ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Özel, Abdullah; Pautz, Tobias; Schmidt, Nikolaus

Article — Published Version

Infrastructure as Code als Maßnahme zur Cloud Automatisierung – Hilfestellung zur Auswahl des richtigen Werkzeugs

HMD Praxis der Wirtschaftsinformatik

Provided in Cooperation with:
Springer Nature

Suggested Citation: Özel, Abdullah; Pautz, Tobias; Schmidt, Nikolaus (2020) : Infrastructure as Code als Maßnahme zur Cloud Automatisierung – Hilfestellung zur Auswahl des richtigen Werkzeugs, HMD Praxis der Wirtschaftsinformatik, ISSN 2198-2775, Springer Fachmedien Wiesbaden, Wiesbaden, Vol. 57, Iss. 5, pp. 936-948, https://doi.org/10.1365/s40702-020-00657-0

This Version is available at:
https://hdl.handle.net/10419/288617

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

https://creativecommons.org/licenses/by/4.0/

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

HMD (2020) 57:936–948
https://doi.org/10.1365/s40702-020-00657-0

SCHWERPUNKT

Infrastructure as Code als Maßnahme zur Cloud Automatisierung – Hilfestellung zur Auswahl des richtigen Werkzeugs

Abdullah Özel · Tobias Pautz · Nikolaus Schmidt

Eingegangen: 22. August 2020 / Angenommen: 29. August 2020 / Online publiziert: 23. Oktober 2020
© Der/die Autor(en) 2020

Zusammenfassung Es stehen mittlerweile mehrere IaC-Tools zur Verfügung, um die Bereitstellung und Konfiguration der Cloud-Infrastruktur zu automatisieren. Mit der Fülle an Auswahlmöglichkeiten gestaltet sich der Auswahlprozess für das geeignete IaC-Tool zunehmend schwieriger. Zusätzlich mangelt es an einer Entscheidungshilfe für den Auswahlprozess des geeigneten IaC-Tools. Wir glauben, dass eine Entscheidungshilfe für den Auswahlprozess des IaC-Tools dazu beitragen kann, bei der Cloud Migration von Anwendungen einen besseren Service zu bieten. Daher zielt diese Forschungsarbeit darauf ab, den Auswahlprozess zu unterstützen, indem eine Hilfestellung bei der Auswahl des richtigen IaC-Tools gegeben wird.

Schlüsselwörter Cloud Computing · DevOps · Infrastruktur · Provisionierung · Konfigurationsmanagement

A. Özel (✉)
Universität zu Köln, Köln, Deutschland
E-Mail: aoezel@outlook.com

Infrastructure as Code—Developing a Framework to Choose the Right Tool

Abstract Several IaC tools are now available to automate the deployment and configuration of the cloud infrastructure. With so many choices available, the process of selecting the right IaC tool is becoming increasingly difficult. In addition, there is a lack of decision support for the selection process of the appropriate IaC tool. We believe that a decision support for the selection process of the IaC tool can help to provide a better service in the cloud migration of applications. Therefore, this research aims to support the selection process by providing guidance in choosing the right IaC tool.

Keywords Cloud computing · DevOps · Infrastructure · Provisioning · Configuration management

Abkürzungen

API	Application Programming Interfaces
CC	Cloud Computing
CSP	Cloud Service Provider
Devs	Softwareentwickler
IaC	Infrastructure-as-Code
IT	Informationstechnologie
KM	Konfigurationsmanagement
KMT	Konfigurationsmanagement-Tool
MVU	Multinationales Versicherungsunternehmen
Ops	IT-Betreiber
OT	Orchestrierungs-Tool

1 Einführung

In den letzten Jahren gab es wohl kein größeres Schlagwort als „Cloud Computing“ (CC). CC verfügt über das Potential einen großen Teil der IT-Branche zu verändern, den Schwerpunkt auf serviceorientierte Ansätze zu verlagern und die IT-Konzeption und den IT-Einkauf neu zu gestalten (Armbrust et al. 2009). Die Auslagerung der IT-Infrastruktur zu Cloud Service Providern (CSP) wie Amazon, Google oder Microsoft erfordert den Einsatz agiler Techniken, da der IT-Markt eine schnelle Produkteinführungzeit anstrebt (Artac et al. 2017). IT-Betreiber stehen vor allem vor der Herausforderung, die für die Bereitstellung und Konfiguration der Cloud-Infrastruktur benötigte Zeit zu verkürzen, damit Software-Entwickler ihre Anwendungen so schnell wie möglich bereitstellen können (Scheuner et al. 2014). Dieses Spannungsverhältnis zwischen IT-Betreibern (Ops) und Softwareentwicklern (Devs) ist allgemein anerkannt. Während Ops darauf abzielen, die Cloud-Infrastruktur stabil zu halten, sind Devs daran interessiert, kontinuierlich neue Anwendungen oder sogar neue Releases für bereits eingesetzte Anwendungen bereitzustellen (Hummer et al. 2013). Es ist nicht ungewöhnlich, dass Ops zögerlich sind, Änderungen an der Cloud-In-

