

# ***Terraform Workshop***





# Was ist Terraform?

- Tool zur Automatisierung und verwalten der
  - Infrastruktur
  - Plattform
  - Services, die auf der Plattform laufen
- Open source
- Deklarativ -> **WHAT**
  - Definition des Endresultats
    - Terraform findet heraus, was man will, und erstellt es
- Was wäre Imperativ? -> **HOW**
  - Definition der genauen Schritte





# ***Tool zur Provisionierung***

***Was bedeutet das?***

- Beispiel
  - Erstellung einer App, die auf GCP (Google Cloud Platform) laufen soll
  - Wie sieht die Infrastruktur aus?



# ***Beispiel Microservice Software in GCP***

App, die aus  
3 Microservices besteht  
und einer Datenbank

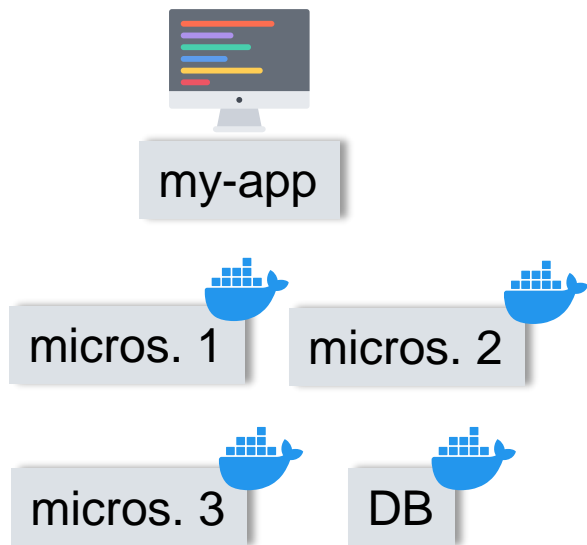




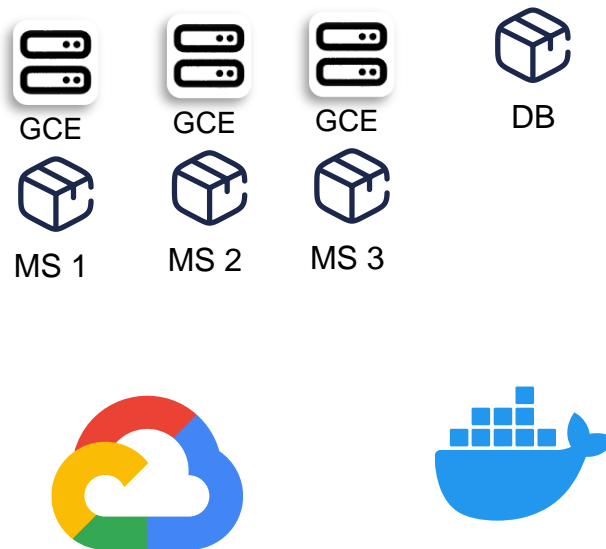
# Beispiel Microservice Software in GCP

*Wie soll es am Ende ausschauen?*

Alle verpackt in  
Docker Container



Infrastruktur





# Beispiel Microservice Software in GCP

*Was muss man dafür machen?*

## Infrastruktur vorbereiten

Private Network

GCE Server Instanzen

Docker und andere Tools installieren

Security Setup (wie Firewalls, networks, etc.)

## Deployment der Applikation

### 1. Provisionierung

- Alles vorbereiten in der Infrastruktur, sodass die Applikation deployed werden kann

### 2. Deployment der Applikation

- Provisionierung u. Deployment sind 2 versch. Dinge!
- Oft versch. Teams/Personen, die Provisionierung u. Deployment machen
  - Provisionierung -> DevOps
  - Deployment -> Softwareentwickler



