Inhaltsverzeichnis

Abkürzungsverzeichnis

VM	Virtuelle Maschine
BS	Betriebssystem
IaC	Infrastructure as Code

1 Einleitung

1.1 Ausgangssituation

In einer zunehmend digitalisierten Welt werden Entwicklungsprojekte zunehmend größer, komplexer und anfälliger für Fehler. Zur Steigerung der Sicherheit und Skalierbarkeit des Entwicklungsprozesses wurde auf virtuelle Entwicklungsumgebungen, sogenannte virtuelle Maschinen (VM), zurückgegriffen. Allerdings muss eine solche Maschine zunächst vorbereitet und aufgesetzt werden, damit sie auch sinnvoll für das Projekt genutzt werden kann. Der hier beschriebene Prozess wird in der Praxis zumeist noch manuell ausgeführt. Dieser manuelle Aufsetzungsprozess ist jedoch zeitintensiv, wodurch wiederum hohe Kosten entstehen, insbesondere bei besonders großen Projekten. Vor diesem Problem stand ein Team der dotSource SE.

Im Rahmen dieser Praxisarbeit soll untersucht werden, inwiefern sich der Aufsetzungsprozess einer VM beschleunigen und automatisieren lässt.

1.2 IST-Zustand

Die bislang für ein Kundenprojekt genutzte VM ist bereits seit einem Jahr in Verwendung. Das verwendete Betriebssystem (BS) sowie der Datenbankstand des Projekts entsprachen nicht mehr dem aktuellen Stand. Aus diesem Grund wurde die Erstellung einer neuen VM initiiert. Als Virtualisierungssoftware findet VirtualBox Anwendung. Im Folgenden werden die Parameter der VM in VirtualBox angegeben. Nach erstmaligem Starten der VM muss das BS initial eingerichtet werden. Im Anschluss ist die Installation und Vorbereitung sämtlicher für die Entwicklung erforderlicher Programme erforderlich. Zu den erforderlichen Programmen zählen unter anderem IntelliJ, SAP Commerce, Git, Docker, SoapUI, Htop und ZSH. Da der Aufsetzungsprozess stets "von Grund auf" erfolgt, ist es erforderlich, die neu erstellte VM am Ende auf ihre Funktionalität und etwaige Fehler zu prüfen und gegebenenfalls zu optimieren. Im Anschluss kann die VM an alle Entwickler verteilt werden. Dies erfolgt typischerweise durch den Export der VM und deren anschließende Veröffentlichung in der Cloud. Nun steht es jedem Entwickler frei, die VM herunterzuladen und in seinen Arbeitsbereich zu

importieren. Der gesamte Vorgang erstreckte sich über einen Zeitraum von drei Monaten und erforderte die Zusammenarbeit von bis zu vier Entwicklern. Dieser Prozess kann durch geeignete Maßnahmen verkürzt werden.

1.3 SOLL-Zustand

Es ist vorgesehen, eine VM vollständig mittels Infrastructure as Code (IaC) aufzubauen, sodass die einzelnen Entwickler lediglich ein Skript ausführen müssen, welches die VM mit allen erforderlichen Eigenschaften erstellt. Im Falle von Erneuerungen oder Änderungen ist ausschließlich eine Modifikation des Skripts erforderlich, welches anschließend erneut ausgeführt wird. Das Ziel ist, die Aktualisierung der VM schnell und einfach zu gestalten. Als IaC-Tool soll dabei Vagrant zum Einsatz kommen.

Im Rahmen dieser Praxisarbeit soll der zuvor skizzierte Plan auf seine Umsetzbarkeit hin geprüft werden. Zu diesem Zweck ist es erforderlich, sich zunächst ausführlich mit den verwendeten Technologien zu befassen. Im Anschluss ist die Erstellung eines VM-Prototyps mit Vagrant vorgesehen, der lediglich die grundlegend notwendigste Software enthält. Die Durchführung dieses Schritts wird in dieser Arbeit dokumentiert. Zudem sollen möglicherweise auftretende Fehler und deren Lösungen angegeben werden.

2 Verwendete Technologien

2.1 Virtuelle Maschinen

2.1.1 Grundlagen

„Eine virtuelle Maschine ist eine Computing-Umgebung, die als isoliertes System mit CPU, Speicher, Netzwerkschnittstelle und Storage fungiert und aus einem Pool von Hardwareressourcen erstellt wurde.“[1] Des Weiteren ist zu erwähnen, dass die virtuelle Maschine über ein eigenes BS verfügt, welches unabhängig vom Host-System läuft. Das Host-System ist unabdingbar für die Erstellung einer virtuellen Maschine. Es handelt sich hierbei um die physische Maschine, auf welcher die VM ausgeführt wird. Das Host-System stellt die für die VM benötigten Ressourcen zur Verfügung. VMs werden in dieser Beziehung als Gast bezeichnet.

