste Version (1:8.2p1-4ubuntu0.2).
0 aktualisiert, 0 neu installiert, 0 zu entfernen und 140 nicht aktualisiert.
melek@melek-VirtualBox:~$
```

Screenshot 11: Der SSH Server wurde auf beiden VMs installiert

Nun benötigen wir ssh keygen + ssh copy- id für den Zugriff auf die 2 Remote Server. Der Beweis das dieser Zugriff ermöglicht wurde, wird in den beiden Screenshots ersichtlich.

```
~/ansible_docker_project -- melek@melek-VirtualBox: ~ -- ssh melek@192.168.99.115
melekozturk@Meleks-MacBook-Pro ansible_docker_project % ssh melek@192.168.99.115
The authenticity of host '192.168.99.115 (192.168.99.115)' can't be established.
ECDSA key fingerprint is SHA256:XLB3QW0xHHYnhR5Z5HbBs4bLM07bd5E1ssVIq/6mzbs.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.99.115' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-53-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

140 Aktualisierungen können sofort installiert werden.
72 dieser Aktualisierung sind Sicherheitsaktualisierungen.
Um zu sehen, wie diese zusätzlichen Updates ausgeführt werden: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Wed Jun  2 17:50:59 2021 from 192.168.99.1
melek@melek-VirtualBox:~$
```

```
~/ansible_docker_project -- melek@melek-VirtualBox: ~ -- ssh melek@192.168.99.121
melekozturk@Meleks-MacBook-Pro ~ % cd ansible_docker_project
melekozturk@Meleks-MacBook-Pro ansible_docker_project % ssh melek@192.168.99.121
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-55-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

187 Aktualisierungen können sofort installiert werden.
85 dieser Aktualisierung sind Sicherheitsaktualisierungen.
Um zu sehen, wie diese zusätzlichen Updates ausgeführt werden: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Jun 11 09:16:53 2021 from 192.168.99.1
melek@melek-VirtualBox:~$
```

Screenshot 12: Zugriff per SSH auf die VMs

Nachdem die SSH Verbindung funktioniert hat, ist es noch wichtig, dass Ansible seine Verbindung zu den Servern aufbauen kann. Denn wenn die Verbindungen nicht gewährleistet sind, ist es nicht möglich, eine Konfiguration vorzunehmen. Mithilfe der Ping-Pong Methode, können wir sehen, dass eine Verbindung aufgebaut wurde.

```

~/ansible_docker_project — melek@melek-VirtualBox: ~ -- zsh
[melekozturk@Meleks-MacBook-Pro ansible_docker_project % ansible ubuntu -m ping
192.168.99.115 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[melekozturk@Meleks-MacBook-Pro ansible_docker_project % ansible ubuntu2 -m ping
192.168.99.121 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
melek@melek-VirtualBox: ~

```

Screenshot 13: Ansible kann sich mit den Servern verbinden

Jetzt sind alle Vorkonfigurationen bereitgestellt, um das Automatisierungskonzept zu starten. Das erste Automatisierungskonzept, besteht darin, dass die Docker-Technologie, auf den VMs installiert wird. In dieser Dokumentation werde ich es anhand einer VM zeigen.

```

! install_docker_ubuntu.yaml ×
! install_docker_ubuntu.yaml > YAML > {} 0 > [ ]tasks
1   - hosts: ubuntu2
2     remote_user : melek
3     become : true
4     become_method : sudo
5
6   tasks:
7
8     # Install Docker
9     # ---
10    #
11    - name: install prerequisites
12      apt:
13        name:
14          - apt-transport-https
15          - ca-certificates
16          - curl
17          - gnupg-agent
18          - software-properties-common
19        update_cache: yes
20
21    - name: add apt-key
22      apt_key:
23        url: https://download.docker.com/linux/ubuntu/gpg
24
25    - name: add docker repo
26      apt_repository:
27        repo: deb https://download.docker.com/linux/ubuntu focal stable
28
29
30    - name: install docker
31      apt:
32        name:
33          - docker-ce
34          - docker-ce-cli
35          - containerd.io
36        update_cache: yes
37
38    - name: add userpermissions
39      shell: "usermod -aG docker melek"

```