# Die Rolle von Terraform

## Wie spielt Terraform in diesem Beispiel eine Rolle?

- Terraform wird für den ersten Teil, der Provisionierung verwendet
  - Infrastruktur erstellen u. vorbereiten
- Wichtig: Alles muss in einer gewissen Reihenfolge passieren, da gewisse Ressourcen von anderen abhängig sind
- Bsp.: Docker kann nur auf einem Server installiert werden, der auch schon existiert 🤪

Erstellung VPC

GCP User erstellen

GCP Rechte Erstellung und Zuweisung

Server erstellen

Tool wie Docker installieren



# Ansible vs. Terraform

*“Ist doch das gleiche?!“*

- **Beides:** Infrastructure as Code
  - Provisionierung, Konfiguration u. Verwaltung der Infrastruktur

- **ABER**

- Terraform
  - Hauptsächlich Provisionierungs Tool
  - Kann App Deployen, ist aber nicht dafür da
- Ansible
  - Hauptsächlich Konfiguration der **schon erstellten** Infrastruktur
  - Deployment
  - Install/Update Software



Erstellung von  
Infrastruktur

VS

Beider kann zusammen  
verwendet werden!



ANSIBLE

Konfiguration der  
Infrastruktur



# Zurück zum Beispiel

## Was ist nach der Provisionierung?

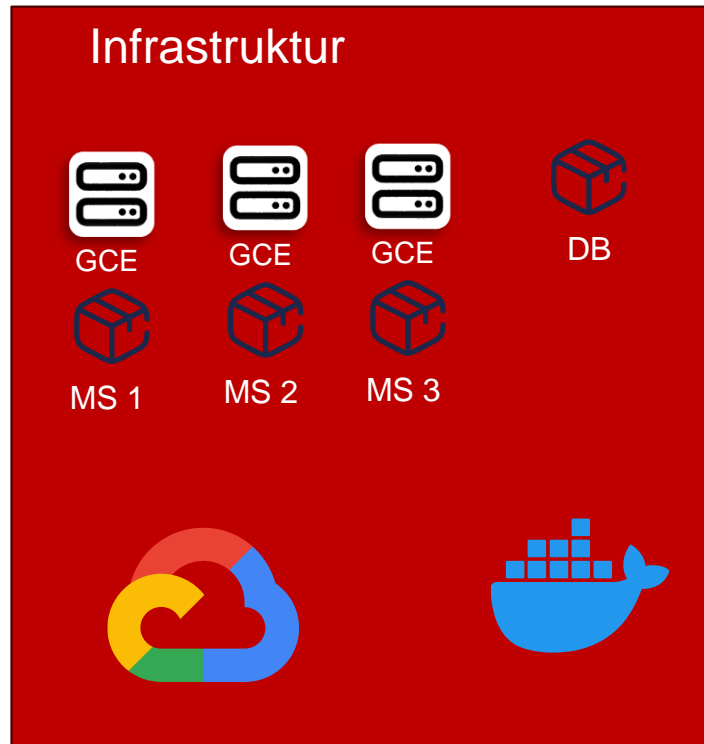
**Annahme:** Alles erstellt

**Neu:** Änderungen werden benötigt

- Mehr Server
- Konfigurationsänderungen
- Weitere User ...
- Entfernung von bestehendem

Änderungen sind einfach mit Terraform u. wichtig!

“Es ist so aufwendig, in GCP noch was zu erstellen und dann danach noch zu Wissen, was man gemacht hat”  
– Quote of a developer





# Zurück zum Beispiel

*Weiteres, oft benötigtes Szenario - Infrastructure Replication*

Oft benötigt man nicht nur eine Umgebung,  
sondern mehrere

DEV

STAGING

PROD

Das kann alles mit Terraform  
automatisiert werden!