2.1.2 Virtualisierung

Der Begriff "Virtualisierung" bezeichnet eine Technologie, welche traditionell hardwaregebundene Ressourcen nutzt, um diese auf mehrere virtuelle Nutzer zu verteilen. Dadurch soll die vollständige Kapazität der physischen Maschine ausgenutzt werden.[2] Bei den virtuellen Nutzern handelt es sich dabei um VMs. Es existieren fünf Arten der Virtualisierung:

1. Bei der Hardwarevirtualisierung wird die gesamte Hardware, inklusive BS und Computerversion, virtualisiert. In der Konsequenz resultiert dies in einem virtuellen, zusammengefassten Primärserver, d. h. einem identischen Klon des Host-Systems.
2. Im Rahmen der Softwarevirtualisierung wird ein Gastsystem mit gleicher Hardware wie das Hostsystem, jedoch mit unterschiedlicher Software erstellt.
3. Unter Speichervirtualisierung wird das Zusammenführen von mehreren physischen Speichergeräten zu einer großen Speichereinheit zur Erhöhung von Performance und Geschwindigkeit verstanden.[3]

[1] [RED 24a]
[2] Vgl. Ebenda
[3] Vgl. [ORA 24]

4. Die Netzwerkvirtualisierung stellt durch das Zuweisen von Bandbreiten zu Kanälen Ressourcen für Server und Geräte in Echtzeit bereit.
5. Die Desktopvirtualisierung ermöglicht den Zugriff auf den Desktop von überall mittels Trennung und Speicherung des Desktops auf einem Remote-Server.[4]

2.1.3 Hypervisor

Die Definition von VMs erfolgt üblicherweise in einer einzelnen Datei. Ein Transfer zwischen verschiedenen Rechnern ist somit möglich. In der Folge kann die Datei an sämtlichen Orten geöffnet und ausgeführt werden, wobei eine konsistente Funktionalität gewährleistet ist. Die Verantwortung hierfür obliegt dem sogenannten Hypervisor.

Die Funktion eines Hypervisors besteht in der Verwaltung der Hardware sowie der Trennung physischer Ressourcen von der virtuellen Umgebung. Sofern erforderlich, können die physischen Ressourcen für die VM freigegeben werden. Der Hypervisor instruiert die physischen Komponenten des Hosts, die Befehle des Gastes auszuführen. In Linux-Umgebungen wird der Hypervisor als Kernel-based Hypervisor bezeichnet. Unter Windows wird die Funktion durch den Hyper-V-Hypervisor ausgeführt. In Bezug auf Hypervisor-Technologien kann zwischen zwei grundlegenden Typen unterschieden werden:

Ein Typ-1-Hypervisor, auch als Bare Metal Hypervisor bezeichnet, schreibt die benötigten Ressourcen direkt der Hardware zu. Ein Beispiel für einen solchen Hypervisor ist der Kernel-based Hypervisor unter Linux.

Ein Typ-2-Hypervisor, auch als Hosted Hypervisor bezeichnet, funktioniert anders. Hierbei werden die von der VM benötigten Ressourcen zunächst beim Host angefordert, bevor dieser die angeforderte Hardware zur Verfügung stellt. Ein Beispiel für einen solchen Hypervisor ist Oracles VirtualBox.[5]

2.1.4 Vorteile von VMs

Die Verwendung von VMs kann aus einer Vielzahl von Gründen heraus begründet werden. Die Zusammenfassung vieler kleiner Workloads auf einen Computer, welche

[4] Vgl. [ORA 24]
[5] Vgl. [RED 24a]

durch Virtualisierung ermöglicht wird, führt zu einer Effizienzsteigerung sowie einer Reduktion der IT-Kosten. Darüber hinaus ist durch den Einsatz von VMs die Anzahl der zu wartenden Hardwarekomponenten geringer. Des Weiteren ermöglichen VMs die kostengünstige Ausführung älterer Anwendungen unter Verwendung aktueller Hardware. Zudem können VMs einfach und schnell zwischen mehreren Geräten ausgetauscht und vervielfältigt werden[6]. Zusätzlich bieten VMs eine sogenannte Desaster-Recovery-Option, welche im Falle eines schwerwiegenden Fehlers innerhalb der VM ein Zurückkehren zu einem vorher erstellten Zurücksetzpunkt ermöglicht. Letztlich ist zu erwähnen, dass die Daten einer VM isoliert vom Rest des Systems sind, wodurch sich VMs gut zum Testen eignen.[7]

2.2 Infrastructure as Code

2.2.1 Grundlagen

Infrastructure as Code (IaC) ist ein IT-Paradigma, welches unter anderem Software, Infrastruktur, Speicher, Rechenleistung und Hardwarestrukturen in Computersprache beschreibt. Einfach ausgedrückt programmiert man mit IaC Hardware- und Softwarestrukturen als ausführbaren Code. Dieser kann dann jederzeit problemlos angepasst, dupliziert, gelöscht, versioniert und verteilt werden.[8] Um das zu erreichen, werden Konfigurationsdateien erstellt. Diese enthalten alle Infrastrukturspezifikationen.[9]

2.2.2 Anforderungen

Über die letzten Jahre stiegen die Ansprüche an Softwareprodukte stark an. Daraus ergaben sich folgende Konsequenzen: Entwicklungszyklen wurden kürzer, das Streben nach maximaler Verfügbarkeit und Flexibilität stieg an. Dafür musste die zugrundeliegende Hardware- und Software-Infrastruktur verbessert werden. Dies ist der Ansatzpunkt für IaC.[10]

---

[6] Vgl. [ORA 24]

[7] Vgl. [RED 24a]

[8] Vgl. [ION 23]

[9] Vgl. [RED 24b]

[10] Vgl. [ION 23]

IaC soll die Qualität und Effizienz der Infrastruktur erhöhen. Um das zu erreichen, muss es folgende Anforderungen erfüllen:

- Manuelle Prozesse sollen automatisiert werden
- Schaffen eines flexiblen Workflows
- Erhöhte Transparenz bei der Änderung von Inhalten
- Hardware-Setups sollen genauso „testbar" sein wie Software[11]