```
! install_docker_ubuntu.yaml ×  
! install_docker_ubuntu.yaml > YAML > {} 0 > [ ]tasks > {} 6 > {} pip > [ ]name  
41 # Installs Docker SDK  
42 # --  
43 #  
44 - name: install python package manager  
45   apt:  
46     name: python3-pip  
47  
48 - name: install python sdk  
49   become_user: melek  
50   pip:  
51     name:  
52       - docker  
53       - docker-compose  
54
```

Screenshot 14: Playbook der Docker, Python und die nötigen Packages für Ubuntu installiert

```
melekozturk@Meleks-MacBook-Pro ansible_docker_project % ansible-playbook -K install_docker_ubuntu.yaml  
BECOME password:  
PLAY [ubuntu] *****  
TASK [Gathering Facts] *****  
ok: [192.168.99.116]  
TASK [install prerequisites] *****  
ok: [192.168.99.116]  
TASK [add apt-key] *****  
ok: [192.168.99.116]  
TASK [add docker repo] *****  
ok: [192.168.99.116]  
TASK [install docker] *****  
ok: [192.168.99.116]  
TASK [add userpermissions] *****  
changed: [192.168.99.116]  
TASK [install python package manager] *****  
ok: [192.168.99.116]  
TASK [install python sdk] *****  
ok: [192.168.99.116]  
PLAY RECAP *****  
192.168.99.116 : ok=8    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  
malekozturk@Meleks-MacBook-Pro ansible_docker_project %
```

Screenshot 15: Installiert Docker, Python und die nötigen Packages für Ubuntu installiert

```
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

141 updates can be installed immediately.
70 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun May 23 16:10:58 2021 from 192.168.99.1
melek@melek-VirtualBox:~$ docker --version
Docker version 20.10.6, build 370c289
melek@melek-VirtualBox:~$ python3 -m pip list | grep docker
docker                  5.0.0
docker-compose           1.29.2
dockerpty                0.4.1
melek@melek-VirtualBox:~$
```

Screenshot 16: Vollständig installiert auf der VM

Die zwei vorletzten Schritte werde ich in diesem Abschnitt zusammenfassen. Nämlich das deployen der Container und die Netzwerkkonfiguration. Die Installation der Docker Technologie wurde vollbracht und funktioniert einwandfrei. Nun können wir die Container mit den gewünschten Anwendungen, auf den VMs laufen lassen.

```
melekozturk@Meleks-MacBook-Pro ~ % cd ansible_docker_project
melekozturk@Meleks-MacBook-Pro ansible_docker_project % ansible-playbook -K build_container_playbook.yaml
BECOME password:

PLAY [Node 1] *****
TASK [Gathering Facts] *****
ok: [192.168.99.115]
TASK [Create Network] *****
ok: [192.168.99.115]

TASK [Deploy the Container Apache] *****
[DEPRECATION WARNING]: The container_default_behavior option will change its default value from "compatibility" to "no_defaults" in community.docker 2.0.0. To remove this warning, please specify an explicit value for it now. This feature will be removed from community.docker in version 2.0.0. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
[DEPRECATION WARNING]: The module will now use the default value for 'network_mode' will change from not specified (which is equal to 'default') to the name of the first network in 'networks' if 'networks' has at least one entry and 'networks.continuous' is 'true'. You can change the behavior now by explicitly setting 'network_mode' to the name of the first network in 'networks', and remove this warning by setting 'network_mode' to 'default'. Please make sure that the value you set to 'network_mode' equals the inspection result for existing containers, otherwise the module will recreate them. You can find out the correct value by running 'docker inspect --format '{{.HostConfig.NetworkMode}}' <container_name>'. This feature will be removed from community.docker in version 2.0.0. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
changed: [192.168.99.115]

TASK [Deploy the Container MYSQL] *****
changed: [192.168.99.115]
TASK [Deploy the Container Wordpress] *****
changed: [192.168.99.115]
TASK [Deploy the Container Mongo DB] *****
changed: [192.168.99.115]

