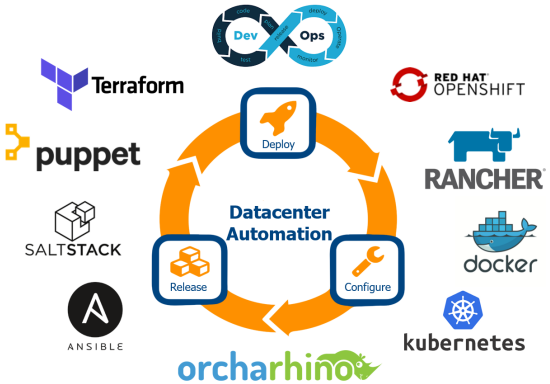
Multi Cloud mit Terraform - Eine Einführung

HashiCorp
Terraform

Martin Grundei

21.08.2022

ATIX AG



ATIX AG

- Open Source Linux & Automatisierung
- Consulting
- Support
- Engineering
- Training

Homepage: `atix.de`

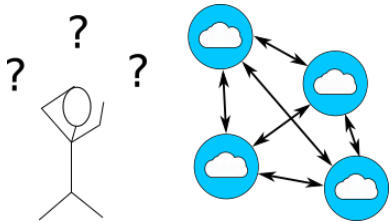
Martin Grundei

Meine Aufgaben als Consultant:

- Linuxadministration
- Konfigurationsmanagement (Puppet, Ansible)
- Cloud (AWS, Azure, Hetzner)
- Infrastructure as Code

Inhalt

- Multi-Cloud - Vorteile und Probleme
- Terraform - Eine kurze Übersicht
- Szenario 1: Azure VM mit AWS Route 53 DNS
- Szenario 2: Site-to-Site VPN Azure - AWS
- Ausblick



Wechseln in die Cloud - Warum?

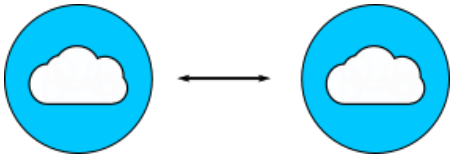
- Flexibilität
- Agilität
- Elastizität
- Verfügbarkeit
- Kosten

Cloud - Szenarien

On premise

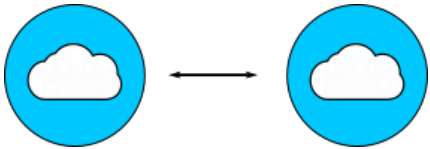
Cloud

Hybrid



Multi-Cloud

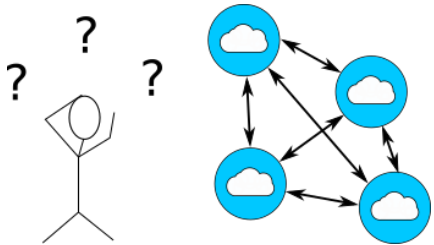
Warum Multi-Cloud?



- Nutzung von Services verschiedener Anbieter
- Bestehende Infrastruktur
- Rechtliche Vorgaben
- Apps in private Clouds
- Kosten

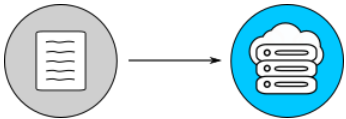
Multi-Cloud - Probleme?

- Wie behalte ich den Überblick?
- Wie verwalte ich meine Toolsets?
- Wie reagiere ich auf Änderungen?



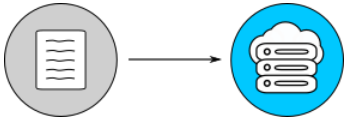
Infrastructure as Code (IaC)

- Keine manuelle Verwaltung der Infrastruktur
- Infrastruktur als Dokumentation/Code
- Automatische Umsetzung der Dokumentation/des Codes