2.2.3 IaC-Ansätze

Es existieren zwei IaC-Ansätze, nach welchen man Infrastruktur erstellen kann. Der Erste ist der deklarative Ansatz. Bei diesem wird der Finalzustand der Infrastruktur definiert[12], inklusive aller benötigten Ressourcen und Eigenschaften. Während der Laufzeit der Infrastruktur wird eine Liste mit allen aktuellen Systemzuständen der Systemobjekte geführt. Diese erleichtert das Terminieren dieser beim Herunterfahren der Infrastruktur.[13] Mit dem deklarativen Ansatz können die einzelnen Bestandteile der Infrastruktur einfach verwaltet und konfiguriert werden.[14]

Der zweite Ansatz ist der imperative Ansatz. Hierbei werden konkrete Befehle angegeben, welche zum Erreichen der gewünschten Konfiguration ausgeführt werden müssen. Dabei ist eine korrekte Angabe und Ausführung der Befehle in der richtigen Reihenfolge erforderlich.[15] Der Vorteil einer solchen Vorgehensweise liegt in der erhöhten Kontrolle über den Erschaffungsprozess. Allerdings birgt die damit einhergehende Komplexität des geschriebenen Codes das Risiko, dass die Umsetzung zeit- und ressourcenintensiv wird. Zudem ist diese Vorgehensweise nur begrenzt skalierbar.[16] Aufgrund dieser Nachteile verwenden viele IaC-Tools heutzutage den deklarativen Ansatz.[17]

---

1 [1]Vgl. [ION 23]

1 [2]Vgl. [IBM 24]

1 [3]Vgl. [RED 24b]

1 [4]Vgl. [IBM 24]

1 [5]Vgl. [RED 24b]

1 [6]Vgl. [IBM 24]

1 [7]Vgl. [RED 24b]

2.2.4 Vorteile von IaC

Die Nutzung von IaC ist mit einer Vielzahl von Vorteilen verbunden. Zum einen kann damit die Effizienz des Entwicklungsprozesses gesteigert werden. So können Ressourcenmanagements durch IaC automatisiert werden, was zur Optimierung des Software Development Life Cycle führt.

Des Weiteren kann der Code zur Erstellung einer Infrastruktur so oft wie gewollt wiederverwendet werden. So muss benötigte Infrastruktur nicht jedes Mal neu „von Hand“ erstellt werden.

Die Nutzung von IaC ermöglicht die Versionsverwaltung der geschriebenen Konfigurationen. Alte Konfigurationen können so einfach wiederhergestellt werden. Dafür muss nur der richtige Code dafür ausgeführt werden. Zu guter Letzt können durch diese Automatisierungen Kosten und Zeit gespart werden.

2.2.5 Nachteile von IaC

Die Nutzung von IaC ist jedoch nicht ohne Nachteile. Zum einen ist das Konzeptionieren und erstmalige Aufsetzen mit einem hohen Aufwand sowie hohen Kosten verbunden.

Zudem ist viel Vorwissen in Bereichen wie beispielsweise Cloud-Infrastruktur, mehreren Programmiersprachen und im Umgang mit APIs erforderlich, was zu einer Erhöhung der Kosten in der Einarbeitungsphase führt.

Schließlich bedeutet ein Wechsel von herkömmlichen Strategien zur Aufsetzung von Infrastruktur hin zu IaC eine große Umstellung für die Administration des Projekts.[18]

2.3 Vagrant

2.3.1 Einführung

Vagrant ist ein Tool zum Erstellen und Verwalten von VMs, welches sich auf Automatisierung und Vereinfachung von Workflows fokussiert. Das Ziel dabei ist es, die Aufsetzungszeit von Entwicklungsumgebungen zu verkürzen. So beschreibt die Firma HashiCorp ihr IaC-Tool Vagrant.[19]

---

[1]
[18] Vgl. [ION 23]
[9] Vgl. [HAS 24a]

HashiCorps Gründer, Mitchell Hashimoto beschreibt Vagrant als „a Swiss Army knife for development environments. It does everything you need to create and manage them, and helps enforce good practices by encouraging the use of automation and an environment that more closely resembles production.“[20]

Vagrant wurde entwickelt, um auf Basis eines Betriebssystems eine VM zu erzeugen und deren physische Eigenschaften mit geringem Aufwand anzupassen. Des Weiteren ermöglicht Vagrant die Etablierung von Netzwerkschnittstellen, wodurch ein Zugriff auf die VM sowohl vom eigenen Computer als auch von anderen Computern im gleichen Netzwerk oder von anderen VMs aus möglich ist. Darüber hinaus ist Vagrant in der Lage, geteilte Ordner zwischen VM und Host zu erstellen, die VM zu booten, den Hostnamen der Maschine zu setzen sowie Software für die Maschine vorzubereiten.

Nach der Erstellung einer VM mit Vagrant kann dieses die Maschine herunterfahren, anhalten, fortsetzen, verpacken und zum Verteilen bereit machen. Ebenso ist das Löschen der Maschine samt virtuelle Festplatte und Metadaten möglich.

Vagrant ist darauf ausgelegt, ein Tool zum Erstellen von komplexen Entwicklungsumgebungen zu sein. Der Zeitaufwand für die Einrichtung einer solchen Umgebung soll mithilfe von Vagrant reduziert werden, um die Produktivität zu steigern. Darüber hinaus zielt Vagrant darauf ab, die Automatisierung von Entwicklungsumgebungen zu fördern.[21]

In den nachfolgenden Kapiteln werden einzelne Vagrant spezifische Konzepte erklärt, welche für die Praxisarbeit von Relevanz sind.