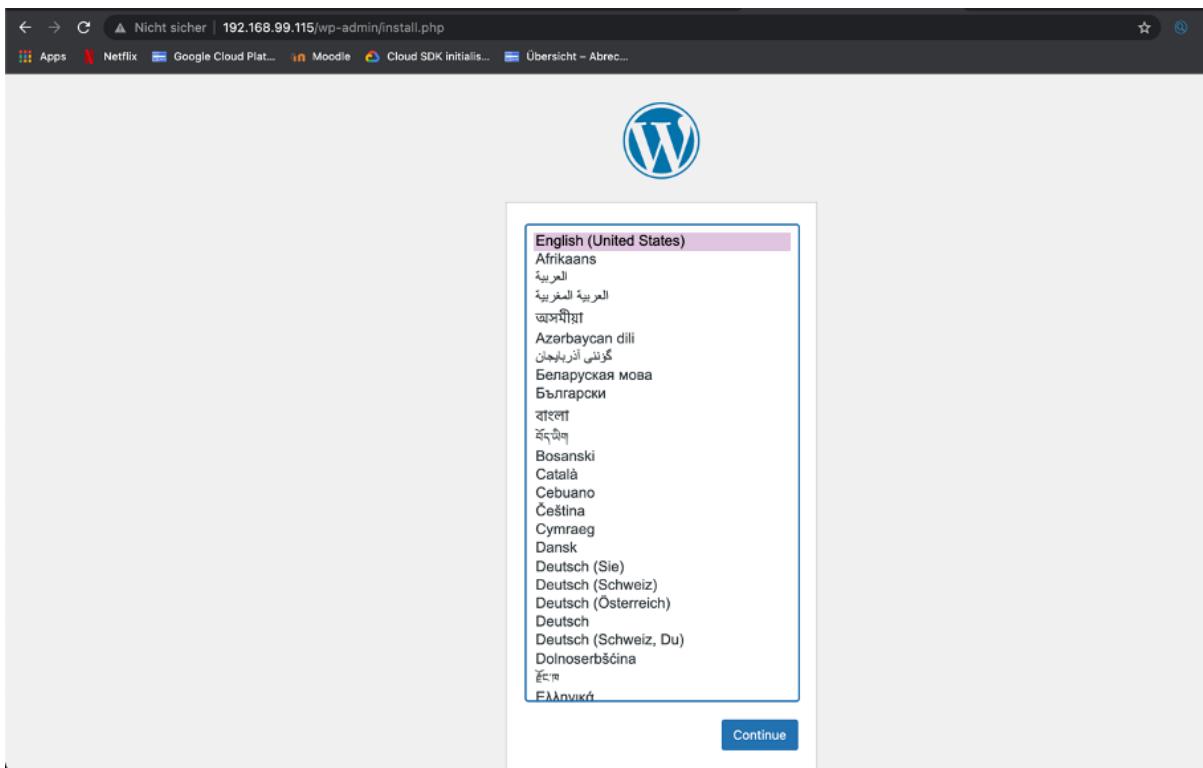
PLAY RECAP *****
192.168.99.115 : ok=6    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
melekozturk@Meleks-MacBook-Pro ansible_docker_project %
```

Screenshot 17: Das Netzwerk für die Docker- Interaktion wurde erstellt und die Container Deployd

```
melek@melek-VirtualBox:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e066bea5a5101 mongo "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:3000->3000/tcp, 27017/tcp mongodb
bfe0f2eb121c wordpress:latest "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:80->80/tcp wordpress
f144d428028c mysql:5.7 "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 3306/tcp, 33060/tcp, 0.0.0.0:4000->4000/tcp db
679464a3372 webserververapache "catalina.sh run" 2 minutes ago Up 2 minutes 80/tcp, 0.0.0.0:8081->8081/tcp meiq

