QA|WARE
SOFTWARE ENGINEERING

Provisionierung von IAAS

Simon Bäumler

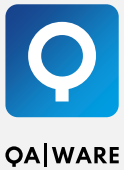
Simon.baeumler@qaware.de

HashiCorp

Terraform

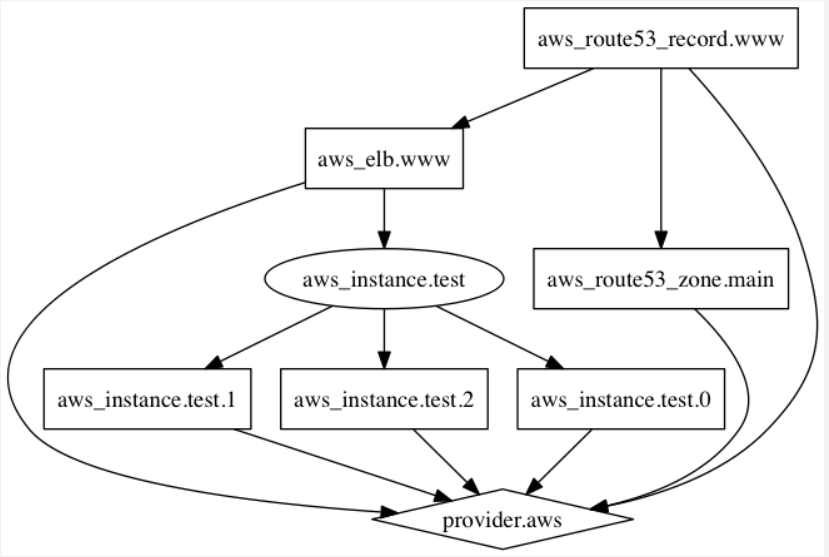
Write, Plan, and Create Infrastructure as Code

Terraform



- Entwickelt von HashiCorp
- Open Source, in Go geschrieben
- Kommandozeilenwerkzeug
- Gedacht um eine Cloud-Infrastruktur zu provisionieren (VMs, VPN, Loadbalancer, Cloud-Storage, etc)
- Nicht zur Installation der Software auf den einzelnen VMs
- Direkte Anbindung vieler Cloud Provider (AWS, Azure, OTC, …)
- Deklarative Programmierung
  - **Write**: Beschreibung Zielzustand über eine domänenspezifische Sprache HCL (HashiCorp Configuration Language)
  - **Plan** (`terraform plan`): Ist-Zustand ermitteln. Notwendige Änderungen planen (entsprechend Abhängigkeiten geordnet und parallelisiert, Unterbrechungen möglichst minimal)
  - **Apply** (`terraform apply`): Idempotente Herstellung des Zielzustands. Der Zustand (*.tfstate* Datei) wird dabei lokal oder in einem Remote Store (S3, HTTP, …) gespeichert

```
terraform/
├── main.tf
└── terraform.tfvars
```

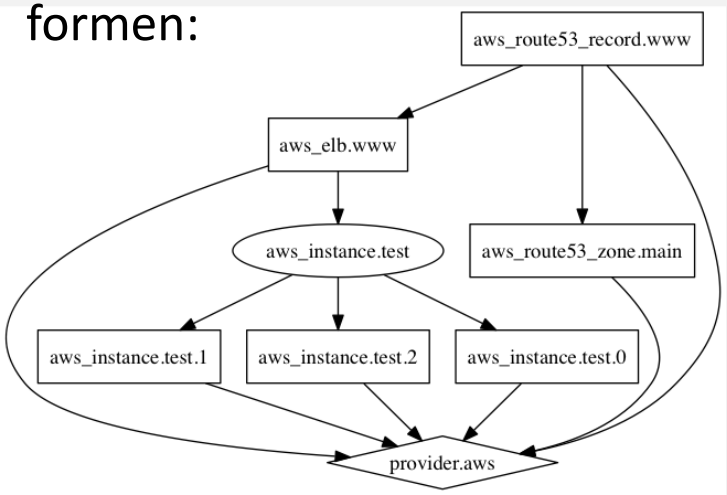


Die Kern-Entitäten eines Terraform-Skripts

Ressource: Provisionierte Komponente der Infrastruktur

```
resource "aws_instance" "web" {
  ami           = "ami-408c7f28"
  instance_type = "t1.micro"
}
```

… haben Abhängigkeiten zueinander, die einen DAG formen:



Provider: Integration der zu provisionierenden Infrastruktur

Alicloud	Archive	AWS
Azure	Bitbucket	CenturyLinkCloud
Chef	Circonus	Cloudflare
CloudScale.ch	CloudStack	Cobbler
Consul	Datadog	DigitalOcean
DNS	DNSMadeEasy	DNSimple
Docker	Dyn	External
Fastly	GitHub	Gitlab

```
provider "aws" {
  access_key = "${var.aws_access_key}"
  secret_key = "${var.aws_secret_key}"
  region     = "us-east-1"
}
```

OpsGenie	Oracle Public Cloud	Oracle Cloud Platform
OVH	Packet	PagerDuty
Palo Alto Networks	PostgreSQL	PowerDNS
ProfitBricks	RabbitMQ	Rancher
Random	Rundeck	Scaleway
SoftLayer	StatusCake	Spotinst
Template	Terraform	Terraform Enterprise
TLS	Triton	UltraDNS
Vault	VMware vCloud Director	VMware NSX-T

Provisioner: Ausführung von Änderungen auf Ressourcen.

```
resource "aws_instance" "web" {
  # ...

  provisioner "local-exec" {
    command = "echo ${self.private_ip} > file.txt"
  }
}
```

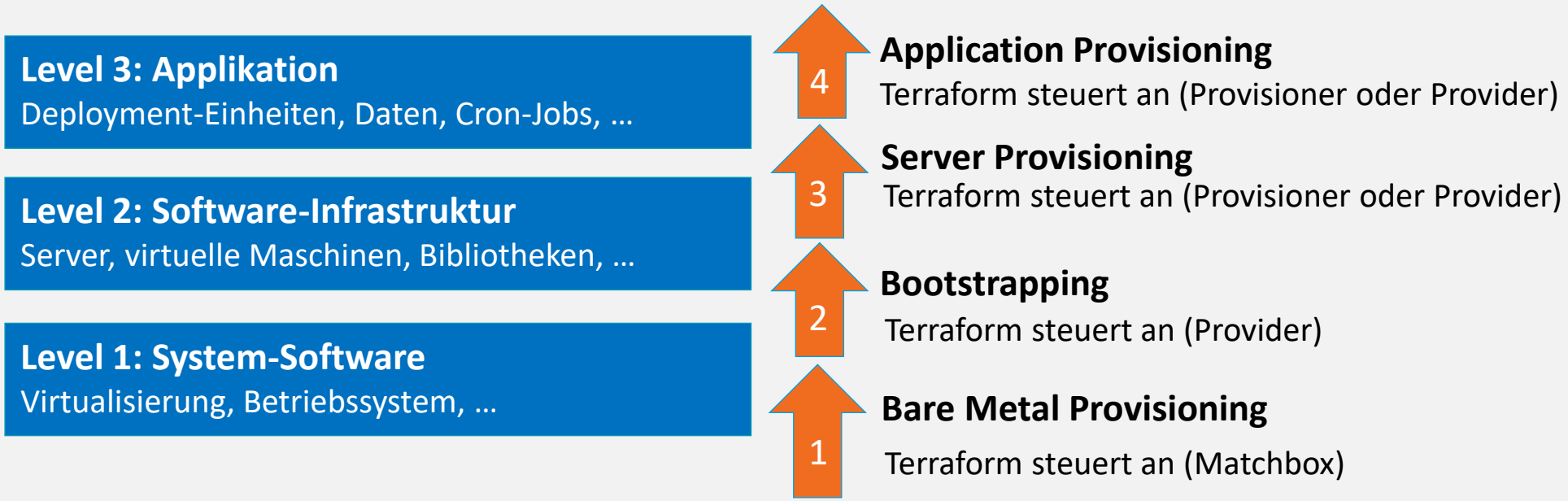
Beispiel


```
# New resource for the S3 bucket our application will use.
resource "aws_s3_bucket" "example" {
  # NOTE: S3 bucket names must be unique across _all_ AWS accounts, so
  # this name must be changed before applying this example to avoid naming
  # conflicts.
  bucket = "terraform-getting-started-guide"
  acl    = "private"
}

# Change the aws_instance we declared earlier to now include "depends_on"
resource "aws_instance" "example" {
  ami           = "ami-2757f631"
  instance_type = "t2.micro"

  # Tells Terraform that this EC2 instance must be created only after the
  # S3 bucket has been created.
  depends_on = ["aws_s3_bucket.example"]
}
```

Provisionierung mit Terraform

QA|WARE

Deployment-Ebenen



Terraform State

Terraform muss den Zustand der Infrastruktur lokal speichern

- Dazu wird das File terraform.tfstate benutzt
- Der State wird im JSON Format geschrieben
- … sollte aber nicht manuell editiert zu werden

Wieso wird der State benötigt?