2.3.2 Vagrantfile

Die Vagrantfile ist eine Datei, welche den Typus einer Maschine sowie deren Provision (vgl. Kapitel 2.3.4 Provisioning) beinhaltet. Sie ist für jedes Vagrant-Projekt unerlässlich und dient als Fundament für die gesamte Maschine. Die Vagrantfile lässt sich als in einem Versionskontrollsystem nutzen und wird zwischen Entwicklern ausgetauscht. Es wird der Syntax der Programmiersprache Ruby verwendet.[22]

---

[20] [HAS 2013], S. 2

[21] Vgl. Ebenda S.2ff

[22] Vgl. [HAS 24d]

2.3.3 Boxen

Boxen sind verpackte Vagrant Umgebungen, welche in der Vagrantfile angegeben werden müssen. Diese Boxen können dabei entweder selbst erstellt werden, oder vorgefertigt von der Vagrant Website genommen werden.[23]

2.3.4 Provisioning

Durch Provisioning können Software, alternative Konfigurationen und mehr automatisch vorgenommen werden. Diese Aufgabe übernimmt der sogenannte Provisioner. Dieser erweist sich als besonders nützlich, da zum Beispiel Boxen nicht immer auf den benötigten Use Case passen, und so weiter angepasst werden können. So müssen die weiteren benötigten Programme nicht „von Hand" installiert werden.[24]

[23]Vgl. [HAS 24c]
[24]Vgl. [HAS 24b]

3 Aufsetzen des Prototyps

3.1 Anforderungen

Unter Berücksichtigung der vorherrschenden Vorteile eines Typ-2-Hypervisors wurde entschieden, VirtualBox für das Hosting zu verwenden. Dadurch sind einige Befehle des folgenden Kapitels VirtualBox-spezifisch. Der Prototyp ist so zu konfigurieren, dass er die gleichen technischen Spezifikationen aufweist wie die aktuelle VM des Projektes. Diesbezüglich sind folgende technische Konfigurationen erforderlich: 20000 MB RAM, 8 Kerne der CPU, 128 MB Grafikspeicher, 3D-Beschleunigung, Kompatibilität mit den USB 1.0, USB 2.0 und USB 3.0 sowie die Deaktivierung von PAE/NX. Die genauen Funktionen der einzelnen Komponenten sind für die vorliegende Arbeit von untergeordneter Bedeutung. Als BS wird eine Version von Ubuntu verwendet.

Standardmäßig soll Xfce4 als Benutzeroberfläche und das deutsche Tastaturlayout eingesetzt werden. Darüber hinaus werden die VirtualBox Guest Addition sowie weitere wichtige Softwarekomponenten installiert, darunter die JetBrains Toolbox und IntelliJ Ultimate Edition, Htop, Git und der SAP Commerce Demoshop. Die Einrichtung dieser Programme soll ebenfalls über Vagrant erfolgen.

3.2 Durchführung

3.2.1 Einrichten von Vagrant

Zunächst muss ein Ordner erstellt werden, in welchem die Vagrant-Dateien gespeichert werden sollen. Dieser wurde "Sitepoint" genannt und liegt auf dem Desktop. In diesem Ordner wird nun mit dem Befehl "vagrant init" eine Vagrantfile mit ersten Probedaten erstellt. Um die Funktionalität von Vagrant zu testen, wurde von dieser Vagrantfile mit dem Befehl "vagrant up" eine Probe-VM erstellt. Dabei wurde jedoch folgender Fehler geworfen:

```
1. There was an error while executing `VBoxManage`, a CLI used by Vagrant
2. for controlling VirtualBox. The command and stderr is shown below.
3.
4. Command: ["startvm", "c12bde4c-5364-4f55-861f-8c68dca1a273", "--type", "headless"]
5.
6. Stderr: VBoxManage.exe: error: The virtual machine
'sitepoint_default_1723616262718_3757' has terminated unexpectedly during startup with exit
code 1 (0x1).  More details may be available in 'C:\Users\[redacted]\VirtualBox
VMs\sitepoint_default_1723616262718_3757\Logs\VBoxHardening.log'
7. VBoxManage.exe: error: Details: code E_FAIL (0x80004005), component MachineWrap,
interface IMachine
```

In der Angegebenen Datei findet sich folgender Fehler:

```
 . 6e94.30d0: Error (rc=-5607):

(0x265000)
 . 6e94.30d0: Error (rc=-5607):

(0x262000) isn't close enough to the mapping size (0x265000)
5. 6e94.30d0: Error -5607 in supR3HardNtChildPurify! (enmWhat=5)
6. 6e94.30d0: supHardenedWinVerifyProcess failed with -5607: ntdll.dll: SizeOfImage
(0x262000) isn't close enough to the mapping size (0x265000)
```

Der Fehler scheint durch die Windows-Version 11 Pro 24H2 bedingt zu sein und ist bereits seit einiger Zeit bekannt.[25] Er konnte durch den Wechsel zu Windows-Version 11 Pro 23H2 behoben werden. Nach der Installation von Vagrant und der Ausführung von Tests mit der Probe-VM wurden keine Fehlermeldungen mehr angezeigt. Dies lässt den Schluss zu, dass die Fehlerursache tatsächlich in der Windows-Version liegt. Bis zum Zeitpunkt der Veröffentlichung wurde der Fehler jedoch noch nicht behoben.