melek@melek-VirtualBox:~$
```

Screenshot 18: Die Container mit den Anwendungen MySQL,Apache Tomcat, Wordpress, MongoDB laufen wie gewünscht in der VM



Screenshot 19: Der Container mit der Anwendung Wordpress

The screenshot shows a web browser window with the URL 127.0.0.1:8081. The page title is "Apache Tomcat/8.5.66". At the top, there's a navigation bar with links to Home, Documentation, Configuration, Examples, Wiki, and Mailing Lists. On the right, there's a "Find Help" button and the Apache Software Foundation logo. A green banner at the top says "If you're seeing this, you've successfully installed Tomcat. Congratulations!". Below the banner is a cartoon cat icon. To the right of the cat, under "Recommended Reading", are links to Security Considerations How-To, Manager Application How-To, and Clustering/Session Replication How-To. To the right of these links are three buttons: "Server Status", "Manager App", and "Host Manager". Under "Developer Quick Start", there are four categories: Tomcat Setup, Realms & AAA, Examples, and Servlet Specifications. Each category has a sub-section with links. The "Managing Tomcat" section includes a note about security and access to the manager webapp. The "Documentation" section links to Tomcat 8.5 Documentation, Configuration, and Wiki. The "Getting Help" section links to FAQ and Mailing Lists, listing several mailing lists like tomcat-announce, tomcat-users, taglibs-user, and tomcat-dev.

Screenshot 20: Der Container mit der Anwendung Apache Tomcat läuft

In der letzten Phase wird ersichtlich, dass die Kommunikation zwischen den Containern klappt.

```
[melek@melek-VirtualBox:~$ docker exec -it me1eq sh
[/opt/apache-tomcat-9.0.46/webapps # ping db
PING db (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.285 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.092 ms
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.089 ms
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.092 ms
64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.093 ms
64 bytes from 172.18.0.3: seq=5 ttl=64 time=0.091 ms
64 bytes from 172.18.0.3: seq=6 ttl=64 time=0.075 ms
64 bytes from 172.18.0.3: seq=7 ttl=64 time=0.079 ms
64 bytes from 172.18.0.3: seq=8 ttl=64 time=0.109 ms
64 bytes from 172.18.0.3: seq=9 ttl=64 time=0.091 ms
64 bytes from 172.18.0.3: seq=10 ttl=64 time=0.083 ms
64 bytes from 172.18.0.3: seq=11 ttl=64 time=0.083 ms
64 bytes from 172.18.0.3: seq=12 ttl=64 time=0.076 ms
64 bytes from 172.18.0.3: seq=13 ttl=64 time=0.106 ms
64 bytes from 172.18.0.3: seq=14 ttl=64 time=0.092 ms
64 bytes from 172.18.0.3: seq=15 ttl=64 time=0.107 ms
64 bytes from 172.18.0.3: seq=16 ttl=64 time=0.106 ms
```

Screenshot 21: Kommunikation zwischen Apache Tomcat und MySQL

