


An abstract graphic on the left side of the slide. It features a light blue silhouette of a person in a climbing pose, reaching upwards. The person is surrounded by several thick, curved lines in various shades of blue, green, and purple, which appear to be ropes or paths. The background is a solid dark blue.

Terraform 101: A Beginner's Guide

Marcel Zehner |  make it noble
Microsoft Azure MVP
@marcelzehner
marcelzehner.ch



About Me

- Microsoft Azure MVP
- Switzerland
- IT & tech geek
- Speaker & blogger
- Community champion
- Experts Live Europe
- Experts Live Switzerland
- World traveller

What you will get

- A quick infrastructure as code overview
- A nice and easy Terraform introduction
 - Live coding demos from level 0 to 100
 - Mainly focused on Azure

Infrastructure as Code

A (really) quick introduction

What is IaC?

- Automated infrastructure management
- Full lifecycle management
 - Initial provisioning
 - In-service maintenance (config changes)
 - Service decommissioning (end of life)
- Don't get confused: IaC does not focus on IaaS only
 - IaaS, PaaS, SaaS, containers, serverless and more ...

Reasons for IaC

- Infra and apps are becoming more complex
 - More time, budget and human resources needed
- Using IaC ..
 - ... reduces costs
 - ... reduces errors
 - ... optimizes security
 - ... reduces risks
 - ... gives you better control
 - ... adds portability and repeatability
 - ... speeds up processes

IaC Approaches

- Imperative
 - What should be changed?
- Declarative
 - How should it look like?

```
New-AzureVM -VM $myVM
New-AzureStorageAccount -StorageAccountName $acct
Set-AzureVNetConfig -ConfigurationPath -Path
```

```
resource "azurerm_virtual_network" "vnet1" {
  name                = "p-vne-corenet-01"
  location            = "West Europe"
  resource_group_name = "p-rgr-corenet-01"
  address_space       = ["10.0.0.0/16"]
  dns_servers         = ["10.0.0.4", "10.0.0.5"]
}
```

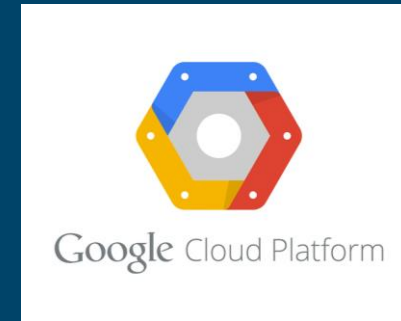
Terraform

A beginner's guide

What is it?

- From Hashicorp
- Tool to provision and manage infrastructure
 - Infrastructure as Code
 - Cloud/platform-agnostic
 - Supports different clouds/platforms
- Declarative approach
 - Using HCL (Hashicorp Configuration Language)
- Full lifecycle management