3.2.2 Konfiguration der Vagrantfile

Nun kann die Vagrantfile konfiguriert werden. Diese Konfiguration lautet wie folgt:

```
 1. # -*- mode: ruby -*-
 2. # vi: set ft=ruby :
 3.
 4. Vagrant.configure("2") do |config|
 5.
 6.   config.vm.provider "virtualbox" do |vb|
 7.     vb.name="Praxisarbeit2"
 8.     vb.memory = "20000"
 9.     vb.cpus = 8
10.     vb.gui = true
11.     vb.customize ["modifyvm", :id, "--vram", "128"]
12.     vb.customize ["modifyvm", :id, "--graphicscontroller", "VMSVGA"]
13.     vb.customize ["modifyvm", :id, "--accelerate3d", "on"]
14.     vb.customize ["modifyvm", :id, "--usb", "on"]
15.     vb.customize ["modifyvm", :id, "--usbehci", "on"]
16.     vb.customize ["modifyvm", :id, "--usbohci", "on"]
17.     vb.customize ["modifyvm", :id, "--usbxhci", "on"]
18.     vb.customize ["modifyvm", :id, "--pae", "off"]
19.   end
20.
21.     config.vm.provision "shell", path: "Provision.sh"
22.     config.vm.hostname="Praxisarbeit"
23.
24.     config.vm.box = "ubuntu/bionic64"
25.
26. end
```

Die ersten beiden Zeilen dienen der Kennzeichnung des nachfolgenden Quelltexts, dass die Syntax von Ruby verwendet wird. Dies ermöglicht eine korrekte Zuordnung des

2

[5] Vgl. [GIT 24]

Quelltexts zu der entsprechenden Programmiersprache, was insbesondere bei der Verwendung von Editoren wie Vim von Vorteil sein kann.

In Zeile vier beginnt die Konfiguration der VM. Die Angabe "Vagrant.configure("2")" bezeichnet die Konfigurationsversion für Vagrant, in diesem Fall die Version "2". Der Ausdruck "do |config|" definiert ein Objekt, welches den Zugriff auf die Vagrant-Optionen ermöglicht. Es stellt die Basis für alle weiteren Konfigurationen dar.

Im Anschluss erfolgt die eigentliche Konfiguration der VM. In den Zeilen acht bis zehn werden der Name, der verfügbare Arbeitsspeicher sowie die Anzahl der CPU-Kerne definiert und das GUI aktiviert. Vagrant bietet hierfür direkte Zugriffsmöglichkeiten, beispielsweise über "vb.memory" für den RAM. Für die nachfolgenden Konfigurationen besteht keine Möglichkeit eines direkten Zugriffs. An dieser Stelle ist der Rückgriff auf den Befehl "vb.customize" erforderlich. In der Folge können weitere Eigenschaften der VM angepasst werden. Die Definition der genannten Parameter erfolgt in eckigen Klammern. Der Befehl "modifyvm" erlaubt die Modifikation einer Eigenschaft der VM. In das nachfolgende Feld ist die ID der gegenwärtig aktiven VM einzutragen. Da Vagrant lediglich auf die VM im aktuellen Ordner zugreifen kann, genügt die Angabe ":id". Vagrant übernimmt die ID der aktuellen VM an dieser Stelle. Der darauffolgende Teil definiert, welche Konfiguration genau geändert werden soll. Zu guter Letzt muss noch festgelegt werden, welche Änderung vorgenommen werden soll. Beispielsweise kann die USB-Kompatibilität aktiviert oder der Grafikcontroller auf VMSVGA festgelegt werden.

Im Anschluss ist eine Provisionierung der VM erforderlich, welche die Installation der Programme übernimmt. Die Verantwortung für die Ausführung dieses Vorgangs trägt der Befehl "config.vm.provision". Dieser kann entweder ein String mit der Provisionierung oder eine eigene Datei, welche die entsprechenden Befehle enthält, übergeben werden. Aus Gründen der Übersichtlichkeit wurde sich für die Verwendung einer eigenen Datei entschieden. Zunächst ist anzugeben, um welchen Dateityp es sich handelt, in diesem Fall eine Shell-Datei. Anschließend ist der Pfad zu dieser Datei anzugeben. Da die Datei im gleichen Ordner wie die Vagrantfile liegt, genügt der Name der Provisioning-Datei. Auf ihren Inhalt wird im nächsten Kapitel eingegangen.

Im Anschluss erfolgt die Definition des Hostnamens mittels des Befehls "config.vm.hostname=Praxisarbeit" in Zeile 22. Infolgedessen kann die VM im Netzwerk unter dem Namen "Praxisarbeit" identifiziert werden.

In Zeile 24 erfolgt schließlich die Definition der Vagrant Box. Dazu wird die Box "bionic64" aus dem Vagrant Cloud Katalog verwendet, welche die Ubuntu Version 18.04.6 LTS bereitstellt. Eine eigens erstellte Box mit einer moderneren Version wäre zwar für dieses Projekt vorteilhafter gewesen, jedoch hätte dies den zeitlichen Rahmen gesprengt.

3.2.3 Konfiguration der Provision.sh

In der vorliegenden Datei besteht die Möglichkeit, Linux-Befehle anzugeben, welche anschließend die erforderlichen Programme installieren und erforderliche Konfigurationen vornehmen sollen. Zuvorderst ist sicherzustellen, dass die Software-Pakete auf dem neuesten Stand sind. Der Befehl "apt-get update" wird verwendet, um die aktuellen Informationen über verfügbare Software-Pakete bereitzustellen. Eine Aktualisierung oder Neuinstallation von Software erfolgt dabei jedoch nicht.

Da in der bionic64 Box standardmäßig nur das US-amerikanische Tastaturlayout installiert und eingerichtet ist, soll nun das deutsche Tastaturlayout installiert und angewendet werden. Dies erfolgt über folgende Befehle:

```
1. #installation und anwendung der Deutschen Tastatur
2. sudo apt-get install -y console-setup
3. sudo localectl set-keymap de
```

Die Installation des erforderlichen Pakets zur Änderung der Tastaturkonfiguration erfolgt mittels des in Zeile 2 aufgeführten Befehls. Mit Hilfe des zuvor installierten Pakets sowie des in Zeile 3 aufgeführten Befehls kann nun das Tastaturlayout auf Deutsch geändert werden.

