

Terraform

Das Praxisbuch für DevOps-Teams und Administratoren

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Auf einen Blick

1	Einleitung	13
2	Was ist Terraform?	21
3	Erste Schritte mit Terraform	29
4	Provider	57
5	State – der Zustand der Umgebung	185
6	Variablen und Datenquellen	213
7	Terraform programmieren – die Sprache HCL	255
8	Updates und Day 2 Operations	353
9	Erfahrungswerte, Tipps und Tricks	387
10	Vorgefertigtes Beispiel	415
11	Ausblick	423
12	Kommandoreferenz	445

Inhalt

Materialien zum Buch	12
1 Einleitung	13
1.1 Gliederung	13
1.1.1 Was Sie schon wissen sollten	14
1.1.2 Was Sie in diesem Buch lernen und nicht lernen werden	15
1.2 Voraussetzungen für das Arbeiten mit Terraform	16
1.3 Zu den Autoren dieses Werks	17
2 Was ist Terraform?	21
2.1 Cloud Computing und seine Auswirkungen	21
2.2 Das Prinzip Infrastructure as Code	22
2.3 Terraform, seine Geschichte und seine Funktionsweise	23
2.4 Wie funktioniert die Cloud?	24
2.5 Ansible, Chef, Salt, Puppet, Terraform – welches Werkzeug für welche Aufgabe?	26
3 Erste Schritte mit Terraform	29
3.1 Installation von Terraform	29
3.1.1 Installation unter Windows	29
3.1.2 Installation unter macOS	31
3.1.3 Installation unter Linux und anderen Unix-artigen Betriebssystemen	33
3.2 Schnelleinstieg: Ein typischer Terraform-Workflow	40
3.2.1 Erstellen eines neuen Projektverzeichnisses	40
3.2.2 Der Befehl terraform init	41
3.2.3 Die erste Ressource	43
3.2.4 Der Befehl terraform plan	44
3.2.5 Der Befehl terraform apply	47

- 3.2.6 Deklarative Beschreibung des Zustands 48
- 3.2.7 Die Datei terraform.tfstate 49
- 3.2.8 Der Befehl terraform destroy 51
- 3.3 Konventionen für Dateinamen und Unterordner 53**
 - 3.3.1 Empfehlungen für kleinere Projekte 53
 - 3.3.2 Empfehlungen für größere Projekte 54
- 3.4 Ordnerstruktur 56**

4 Provider 57

- 4.1 Provider finden und installieren 57**
 - 4.1.1 Provider finden 57
 - 4.1.2 Provider und deren Versionen 61
 - 4.1.3 Provider verwenden 61
- 4.2 Konfiguration und Authentifizierung 66**
- 4.3 Szenarien: So deployen Sie Terraform 68**
 - 4.3.1 AWS 69
 - 4.3.2 Google Cloud Platform 89
 - 4.3.3 Azure 113
 - 4.3.4 OpenStack 137
 - 4.3.5 libvirt 158
 - 4.3.6 VMware 170
 - 4.3.7 Weitere Provider im Überblick 181

5 State – der Zustand der Umgebung 185

- 5.1 Interaktion mit dem State 187**
- 5.2 Format 189**
- 5.3 Speicherort des State 190**
 - 5.3.1 Remote Backends – Beispiele für entfernte Speicherorte 191
 - 5.3.2 Einrichten eines Remote Backend 192
- 5.4 State durch Locking verriegeln 208**
- 5.5 Sicherheit 210**
 - 5.5.1 Backends 210
 - 5.5.2 Zugangsdaten 211