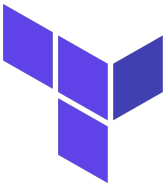
Vorteile von IaC

- deklarativ
- nachvollziehbar
- vollständig
- automatisierbar



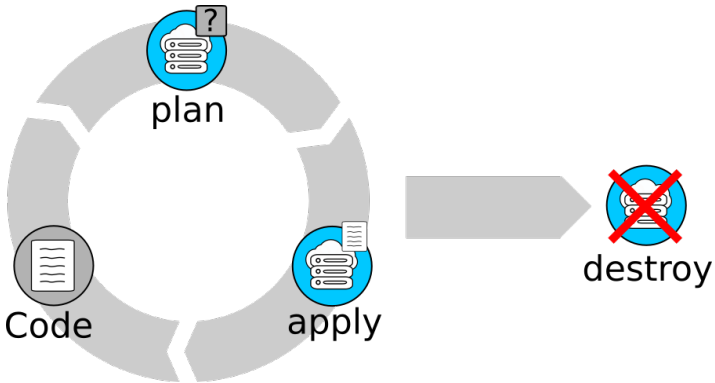
Terraform - Übersicht

- Open-Source by HashiCorp
- Passt interaktiv die bestehende Infrastruktur dem Code an.
- Interagiert mit einer Vielzahl an Cloud/On-Premise APIs.



Terraform Workflow

1. Code schreiben
2. `terraform plan`: Infrastruktur planen
3. `terraform apply`: Infrastruktur aufsetzten
4. `terraform destroy`: Infrastruktur löschen



Terraform - Resourcen

- Resourcen in 'human-readable' Dateien (`.tf`)
- Resourcen beliebig auf Dateien aufteilbar

Beispiel: Erzeugung von SSH Keys

```
resource "tls_private_key" "this" {
  algorithm = "RSA"
  rsa_bits  = 2048
}
```

Reihenfolge in Terraform

- Reihenfolge implicit aus Abhängigkeiten der Resourcen
- Art der Resource beliebig

Beispiel: Ablegen von SSH Key files

```
resource "local_file" "public_key" {
  file_permission = "0666"
  content         = tls_private_key.this.public_key_openssh
  filename = "${path.root}/local/${var.ssh_key_name}.pub"
}
```

Terraforms Gedächtnis

Terraform ’merkt’ sich die erzeugte Infrastruktur.

- Infrastruktur in `.terraform.tfstate` file
- Resourcen rücksetzbar/wiederherstellbar
- Automatisches Löschen der Infrastruktur möglich

Terraform State

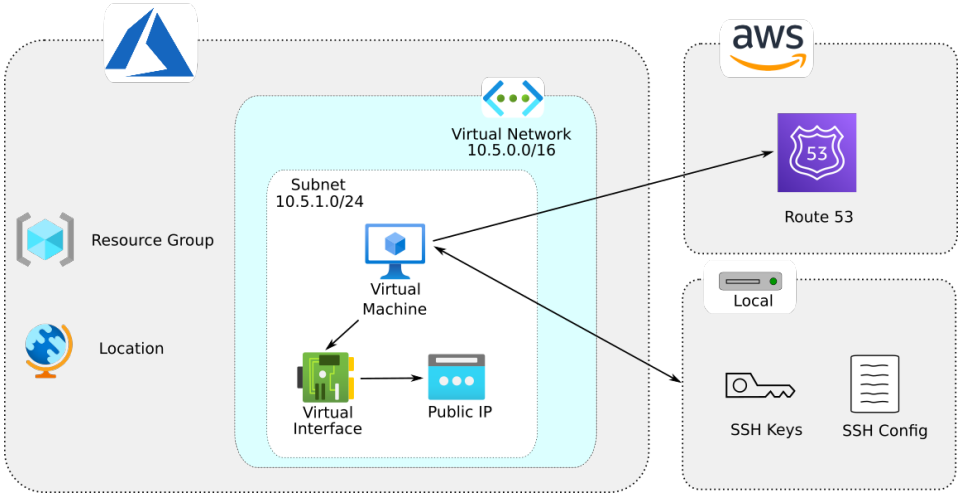
Enthält sensitive Daten im **Klartext**.

`terraform.fstate` sicher aufbewahren:

- Zugang beschränken
- Verschlüsseln
- Backend benutzen (v.a. im Team)

Backends für Gitlab, AWS S3 Bucket, Azure, Kubernetes, Consul, etc.

