Dokumentation: Automatisierte Infrastruktur auf AWS mit Terraform: Skalierbare & Kosteneffiziente Bereitstellung

---

Titelblatt

Titel: Automatisierte Infrastrukturbereitstellung auf AWS mit Terraform: Entwicklung eines skalierbaren und kosteneffizienten Cloud-Systems
Autor: [Sodbilegt Barsbold]
Datum: [01.02.2025]
Kurs: [Web Technology - Aktuelle Themen 2: DevOps and Site Reliability Engineering]

Abstract

Diese Arbeit dokumentiert die Entwicklung einer Infrastructure-as-Code (IaC)-Lösung für AWS mittels Terraform. Das Projekt zielte darauf ab, ein skalierbares, wiederverwendbares und kosteneffizientes System zur Bereitstellung von EC2-Instanzen, RDS-Datenbanken und Netzwerkressourcen zu schaffen. Der Fokus lag auf der Anwendung von Best Practices wie Modularisierung, Validierung und Nutzung von AWS Free Tier. Die Dokumentation reflektiert den Lernprozess, technische Herausforderungen und Lösungsansätze.

Inhaltsverzeichnis

---

1. Einleitung

Hintergrund:
Infrastructure-as-Code (IaC) revolutioniert die Cloud-Infrastrukturverwaltung durch Automatisierung und Reproduzierbarkeit. AWS und Terraform sind führende Tools in diesem Bereich.

Zielsetzung:

- Entwicklung eines IaC-Systems für AWS mit Terraform.
- Bereitstellung einer statischen Webanwendung auf EC2 und einer PostgreSQL-Datenbank (RDS).
- Einhaltung von Best Practices (Skalierbarkeit, Kosteneffizienz, Sicherheit).

Problemstellung:

„Viele Unternehmen stehen vor der Herausforderung, ihre Cloud-Infrastruktur effizient zu verwalten, insbesondere in Bezug auf Kosten und Skalierbarkeit. Dieses Projekt adressiert diese Problematik durch den Einsatz von Terraform und AWS."

- Begrenzte Vorkenntnisse in AWS und Terraform.
- Komplexität der Integration von Netzwerken, Sicherheitsgruppen und Ressourcen.

---

2. Literaturrecherche & Theoretische Grundlagen

2.1 Terraform & IaC

- **Terraform:** Deklarative Sprache zur Infrastrukturdefinition (HashiCorp, 2023).
- **Online Kurs:**
  - Terraform: The Complete Guide from Beginner to Expert (Udemy, 2023) vermittelte grundlegende bis fortgeschrittene Konzepte, darunter Modulentwicklung, State Management und AWS-Integration. Der Kurs diente als strukturierte Einführung in Best Practices und Fehlerbehebung.
  - **Link:** Terraform: The Complete Guide

2.2 AWS-Komponenten

- **EC2:** Skalierbare virtuelle Server (AWS, 2023).
- **RDS:** Managed SQL-Datenbanken.
- **VPC:** Isolierte Netzwerkumgebung.

2.3 Vergleich mit Alternativen

- **AWS CloudFormation:** AWS-spezifisch, weniger flexibel (Terraform-Registry, 2023).
- **Ansible:** Konfigurationsmanagement, kein vollständiges IaC.

---

3. Methodik & Projektplanung

3.1 Tools & Setup

- **Terraform CLI:** Installation und Konfiguration.
- **AWS-CLI:** Authentifizierung via IAM-Rollen.
- **IDE:** Visual Studio Code mit Terraform-Erweiterungen.

3.2 Phasen des Projekts

1. **Grundlagenforschung:**
    - AWS Free Tier-Nutzung.
    - Terraform-Dokumentation (HashiCorp, 2023).

2. **Strukturiertes Lernen:**
    - Absolvierung des Udemy-Kurses Terraform: The Complete Guide zur Vertiefung von Konzepten wie:

        - Terraform-State und Backend-Konfiguration.
        - Modularisierung und Wiederverwendbarkeit.
        - Integration mit AWS-Ressourcen (EC2, VPC, RDS).