6	Variablen und Datenquellen	213
6.1	Was sind Variablen?	214
6.2	Eingabevariablen setzen und verwenden	214
6.2.1	Einfache Eingabevariablen	215
6.2.2	Nutzung von Variablen	215
6.2.3	Syntax bei der Nutzung von Variablen	216
6.2.4	Übergeben von Variablen auf der Kommandozeile	217
6.2.5	Eingabevariablen mit Standardwert	218
6.2.6	Übergeben von Variablen über tfvars-Dateien	219
6.2.7	Zuweisen von Werten über Umgebungsvariablen	221
6.2.8	Präzedenz von Variablenzuweisungen	222
6.2.9	Empfehlungen zum Zuweisen von Variablen	223
6.2.10	Beschränkungen bei der Wahl von Variablennamen	225
6.2.11	Variablendeklaration im Detail	225
6.3	Ausgabewerte (Outputs)	238
6.3.1	Verwendung von Ausgabewerten	238
6.3.2	Beispiel eines Ausgabewerts	239
6.3.3	Ausgabewerte mit Beschreibung	240
6.3.4	Ausgabewerte mit vertraulichen Daten	240
6.4	Lokale Variablen oder Werte	241
6.5	Datenquellen (Data Sources)	243
6.5.1	Beispiel: Auflistung von Maschinentypen in GCP	244
6.5.2	Beispiel: Übersetzung von Ressourcennamen in IDs	244
6.5.3	Beispiel: Einlesen von Dateien	245
6.5.4	Beispiel: Externer State als Datenquelle	245
6.6	Interne Werte und Block-lokale Werte	247
6.6.1	Interne Werte	247
6.6.2	Block-lokale Werte	251
7	Terraform programmieren – die Sprache HCL	255
7.1	Logikfunktionen	255
7.1.1	Die count-Anweisung	255
7.1.2	Die for_each-Anweisung	258
7.1.3	Die for-Anweisung	261
7.1.4	Bedingungen (Conditionals)	263
7.1.5	Dynamische Blöcke (Dynamic Blocks)	265

7.2	Provisioner als Schnittstellen zum Konfigurationsmanagement	268
7.2.1	Nutzung eines Provisioners mit einer null_resource	269
7.2.2	Ausführen eines Provisioners beim Erstellen oder Entfernen von Ressourcen	270
7.2.3	Konfiguration der Verbindungsparameter für die Provisioner file und remote-exec	271
7.2.4	Der local-exec-Provisioner	275
7.2.5	Der file-Provisioner	280
7.2.6	Der remote-exec-Provisioner	283
7.2.7	Verfügbarkeit über Provisioner prüfen	286
7.2.8	Der Block-lokale Wert self	288
7.2.9	Alternative Ansätze zur Konfiguration virtueller Maschinen über Provisioner	288
7.3	Abhängigkeiten verwalten	290
7.3.1	Abhängigkeiten von Modulen oder Providern	290
7.3.2	Abhängigkeiten von Ressourcen	292
7.3.3	Explizite Abhängigkeiten von Ressourcen	293
7.4	Arbeiten mit Terraform-Modulen	295
7.4.1	Eigenen Code in Module auslagern	295
7.4.2	Variablen und Rückgabewerte übergeben	300
7.4.3	Nutzung von Modulen aus der Terraform Registry	302
7.5	Deployments versionieren	304
7.5.1	Hochverfügbarkeit und Vorüberlegungen zum Aufbau der Infrastruktur	304
7.5.2	Hochverfügbarkeit auf Terraform-Ebene	305
7.5.3	Providerspezifische Besonderheiten in Bezug auf Hochverfügbarkeit	308
7.6	Code modularisieren und strukturieren: Best Practices für Terraform	309
7.6.1	Versionsverwaltung Ihres Terraform-Codes	309
7.6.2	Module einbinden	310
7.7	Funktionen	315
7.8	Ausdrücke und Expressions	332
7.8.1	Zeichenketten und Interpolation in Zeichenketten	332
7.8.2	Operatoren	334
7.8.3	Platzhalter (Splat Expressions)	337
7.9	Nützliche Funktionen und Beispiele	338
7.9.1	Die element-Funktion	338
7.9.2	Verschachtelte Variablen	339
7.9.3	Die abspath-Funktion	340
7.9.4	Die pathexpand-Funktion	340

7.9.5	Die lookup-Funktion	341
7.9.6	Der count-Parameter	343
7.9.7	Der Block-lokale Wert count.index	344
7.9.8	Validierung von Benutzereingaben	345
7.9.9	Typprüfungen zur Validierung von Benutzereingaben	348
7.9.10	TF_VAR_-Umgebungsvariablen	351
8	Updates und Day 2 Operations	353
8.1	Backup, Backup, Backup	353
8.2	Überprüfung und automatische Qualitätskontrollen für Terraform-Code	354
8.3	Mit Terraform arbeiten	355
8.4	Überwachung (Monitoring) der Umgebung	355
8.4.1	Horizontale Skalierung bei Lastspitzen	356
8.4.2	Vertikale Skalierung von virtuellen Maschinen	357
8.4.3	Wenn horizontale Skalierung nicht mehr ausreicht	359
8.5	Updates und Änderungen einspielen	359
8.5.1	Immutable Infrastructure	360
8.5.2	Koordiniertes Ausrollen neuer Versionen	362
8.5.3	Umsetzung der Release-Strategien mit Terraform	368
8.5.4	Ausrollen neuer Betriebssystemabbilder	368
8.6	Lifecycle-Management mit Terraform	369
8.6.1	Der lifecycle-Parameter	370
8.6.2	Erzwingen des Neuerstellens von Ressourcen	376
8.7	Day 2 Operations für Terraform	379
8.7.1	Workflow zum Arbeiten mit Terraform	380
8.7.2	Backups von Terraform und dem State	380
9	Erfahrungswerte, Tipps und Tricks	387
9.1	Versionsverwaltung nutzen	387
9.1.1	Dateien aus der Versionsverwaltung ausschließen	388
9.2	Remote Backends nutzen	390

