Praxishandbuch Terraform

Infrastructure as Code für DevOps, Administration und Entwicklung

DAS VORWORT

Vorwort

Vor langer Zeit in einem fernen Datacenter hatte eine alte Gruppe mächtiger Wesen, die als »Sysadmins« bekannt waren, die Aufgabe, Infrastruktur manuell zu deployen. Jeder Server, jede Datenbank, jeder Load Balancer und jedes Element der Netzwerkinfrastruktur wurde von Hand geschaffen und verwaltet. Es war ein dunkles und schreckliches Zeitalter: Man fürchtete sich vor Downtime, vor unabsichtlicher Fehlkonfiguration, vor langsamen und anfälligen Deployments und vor dem, was passieren würde, wenn die Sysadmins auf die dunkle Seite der Macht gezogen würden (also Urlaub machten). Die frohe Nachricht ist, dass es dank der DevOps-Bewegung mittlerweile einen besseren Weg gibt, all das zu erledigen: *Terraform*.

Terraform (*https://www.terraform.io/*) ist ein Open-Source-Tool, das von HashiCorp geschaffen wurde. Es ermöglicht Ihnen, Ihre Infrastruktur als Code durch eine einfache, deklarative Sprache zu definieren und diese Infrastruktur auf einer Vielzahl öffentlicher Cloud-Provider (zum Beispiel *Amazon Web Services* [AWS], Microsoft Azure, Google Cloud Platform oder DigitalOcean) und auf privaten und Virtualisierungsplattformen (zum Beispiel OpenStack oder VMware) mit einer Handvoll Befehlen zu deployen und zu managen. Statt beispielsweise manuell auf einer Webseite herumzuklicken oder Dutzende von Befehlen auszuführen, ist dies der gesamte Code, den Sie brauchen, um auf AWS einen Server zu konfigurieren:

```
provider "aws" {
  region = "us-east-2"
}

resource "aws_instance" "example" {
  ami = "ami-0fb653ca2d3203ac1"
  instance_type = "t2.micro"
}
```

Zum Deployen führen Sie nur Folgendes aus:

```
$ terraform init
$ terraform apply
```

Dank seiner Einfachheit und Leistungsfähigkeit ist Terraform zu einem zentralen Teil der DevOps-Welt aufgestiegen. Sie können damit die mühsamen, zerbrechli-

chen und manuellen Elemente des Infrastrukturmanagements durch eine solide und automatisierte Grundlage ersetzen, auf der Sie dann all Ihre anderen DevOps-Praktiken (zum Beispiel automatisiertes Testen, Continuous Integration oder Continuous Delivery) und die zugehörigen Tools (zum Beispiel Docker, Chef oder Puppet) aufsetzen.

Dieses Buch ist der schnellste Weg, um sich mit Terraform bekannt zu machen und es einzusetzen.

Sie werden mit dem einfachsten »Hallo Welt«-Beispiel in Terraform beginnen (tatsächlich haben Sie es gerade schon gesehen) und sich bis zu einem Full Tech Stack vorarbeiten (virtuelle Server, Kubernetes-Cluster, Docker-Container, Load Balancer und Datenbanken), mit dem sehr viel Traffic und ein großes Entwicklungsteam bedient werden können – all das innerhalb weniger Kapitel. Dies ist ein praxisnahes Tutorium, das Ihnen nicht nur die Prinzipien von *DevOps* und *Infrastructure as Code* (IaC) nahebringt, sondern auch Dutzende Codebeispiele enthält, die Sie zu Hause ausprobieren können – sorgen Sie also dafür, Ihren Computer zur Hand zu haben.

Wenn Sie das Buch durchgearbeitet haben, sind Sie dazu bereit, Terraform in der realen Praxis einzusetzen.