Azure - Route 53



Azure - Route 53

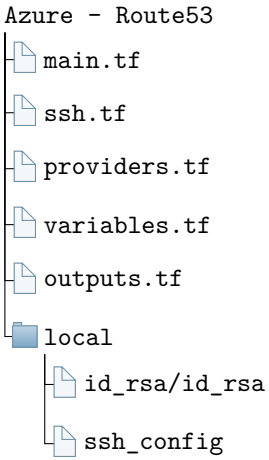
Simples Beispiel-Szenario:

- Linux VM in Azure
- DNS mit Route 53
- Login über SSH

Was wird benötigt:

- SSH Key
- Azure Virtual Network
- Azure Resource Group
- Azure Subnet
- Azure Linux VM
- Azure Network Interface
- Azure Public IP
- AWS Route 53 A Record

Azure - Route 53 - Code Files

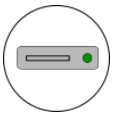


- `main.tf`: Hauptteil des Codes, Cloud Resourcen
- `providers.tf`: Provider, Terraform Konfiguration, Grundelgende Datentypen
- `variables.tf`: Variablen (Mehr dazu später)
- `outputs.tf`: Output (Auch mehr dazu später)
- `local`: Lokale Dateien (SSH Keys, SSH Config File)

Provider Block

Provider interagieren mit der API.

```
provider "azurerm" {
  features {}
  skip_provider_registration = true
}

provider "aws" {
  region = var.aws_provider_region
}

provider "local" {}

provider "tls" {}
```



Providers

`hashicorp/azurerm`

- Resource Manager API
- Computing
- Storage
- Networking
- ...

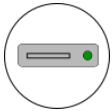
(Für Users/Groups/AD/... s. `hashicorp/azuread`)

`hashicorp/aws`

- Computing
- Datenbanken
- IAM
- Networking
- Container
- ...

`hashicorp/local`

- Lokale Dateien



`hashicorp/tls`

- SSH keys

terraform plan/terraform apply

- Vergleich Ist-Zustand mit Soll-Zustand mit Übersicht
- Bestätigung für das Deployment nötig

```
[user@laptop]$ terraform apply
  # tls_private_key.this will be created
  + resource "tls_private_key" "this" {
      + algorithm                     = "RSA"
      + ecdsa_curve                   = "P224"
      ...
    }
  ...
Plan: 10 to add, 0 to change, 0 to destroy.
  ...
Do you want to perform these actions?
```

terraform plan/terraform apply

- Output nach `terraform apply`

```
Apply complete! Resources: 10 added, 0 changed, 0 destroyed
    .

Outputs:

azure_admin_username = <sensitive>
azure_vm_fqdn = "testubuntuvm.atix-training.de"
public_ip_ip_address = "13.80.22.203"
```

Endergebnis

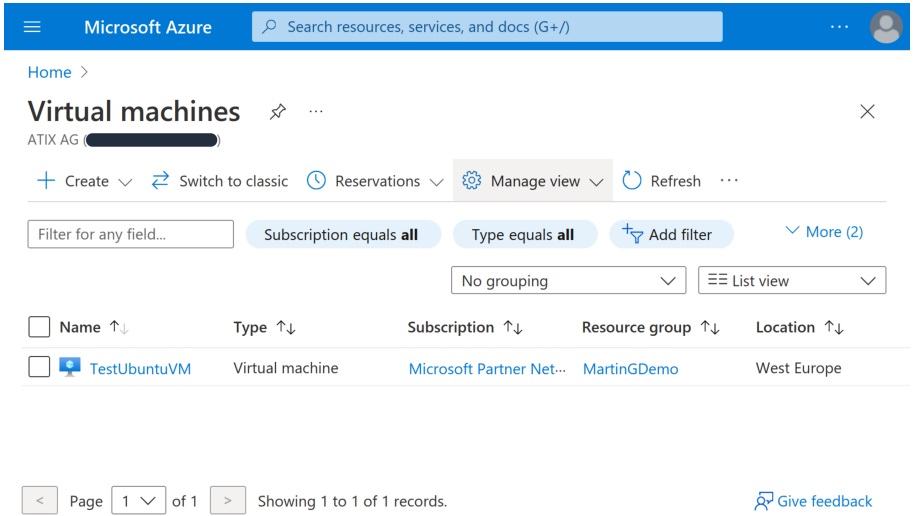
Nach dem Deployment erhält man folgendes SSH Config File

```
Host azure_vm_fqdn_login
  HostName testubuntuvm.atix-training.de
  User atix
  IdentityFile id_rsa
```

login ist möglich mit

```
user@laptop:~$ ssh -F ssh_config azure_vm_fqdn_login
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.15.0-1014-azure x86_64)
...
atix@TestUbuntuVM:~$
```