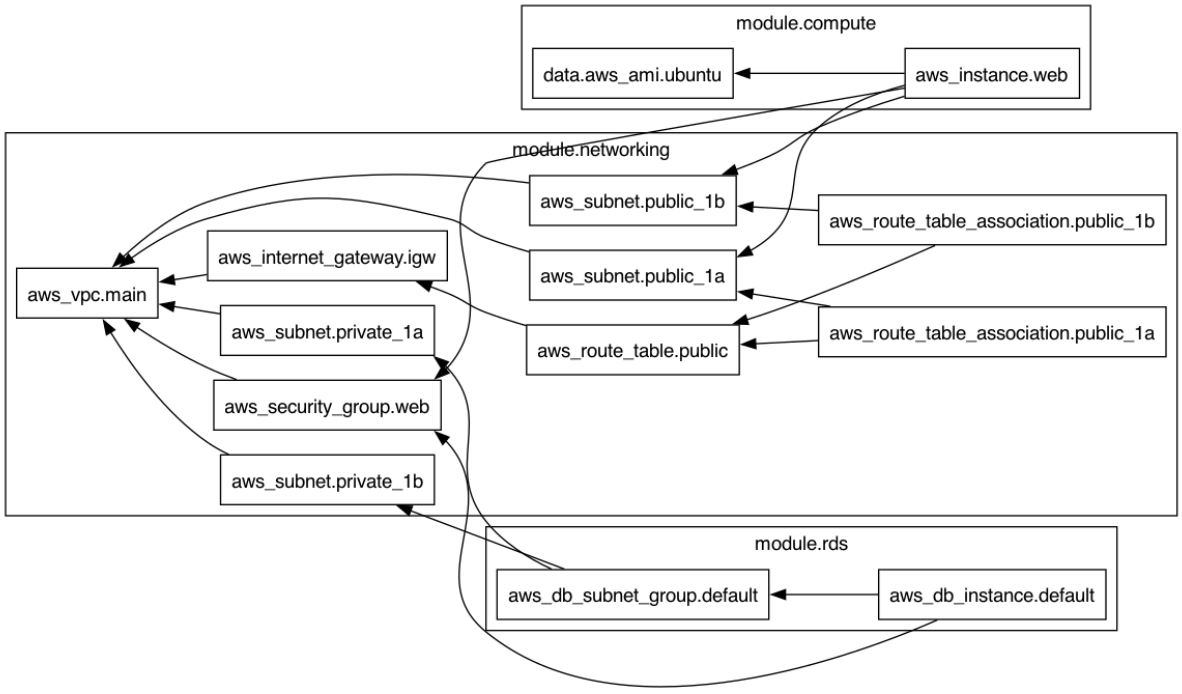
3. **Experimente:**
    - Erstellung von S3-Buckets, EC2-Instanzen.
    - Nutzung von Data Sources für AMI-IDs.
4. **Strukturierung:**
    - Modularisierung (Netzwerk, Compute, RDS).
    - Variablenvalidierung und tfvars-Dateien.
5. **Testing:**
    - Iteratives terraform apply und destroy.

---

4. Implementierung

4.1 Architekturdiagramm

VPC mit öffentlichen/privaten Subnetzen, EC2, RDS und Sicherheitsgruppen.



4.2 Code-Struktur

```
terraform-project/
 ├── modules/
|    ├── networking/  # VPC, Subnetze, Security Groups
|    ├── compute/     # EC2-Instanzen mit NGINX
|    └── rds/         # PostgreSQL-Datenbank
 ├── main.tf          # Modulintegration
 ├── variables.tf     # Globale Variablen
└── terraform.tfvars # Sensible Daten
```

4.3 Schlüsselcode-Snippets

Netzwerkmodul (modules/networking/main.tf):

```
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr  # VPC-CIDR-Block (z.B. 10.0.0.0/16)
  tags = {
    Name = "main-vpc"
  }
}
```

Compute-Modul (modules/compute/main.tf):

```
data "aws_ami" "ubuntu" {
  most_recent = true
  owners      = ["099720109477"]  # Canonical-AMI für Ubuntu 22.04
}

resource "aws_instance" "web" {
  ami           = data.aws_ami.ubuntu.id
  instance_type   = "t2.micro"    # Free Tier-eligible
  user_data       = file("user-data.sh")  # NGINX-Installation
}
```

---

5. Ergebnisse & Diskussion

5.1 Erfolge

- **Funktionierende Infrastruktur:**
  - EC2-Instanz mit NGINX unter http://<PUBLIC_IP>.
  - RDS-PostgreSQL-Datenbank mit privaten Subnetzen.
- **Kosteneffizienz:**
  - Nutzung von Free Tier-Ressourcen (t2.micro, db.t3.micro).
- **Wiederverwendbarkeit:**
  - Modularer Code für zukünftige Projekte.

5.2 Herausforderungen & Lösungen

Herausforderung	Lösung
Fehlende Internet Gateway-Konfiguration	Hinzufügen von aws_internet_gateway und Routentabellen.
Ungültige PostgreSQL-Benutzernamen	Validierung mit validation-Blöcken.
SSH-Zugriffsprobleme	Sicherheitsgruppenregeln für Port 22.

5.3 Ethische & Sicherheitsaspekte

- **Sicherheit:**
  - Sensible Daten (Passwörter) via tfvars geschützt.
  - Eingeschränkte Security Group-Regeln (nur Port 80/22).
- **Kostenmanagement:**
  - Regelmäßiges terraform destroy zur Vermeidung von unerwarteten Gebühren.

---

6. Fazit & Ausblick

Zusammenfassung:
Das Projekt demonstriert die Machbarkeit einer vollständig automatisierten AWS-Infrastruktur mit Terraform. Durch Modularisierung und Validierung wurde eine robuste Lösung geschaffen.

Ausblick:

- Integration von CI/CD-Pipelines (GitHub Actions).
- Nutzung von Terraform Cloud für Remote State Management.

---

7. Literaturverzeichnis

- HashiCorp. (2023). *Terraform Documentation*. https://www.terraform.io/docs
- AWS. (2023). *Amazon Web Services Documentation*. https://docs.aws.amazon.com
- Udemy-Kurs: Zeal. (2023). Terraform: The Complete Guide from Beginner to Expert. Terraform: The Complete Guide

---

8. Anhang

A. Vollständiger Terraform-Code

- Siehe beigefügte Dateien: main.tf, variables.tf, usw.

B. AWS-Architekturdiagramm

C. Testprotokolle