```
/opt/apache-tomcat-9.0.46/webapps # ping wordpress
PING wordpress (172.18.0.4): 56 data bytes
64 bytes from 172.18.0.4: seq=0 ttl=64 time=0.083 ms
64 bytes from 172.18.0.4: seq=1 ttl=64 time=0.193 ms
64 bytes from 172.18.0.4: seq=2 ttl=64 time=0.133 ms
64 bytes from 172.18.0.4: seq=3 ttl=64 time=0.170 ms
64 bytes from 172.18.0.4: seq=4 ttl=64 time=0.116 ms
64 bytes from 172.18.0.4: seq=5 ttl=64 time=0.126 ms
64 bytes from 172.18.0.4: seq=6 ttl=64 time=0.095 ms
64 bytes from 172.18.0.4: seq=7 ttl=64 time=0.130 ms
64 bytes from 172.18.0.4: seq=8 ttl=64 time=0.183 ms
64 bytes from 172.18.0.4: seq=9 ttl=64 time=0.101 ms
64 bytes from 172.18.0.4: seq=10 ttl=64 time=0.132 ms
64 bytes from 172.18.0.4: seq=11 ttl=64 time=0.087 ms
^C
--- wordpress ping statistics ---
12 packets transmitted, 12 packets received, 0% packet loss
round-trip min/avg/max = 0.083/0.129/0.193 ms
/opt/apache-tomcat-9.0.46/webapps #
```

Screenshot 22: Kommunikation zwischen Apache Tomcat und WordPress

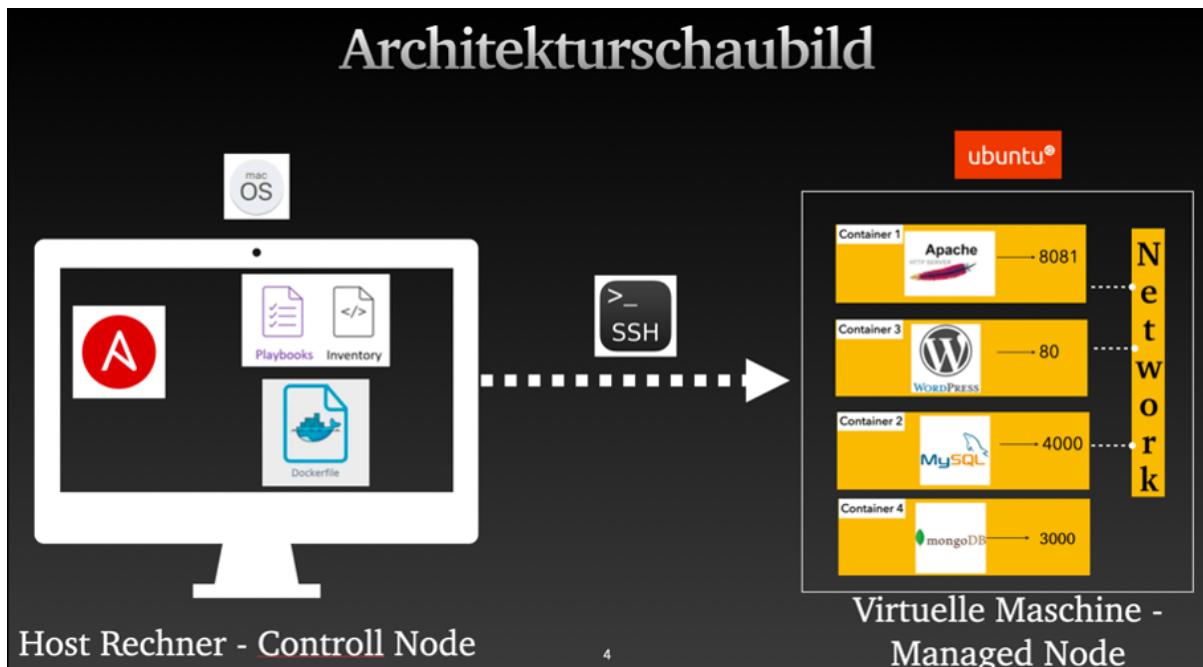
```
melek@melek-VirtualBox:~$ docker exec -it me1eq sh  
/opt/apache-tomcat-9.0.46/webapps # ping mongodb  
ping: bad address 'mongodb'
```

Screenshot 23: Die Datenbank mongodb hat nicht denselben Netzwerk zugewiesen bekommen deshalb funktioniert die Verbindung nicht von Apache Tomcat zu MongoDB