Endergebnis



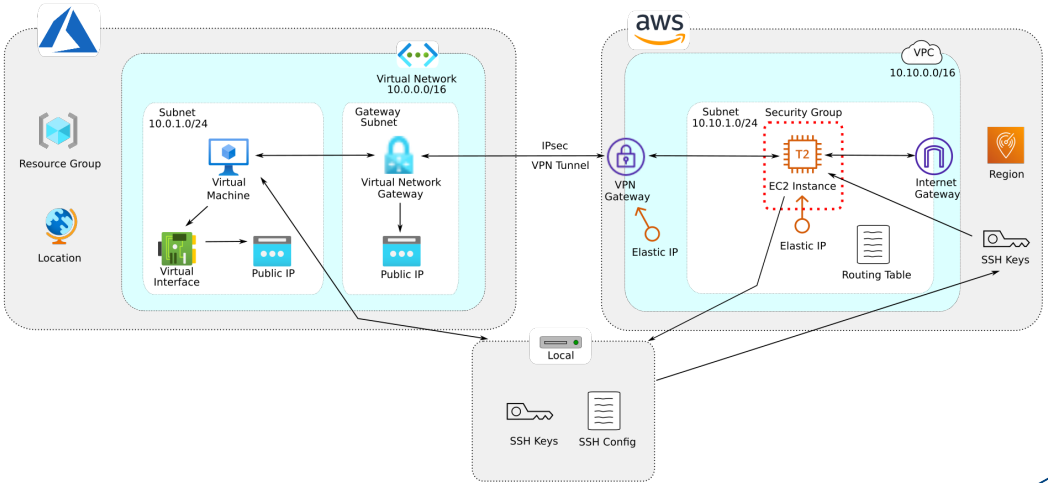
terraform destroy

`terraform destroy` löscht die komplette Infrastruktur.

```
Plan: 0 to add, 0 to change, 10 to destroy.
...

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value:
```

```
Destroy complete! Resources: 10 destroyed.
```

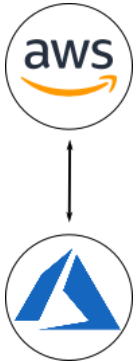
Azure - AWS - Site-to-Site - VPN



Azure - AWS - Site-to-Site - VPN

Szenario:

- Linux VM in Azure subnet `10.0.1.0/24`
- Linux VM in AWS subnet `10.10.1.0/24`
- VPN Site-to-Site Connection zwischen beiden Subnetzen
- ICMP und SSH zwischen beiden VMs über lokales Netzwerk



Azure - AWS - Site-to-Site - VPN

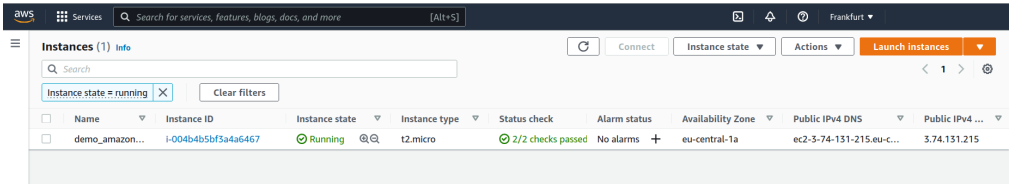
- Setup Azure VM:
  - s. Szenario 1
- Setup AWS VM:
  - VPC
  - Subnet
  - Elastic IP
  - Security Group
  - Internet Gateway
  - EC2 Instance
  - SSH Keys
- AWS VPN Setup:
  - VPN Gateway
  - Elastic IP
  - Routing Table
- Azure VPN Setup:
  - Gateway Subnet
  - Virtual Network Gateway
  - Public IP
- Azure-AWS-connection:
  - 2 Azure Local Gateways
  - AWS Customer Gateway