Wer dieses Buch lesen sollte

Dieses Buch ist für alle gedacht, die für Code verantwortlich sind, nachdem er geschrieben wurde. Dazu gehören Sysadmins, Operations Engineers, Release Engineers, Site Reliability Engineers, DevOps Engineers, Infrastructure-Entwicklerinnen, Full-Stack-Entwickler, Engineering Manager und CTOs. Egal wie Ihr Titel lautet – wenn Sie dafür zuständig sind, Infrastruktur zu managen, Code zu deployen, Server zu konfigurieren, Cluster zu skalieren, Daten zu sichern, Apps zu überwachen und um drei Uhr nachts auf Alerts zu reagieren, ist dieses Buch etwas für Sie.

Alle diese Aufgaben werden normalerweise zu *Operations* zusammengefasst. In der Vergangenheit war es üblich, Entwicklerinnen und Entwickler zu finden, die wussten, wie man Code schreibt, die aber keine Ahnung von Operations hatten – und umgekehrt suchte man Sysadmins, die Operations verstanden, aber nicht wussten, wie Code zu schreiben ist. Damals konnte man mit dieser Unterteilung zurechtkommen, aber in der modernen Welt, in der Cloud Computing und die DevOps-Bewegung allgegenwärtig werden, muss so gut wie jeder Entwickler Operations-Skills erlernen und sich jede Sysadmin Coding-Skills aneignen.

Dieses Buch geht nicht davon aus, dass Sie schon ein Expert Coder oder Expert Sysadmin sind – eine grundlegende Vertrautheit mit Programmieren, der Befehlszeile und serverbasierter Software (zum Beispiel Websites) sollte ausreichen. Alles andere werden Sie auf dem Weg erfahren, sodass Sie am Ende des Buchs eine solide Grundlage für einen der zentralsten Aspekte von Operations und moderner Entwicklung haben: Infrastruktur als Code zu managen.

Tatsächlich werden Sie nicht nur lernen, wie Sie mit Terraform Infrastruktur als Code managen, sondern auch, wie all das in die gesamte DevOps-Welt passt. Dies sind ein paar der Fragen, die Sie am Ende werden beantworten können:

- Warum sollte man IaC überhaupt nutzen?
- Was sind die Unterschiede zwischen Konfigurationsmanagement, Orchestrierung, Provisionierung und Server Templating?
- Wann sollten Sie Terraform, Chef, Ansible, Puppet, Pulumi, CloudFoundation, Docker, Packer oder Kubernetes einsetzen?
- Wie funktioniert Terraform, und wie nutzen Sie es, um Ihre Infrastruktur zu managen?
- Wie erstellen Sie wiederverwendbare Terraform-Module?
- Wie managen Sie Secrets sicher bei der Arbeit mit Terraform?
- Wie nutzen Sie Terraform mit mehreren Regionen, Accounts und Clouds?
- Wie schreiben Sie Terraform-Code, der zuverlässig genug für den produktiven Einsatz ist?
- Wie testen Sie Ihren Terraform-Code?
- Wie machen Sie Terraform zu einem Teil Ihres automatisierten Deployment-Prozesses?
- Was sind die Best Practices für die Verwendung von Terraform im Team?

Die einzigen Werkzeuge, die Sie brauchen, sind ein Computer (Terraform läuft auf den meisten Betriebssystemen), eine Internetverbindung und den Wunsch, etwas zu lernen.

Warum ich dieses Buch geschrieben habe

Terraform ist ein leistungsfähiges Tool. Es arbeitet mit allen bekannteren Cloud-Providern zusammen. Es besitzt eine saubere, einfache Sprache und eine umfassende Unterstützung für Wiederverwendung, Testen und Versionierung. Es ist Open Source und hat eine freundliche und aktive Community. Aber es gibt einen Bereich, der Luft nach oben hat: Ausgereiftheit.