frastruktur vorzunehmen und den Softwarecode langsamer verarbeiten als von Devs gewünscht (Hummer et al. 2013). Infolgedessen haben Unternehmen DevOps-Praktiken eingeführt, um dieser Tendenz entgegenzuwirken (Hüttermann 2012). DevOps zielt darauf ab organisatorische Distanzen zu verringern und die kontinuierliche Zusammenarbeit zwischen Ops und Devs zu fördern (Artac et al. 2017). Ein Grundstein dafür stellt Infrastructure-as-Code (IaC) dar (Morris 2016). IaC übernimmt die Bereitstellung und Konfiguration der gesamten Cloud-Infrastruktur im Quellcode, so dass manuelle Tätigkeiten kaum noch notwendig sind (Scheuner et al. 2014). Dieser Ansatz stärkt die Verwendung konsistenter und wiederholbarer Routinen (Sandobalin et al. 2017), wodurch die Entwicklung einer Automatisierungslogik für die Bereitstellung und Konfiguration der Cloud-Infrastruktur erleichtert wird (Hummer et al. 2013). Die Bereitstellung und Konfiguration der Cloud-Infrastruktur ist die Grundlage einer laufenden Cloud-Umgebung (Masek et al. 2018). Moderne IaC-Tools bieten Entwicklern mehrere Möglichkeiten, die Bereitstellung und Konfiguration der Cloud-Infrastrukturen zu automatisieren (Hummer et al. 2013). Terraform[1], CloudFormation[2], Chef[3], Puppet[4] oder Ansible[5] sind nur einige der vielen Tools, die helfen IaC in den Entwicklungsprozess zu integrieren. Obwohl IaC den Entwicklern den direkten Zugriff auf die CSP-Plattform erspart (mit Ausnahme von CloudFormation), benötigt es die manuelle Erstellung des Bereitstellungs- und Konfigurationscodes, um die Cloud-Infrastruktur zu verwalten (Bhattacharjee et al. 2018). Da jedes der IaC-Tools ein spezifisches Design hinsichtlich Syntax, Semantik und Formatierungsregeln hat, müssen Entwickler einen Code schreiben, der die Anforderungen des ausgewählten Tools erfüllt (Bhattacharjee et al. 2018). Je nach spezifischem Anwendungsfall sind einige von ihnen möglicherweise besser geeignet als andere. Daraus ergibt sich das folgende Problem in der Praxis:

> Es stehen mehrere IaC-Tools zur Verfügung, um die Bereitstellung und Konfiguration der Cloud-Infrastruktur zu automatisieren und mit der Fülle an Auswahlmöglichkeiten wird der Auswahlprozess des geeigneten IaC-Tools schwierig.

Daraus lässt sich folgende Frage ableiten, welche in diesem Artikel thematisiert wird:

> Wie ist es möglich, den Auswahlprozess für das geeignete IaC-Tool zu unterstützen?

[1] cf. https://www.terraform.io/.

[2] cf. https://aws.amazon.com/cloudformation/.

[3] cf. https://www.chef.io/.

[4] cf. https://puppet.com/.

[5] cf. https://www.ansible.com/.

2 Theoretischer Hintergrund

Im Zuge der Weiterentwicklung des CC hat sich der IaC-Ansatz etabliert und die Management-Performance von Cloud-Infrastrukturen hat sich verbessert (Alt et al. 2017). Das Ziel von IaC ist es, den IT-Betrieb zu vereinfachen, indem der Zeit- und Arbeitsaufwand für die Bereitstellung, Konfiguration, Aktualisierung und Wartung der Dienste verringert wird (Morris 2016). Aufkommende Probleme sollen schnell erkannt und behoben werden und alle Systeme sollen konsistent und auf dem neuesten Stand gehalten werden (Morris 2016). Der Einsatz von IaC soll die Hürden für Änderungen an der Infrastruktur senken. Das IT-Personal sollte dann weniger Zeit für Routineaufgaben und mehr Zeit für die Erfüllung der sich ständig ändernden Anforderungen der heutigen Geschäftswelt aufwenden können (Morris 2016). IaC folgt den Grundprinzipien der Softwareentwicklung und zielt auf Automatisierung für die Bereitstellung und Konfiguration von virtuellen und physischen Infrastrukturkomponenten ab (Hummer et al. 2013; Spinellis 2012). Es stellt einheitliche, reproduzierbare Muster für die Bereitstellung und Modifikation von Cloud-Infrastrukturen und deren Konfiguration zur Verfügung (Morris 2016). IaC ermöglicht es Unternehmen, Tools zur Softwareentwicklung wie Versionskontrollsysteme, automatisierte Testbibliotheken und verschiedene andere Tools zur Verwaltung ihrer Cloud-Infrastruktur einzusetzen (Morris 2016). Die Bereitstellung und Konfiguration der Cloud-Infrastruktur kann durch die Spezifikation des Programmcodes innerhalb der IaC-Tools festgelegt werden (Spinellis 2012). Zu den Bereitstellungs- und Konfigurationsspezifikationen gehören die erforderlichen Bibliotheken oder Server, die Menge des RAM oder die CPU-Geschwindigkeit für eine virtuelle oder physische Infrastrukturkomponente (Jiang und Adams 2015). Unternehmen verwenden IaC-Tools, um neue Cloud-Infrastrukturen einzurichten, bestehende zu erweitern oder solche zu reparieren, deren Einstellungen nicht mehr gültig sind (Spinellis 2012). In allen Fällen liefert die Spezifikation eine präzise, ausführbare Dokumentation der Einstellungen der Cloud-Infrastruktur (Spinellis 2012).