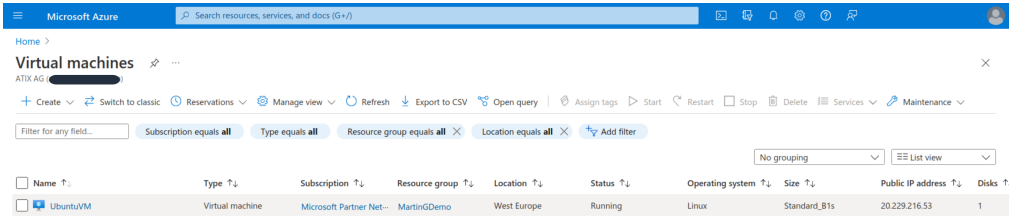
Azure - AWS - Site-to-Site - VPN

terraform apply:

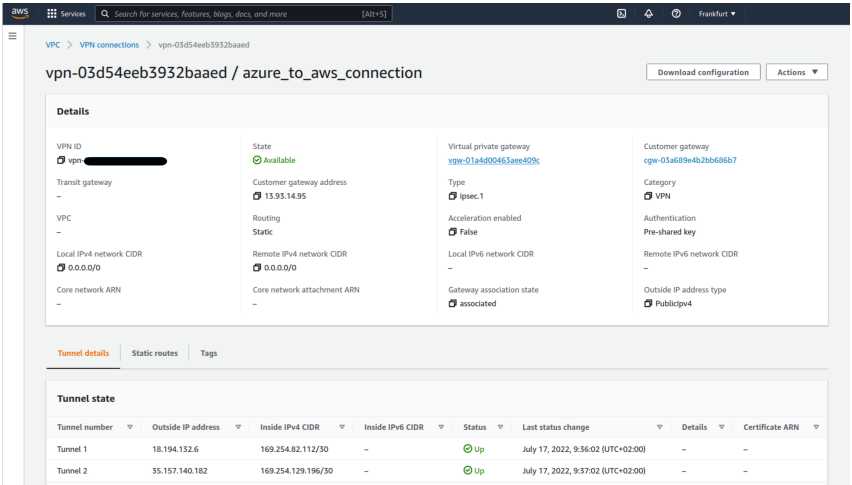
```
Apply complete! Resources: 29 added, 0 changed, 0 destroyed.

Outputs:

aws_elastic_ip_ip_address = "3.74.131.215"
awsvm_local_ip = "10.10.1.220"
azure_admin_username = <sensitive>
azure_public_ip_ip_address = "20.229.216.53"
azurevm_local_ip = "10.0.1.4"
```