Terraform erfreut sich sehr großer Beliebtheit, aber es ist immer noch eine recht neue Technologie, und trotz seiner Bekanntheit ist es schwierig, Bücher, Blogposts oder Experten und Expertinnen zu finden, die Ihnen dabei helfen, das Tool zu meistern. Die offizielle Terraform-Dokumentation ist recht gut, wenn es darum geht, Sie in die grundlegende Syntax und die einfacheren Features einzuführen, aber sie enthält nur wenig über idiomatische Patterns, Best Practices, Testen, Wiederverwendbarkeit oder Team-Workflows. Es ist, als würde man versuchen, fließend Französisch zu sprechen, indem man nur die Vokabeln erlernt, aber sich weder um Grammatik noch um Redewendungen kümmert.

Ich habe dieses Buch geschrieben, um Entwicklerinnen und Entwicklern dabei zu helfen, fließend Terraform zu sprechen. Sechs der sieben Jahre, die es Terraform

schon gibt, habe ich es verwendet – meist in meiner Firma Gruntwork (*https://www.gruntwork.io*), in der Terraform eines der zentralen Werkzeuge ist, das wir verwendet haben, um eine Bibliothek mit mehr als 300.000 Zeilen wiederverwendbarem, erprobtem Infrastrukturcode zu schaffen, die von Hunderten von Firmen produktiv eingesetzt wird. Das Schreiben und Betreuen dieses vielen Infrastrukturcodes über all die Jahre und dessen Einsatz in so vielen Unternehmen und bei so vielen Anwendungsfällen hat uns diverse harte Lektionen gelehrt. Mein Ziel ist, diese Lektionen mit Ihnen zu teilen, sodass Sie diesen langatmigen Prozess abkürzen können und in wenigen Tagen vertraut mit Terraform werden.

Natürlich können Sie nicht einfach nur durch Lesen meisterlich werden. Um fließend Französisch sprechen zu können, müssen Sie mit französischen Muttersprachlern sprechen, französische TV-Shows schauen und französischer Musik lauschen. Um fließend Terraform zu sprechen, müssen Sie echten Terraform-Code schreiben, damit echte Software managen und diese auch auf echten Servern deployen. Machen Sie sich daher dafür bereit, viel Code zu lesen, zu schreiben und auszuführen.

Was Sie in diesem Buch finden werden

Dies sind die Themen, die in diesem Buch behandelt werden:

Kapitel 1 »Warum Terraform?«
: Wie DevOps unsere Art und Weise, Software zu betreiben, verändert; ein Überblick über die Tools zu Infrastructure as Code, unter anderem zu Konfigurationsmanagement, Server Templating, Orchestrierung und Provisionierung; die Vorteile von Infrastructure as Code; ein Vergleich von Terraform, Chef, Puppet, Ansible, Pulumi, OpenStack Heat und CloudFoundation; wie sich Tools wie Terraform, Packer, Docker, Ansible und Kubernetes kombinieren lassen.

Kapitel 2 »Einstieg in Terraform«
: Terraform installieren; ein Überblick über die Syntax; ein Überblick über das Befehlszeilentool von Terraform; wie ein einzelner Server deployt wird; wie Sie einen Webserver deployen; wie ein Cluster aus Webservern deployt wird; wie Sie einen Load Balancer deployen; wie Ressourcen aufgeräumt werden, die Sie angelegt haben.

Kapitel 3 »Wie Sie den Terraform-Status managen«
: Was der Terraform-Status ist; wie Sie den Status ablegen, sodass mehrere Teammitglieder darauf zugreifen können; wie Sie Statusdateien sperren, um Race Conditions zu verhindern; wie Statusdateien isoliert werden, um den Schaden durch Fehler zu begrenzen; wie Sie Terraform-Workspaces nutzen; ein Best-Practices-Datei- und -Ordnerlayout für Terraform-Projekte; wie Sie schreibgeschützten Status nutzen.

Kapitel 4 »Wie man wiederverwendbare Infrastruktur mit Terraform-Modulen erzeugt«

Was Module sind; wie Sie ein einfaches Modul erstellen; wie ein Modul durch Eingabe- und Ausgabeparameter konfigurierbar wird; lokale Werte; versionierte Module; Modulfallstricke; Verwenden von Modulen zum Definieren wiederverwendbarer, konfigurierbarer Infrastrukturelemente.