Mit IaC ist es möglich, jedes beliebige Element der Cloud-Infrastruktur schnell zu reproduzieren, da alles innerhalb eines Skripts erfasst ist (Morris 2016). Durch diese Ermöglichung können die Risiken und Ängste, die sonst mit Änderungen verbunden sind, eliminiert werden (Morris 2016). IaC erfordert, dass die Systeme unter der Annahme entworfen werden, dass sich die Cloud-Infrastruktur im Laufe der Zeit ändert und sich an die ändernden Bedürfnisse anpasst (Morris 2016). Selbst wenn Server wegfallen oder skaliert werden, sollen Anwendungen weiterhin laufen (Morris 2016). Die nützliche Eigenschaft von IaC-Tools, Cloud-Infrastruktur-Ressourcen einfach zu erstellen, zu löschen, zu ersetzen, zu skalieren und zu verschieben, erleichtert die Verbesserung und Anpassung der laufenden Cloud-Infrastruktur (Morris 2016). Auf diese Weise werden die Services ausfallsicherer (Morris 2016). Ein weiteres Prinzip des IaC ist die Konsistenz der Cloud-Systeme (Morris 2016). Die Konfiguration von Cloud-Elementen, die einen ähnlichen Dienst bereitstellen, sollte identisch sein (Morris 2016). Da es keine Inkonsistenzen gibt, können Unternehmen ihrer Automatisierung vertrauen (Morris 2016). Zudem basiert IaC auf dem Prinzip der Reproduzierbarkeit, was bedeutet, dass alle Maßnahmen, die an der Cloud-Infrastruktur durchgeführt werden, wiederholbar sind (Morris 2016).

3 Research Design

Die Automatisierung der Bereitstellung und Konfiguration von Cloud-Infrastrukturen ist ein relativ neues Forschungsgebiet, und es gibt keine umfassenden Untersuchungen zur Auswahl des geeigneten IaC-Tools. Daher wurde eine Fallstudie durchgeführt, die dem Leitfaden von (Eisenhardt 1989) folgt, der einen Fahrplan für die Entwicklung von Theorien aus der Fallstudienforschung vorschlägt. Abb. 1 fasst den Forschungsansatz zusammen.

In dieser Studie wird ein multinationales Versicherungsunternehmen (MVU) untersucht welches mit über 9000 Mitarbeitern zu den führenden Versicherern in Deutschland zählt. Die Studie folgt einem qualitativen Ansatz und die Daten wurden nach (Eisenhardt 1989) durch eine Kombination aus Beobachtungen, Dokumentation und Interviews erhoben. Diese Daten stammen aus der Abteilung „IT-Produkte und -Dienstleistungen“ und wurden in einem Zeitraum von sechs Monaten erhoben. Der Forscher nahm am täglichen Leben des MVU teil und beobachtete bestimmte Verhaltensweisen, darunter Gespräche mit Mitarbeitern, Besprechungen und Aktivitäten. Die Teilnahme des Forschers am täglichen Leben des Unternehmens erleichterte auch die Sammlung dokumentarischer Informationen, wie z. B. Präsentationen im Zusammenhang mit der CC-Toollandschaft, Prozessbeschreibungen, Tagesordnungen und Ankündigungen.

Darüber hinaus wurden Interviews mit Experten aus dem CC-Bereich geführt, wie sie in der Fallstudienforschung (Yin 2014) üblich sind. Ziel der Interviews in dieser Untersuchung war es, ein tieferes Verständnis des aktuellen Prozesses der Bereitstellung von Cloud-Infrastrukturen und der Konfigurationsautomatisierung im MVU zu gewinnen. Dabei ging es darum zu verstehen, welche Aspekte der verwendeten IaC-Tools die Organisation dazu veranlasst haben, diese zu verwenden. Darüber hinaus werden Aspekte identifiziert, die aus Sicht der Befragten zu Unzufriedenheit führen, und es werden, wenn möglich, Vorschläge zur Verbesserung des aktuellen Prozesses und der verwendeten IaC-Tools ermittelt. Zur Erreichung dieser Ziele wurden sechs halbstrukturierte Interviews mit Mitarbeitenden durchgeführt, die entweder direkt mit den IaC-Instrumenten oder im CC-Umfeld arbeiten und daher über Expertenwissen verfügen. Die Interviewlänge betrug jeweils zwischen 30 und 90 min. Die Interviews wurden im Zeitraum von Januar bis Februar 2020 durchgeführt. Tab. 1 enthält allgemeine Informationen zu den Interviews und den Interviewpartnern.

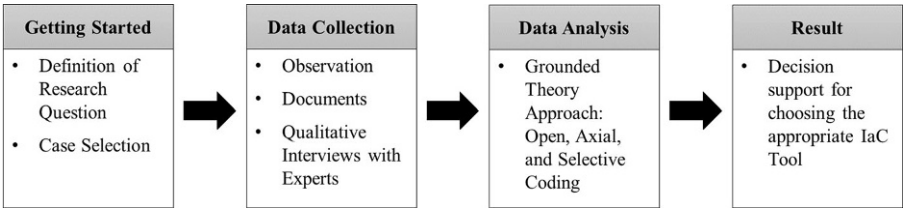

Abb. 1 Research Design

Tab. 1 Experteninterview Übersicht

ID	Jobtitel & Level	Land	Datum	Interviewdauer
I1	Cloud Engineer & Technical Architect	Deutschland	17.01.2020	36:26 min
I2	Consultant & Solutions Architect	Deutschland	23.01.2020	1:28:42 h
I3	Cloud Engineer	Deutschland	27.01.2020	47:27 min
I4	Cloud Engineer	Deutschland	27.01.2020	28:18 min
I5	Product Owner	Deutschland	04.02.2020	28:07 min
I6	Consultant & Solutions Architect	Frankreich	07.02.2020	38:36 min

4 Framework zur Auswahl des Werkzeugs

4.1 State-of-the-Art IaC Tools

Zur Automatisierung der Bereitstellung von Cloud-Infrastrukturen stehen mehrere hochmoderne IaC-Tools mit unterschiedlichen Merkmalen und Funktionen zur Verfügung. In den Interviews erwähnten die Experten IaC-Werkzeuge/IaC-Tools wie CloudFormation, Terraform, Chef, Puppet und Ansible. Natürlich gibt es noch einige weitere Tools, aber um den Rahmen nicht zu sprengen, beschränkt sich die Studie auf diese fünf.