Im nächsten Schritt erfolgt die Installation der Benutzeroberfläche Xfce4, da die virtuelle Maschine derzeit lediglich über eine Kommandozeile steuerbar ist. Die Funktionsfähigkeit von Xfce4 setzt voraus, dass neben Xfce4 auch die erforderlichen VM Guest Additions installiert werden. Im Anschluss ist sicherzustellen, dass jeder Nutzer die Benutzeroberfläche nutzen kann. Dies erfolgt über den sogenannten X-Server, hier

X11, dessen Start durch jeden Nutzer zu ermöglichen ist. Um diese beiden Ziele zu erreichen, müssen folgende Befehle der Provision.sh-Datei hinzugefügt werden:

```
1. #installieren und aktivieren von xfce4 und Guest Addition
2. sudo apt-get install -y xfce4 xinit virtualbox-guest-dkms virtualbox-guest-
utils virtualbox-guest-x11
3. sudo sed -i 's/allowed_users=.*$/allowed_users=anybody' /etc/X11/Xwrapper.config
```

Um die Kompatibilität von Xfce4 mit dem zuvor installierten deutschen Tastaturlayout zu gewährleisten, ist es erforderlich, das Tastaturlayout für das X11-System ebenfalls auf Deutsch zu ändern. Dies kann mit dem folgenden Befehl durchgeführt werden:

```
1. sudo localectl set-x11-keymap de
```

Im nächsten Schritt ist die Erstellung eines neuen Benutzers für die Entwickler vorgesehen. Dieser soll den Namen "developer" und das Passwort "passwort" erhalten. Zusätzlich ist die Vergabe von Rechten zur Ausführung von sudo-Befehlen erforderlich. Dazu sind folgende Befehle notwendig:

```
1. #einrichten neuer User
2. sudo useradd -m -s /bin/bash developer
3. echo 'developer:passwort' | sudo chpasswd
4. sudo usermod -aG sudo developer
```

Die Überwachung laufender Programme auf der VM kann mit Hilfe des Programmes "Htop" gewährleistet werden. Dazu ist lediglich die Ausführung des folgenden Befehls zur Installation erforderlich:

```
1. sudo apt-get install htop
```

Auch der Chrome Browser und das Programm "unzip" können auf diese Art installiert werden. Dafür müssen folgende Befehle ausgeführt werden:

```
1. sudo apt-get install -y chromium-browser
2. sudo apt-get install unzip
```

Im nächsten Schritt erfolgt die Installation und Konfiguration von Git. Zur Installation wird wie im vorherigen Schritt der Befehl "apt-get install" genutzt. Anschließend müssen die Benutzerdaten, d. h. der Benutzername und die E-Mail-Adresse, eingegeben werden. Um dies zu bewerkstelligen, muss der folgende Code in die Provision.sh-Datei eingefügt werden:

```
1. #installieren und Einrichten Git
2. sudo apt-get install -y git
3. git config --global user.name "Name einfügen"
4. git config --global user.email "E-Mail einfügen"
```

Im Rahmen der vorliegenden Projektarbeit wurden der Name sowie die E-Mail-Adresse aus datenschutzrechtlichen Erwägungen sowie zur Gewährleistung der Firmensicherheit entfernt.

Als nächstes muss die JetBrains Toolbox und IntelliJ installiert werden. Folgende Befehle müssen ausgeführt werden, um die JetBrains Toolbox zu installieren und einzurichten:

```
1. #installieren von Jetbrains Toolbox
2. wget -c https://download.jetbrains.com/toolbox/jetbrains-toolbox-2.4.2.32922.tar.gz
3. sudo tar -xzf jetbrains-toolbox-2.4.2.32922.tar.gz -C /opt
```

Das Tool "wget" ermöglicht den Download von Dateien aus der angegebenen URL. In diesem Fall wird dadurch die zum Zeitpunkt der Veröffentlichung der Arbeit aktuelle Version der JetBrains Toolbox installiert. Bei der heruntergeladenen Datei handelt es sich um eine .tar.gz-Datei, das heißt, sie ist verpackt und muss erst mittels des "tar"-Befehls entpackt werden. Anschließend kann die entpackte Datei unter /opt gespeichert werden. Dort kann sie dann in der VM ausgeführt werden, um die JetBrains Toolbox zu starten.

Im Anschluss ist die Installation von IntelliJ Idea Ultimate mittels der JetBrains Toolbox geplant. Eine Installation von IntelliJ über die JetBrains Toolbox ist jedoch nur über eine grafische Oberfläche und nicht über Kommandozeilenbefehle möglich, was dem Ziel der Automatisierung widerspricht. Nach der Verteilung der VM müsste jeder Entwickler IntelliJ händisch installieren. IntelliJ kann folglich nicht über die JetBrains Toolbox installiert werden. Die Installation erfolgt demnach unabhängig von der JetBrains Toolbox. Dazu ist folgender Befehl erforderlich:

```
1. sudo snap install intellij-idea-ultimate --classic
```

Im Folgenden soll die Installation und Einrichtung von SAP Commerce inklusive des Electronics Test Shops erläutert werden. Zuvor ist jedoch die Installation der SAP Machine erforderlich, welche eine von der SAP veröffentlichte Version des Open Java Development Kits darstellt.[26] Die Installation erfolgt nach demselben Schema wie die Installation der JetBrains Toolbox. Der auszuführende Code lautet demnach wie folgt:

```
1. #SAP Machine installieren
2. wget https://github.com/SAP/SapMachine/releases/download/sapmachine-17.0.12/sapmachine-
jdk-17.0.12_linux-x64_bin.tar.gz
3. sudo tar -xzf sapmachine-jdk-17.0.12_linux-x64_bin.tar.gz -C /opt
```