Kapitel 5 »Tipps und Tricks zu Terraform: Schleifen, if-Anweisungen, Deployment und Fallstricke«

Schleifen mit dem Parameter `count`, `for_each`- und `for`-Ausdrücke und die `if`-String-Direktive; eingebaute Funktionen; Zero-Downtime-Deployment; häufige Fallstricke und Probleme in Terraform, unter anderem Grenzen von `count` und `for_each`, Hindernisse beim Zero-Downtime-Deployment, wie vernünftige Pläne fehlschlagen können und wie Sie Terraform-Code sicher refaktorieren.

Kapitel 6 »Secrets mit Terraform managen«

Eine Einleitung in das Secret-Management; Überblick über verschiedene Arten von Secrets, unterschiedliche Wege, Secrets abzulegen und darauf zuzugreifen; Vergleich häufig genutzter Secret-Management-Tools wie HashiCorp Vault, AWS Secrets Manager und Azure Key Vault; Managen von Secrets bei der Arbeit mit Providern, einschließlich der Authentifizierung über Umgebungsvariablen, IAM-Rollen und OIDC; Managen von Secrets bei der Arbeit mit Ressourcen und Datenquellen, einschließlich des Einsatzes von Umgebungsvariablen, verschlüsselten Dateien und zentralen Secret Stores; wie sicher mit Statusdateien und Plandateien umgegangen wird.

Kapitel 7 »Arbeiten mit mehreren Providern«

Ein genauerer Blick darauf, wie Terraform-Provider arbeiten, einschließlich ihrer Installation, des Steuerns der Version und des Einsatzes in Ihrem Code; wie Sie mehrere Kopien des gleichen Providers nutzen, einschließlich des Deployens auf mehrere AWS-Regionen und auf mehrere AWS-Konten sowie des Bauens wiederverwendbarer Module, die mehrere Provider nutzen können; wie mehrere verschiedene Provider gemeinsam genutzt werden, einschließlich eines Beispiels für den Einsatz von Terraform zum Betreiben eines Kubernetes-Clusters (EKS) in AWS und das Deployen dockerisierter Apps im Cluster.

Kapitel 8 »Produktiv nutzbarer Terraform-Code«

Warum DevOps-Projekte immer länger brauchen, als Sie erwartet haben; die Checkliste für produktiv nutzbare Infrastruktur; wie Sie Terraform-Module für die Produktivumgebung bauen; kleine Module; zusammenstellbare Module; testbare Module; releasbare Module; Terraform-Registry; Variablenvalidierung; Versionieren von Terraform, Terraform-Providern, Terraform-Modulen und Terragrunt; Notausstiege in Terraform.

Kapitel 9 »Wie Sie Terraform-Code testen«

Manuelle Tests für Terraform-Code; Sandbox-Umgebungen und Aufräumen; automatisierte Tests für Terraform-Code; Terratest; Unit Tests; Integrationstests; End-to-End-Tests; Dependency Injection; Tests parallel ausführen; Test Stages; Retries; die Test-Pyramide; statische Analysen; Plantests; Servertests.

Kapitel 10 »Wie Sie Terraform im Team verwenden«
: Wie Sie Terraform im Team übernehmen; wie Sie Ihre Vorgesetzten überzeugen; ein Workflow zum Deployen von Anwendungscode; ein Workflow zum Deployen von Infrastrukturcode; Versionsverwaltung; die Goldene Regel von Terraform; Code Reviews; Coding-Richtlinien; Terraform-Stil; CI/CD für Terraform; der Deployment-Prozess.

Sie können das Buch von vorne bis hinten durchlesen oder in die Kapitel springen, die Sie am meisten interessieren. Beachten Sie, dass die Beispiele in jedem Kapitel auf die Beispiele der vorherigen Kapitel verweisen und darauf aufbauen – wenn Sie also herumspringen, nutzen Sie die Open-Source-Codebeispiele (wie in »Open-Source-Codebeispiele« auf Seite 15 beschrieben), um alles zum Laufen zu bringen. Am Ende des Buchs finden Sie in Anhang eine Liste mit empfehlenswerter Lektüre, in der Sie mehr über Terraform, Operations, Infrastructure as Code und DevOps erfahren können.