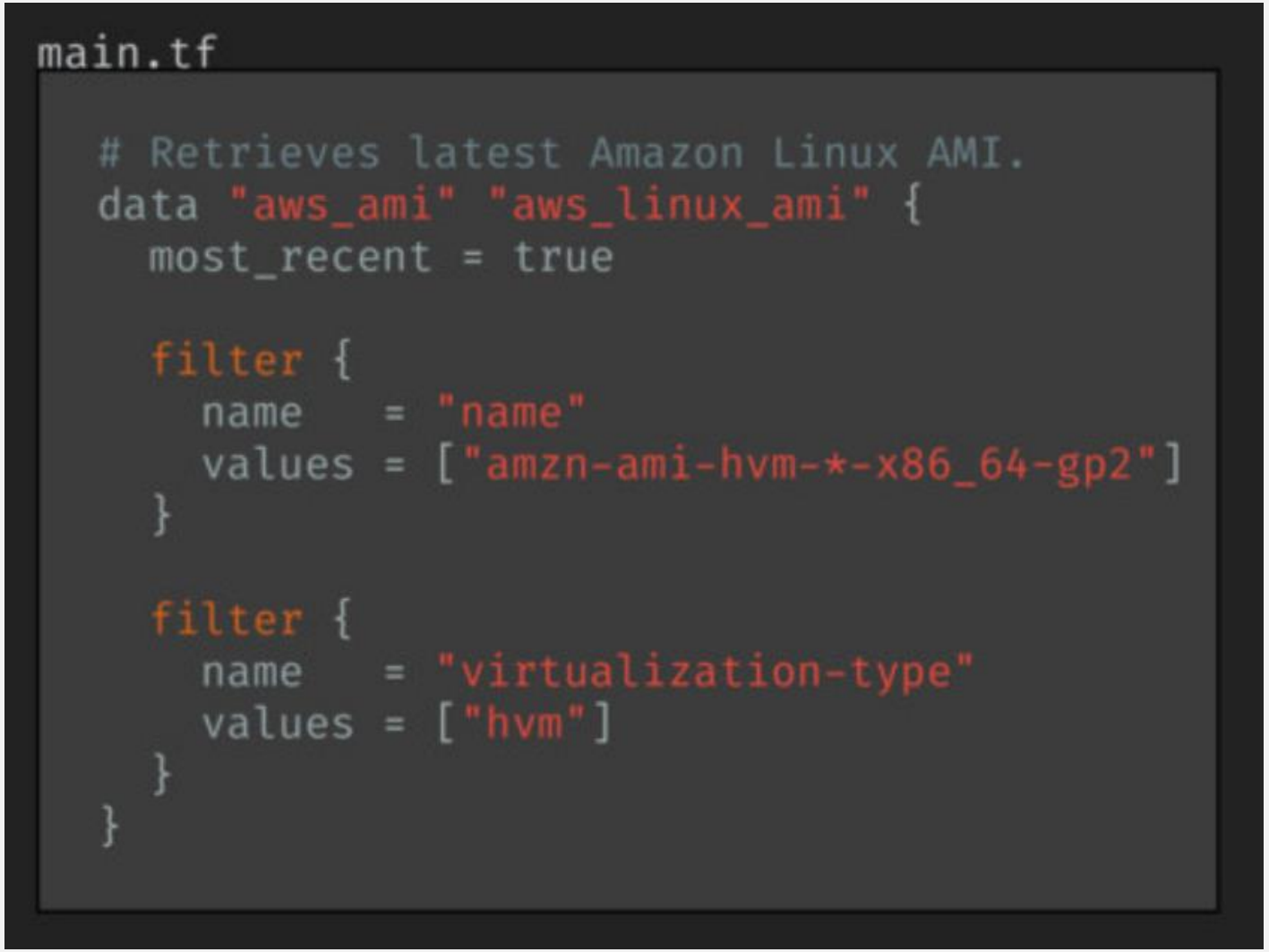
- Mapping von Terraform Identifier zu Cloud Resourcen Identifier
  - Z.B. resource „aws_instance“ „foo“ -> „i-abcd5678“
- Tracking von Abhängigkeiten
- Speedup
  - State minimiert die Anzahl der Cloud-API Aufrufe