9.3	Terraform-Code strukturieren durch Nutzung von Modulen	390
9.3.1	Erster Schritt: Statischer Terraform-Code	391
9.3.2	Zweiter Schritt: Module nutzen	391
9.3.3	Dritter Schritt: Weitere Parametrisierung des Terraform-Codes	393
9.3.4	Vierter Schritt: Providerunterschiede abstrahieren	394
9.3.5	Fünfter Schritt: Mehrfachnutzung für verschiedene Kunden	396
9.4	Terraform-Code und Repository logisch strukturieren	398
9.4.1	Empfehlungen für Dateinamen	398
9.4.2	Trennung nach Landschaften oder Systemumgebungen	399
9.4.3	Benennung von Ressourcen	403
9.5	Terraform-Code prüfen, kontrollieren und testen	404
9.5.1	Prüfung auf korrekte Syntax	406
9.5.2	Prüfung auf Einhaltung von Codekonventionen	407
9.5.3	Beispiel für ein Makefile	408
9.6	Variablen mit Standardwert null	411
9.7	Zugriff auf geschachtelte Listen	412
10	Vorgefertigtes Beispiel	415
<hr/>		
10.1	Szenario	415
10.2	Vorbereitung	416
10.2.1	SSH-Schlüssel anlegen	417
10.2.2	Infrastruktur auswählen	417
10.3	Das Deployment	419
11	Ausblick	423
<hr/>		
11.1	Automate the Automation	423
11.2	Terraform Cloud und Enterprise	425
11.2.1	Nutzung von Terraform Cloud und Terraform Enterprise	426
11.2.2	Benutzerzugang erstellen und testen	429
11.3	Terragrunt, Timon und weitere Tools	437
11.3.1	Terragrunt	437
11.3.2	Timon	440
11.3.3	Terratest	440

11.3.4	Werkzeuge zum Importieren bestehender Infrastruktur	440
11.4	Cloud Development Kit Terraform (CDKTF)	442
12	Kommandoreferenz	445
12.1	Allgemeine Parameter	445
12.1.1	Der Parameter <code>-chdir</code>	445
12.1.2	Hilfe anzeigen per <code>-help</code>	446
12.1.3	Der Parameter <code>-version</code>	446
12.2	Wichtige Befehle	446
12.2.1	Der Befehl <code>terraform init</code>	447
12.2.2	Der Befehl <code>terraform validate</code>	450
12.2.3	Der Befehl <code>terraform plan</code>	451
12.2.4	Der Befehl <code>terraform apply</code>	454
12.2.5	Der Befehl <code>terraform destroy</code>	457
12.3	Weniger häufig verwendete Befehle	458
12.3.1	Der Befehl <code>terraform console</code>	458
12.3.2	Der Befehl <code>terraform fmt</code>	461
12.3.3	Der Befehl <code>terraform force-unlock</code>	462
12.3.4	Der Befehl <code>terraform get</code>	463
12.3.5	Der Befehl <code>terraform graph</code>	464
12.3.6	Der Befehl <code>terraform import</code>	465
12.3.7	Der Befehl <code>terraform login</code>	467
12.3.8	Der Befehl <code>terraform logout</code>	468
12.3.9	Der Befehl <code>terraform output</code>	468
12.3.10	Der Befehl <code>terraform providers</code>	470
12.3.11	Der Befehl <code>terraform refresh</code>	472
12.3.12	Der Befehl <code>terraform show</code>	472
12.3.13	Der Befehl <code>terraform state</code>	473
12.3.14	Der Befehl <code>terraform taint</code>	480
12.3.15	Der Befehl <code>terraform untaint</code>	481
12.3.16	Der Befehl <code>terraform version</code>	483
12.3.17	Der Befehl <code>terraform workspace</code>	483
12.4	Konfigurationsoptionen für die Terraform-CLI (alias Terraform Settings)	485
12.4.1	Speicherort der Konfigurationsdatei	485
12.4.2	Aufbau der Konfigurationsdatei	486
Index	491