Was Sie in diesem Buch nicht finden werden

Dieses Buch soll keine allumfassende Referenz für Terraform sein. Ich behandle nicht alle Cloud-Provider und nicht alle Ressourcen, die von einem Cloud-Provider unterstützt werden, und auch nicht alle Terraform-Befehle. Für diese Details verweise ich stattdessen auf die Terraform-Dokumentation (*https://www.terraform.io/docs*).

Die Dokumentation enthält viele nützliche Antworten, aber wenn Terraform, Infrastructure as Code und Operations für Sie neu sind, werden Sie nicht einmal wissen, was Sie fragen sollten. Daher konzentriert sich dieses Buch darauf, was die Dokumentation *nicht* behandelt – vor allem, wie Sie über die einführenden Beispiele hinausgehen und Terraform in einer realen Situation einsetzen. Mein Ziel ist, schnell in Fahrt zu kommen, indem ich zunächst erkläre, warum Sie Terraform nutzen sollten, wie es in Ihren Workflow passt und welche Praktiken und Patterns am besten funktionieren.

Um diese Patterns zu demonstrieren, habe ich eine Reihe von Codebeispielen aufgenommen. Ich habe versucht, es Ihnen so leicht wie möglich zu machen, diese Beispiele zu Hause auszuprobieren, indem Abhängigkeiten zu dritter Seite minimiert wurden. Darum verwenden so gut wie alle Beispiele mit AWS nur einen Cloud-Provider, sodass Sie sich auch nur für einen einzigen Fremdservice anmelden müssen (und AWS bietet eine großzügige Ausprobierphase an, daher sollte Sie das Ausführen des Beispielcodes nicht viel kosten). Das Buch und der Beispielcode gehen deshalb nicht auf die kostenpflichtigen Services Terraform Cloud und Terraform Enterprise von HashiCorp ein. Und ich biete auch alle Codebeispiele als Open Source an.

Open-Source-Codebeispiele

Sie finden alle Codebeispiele aus diesem Buch unter folgender URL:

https://github.com/brikis98/terraform-up-and-running-code

Vielleicht wollen Sie erst einmal dieses Repo auschecken, bevor Sie weiterlesen, damit Sie alle Beispiele auf Ihrem eigenen Computer nachvollziehen können:

```
git clone https://github.com/brikis98/terraform-up-and-running-code.git
```

Die Codebeispiele befinden sich in diesem Repo im Ordner *code*. Auf oberster Ebene sind sie nach dem Tool oder der Sprache organisiert (zum Beispiel Terraform, Packer oder OPA), dann nach Kapiteln. Die einzige Ausnahme ist der Go-Code für die automatisierten Tests in Kapitel 9, die sich im Ordner *terraform* finden, um sich am Ordnerlayout *examples*, *modules* und *test* zu orientieren, das in diesem Kapitel empfohlen wird. In Tabelle 1-1 sind ein paar Beispiele dazu aufgeführt, wo welche Arten von Codebeispielen im Repo zu finden sind.

Tabelle 1-1: Wo Sie die verschiedenen Arten von Codebeispielen im Beispiel-Repo finden

Art von Code	Kapitel	Ordner im Beispiel-Repo
Terraform	Kapitel 2	*code/terraform/02-intro-to-terraform-syntax*
Terraform	Kapitel 5	*code/terraform/05-tips-and-tricks*
Packer	Kapitel 1	*code/packer/01-why-terraform*
OPA	Kapitel 9	*code/opa/09-testing-terraform-code*
Go	Kapitel 9	*code/terraform/09-testing-terraform-code/test*

Es sei darauf hingewiesen, dass die meisten Beispiele zeigen, wie der Code am *Ende* eines Kapitels aussieht. Wollen Sie möglichst viel lernen, schreiben Sie den Code besser von Grund auf selbst und prüfen nur die »offiziellen« Lösungen ganz am Schluss.