2

[6]Vgl. [SAP 24]

Anschließend kann SAP Commerce installiert werden. Der dafür benötigte Code lautet wie folgt:

```
 1. wget --user="Name einfügen"--password="Passwort
einfügen" [redacted]/repository/hybris/hybris/commerce/2205.4/commerce-
2205.4.ZIP
 2. cd /home/developer && mkdir SAP
 3. mv /home/vagrant/commerce-2205.4.ZIP /home/developer/SAP
 4. cd /home/developer/SAP && unzip commerce-2205.4.ZIP
 5. export JAVA_HOME="/opt/sapmachine-jdk-17.0.12"
 6. export PATH="$PATH:/opt/sapmachine-jdk-17.0.12/bin"
 7. echo export JAVA_HOME="/opt/sapmachine-jdk-17.0.12" >> /home/developer/.bashrc
 8. echo export PATH="$PATH:/opt/sapmachine-jdk-17.0.12/bin" >> /home/developer/.bashrc
 9. git clone [redacted]/hybris/slim-install-
recipe.git /home/developer/SAP/installer/recipes/slim-install-recipe
10. cd /home/developer/SAP/installer && ./install.sh -r slim-install-recipe setup
```

Der Download von SAP Commerce erfolgt zunächst als ZIP-Archiv aus einem Nexus, ein firmeninternes Tool zum Managen von Repositorys. Für die Ausführung des Befehls "wget" ist die Angabe von Benutzername und Passwort für Nexus erforderlich. Auch hier wurden der Name sowie das Passwort aus datenschutzrechtlichen Erwägungen sowie zur Gewährleistung der Firmensicherheit entfernt.

In der Folge wird mittels der Befehle in Zeile zwei und drei ein neuer Ordner mit dem Namen "SAP" im Verzeichnis "/home/developer" erstellt. Im Anschluss wird die ZIP-komprimierte SAP Commerce-Datei in den zuvor erstellten Ordner verschoben und dort entpackt.

Im Anschluss ist eine Neusetzung der Variablen PATH und JAVA_HOME erforderlich. Diese beiden Umgebungsvariablen sind erforderlich, um den Java-Code zu kompilieren und auszuführen.[27] Im Anschluss werden die Variablen in den Zeilen fünf und sechs mit dem Befehl "export" angelegt. Die Variable JAVA_HOME wird dabei auf den Speicherort des SAP Machine gesetzt, während die Variable PATH auf den aktuellen Wert von PATH (mittels $PATH) festgelegt wird, wobei der Pfad zum /bin-Verzeichnisses des SAP Machine angehängt wird. Um auch zukünftig verfügbar zu sein, müssen diese Variablen der Datei .bashrc hinzugefügt werden. Dies erfolgt über die Befehle aus den Zeilen sieben und acht.

Nachdem die Installation von SAP Commerce abgeschlossen ist, muss im nächsten Schritt der Electronics Test Shop eingerichtet werden. Zu diesem Zweck wird zunächst in Zeile 9 ein Git-Repository geklont, welches das dafür erforderliche Installationsrezept

---

[2] [7]Vgl. [ARU 23]

beinhaltet. Die angezeigte Adresse wurde aus datenschutzrechtlichen Erwägungen sowie zur Gewährleistung der Firmensicherheit für diese Praxisarbeit abgeändert. Das Repository ist in das Verzeichnis "./installer/recipies" zu klonen, in welchem alle Installationsrezepte von SAP Commerce abgelegt sind. Im Gegensatz zu den von SAP angelegten Rezepten erzeugt dieses lediglich den Electronics Test Shop, beinhaltet einige nützliche Erweiterungen und modifiziert einige Einstellungen, um die Anwendung benutzerfreundlicher zu gestalten.

Mit dem Befehl aus Zeile zehn wird das Installationsrezept ausgeführt. Nach Abschluss dieses Schritts ist die Provision.sh-Datei für die Zwecke dieser Praxisarbeit vollständig.

4 Fazit

4.1 Zusammenfassung

Die Anwendung von Vagrant eröffnet das Potential, den Prozess der VM-Erstellung und deren Verteilung zu automatisieren, optimieren und zu beschleunigen. Nach dem Schreiben der benötigten Dateien können diese schnell und einfach verteilt werden. Dies ist unter anderem darauf zurückzuführen, dass die Dateien im Vergleich zu einer exportierten VM eine geringere Größe aufweisen. Des Weiteren können Änderungen zeitnah und ohne großen Aufwand implementiert werden. Anstatt die vorherigen Programme zu deinstallieren und neue zu installieren, genügt eine Anpassung der Provision.sh-Datei und eine erneute Ausführung, um die neuen Programme zu integrieren. Dies spart in der Entwicklung Zeit und Kosten. Zum Vergleich: Beim „händischen“ Aufsetzen einer VM werden mehre Stunden benötigt, hingegen das vollständige Ausführen der Vagrantfile dauerte 18 Minuten. Zudem ist Vagrants einfache Benutzbarkeit zu erwähnen, die hauptsächlich auf der Vermeidung komplexer grafischer Benutzeroberflächen und der vollständigen Ausführung in der Kommandozeile basiert.