Syncronisation

- Im Default wird der State lokal gespeichert
  - Das ist nicht empfohlen für Produktivsysteme, da der Zustand verloren gehen kann
- Empfehlung: Nutzung von Remote State
  - Z.B. wird das Speichern des Zustandes in AWS S3 unterstützt (inkl Locking)

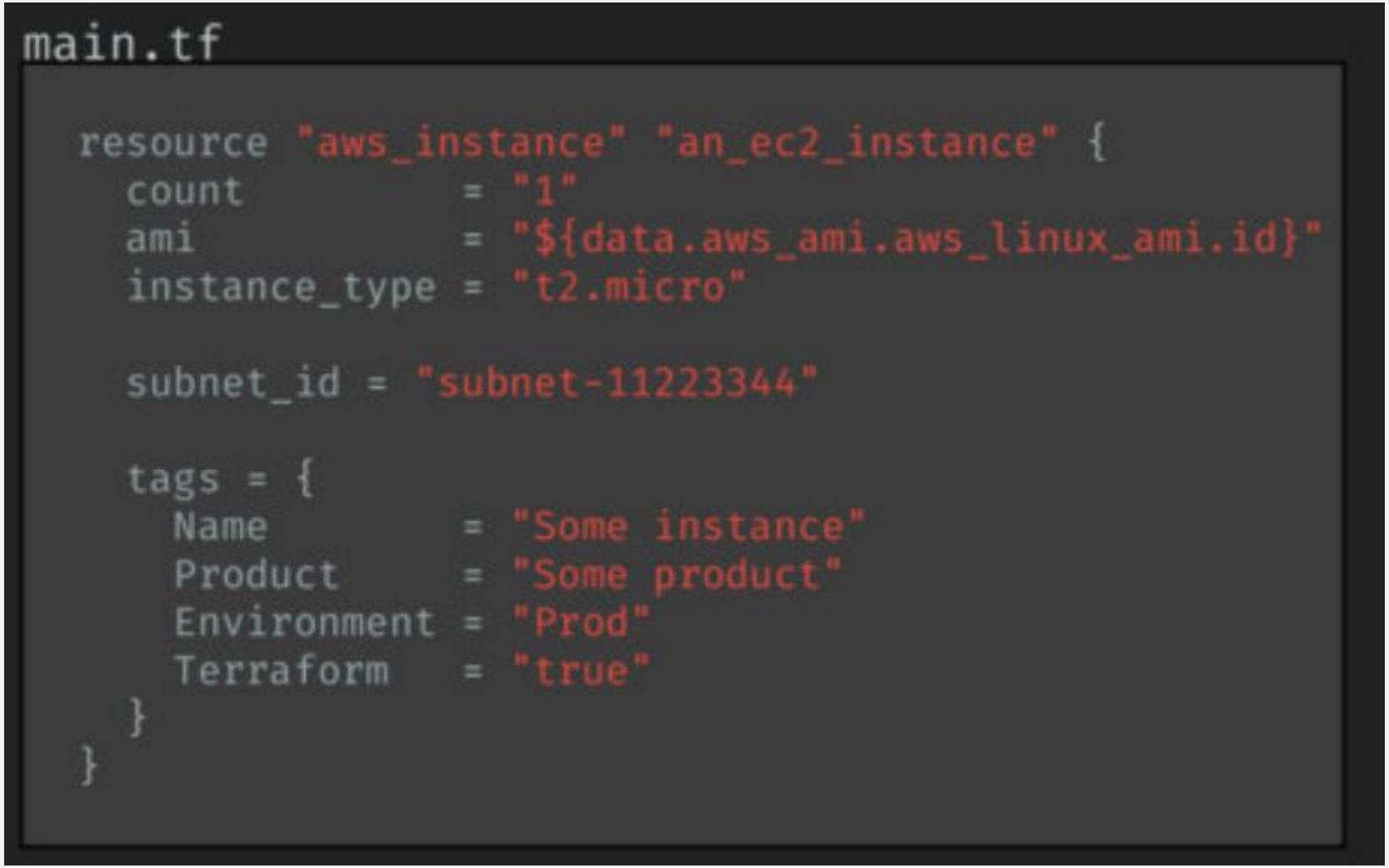
Beispiel: Provider

Beispiel: Data



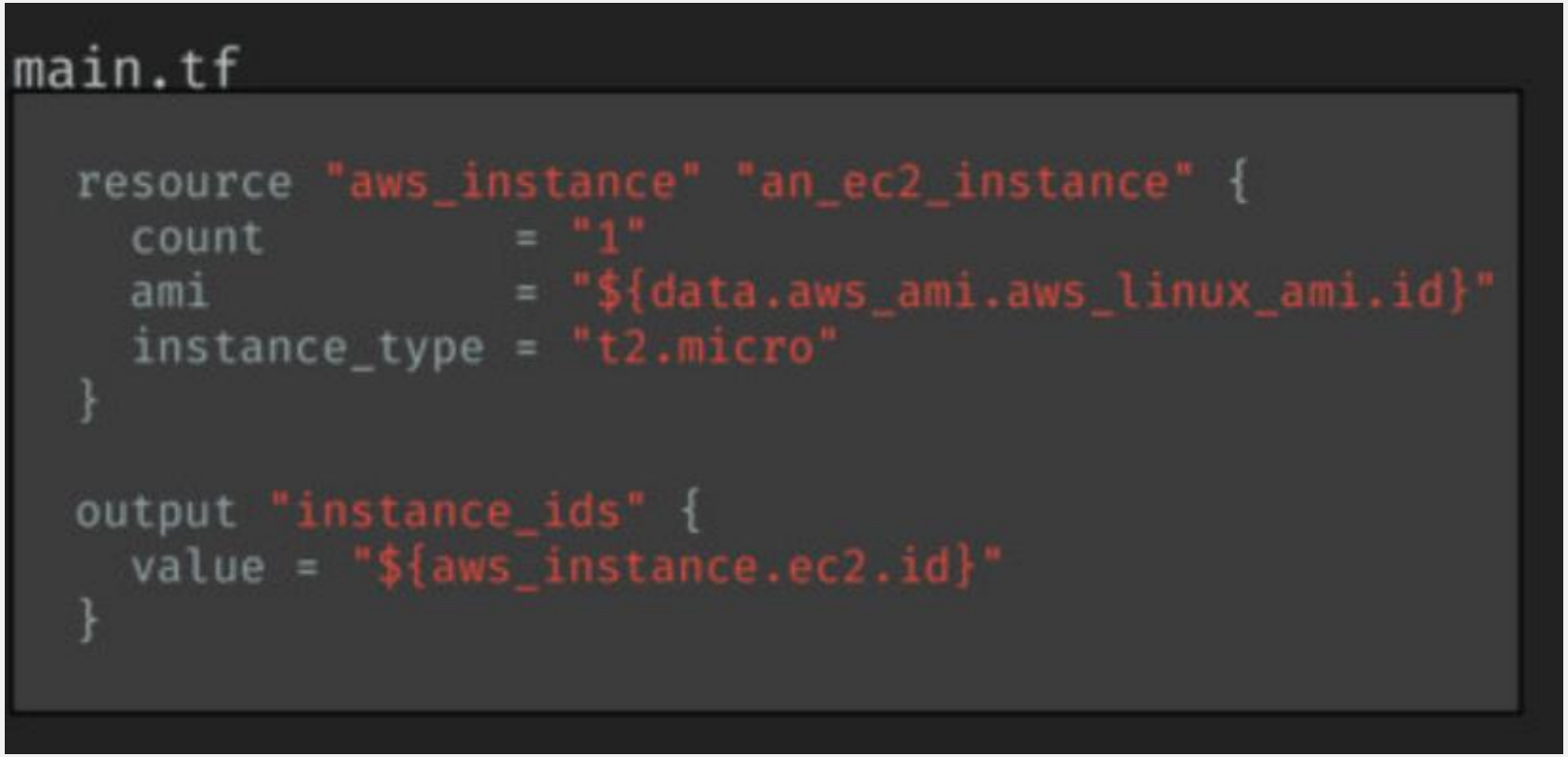
QA|WARE

Beispiel: Resources



QA|WARE

Beispiel: Output



Beispiel: Struktur

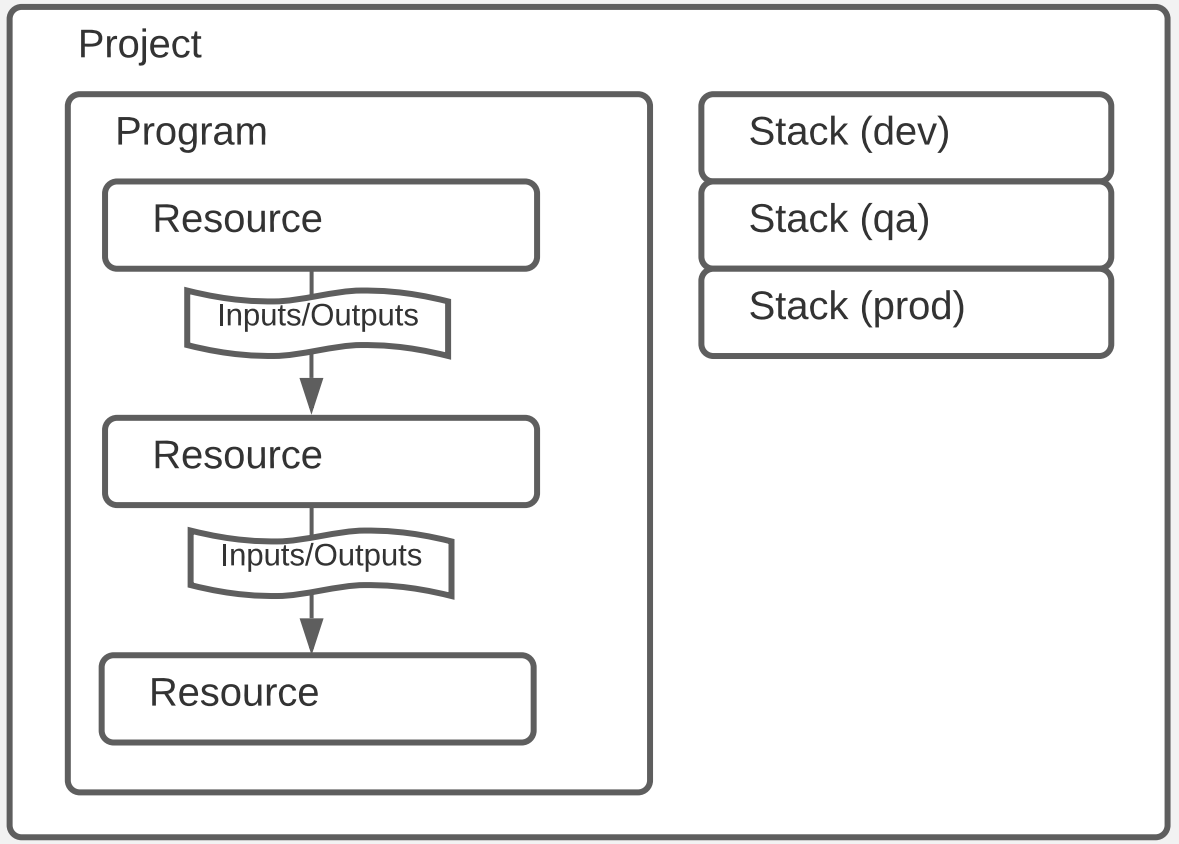
QA|WARE

Alternative zu Terraform: Pulumi

Pulumi - Überblick

Pulumi

- Definition der Infrastruktur in Programm-Syntax
- Unterstützte Programmiersprachen:
  - Javascript / Typescript
  - Python
  - Go
  - C#
- Vorteile:
  - Einfachere Einarbeitung – es muss keine neue Syntax gelernt werden
  - Mächtige Programmiersprachen, z.B. einfache nutzung von Loops um mehrere gleichartige Objekte zu erzeugen



https://www.pulumi.com/docs/intro/concepts/

Pulumi – Python Beispiel

```
import pulumi
import pulumi_aws

group = pulumi_aws.ec2.SecurityGroup(
           ‚secgroup‘,
           description='Enable HTTP access‘,
           ingress=[
                      { 'protocol': 'tcp‘,
                        'from_port': 80,
                        'to_port': 80,
                        'cidr_blocks': ['0.0.0.0/0'] }
           ])

server = pulumi_aws.ec2.Instance(
           'server‘,
           ami='ami-023bffe58a8016c81‘,
           instance_type=‚t2.medium‘,
           vpc_security_group_ids=[group.name])

pulumi.export('public_ip', server.public_ip)
pulumi.export('public_dns', server.public_dns)
```