```
        ],  
        "SandboxKey": "/var/run/docker/netns/d735a7b31759",  
        "SecondaryIPAddresses": null,  
        "SecondaryIPv6Addresses": null,  
        "EndpointID": "",  
        "Gateway": "",  
        "GlobalIPv6Address": "",  
        "GlobalIPv6PrefixLen": 0,  
        "IPAddress": "",  
        "IPPrefixLen": 0,  
        "IPv6Gateway": "",  
        "MacAddress": "",  
        "Networks": {  
            "connection": {  
                "IPAMConfig": null,  
                "Links": null,  
                "Aliases": [  
                    "972c5204fc58"  
                ],  
                "NetworkID": "d75185c6eab182fc67bcdee4a0dad11a3636b31c5887771ae  
3c0705d99fc8224",  
                "EndpointID": "b48cdee881d2c202b798b3692a8edfe9a182cff27bf4f990  
a93f85824d63941b",  
                "Gateway": "172.18.0.1",  
                "IPAddress": "172.18.0.2",  
                "IPPrefixLen": 16,  
                "IPv6Gateway": "",  
                "GlobalIPv6Address": "",  
                "GlobalIPv6PrefixLen": 0,  
                "MacAddress": "02:42:ac:12:00:02",  
                "DriverOpts": null  
            }  
        }  
    }  
]  
melek@melek-VirtualBox:~$ █
```

Screenshot 24: Hier ist das Netzwerk namens „connection“ ersichtlich, an den sich 3 Container bedienen

6. Architektschaubild



Im Folgenden werde ich das Architektschaubild kurz erläutern. Denn die Schritte wurden oben bereits ausgiebig vorgestellt. In dieser Darstellung ist ersichtlich, dass ich auf meinem Host Rechner mit dem macOS Betriebssystem Ansible installiert habe. Mein Rechner stellt in der Ansible Architektur den Controller Node. Der Controller Node ist dafür zuständig die Definitionsdateien wie Playbooks, Inventory Datei und Dockerfile auf den Servern auszuführen. Dafür benötigt es eine sichere Verbindung und das stellt SSH Verbindung dar. Nachdem die SSH Verbindung realisiert wurde, können die individuellen Konfigurationen auf den gewünschten Virtuellen Maschinen bereitgestellt werden. In dem Schaubild wurde eine virtuelle Maschine angelegt und darauf das Betriebssystem Ubuntu installiert. Die virtuelle Maschine stellt den Managed Node dar. Denn das ist der Knoten der von dem Host Rechner konfiguriert wird. Die Infrastruktur soll folgende Container beinhalten:

- Apache Tomcat Webserver
- WordPress-Software
- MySQL
- mongoDB

Diese Container laufen auf unterschiedlichen Ports. Jedoch nutzen sie dasselbe Netzwerk. Diese Konfiguration dient zur Kommunikation zwischen den Containern auf verteilten Anwendungen. Der vierte Container mongoDB wurde von mir bewusst nicht mit in das Netzwerk miteingeschlossen, da ich demonstrieren wollte, wie es ist, wenn einem Container kein Netzwerk zugeordnet ist. Nämlich sie

besitzen dann den default Netzwerk von Docker. Dies führt dazu, dass der vierte Container nicht mit den anderen Containern interagieren kann.

7. Erkenntnisse

Das Verständnis der beiden Technologien wurde durch das Projekt positiv gestärkt. Inzwischen verstehe ich warum Cloud Computing der Mega-Trend der IT-Branche ist. In den diversen IT-Seiten wie zum Beispiel Heise, habe ich oft zu den einzelnen Automatisierungstools etwas gelesen. Jetzt kann ich das gelesene bestätigen, denn mit der Anwendung habe ich gemerkt, dass die Vorteile wie, zeitsparend, effektiv, effizient, transparent, dynamisch, skalierbar sich bekräftigt hat. Das Verwenden von qualitativen und fertig konfigurierten Skripten senkt das Fehlerpotenzial erheblich, dass bei einer manuellen Administration gegeben ist. Das wichtigste, dass ich aus diesem Projekt mitnehme ist, dass ich nicht die ganze Infrastruktur auf den Kopf stellen muss, wenn ich etwas anpassen, ändern oder ein Teil neu konfigurieren muss. Zusätzlich ist es ratsam immer auf Provisierungstool setzen, wenn die Umgebungen immer gleich aufgesetzt werden muss.

8. Fazit

Zusammenfassend lässt sich sagen, dass es mir große Freude bereitet hat, dass ich meine Idee in die Tat umsetzen konnte. Dabei habe ich gemerkt, dass Cloud Computing meine Neugier geweckt hat . Deshalb möchte ich mich in Zukunft mehr damit auseinandersetzen. Im Wintersemester 21/22 werde ich voraussichtlich meine Thesis angehen und bin fest entschlossen, dass ich mein Themengebiet in dem Bereich Cloud Computing eingrenzen möchte.