Allerdings sind mit der Nutzung von Vagrant auch einige Nachteile verbunden. Obgleich Vagrant nach erstmaliger Ausarbeitung Zeit einspart, ist für die erstmalige Erstellung ein beträchtlicher Zeitaufwand zu veranschlagen. Diesbezüglich ist zu erwähnen, dass insbesondere für Entwickler, die bislang keine Erfahrung mit Vagrant oder ähnlichen IaC-Tools gesammelt haben, die Einarbeitungsphase einen beträchtlichen Zeitaufwand mit sich bringen kann. Des Weiteren ist zu bemerken, dass Vagrant zwar eine Vielzahl von Prozessen automatisieren kann, jedoch nicht alle. So konnte beispielsweise IntelliJ nicht über die JetBrains Toolbox installiert werden, sondern musste separat heruntergeladen werden. Obgleich eine Lösung für das dargestellte Problem gefunden werden konnte, wäre der Ansatz der Automatisierung überflüssig, sofern auch nur eine Aufgabe nicht automatisiert werden könnte. Folglich wäre entweder jeder Entwickler dazu verpflichtet, die nicht automatisierte Software selbst zu installieren beziehungsweise einzurichten, alternativ müsste sie im Vorfeld manuell installiert werden. In diesem Fall kann die VM lediglich als exportierte Datei an andere Entwickler weitergegeben werden, nicht jedoch als IaC in der Vagrantfile beziehungsweise der

Provision.sh-Datei. Derzeit stellt die fehlende Kompatibilität mit der aktuellen Windows 11 Pro Version 24H2 das größte Problem für Vagrant dar. Da sich VMs in dieser Version nicht über Vagrant erzeugen und starten lassen, ist Vagrant in diesem aktuellen Zustand nicht für die dotSource SE zu gebrauchen, da alle konventionell genutzten Geräte mit dieser Version laufen.

4.2 Ausblick in die Zukunft

Die Anwendung von Vagrant stellt ein erhebliches Potenzial zur Einsparung von Zeit und Ressourcen bei der Firma dotSource SE dar. Allerdings kann erst dann von einer tatsächlichen Nutzung ausgegangen werden, sobald die Kompatibilität mit der Windows 11 Pro Version 24H2 und höher gewährleistet ist. Bis zu diesem Zeitpunkt ist die Verwendung von Vagrant als IAC-Tool nicht möglich. Im Anschluss an die vorangehend beschriebene Arbeit kann eine Auseinandersetzung mit alternativen IaC-Tools erfolgen, welche eine Alternative zu Vagrant darstellen und mit aktuellen Windows-Versionen kompatibel sind. Sollte sich herausstellen, dass Vagrant von aktuellen Windows-Versionen unterstützt wird, wäre es möglich, sich erneut mit diesem Tool zu befassen. Im Anschluss an die Erstellung des ersten Prototyps ist eine Prüfung der Einrichtung und Kompatibilität weiterer Komponenten erforderlich. Dies umfasst beispielsweise Docker oder Datenbankimporte.

Literaturverzeichnis

[ARU 23] Arunachalam, Palaniappan: JAVA_HOME vs PATH Environment Variables, 2023, https://www.baeldung.com/java-home-vs-path-env-var, Abgerufen am: 2024.09.12

[GIT 24] GitHub: Vagrant up fails with supHardenedWinVerifyProcess failed with -5607: ntdll.dll on Windows 11 Pro Insider · Issue #13163 · hashicorp/vagrant, 2024, https://github.com/hashicorp/vagrant/issues/13163, Abgerufen am: 2024.08.29

[HAS 24a] HashiCorp: Introduction | Vagrant | HashiCorp Developer, 2024, https://developer.hashicorp.com/vagrant/intro, Abgerufen am: 2024.08.06

[HAS 24b] HashiCorp: Provisioning | Vagrant | HashiCorp Developer, 2024, https://developer.hashicorp.com/vagrant/docs/provisioning, Abgerufen am: 2024.08.13

[HAS 24c] HashiCorp: Vagrant Boxes | Vagrant | HashiCorp Developer, 2024, https://developer.hashicorp.com/vagrant/docs/boxes, Abgerufen am: 2024.08.13

[HAS 24d] HashiCorp: Vagrantfile | Vagrant | HashiCorp Developer, 2024, https://developer.hashicorp.com/vagrant/docs/vagrantfile, Abgerufen am: 2024.08.13

[HAS 2013] Hashimoto, M.: Vagrant: up and running, 1. ed., O'Reilly, Beijing, Köln, 2013

[IBM 24] IBM Technology: What is Infrastructure as Code?, 2024, https://www.youtube.com/watch?v=zWw2wuiKd5o, Abgerufen am: 2024.08.05

[ION 23] IONOS: Infrastructure as Code (IaC), 2023, https://www.ionos.de/digitalguide/server/knowhow/infrastructure-as-code/, Abgerufen am: 2024.08.02

[ORA 24] Oracle: Was ist eine virtuelle Maschine?, 2024, https://www.oracle.com/de/cloud/compute/virtual-machines/what-is-virtual-machine/, Abgerufen am: 2024.08.01

[RED 24a] RedHat: Was sind virtuelle Maschinen (VM) und wie funktionieren sie?, 2024, https://www.redhat.com/de/topics/virtualization/what-is-a-virtual-machine, Abgerufen am: 2024.07.31

[RED 24b] RedHat: Was ist IaC (Infrastructure as Code)?, 2024, https://www.redhat.com/de/topics/automation/what-is-infrastructure-as-code-iac, Abgerufen am: 2024.08.05

[SAP 24] SAP: SapMachine, 2024, https://sap.github.io/SapMachine/, Abgerufen am: 2024.09.12