Multiple Cloud Providers



ARM Templates

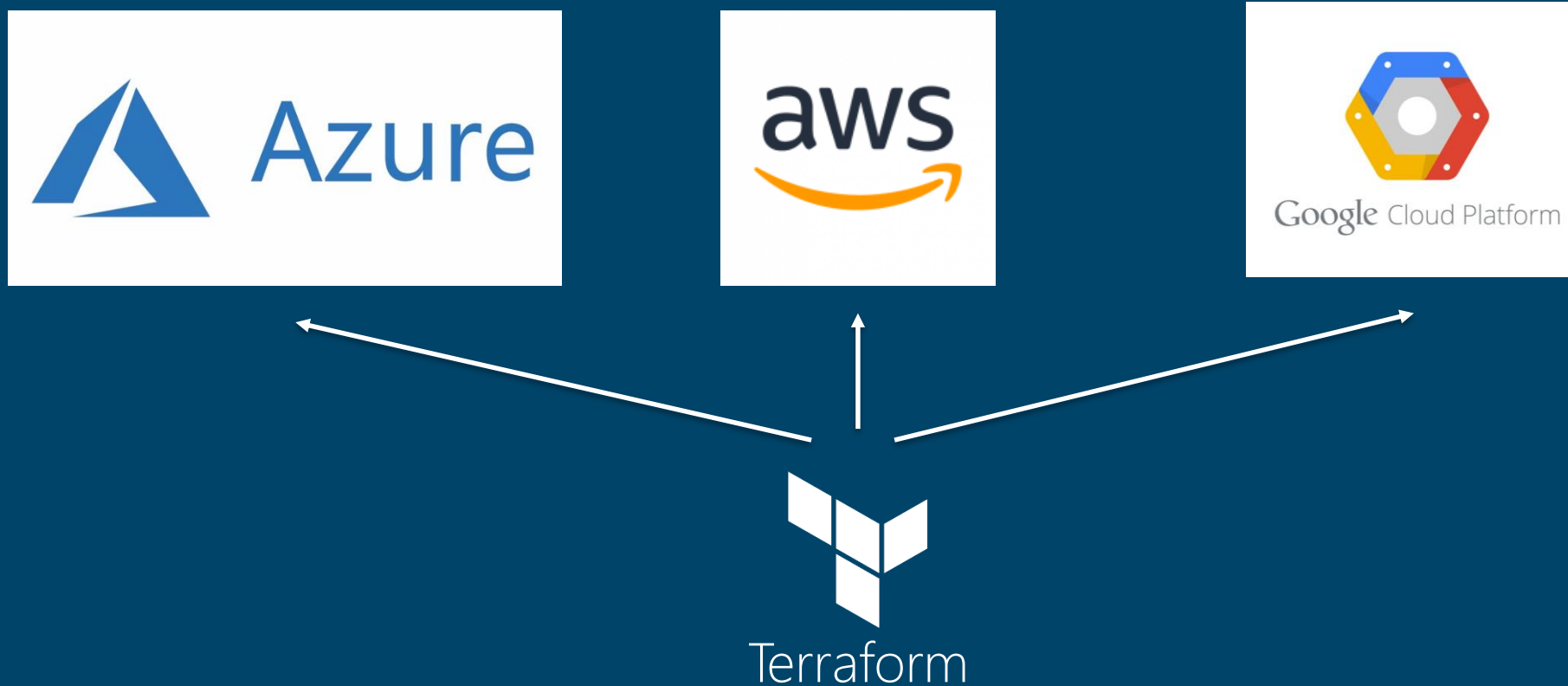


CloudFormation



Deployment Manager

Multiple Cloud Providers



Full Lifecycle Management

New Service

- Initial service deployment
- From nothing to V1

In Service

- Reconfigure and change V1 service
- Extend service
- From V1 to V2, V3, V4, V5 etc.

Out Of Service

- Destroy service

Configuration Files

- Describe the desired state of a service
 - .tf-files
 - One or multiple files
- Use tool of your choice
 - VS Code with a Terraform extension is your friend

```
1 #####
2 # Providers
3 #####
4 provider "azurerm" { ...
7
8 #####
9 # Variables
10 #####
11 variable "location" { ...
15 variable "aadgroup1" { ...
19 variable "policydefinition1" { ...
23 variable "owner" {default = "Marcel Zehner"}
24 variable "session" {default = "Mastering Azure Resources"}
25
26 #####
27 # Resource Groups
28 #####
29 resource "azurerm_resource_group" "rg1" { ...
37 resource "azurerm_resource_group" "rg2" { ...
45 resource "azurerm_resource_group" "rg3" { ...
53 resource "azurerm_resource_group" "rg4" { ...
61
62
63 #####
64 # Resource Groups
65 #####
```

Configuration Files Components

Providers

- "Connector" to cloud/platform
- API interaction & auth

Resources

- Describes resources
- Knows about resource dependencies

Variables

- Parametrization of deployment
- Enhances flexibility

Data Sources

- Get data from existing system or service
- Use in configuration file

Modules

- Complexity abstraction
- Multiple resources
- `registry.terraform.io`

Outputs

- Output data from config
- Generated values

State

- Stores the state about the managed infrastructure
 - Map Terraform id's to real resources
 - Dependencies (in case of resource deletion)
 - Better performance than live querying
- Local state (default) or remote state
 - .tfstate files
- Used to create execution plans

Terraform Process (simplified)

Write
Config
File

- One or multiple files
- Multiple files will be merged

Terraform
Init

- Reads configuration files
- Downloads providers, modules etc. that are specified

Terraform
Plan

- Creates an execution plan based on the configuration files
- Shows "What If"
- Refreshes the known state from the real world (if service already exists)

Terraform
Apply

- Applies the changes to reach the desired state

How to kickoff your journey?

- Download Terraform
 - Free version for individuals and small teams available
 - Single executable > terraform.exe
 - Ready and usable in 1 minute
- Azure alternative
 - Use Azure Cloud Shell
 - shell.azure.com
 - Terraform is pre-installed

Terraform


Live demo

Key Takeaways

- Invest in IaC (and automation in general)
- Terraform is cool
- Terraform is capable to manage multiple clouds and platforms
- Terraform HCL is easy to learn
- ... any maybe a good friend for your future cloud adventures

An abstract graphic on the left side of the slide. It features a light blue silhouette of a person in a climbing pose, reaching upwards. The person is surrounded by several thick, curved lines in various shades of blue, green, and purple, which appear to be ropes or paths. The background is a solid dark blue.

Terraform 101: A Beginner's Guide

Marcel Zehner |  make it noble
Microsoft Azure MVP
@marcelzehner
marcelzehner.ch