CloudFormation ist ein Closed-Source-Orchestrierungstool, das von Amazon Web Services (AWS) bereitgestellt wird. CloudFormation ermöglicht die Kodierung der Cloud-Infrastruktur, um die Bereitstellung und Verwaltung von AWS-Ressourcen zu automatisieren (Chan 2018). Außerdem ist CloudFormation nur auf AWS anwendbar (Chan 2018). Terraform ist ein Open-Source IaC-Tool, das in der Programmiersprache Go geschrieben ist und von der Firma HashiCorp bereitgestellt wird (Brikman 2019). Terraform ermöglicht die Automatisierung der Bereitstellung von Cloud-basierten-Infrastrukturen und Onsite-Infrastrukturen in einer deklarativen Sprache (IBM Cloud Education 2019). Es kann Diagramme der verwendeten Ressourcen erstellen und automatisiert Änderungen an diesen Ressourcen mit geringem Bedarf an menschlicher Interaktion (Chan 2018). Chef ist ein Open-Source-Tool und wurde von Opscode 2008 entwickelt (Meyer et al. 2013). Bei Chef wird ein prozeduraler Ansatz für das KM angewendet und gehört zur Gruppe der cloud-agnostischen Tools (Chan 2018). Puppet, das sich seit 2005 in der Entwicklung befindet, ist ein klassisches Konfigurationsmanagement-Tool (KMT) und unterstützt Ingenieure bei der kontinuierlichen Bereitstellung von Software (Chan 2018; Meyer et al. 2013). Puppet arbeitet sowohl mit cloud-basierten Implementierungen als auch mit physischen Maschinen und richtet sich in erster Linie an Systemadministratoren (vgl. I5, l. 142–43), während z. B. Chef in erster Linie für Entwickler gedacht ist (Meyer et al. 2013). Puppet ist als Server-Client- oder serverlose Umgebung erhältlich (Önnberg 2012). Es folgt einem deklarativen Ansatz und gehört ebenfalls zur Gruppe der cloud-agnostischen Tools (Chan 2018). Ansible soll Unternehmen bei der Automatisierung des KM und der Anwendungsbereitstellung unterstützen und wurde von RedHat, dem Anbieter von Open-Source-Technologie für Unternehmen, entwickelt (Chan 2018; IBM Cloud Education 2019). Ansible bildet die Cloud-Infrastruktur ab, indem es definiert, wie Komponenten und Systeme miteinander interagieren, anstatt

Systeme unabhängig voneinander zu verwalten (Chan 2018). Außerdem ist Ansible die erste Wahl, wenn es darum geht, die Bereitstellung von Docker-Containern und Kubernetes-Implementierungen zu automatisieren (IBM Cloud Education 2019).

4.2 Infrastructure-as-Code Tool Selektions Kriterien

Je nach Anwendungsfall kann ein IaC-Tool besser geeignet sein als ein anderes, um unternehmensinterne Anforderungen zu erfüllen (Masek et al. 2018; Strutt 2018). In diesem Sinne erfordert die Entscheidung über die bestmögliche Tool-Auswahl ein tiefes Verständnis der vielen Aufgaben, die mit dem Einsatz von Cloud-Infrastrukturen verbunden sind (Strutt 2018). Die Experteninterviews ermöglichten die Ableitung mehrerer Auswahlkriterien. Anschließend wurden die identifizierten IaC-Tool-Auswahlkriterien mit der Literatur verglichen und bis zu einem gewissen Grad mit (Brikman 2019) ergänzt. Tab. 2 zeigt einen vergleichenden Überblick über die IaC-Instrumente und ihre spezifischen Kriterien.

4.2.1 Konfigurationsmanagement vs. Orchestrierung

Die IaC-Tools lassen sich in die Bereiche Orchestrierung und KM unterteilen und gehören in den Bereich der Automatisierung von Cloud-Infrastrukturen (Chan 2018). Die OTs stellen die Infrastrukturen auf einer höheren Ebene als beim KM bereit (Strutt 2018). Der Schwerpunkt liegt hier vorzugsweise auf der Koordination von Konfigurationen in komplexen Umgebungen (Strutt 2018). Die Konfiguration der einzelnen Server wird den KMTs überlassen (Strutt 2018). Diese Tools helfen bei der Konfiguration der Software und Systeme auf der bereits bereitgestellten Infrastruktur (Chan 2018). Dazu gehört die Installation von Paketen, das Starten von Diensten oder die Installation von Skripten oder Konfigurationsdateien auf den Instanzen (Strutt 2018). Mit den KMTs ist es für die Instanzen möglich geworden, ihre Arbeit zu tun, ohne dass der Benutzer die genauen Befehle manuell oder mit Ad-hoc-Skripten angeben muss (Strutt 2018). Zusammenfassend lässt sich sagen, dass KMTs auf der Konfigurationsebene gut funktionieren, aber nicht ideal sind, um