Sie beginnen mit dem Schreiben von Code in Kapitel 2, wo Sie lernen, wie Sie Terraform zum Deployen eines einfachen Clusters aus Webservern verwenden. Danach folgen Sie in den anschließenden Kapiteln den Anweisungen zum Entwickeln und Verbessern dieses Webserver-Clusters. Nehmen Sie die Änderungen wie beschrieben vor, versuchen Sie, den gesamten Code selbst zu schreiben, und nutzen Sie den Beispielcode im GitHub-Repo nur zum Überprüfen Ihrer Arbeit oder um sich aus einer Sackgasse zu befreien.

Ein Hinweis zu Versionen

Alle Beispiele in diesem Buch wurden mit Terraform 1.x und AWS Provider 4.x getestet – den neuesten Major-Releases beim Schreiben dieser Zeilen. Weil es sich bei Terraform um ein recht neues Tool handelt, ist es möglich, dass zukünftige Releases Änderungen enthalten, die nicht abwärtskompatibel sind, und dass sich manche der Best Practices mit der Zeit ändern oder weiterentwickeln.

Ich werde versuchen, so oft wie möglich Updates bereitzustellen, aber das Terraform-Projekt schreitet schnell voran, daher werden Sie auch selbst Aufwand investieren müssen. Die neuesten News, Blogposts und Gespräche über Terraform und DevOps finden Sie auf der Website zum Buch (*http://www.terraformupandrunning.com/*), oder Sie melden sich für den (englischsprachigen) Newsletter unter *http://www.terraformupandrunning.com/#newsletter* an!

Verwenden von Codebeispielen

Dieses Buch dient dazu, Ihnen bei der Erledigung Ihrer Arbeit zu helfen. Im Allgemeinen dürfen Sie die Codebeispiele aus diesem Buch in Ihren eigenen Programmen und der dazugehörigen Dokumentation verwenden. Sie müssen uns dazu nicht um Erlaubnis bitten, solange Sie nicht einen beträchtlichen Teil des Codes reproduzieren. Beispielsweise benötigen Sie keine Erlaubnis, um ein Programm zu schreiben, in dem mehrere Codefragmente aus diesem Buch vorkommen. Wollen Sie dagegen eine CD-ROM mit Beispielen aus Büchern von O'Reilly verkaufen oder verbreiten, benötigen Sie eine Erlaubnis. Eine Frage zu beantworten, indem Sie aus diesem Buch zitieren und ein Codebeispiel wiedergeben, benötigt keine Erlaubnis. Eine beträchtliche Menge Beispielcode aus diesem Buch in die Dokumentation Ihres Produkts aufzunehmen, bedarf hingegen unserer ausdrücklichen Zustimmung.

Wir freuen uns über Zitate, verlangen diese aber nicht. Ein Zitat enthält Titel, Autor, Verlag und ISBN. Beispiel: »*Praxishandbuch Terraform. Infrastructure as Code für DevOps, Administration und Entwicklung von Yevgeniy Brikman (O'Reilly), Copyright 2023 dpunkt.verlag. ISBN 978-3-96009-219-3.*«

Wenn Sie glauben, dass Ihre Verwendung von Codebeispielen über die übliche Nutzung hinausgeht oder außerhalb der oben vorgestellten Nutzungsbedingungen liegt, kontaktieren Sie uns bitte unter *kommentar@oreilly.de*.

Konventionen, die in diesem Buch genutzt wurden

Die folgenden typografischen Konventionen kommen in diesem Buch zum Einsatz:

Kursiv
: Steht für neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateierweiterungen.

`Nichtproportional`
: Wird für Programmlistings verwendet, aber auch innerhalb von Absätzen, um sich auf Programmelemente wie Variablen oder Funktionsnamen, Datenbanken, Datentypen, Umgebungsvariablen, Anweisungen und Schlüsselwörter zu beziehen.