Azure - AWS - Site-to-Site - VPN

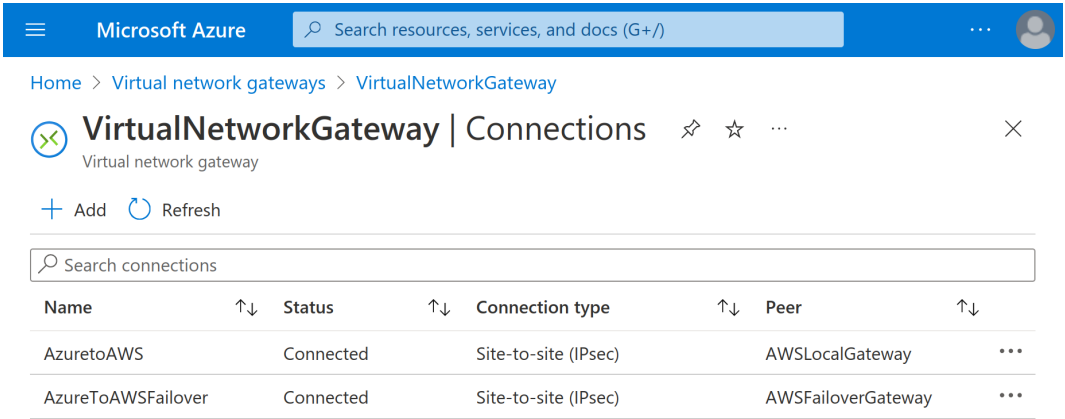




Azure - AWS - Site-to-Site - VPN



Azure - AWS - Site-to-Site - VPN



Azure - AWS - Site-to-Site - VPN

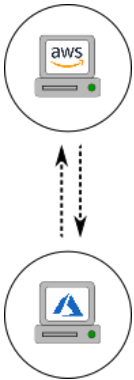
ssh configuration file:

```
Host azure_vm
  HostName 20.229.216.53
  User atix
  IdentityFile id_rsa

Host aws_vm_jump
  HostName 10.10.1.220
  User atix
  IdentityFile id_rsa
  ProxyJump azure_vm
```

Azure - AWS - Site-to-Site - VPN

SSH Proxy Jump:

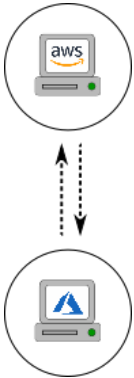
```
[mg@mg-XPS-15-9500 local (develop)]$ ssh -F ssh_config
    azure_vm_jump
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.15.0-1014-azure
    x86_64)
...
Last login: Sun Jul 17 10:24:11 2022 from 10.10.1.220
To run a command as administrator (user "root"), use "sudo
    <command>".
See "man sudo_root" for details.

atix@UbuntuVM:~$
```



Azure - AWS - Site-to-Site - VPN

ICMP:

```
atix@UbuntuVM:~$ ping 10.0.1.4 -c 3
PING 10.0.1.4 (10.0.1.4) 56(84) bytes of data.
64 bytes from 10.0.1.4: icmp_seq=1 ttl=64 time=0.029 ms
64 bytes from 10.0.1.4: icmp_seq=2 ttl=64 time=0.050 ms
64 bytes from 10.0.1.4: icmp_seq=3 ttl=64 time=0.051 ms

--- 10.0.1.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time
    2050ms
```



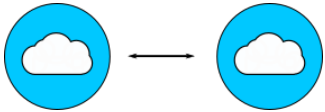
Zusammenfassung

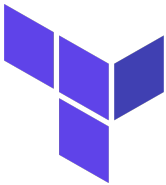
Multi-Cloud

- Ursache: Services, Rechtliche Vorgaben, Kosten,...
- Problem: Wie manage ich meine Umgebungen?
- Lösung: IaC

Terraform

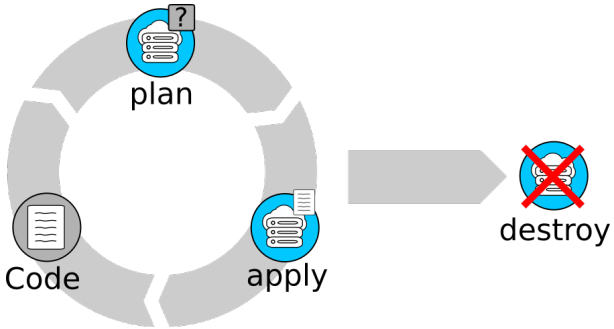
- Resourcen einfach deployen
- Bestimmt automatisch die Reihenfolge
- ‘Merkt’ sich die Infrastruktur





Zusammenfassung

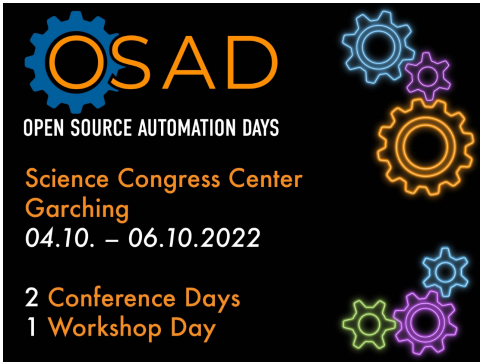
Workflow



Produktiv gehen

- Versionsverwaltung
- Pipelines
- Terraform Backend(!)

Vielen Dank fürs Zuhören



- Documentation: `https://www.terraform.io/docs/`
- HCL specific: `https://www.terraform.io/docs/language/`
- Providers and Modules: `https://registry.terraform.io/`