Tab. 2 Übersicht und Vergleich von IaC-Tools

	Cloud-Formation	Terraform	Chef	Puppet	Ansible
Typ	Orchestrierung	Orchestrierung	Konfig. Management	Konfig. Management	Konfig. Management
Infrastruktur	Unwandelbar	Unwandelbar	Wandelbar	Wandelbar	Wandelbar
Sprache	Deklarativ	Deklarativ	Prozedural	Deklarativ	Prozedural
Cloud	Cloud-Nativ	Cloud-Agnostisch	Cloud-Agnostisch	Cloud-Agnostisch	Cloud-Agnostisch
Source	Closed/ Enterprise	Open/ Enterprise	Open/ Enterprise	Open/ Enterprise	Open/ Enterprise
Master Server	Masterlos	Masterlos	Master	Master	Masterlos
Client Agent	Agentlos	Agentlos	Agent	Agent	Agentlos

komplexe Aufgaben zu koordinieren (Strutt 2018). Chef, Puppet und Ansible sind Tools, die zur Kategorie der KMTs gehören, CloudFormation und Terraform sind OTs (vgl. I2, l. 577; I6, l. 185–186). Es gibt jedoch eine Überschneidung der Rollen und Fähigkeiten beider Tools. Einige OTs können bis zu einem gewissen Grad mit dem KM umgehen und einige KMTs können umgekehrt bestimmte Orchestrierungsaufgaben übernehmen (Chan 2018; Strutt 2018). Bei der Auswahl des richtigen IaC-Tools ist zudem entscheidend, ob Server-Templating-Programme wie Docker oder Packer im Einsatz sind (Brikman 2019). Wenn dies der Fall ist, kümmern sich diese Programme um die Anforderungen des KM (Brikman 2019). Es muss nur noch ein OT vorhanden sein, mit dem die Infrastruktur gestartet wird (Brikman 2019). Falls keine Server-Templating-Programme im Einsatz sind, ist es eine gute Entscheidung ein KMT gemeinsam mit einem OT einzusetzen (Brikman 2019). Das OT startet dann den Server und das KMT konfiguriert ihn (Brikman 2019).

4.2.2 Wandelbare Infrastruktur vs. Unwandelbare Infrastruktur

Innerhalb des Paradigmas der wandelbaren Infrastruktur startet ein Server normalerweise mit einem leeren Image (vgl. I5, l. 176–177). Ein KMT läuft dann über diesen Server und installiert die gewünschten Konfigurationen (vgl. I5, L. 177–178). In diesem Fall wird die Cloud-Infrastruktur nur einmal mit einem OT eingerichtet (vgl. I4, L. 232–233). Falls sich im Laufe der Zeit Änderungen an den Konfigurationen ergeben, werden diese mit Hilfe der KMTs vorgenommen (vgl. I4, L. 100–101). Da immer mehr Aktualisierungen stattfinden, kann ein Phänomen auftreten, das als Konfigurationsdrift bezeichnet wird (vgl. I2, l. 329–340). Konfigurationsdrift tritt auf, wenn sich jeder Server aufgrund seiner einzigartigen Änderungshistorie von allen anderen Servern unterscheidet, was zu subtilen Konfigurationsfehlern führt, die schwer zu erkennen und fast unmöglich zu reproduzieren sind (Brikman 2019). KMTs wie Chef, Puppet und Ansible eignen sich daher besser für das Paradigma der wandelbaren Infrastruktur, das bis zu einem gewissen Grad gängige Praxis ist (vgl. I5, l. 159–162). Im Gegensatz dazu setzt das unveränderliche Infrastruktur-Paradigma die vorgefertigten Images ein, die von Docker oder Packer mit einem OT wie Terraform oder CloudFormation erstellt werden (Brikman 2019). Alle Konfigurationen sind bereits auf dem Image installiert, wenn der Bereitstellungsprozess stattfindet, und es besteht keine Notwendigkeit für ein KMT (vgl. I5, l. 189–190). Wenn Änderungen auftreten, erstellt Docker oder Packer ein neues Image (Brikman 2019). Dieses Image wird dann auf einer neuen Cloud-Infrastruktur bereitgestellt, und sobald es ordnungsgemäß funktioniert, werden die alten Instanzen heruntergefahren (Brikman 2019). Dieser Ansatz verringert das Auftreten von Konfigurationsdrifts und vereinfacht die Erkennung, welche Version der Software auf der Infrastruktur läuft (Brikman 2019).

4.2.3 Prozedurale Sprache vs. Deklarative Sprache

Chef und Ansible entsprechen dem programmatischen Stil der prozeduralen Sprache (Chan 2018). Dieser Ansatz konzentriert sich eher auf das „Wie“ als auf das „Was“. Für die prozedurale Sprache wird ein Code geschrieben, der die genauen Schritte