`Nichtproportionalschrift fett`
: Steht für Befehle oder anderen Text, der genau so einzugeben ist.

Tipp
Dieses Element enthält einen Tipp oder Vorschlag.

Hinweis
Dieses Element enthält einen allgemeinen Hinweis.

Warnung
Dieses Element enthält eine Warnung.

Danksagung

Josh Padnick

Dieses Buch wäre ohne dich nicht entstanden. Du warst der, der mich als Erster mit Terraform bekannt gemacht hat, mir alle Grundlagen beibrachte und mir dabei half, all die fortgeschritteneren Themen zu meistern. Vielen Dank für die Unterstützung, während ich unsere gemeinsamen Erfahrungen in ein Buch umgewandelt habe. Vielen Dank dafür, dass du so ein famoser Mitgründer bist und es möglich gemacht hast, ein Start-up zu leiten und gleichzeitig Spaß am Leben zu haben. Und vor allem vielen Dank dafür, dass du ein guter Freund und ein guter Mensch bist.

O'Reilly Media

Vielen Dank für das Veröffentlichen eines weiteren meiner Bücher. Lesen und Schreiben haben mein Leben grundlegend geändert, und ich bin stolz darauf, dass ihr mir geholfen habt, einige meiner Texte mit anderen teilen zu können. Ein besonderer Dank geht an Brian Anderson, Virginia Wilson und Corbin Collins für eure Hilfe bei der ersten, zweiten und dritten Auflage.

Gruntwork-Mitarbeitende

Ich kann euch allen gar nicht genug dafür danken, dass ihr (a) bei unserem winzigen Start-up angefangen habt, (b) erstaunliche Software baut, (c) die Stellung gehalten habt, während ich an der dritten Auflage schrieb, und (d) tolle Kollegen und Freundinnen seid.

Gruntwork-Kundinnen und -Kunden

Vielen Dank, dass ihr einer kleinen, unbekannten Firma eine Chance gabt und euch dazu bereit erklärt habt, als Versuchskaninchen für unsere Terraform-Experimente zu dienen. Die Mission von Gruntwork ist, es zehnmal leichter zu machen, Software zu verstehen, zu entwickeln und zu deployen. Wir waren damit nicht immer erfolgreich (viele unserer Fehler kommen in diesem Buch vor!), daher bin ich für eure Geduld dankbar und für euren Willen, Teil unseres gewagten Versuchs zu sein, die Welt der Software zu verbessern.

HashiCorp

Vielen Dank für das Bauen einer erstaunlichen Sammlung von DevOps-Tools, unter anderem Terraform, Packer, Consul und Vault. Ihr habt die Welt von DevOps und damit das Leben von Millionen Softwareentwicklern und -entwicklerinnen verbessert.

Rezensenten

Vielen Dank an Kief Morris, Seth Vargo, Mattias Gees, Ricardo Ferreira, Akash Mahajan, Moritz Heiber, Taylor Dolezal und Anton Babenko für das Lesen früher Versionen dieses Buchs und das Geben von viel detailliertem und konstruktivem Feedback. Eure Vorschläge haben dieses Buch deutlich verbessert.

Leserinnen und Leser der ersten und zweiten Auflage

Diejenigen, die die erste und zweite (englischsprachige) Auflage dieses Buchs kauften, haben diese dritte Auflage erst möglich gemacht. Vielen Dank. Ihr Feedback, Fragen, Pull Requests und Ihr konstantes Anstupsen zu Updates haben die Motivation zu so viel neuen und aktualisierten Inhalten geliefert. Ich hoffe, Sie finden die Aktualisierungen hilfreich, und ich freue mich auf weiteres Anstupsen.

Mama, Papa, Larisa, Molly

Ich habe aus Versehen ein weiteres Buch geschrieben. Das heißt vermutlich, dass ich nicht so viel Zeit mit euch verbracht habe, wie ich wollte. Vielen Dank, dass ihr es trotzdem mit mir aushaltet. Ich liebe euch.