x3

Infrastruktur



GCE



GCE



GCE



DB



MS 1



MS 2



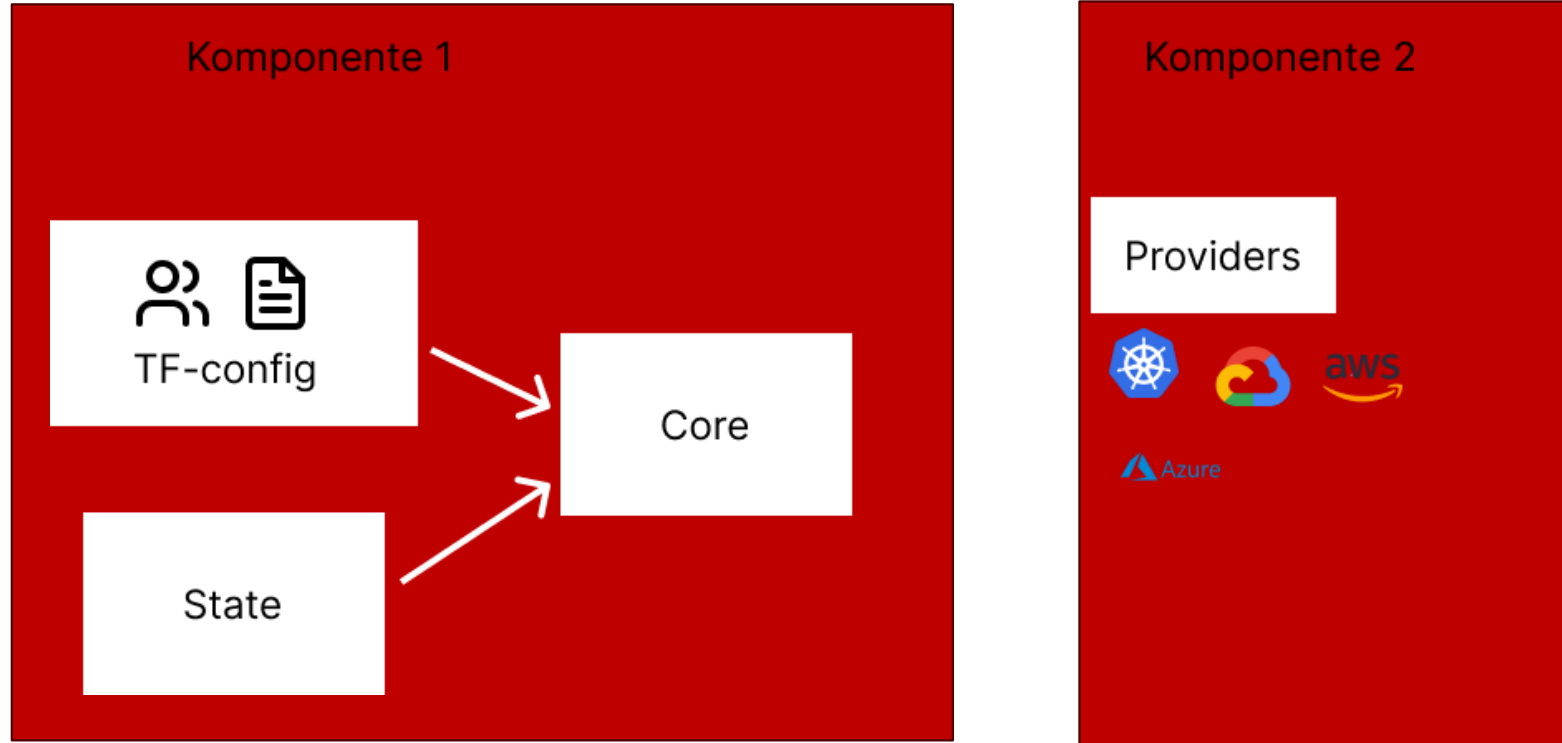
MS 3





# Funktionsweise Terraform

*Wie kann das Terraform überhaupt?*





# Beispiel Syntax

*main.tf file aus dem hands-on Teil*

```
1 terraform {
2   required_providers {
3     docker = {
4       source = "kreuzwerker/docker"
5       version = "~> 3.0.1"
6     }
7   }
8 }
9
10 provider "docker" {
11   host = "npipe:////./pipe/docker_engine"
12 }
13
14 resource "docker_image" "nginx" {
15   name = "nginx"
16   keep_locally = false
17 }
18
19 resource "docker_container" "nginx" {
20   image = docker_image.nginx.image_id
21   name = "tutorial"
22
23   ports {
24     internal = 80
25     external = 8080
26   }
27
28   // for some reason, i had to specify network_mode, pid_mode and ulimits
29   // to have no changes after executing applying terraform apply a second time
30
31   network_mode = "bridge"
32   pid_mode = "private"
33   ulimit { // This sets the maximum number of open file descriptors.
34     hard = 1048576 // This is the soft limit for the resource. It can be increased up to the hard limit without requiring special permissions.
35     name = "RLIMIT_NOFILE" // This specifies the type of resource limit being set. In your configuration
36     soft = 1048576 // This is the hard limit for the resource. It is the maximum value that the soft limit can be set to. Only privileged processes can increase the hard limit.
37   }
38   ulimit { // This sets the maximum number of processes
39     hard = 127690
40     name = "RLIMIT_NPROC"
41     soft = 127690
42   }
43 }
44
```



# Configuration Syntax


## What are all these things in a .tf file? – Low Level Syntax

**Arguments:** Weisen bestimmten Namen innerhalb einer Konfiguration Werte zu.



```
image_id = "abc123"
```

**Blocks:** **Container** für andere Inhalte, die eine Hierarchie definieren.



```
resource "aws_instance" "example" {  
  ami = "abc123"  
  
  network_interface {  
    # ...  