zum Erreichen des gewünschten Endzustandes definiert (Chan 2018). Im Gegenzug entsprechen Terraform, CloudFormation und Puppet der deklarativen Sprache, die sich auf das „Was“ konzentriert. Hierbei spezifiziert der geschriebene Code den gewünschten Endzustand der Cloud-Infrastruktur, und das IaC-Tool sorgt selbst dafür, dass dieser erreicht wird (vgl. I1, 186–188; I3, l. 339–340). Beide Sprachansätze haben ihre Vor- und Nachteile. So ist im Falle des prozeduralen Codes der Zustand der Cloud-Infrastruktur nicht vollständig sichtbar, da die Ausführung der Sequenzen nicht bekannt ist (Brikman 2019). Darüber hinaus ist die Wiederverwendbarkeit des prozeduralen Codes begrenzt, da der aktuelle Zustand der Cloud-Infrastruktur aufgrund von Änderungen des Codes im Laufe der Zeit nur schwer nachvollziehbar ist (Brikman 2019). Im deklarativen Ansatz repräsentiert der Code immer den neuesten Stand der Cloud-Infrastruktur, wodurch es einfacher wird, wiederverwendbaren Code zu erstellen (Brikman 2019). Darüber hinaus ist der deklarative Ansatz für Nicht-Entwickler leichter zu verstehen und zu bedienen (vgl. I5, l. 115, l. 126–127). Allerdings erweist sich der deklarative Ansatz bei der Verwaltung großer komplexer Cloud-Infrastrukturen als nachteilig (vgl. I5, l. 127–129). Der Grund liegt in der begrenzten Aussagekraft des deklarativen Stils ohne programmatischen Sprachansatz (Brikman 2019) (Abb. 2).

4.2.4 Cloud-Nativ vs. Cloud-Agnostisch

Terraform, Chef, Puppet und Ansible sind allesamt cloud-agnostische Tools, die sich in viele CSPs wie AWS, Microsoft Azure oder Google Cloud Platform (Chan 2018) integrieren lassen. Im Gegenzug ist CloudFormation ein cloud-natives Tool und lässt sich nur vollständig in AWS integrieren (vgl. I2, l. 856–859). Der Vorteil eines cloud-agnostischen Tools wird deutlich, wenn mit mehreren CSPs gleichzeitig gearbeitet wird (Chan 2018). In diesem Fall ist es möglich, mit nur einem einzigen Tool mehrere Plattformen gleichzeitig anzusprechen. Dies bedeutet jedoch nicht, dass der geschriebene Code zwischen den Plattformen übertragbar ist. Jede Plattform hat spezifische Namen für die Ressourcen, weshalb der Code modifiziert werden muss (vgl. I3, l. 407–409). Der Vorteil eines cloud-nativen Tools wie CloudFormation besteht darin, dass es eng in die Cloud-Plattform integriert ist und in der Regel als kostenloser eingebetteter Dienst angeboten wird, wodurch Kosteneinsparungen möglich sind (Chan 2018).

Declarative: CloudFormation	Procedural: AWS CLI
S3Bucket: Type: AWS::S3::Bucket Properties: BucketName: test	$ aws s3api create-bucket --bucket “test"

Abb. 2 Beispiel von deklarativer und prozeduraler Sprache

4.2.5 Open Source vs. Enterprise

Terraform, Chef, Puppet und Ansible sind sowohl als Open-Source- als auch als Enterprise-Editionen erhältlich. CloudFormation hingegen ist Teil der AWS-Plattform und Closed-Source (vgl. I2, l. 856–859). Bei der Wahl des geeigneten IaC-Tools kann es sicherlich von Bedeutung sein, ob das eingesetzte Tool über einen Support bei der Behebung von Problemen verfügt (vgl. I1, l. 391-394; I3, l. 271–273). Dies erfordert eine Enterprise-Version, die mit Kosten verbunden ist. Sollen die Kosten jedoch so niedrig wie möglich gehalten werden, könnte ein Open-Source-Tool die bessere Option sein (vgl. I2, l. 862–864).

4.2.6 Master vs. Masterlos

Standardmäßig sind CloudFormation, Terraform und Ansible alle masterlos (Brikman 2019). Im Gegensatz benötigen Chef und Puppet einen Master-Server, um den Zustand der Cloud-Infrastruktur zu speichern und Aktualisierungen zu verteilen (Brikman 2019). Der Betrieb eines Master-Servers hat mehrere Vorteile. Zum einen ist der Master-Server ein einziger zentraler Ort, an dem der Status der Cloud-Infrastruktur sowohl einsehbar als auch verwaltbar ist (Brikman 2019). Auf der anderen Seite ist es auch möglich, dass mehrere Master-Server permanent im Hintergrund laufen und die Konfiguration erzwingen, so dass sogar manuelle Änderungen rückgängig gemacht werden können, um ein Abdriften der Konfiguration zu verhindern (Brikman 2019). Dennoch hat der Einsatz eines Master-Servers auch einige gravierende Nachteile. Beispielsweise müssen für den Betrieb des Masters zusätzliche Server für hohe Verfügbarkeit und Skalierbarkeit bereitgestellt werden (Brikman 2019). Darüber hinaus muss der Master kontinuierlich gewartet werden und es muss ein sicherer Kommunikationspfad für die Clients mit dem Master-Server bereitgestellt werden (Brikman 2019). Es gibt jedoch auch ein gewisses Maß an Unterstützung für masterlose Modi innerhalb von Chef and Puppet (Brikman 2019).

4.2.7 Agent vs. Agentlos

Chef and Puppet werden auch verlangen, dass auf jedem Server, den sie konfigurieren sollen, eine Agentensoftware installiert wird (Brikman 2019). Normalerweise arbeitet der Agent im Hintergrund auf jedem Server, der für die Installation der neuesten Updates für das Konfigurationsmanagement vorgesehen ist (Brikman 2019). Auch diese Agentensoftware muss ständig aktualisiert und, falls verfügbar, permanent mit dem Master-Server synchronisiert werden (Brikman 2019). Für die Kommunikation mit dem Server ist die Einrichtung eines sicheren Authentifizierungskanals erforderlich (Brikman 2019). Andernfalls könnten Hacker Unheil anrichten (Brikman 2019). Es gibt jedoch wieder ein gewisses Maß an Unterstützung für agentenlose Modi innerhalb von Chef and Puppet (Brikman 2019).

