MAYFLOWER

Terraform & AWS

Hands-On, 05. Juli 2023

HashiCorp
Terraform

Willkommen bei Terraform

- Infrastructure as Code (**IaC**) Tool
- Entwickelt von **HashiCorp** (Vagrant, Vault, Packer)
- Zum Erstellen, Ändern und Versionieren von **Cloud Infrastruktur**
- Cloud Provider agnostisch (e.g. AWS, Azure, Google Cloud)
- Ermöglicht das automatisierte Deployen, Provisionieren und Verwalten von Cloud Infrastruktur und Cloud Ressourcen in Clouds oder Datenzentren (**CI/CD**)

Imagine geht in die Cloud

Imagine hat diverse Web Applikationen für ihre Firma entwickelt die alle auf unterschiedlichen Tech Stacks basieren. Jetzt möchte sie die Projekte in die Cloud deployen und so ihre Apps über public IPs öffentlich zugänglich machen.

Sie hört von dem Tool Terraform, einem Infrastructure as Code Tool, mit dem die Provisionerung einer Cloud Infrastruktur automatisiert werden kann – und lernt das Tool in der Praxis kennen.

Alle handelnden Personen sind frei erfunden

Architektur & Arbeitsweise

- Terraform CLI
- Deklaration aller Ressourcen in **HCL**
  (HashiCorp Configuration Language)
- Wichtigste CLI Kommandos: **init**, **plan**, **apply**, **destroy**
- .terraform directory, Lock-File (add to VCS), State-File

Beispiel 1:
AWS Availability Zones

- Anzeige aller verfügbaren AWS Availability Zones für den AWS Account der Firma
- Referenz Availability Zones
- Code Beispiel:

availability-zones.tf

Beispiel 2: Tiernamen

- Erstellen zufälliger Tiernamen die als Bezeichner für zukünftige Cloud Ressourcen vergeben werden können
- Referenz Pet Resource
- Code Beispiel:

random-pets.tf

AWS Cloud Deployment

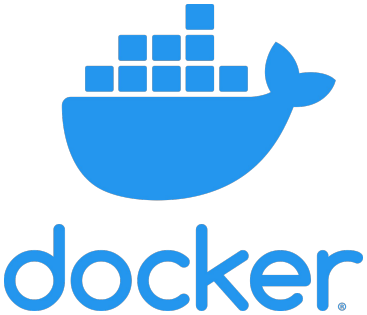
Imagine kennt nun das Tool Terraform und dessen wichtigste Befehle.

Ihr nächstes Ziel ist nun die automatisierte Provisionierung aller Cloud Ressourcen, die für den Betrieb ihrer Web Applikationen in der AWS Cloud erforderlich sind – um so die Apps über public IPs verfügbar zu machen.

MAYFLOWER

Terraform & AWS Hands-On

Hands-On, 05. Juli 2023



Hands-On: Big Picture

- Deployment der Node.js Web Applikation in einem lokalen Docker Container
- Starten des lokalen Docker Containers
- Betrieb und Requesten der Web Applikation im lokal laufenden Docker Container
- Deployment des Docker Application Containers in die AWS Cloud

HashiCorp
Terraform

aws

Hands-On: Agenda

1. Anforderungen
   (Tools und Account Setup)
2. Erstellen einer neuen Terraform Konfiguration zum Deployen des Node.js Containers in die AWS Cloud
3. Weitere Terraform Features
   (Eingabe- und Ausgabevariablen)
4. Deployment des zweiten Application Containers mit der nginx Web Applikation
5. Aufschalten des dritten Application Containers mit php-fpm auf den nginx Container
6. Weitere nützliche Terraform Kommandos
7. Terraform Cloud

Terraform Cloud

- Die Provisionierung und Verwaltung aller Cloud Ressourcen wird für das gesamte Team zugänglich und transparent
- Protokollierung aller geplanten und durchgeführten Änderungen an der Cloud Infrastruktur
- Serverseitiges Durchführen der Cloud Provisionierung

Vorteile von Terraform

Hochgradig Erweiterbar

Developer, Würzburg

Open Source und einheitliche Syntax für Cloud Infrastruktur unterschiedlicher Cloud Provider

Developer, Munich

Zitate dienen nur Veranschaulichungszwecken

Nachteile von Terraform

Keine automatische Rollback Funktionalität für fehlerhafte Änderungen an Cloud Ressourcen

Developer, Berlin

Support für diverse Cloud Lösungen, Kollaborations- und Sicherheits-Features nur verfügbar in kosten-pflichtigen Enterprise Plänen

Developer, Würzburg

Zitate dienen nur Veranschaulichungszwecken


Herzlichen Dank für's
Mitmachen beim
Terraform & AWS Hands-On

Quellen und weitere Infos

Terraform by HashiCorp
Terraform Cloud by HashiCorp
Amazon Web Services
GitHub Repository: Terraform AWS Workshop