  }  
}
```

**Identifiers:** Namen für Argumente, Blocktypen und andere Konstrukte. Sie können **Buchstaben**, **Ziffern**, **Unterstriche** und **Bindestriche** enthalten, dürfen aber **nicht mit einer Ziffer beginnen**.

**Comments:** Notizen innerhalb der Konfiguration, die von Terraform ignoriert werden. Einzeilige Kommentare beginnen mit **#** oder **//**, und mehrzeilige Kommentare werden in **/\* ... \*/** eingeschlossen.

### Character Encoding and Line Endings:

Konfigurationsdateien müssen **UTF-8**-kodiert sein und können entweder Unix-Stil (**LF**) oder Windows-Stil (**CRLF**) Zeilenenden verwenden, wobei Unix-Stil bevorzugt wird.

# JSON Configuration Syntax

## Terraform language VS. JSON syntax

- Die Terraform Language kann auch als JSON syntax ausgedrückt werden
- Pro: einfacher zum parsen u. generieren
- Contra: “schwieriger” zu lesen
- Wie verwendet man das dann? Andere File Suffixe!
  - .tf
  - .tf.json
- Nicht immer die gleiche Umwandlung von Terraform Syntax zu Json Repräsentation!
  - [Block-type-specific Exceptions](#)

```
variable "example" {  
  default = "hello"  
}  
  
resource "aws_instance" "example" {  
  instance_type = "t2.micro"  
  ami           = "ami-abc123"  
}
```

```
{  
  "resource": {  
    "aws_instance": {  
      "example": {  
        "instance_type": "t2.micro",  
        "ami": "ami-abc123"  
      }  
    }  
  }  
}
```





# *Themen für das nächste mal*

- Erweitertes Beispiel (Praktische Übung)
  - Bereitstellung Ihrer eigenen Infrastruktur lokal -> nginx in einem Docker-Container
  - Lokales Dateisystem "Infrastruktur"
- Wo soll das State file gespeichert werden? Warum ist das wichtig?
- GCP Cloud-Anbieter (und möglicherweise andere?) -
  - > Betrachtung der Otto-Infrastruktur und Beantwortung der Frage ...
  - Wie erstellt man Buckets?
  - Importieren von Ressourcen und vorhandenen TFstates
  - Verwaltung von VMs
  - Erstellen von Benutzern
  - Erstellen von Netzwerken
  - Best practices
- Any other topics?