Tab. 3 Vergleich von IaC-Tool Communities. (Brikman 2019)

	Contributors	Sterne	Commits (30 days)	Bugs (30 days)	Libraries	Stack-Overflow[a]
CloudFormation	–	–	–	–	377[b]	3315
Terraform	1261	16.837	173	204	1462[c]	2730
Chef	562	5794	435	86	3832[d]	5982
Puppet	515	5299	94	314[e]	6110[f]	3585
Ansible	4386	37.161	506	523	20.677[g]	20.677

[a]Cf. https://stackoverflow.com/
[b]Anzahl der Templates im awslabs GitHub-Konto
[c]Anzahl der Module in der Terraform Registry
[d]Anzahl der cookbooks im Chef Supermarket
[e]Basierend auf dem JIRA-Konto von Puppet Labs
[f]Anzahl von Modulen im Puppet Forge
[g]Anzahl von wiederverwendbaren Rollen in Ansible Galaxy

4.2.8 Große Community vs. Kleine Community

Bei der Entscheidung über ein Tool ist es auch wichtig, die Nutzergemeinschaft zu berücksichtigen (Brikman 2019). In vielen Fällen kann das Ökosystem um das Tool herum einen weitaus größeren Einfluss auf die Benutzerfreundlichkeit haben als die mit dem Tool verbundene Qualität selbst (Brikman 2019). Die Gemeinschaft bestimmt die Attraktivität des Tools, was sich zum Beispiel an der Anzahl der vorhandenen Plug-ins, Integrationen und Erweiterungen zeigt (Brikman 2019). Tab. 3 zeigt einen Vergleich der untersuchten IaC-Tools.

5 Diskussion

Da die Automatisierung der Bereitstellung und Konfiguration von Cloud-Infrastrukturen ein relativ neues Forschungsgebiet ist, haben sich nur wenige Artikel mit dieser Technologie befasst. Insbesondere der Mangel an Unterstützung bei der Auswahl von IaC-Tools zeigte, dass weitere Forschung auf diesem Gebiet wünschenswert ist. Daher konzentrierte sich diese Forschung auf diesen speziellen Bereich, erweiterte das Forschungsthema und lieferte wertvolle Erkenntnisse, die es Forschern ermöglicht, neue Forschungslücken zu identifizieren. Die Auswahlkriterien für IaC-Tools und der vergleichende Überblick, können Unternehmen bei der Auswahl des geeigneten Instruments helfen und Fehlentscheidungen verhindern, die sonst zu Ineffizienz oder Unzufriedenheiten führen. Darüber hinaus kann kein IaC-Tool alle Bedürfnisse eins Unternehmens vollständig erfüllen (vgl. I2, l. 460–463). Die meisten Tools haben in einem hochdynamischen Umfeld noch immer ihre Schwächen (vgl. I3, L. 241–233). Je nach Komplexität der Cloud-Infrastruktur ist es erforderlich, dass mehrere Tools gleichzeitig eingesetzt werden oder die fehlende Funktionalität selbst entwickelt wird (vgl. I3, L. 361–363). Diese Untersuchung versucht, den Auswahlprozess durch eine Anleitung zur Auswahl des richtigen Werkzeugs zu unterstützen. Die daraus resultierenden Erkenntnisse sind insbesondere für Unternehmen, die eine Automatisierung der Cloud-Infrastruktur anstreben, von Bedeutung. Darüber hinaus

tragen die Ergebnisse auch zur zukünftigen Forschung in diesem speziellen Bereich bei. Die Generalisierbarkeit dieser Forschung ist begrenzt, da der Forschende die Ergebnisse in einer einzigen Fallstudie erfasst hat, was zur Repräsentation einer einzigen empirischen Tatsache führt. Die Verallgemeinerbarkeit der Ergebnisse eines einzelnen Falles in anderen empirischen Fällen ist möglich, indem zusätzliche Studien durchgeführt und diese Ergebnisse in anderen Fallstudien bestätigt werden (Darke et al. 1998). Daher ist es ratsam, dieses Forschungsthema auch in anderen Branchen zu untersuchen, in denen IaC-Tools zur Automatisierung der Cloud-Infrastruktur eingesetzt werden.

Funding Open Access funding provided by Projekt DEAL.

Open Access Dieser Artikel wird unter der Creative Commons Namensnennung 4.0 International Lizenz veröffentlicht, welche die Nutzung, Vervielfältigung, Bearbeitung, Verbreitung und Wiedergabe in jeglichem Medium und Format erlaubt, sofern Sie den/die ursprünglichen Autor(en) und die Quelle ordnungsgemäß nennen, einen Link zur Creative Commons Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden.

Die in diesem Artikel enthaltenen Bilder und sonstiges Drittmaterial unterliegen ebenfalls der genannten Creative Commons Lizenz, sofern sich aus der Abbildungslegende nichts anderes ergibt. Sofern das betreffende Material nicht unter der genannten Creative Commons Lizenz steht und die betreffende Handlung nicht nach gesetzlichen Vorschriften erlaubt ist, ist für die oben aufgeführten Weiterverwendungen des Materials die Einwilligung des jeweiligen Rechteinhabers einzuholen.

Weitere Details zur Lizenz entnehmen Sie bitte der Lizenzinformation auf http://creativecommons.org/licenses/by/4.0/deed.de.

Literatur

Alt R, Auth G, Kögler C (2017) Innovationsorientiertes IT-Management mit DevOps. In: Alt R, Auth G, Kögler C (Hrsg) essentials. Innovationsorientiertes IT-Management mit DevOps: IT im Zeitalter von Digitalisierung und Software-defined Business. Springer Gabler, Wiesbaden, S 21–32 https://doi.org/10.1007/978-3-658-18704-0_3

Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Stoica I et al (2009) Above the clouds: a Berkeley view of cloud computing. Technical Report No. UCB/EECS-2009-28. University of California, Berkeley

Artac M, Borovssak T, Di Nitto E, Guerriero M, Tamburri DA (2017) DevOps: Introducing Infrastructure-as-Code. In: The Institute of Electrical and Electronics Engineers (Hrsg) 2017 IEEE/ACM 39th International Conference on Software Engineering companion proceedings ICSE-C 2017, Buenos Aires, Argentina, 20–28 May 2017 IEEE, Piscataway, NJ, S 497–498 https://doi.org/10.1109/ICSE-C.2017.162

Bhattacharjee A, Barve Y, Gokhale A, Kuroda T (Hrsg) (2018) (wip) cloudcamp: Automating the deployment and management of cloud services. In 2018 IEEE International Conference on Services Computing (SCC), IEEE, S 237–240

Brikman Y (2019) Terraform: Up & Running: Writing Infrastructure as Code. O'Reilly Media

Chan M (2018) 15 Infrastructure as Code tools you can use to automate your deployments. https://www.thorntech.com/2018/04/15-infrastructure-as-code-tools/. Zugegriffen: 30. Mai 2020

Darke P, Shanks G, Broadbent M (1998) Successfully completing case study research: combining rigour, relevance and pragmatism. Inform Syst J 8(4):273–289. https://doi.org/10.1046/j.1365-2575.1998.00040.x

Eisenhardt KM (1989) Building theories from case study research. AMR 14(4):532–550. https://doi.org/10.5465/amr.1989.4308385

Hummer W, Rosenberg F, Oliveira F, Eilam T (2013) Testing idempotence for infrastructure as code. In: Eyers D, Schwan K (Hrsg) Middleware 2012: ACM/IFIP/USENIX, 14th International Middleware

Conference Beijing, China, December 9–13, 2013. LNCS sublibrary. SL 2, Programming and software engineering, Bd. 8275. Springer, Heidelberg, S 368–388 https://doi.org/10.1007/978-3-642-45065-5_19

Hüttermann M (2012) Devops for developers. The expert's voice in web development. Apress, Berkeley, CA https://doi.org/10.1007/978-1-4302-4570-4

IBM Cloud Education (2019) Infrastructure as Code (IaC). https://www.ibm.com/cloud/learn/infrastructure-as-code. Zugegriffen: 2. Juni 2020

Jiang Y, Adams B (2015) Co-evolution of infrastructure and source code—an empirical study. In: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories (MSR) Florence, Italy, 16–17 May 2015 IEEE, Piscataway, NJ, S 45–55 https://doi.org/10.1109/MSR.2015.12

Masek P, Stusek M, Krejci J, Zeman K, Pokorny J, Kudlacek M (2018) Unleashing full potential of ansible framework: university labs administration. In: Balandin SI (Hrsg) Proceedings of the 22nd Conference of Open Innovations Association FRUCT Jyväskylä, Finland, 15–18 May 2018 IEEE, Piscataway, NJ, S 144–150 https://doi.org/10.23919/FRUCT.2018.8468270

Meyer S, Healy P, Lynn T, Morrison J (2013) Quality assurance for open source software configuration management. In: 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. IEEE, Piscataway, NJ, S 454–461 https://doi.org/10.1109/SYNASC.2013.66

Morris K (2016) Infrastructure as code: managing servers in the cloud, 1. Aufl. O'Reilly, Sebastopol, CA

Önnberg F (2012) Software configuration management: a comparison of Chef, CFengine and Puppet

Sandobalin J, Insfran E, Abrahao S (2017) An infrastructure modelling tool for cloud provisioning. In: Liu XF, Bellur U (Hrsg) IEEE 14th International Conference on Services Computing SCC 2017, Honolulu, Hawaii, USA, 25–30 June 2017 IEEE Computer Society, Los Alamitos, California, S 354–361 https://doi.org/10.1109/SCC.2017.52

Scheuner J, Leitner P, Cito J, Gall H (2014) Cloud work bench—infrastructure-as-code based cloud benchmarking. In: IEEE 6th International Conference on Cloud Computing Technology and Science CloudCom 2014, Singapore, 15–18 December 2014. Bd. 2014. IEEE Computer Society, Los Alamitos, California, Washington, Tokyo, S 246–253 https://doi.org/10.1109/CloudCom.2014.98

Spinellis D (2012) Don't install software by hand. IEEE Softw 29(4):86–87. https://doi.org/10.1109/MS.2012.85

Strutt S (2018) Infrastructure as code: Chef, Ansible, Puppet, or Terraform? https://www.ibm.com/cloud/blog/chef-ansible-puppet-terraform. Zugegriffen: 31. Mai 2020

Yin RK (2014) Case study research design and methods. SAGE, Thousand Oaks, CA