aufgabe-02.md

AWS Well-Architected Framework - Aufgabe 2

Szenario

Du arbeitest als Cloud Architect bei einem großen Unternehmen. Dein Team ist für die Bereitstellung von Cloud-Infrastruktur verantwortlich. Du hast die Aufgabe, die Infrastruktur für eine neue E-Commerce Webanwendung zu erstellen.

Der Online Shop soll auch in Zukunft bei Fehlern und Ausfällen der Infrastruktur verfügbar sein. Die Anwendung soll auch bei steigender Anzahl von Kunden und Anfragen weiterhin schnell reagieren.

Voraussetzungen

Für den folgenden Praxisteil ist eine AWS Sandbox und folgende Programme notwendig:

- aws-cli (Konfiguriert mit Access Key und Secret Key)
- git

Aufgabe 1 - Infrastruktur als Code

- Um unsere Infrastruktur zentral zu verwalten, wollen wir die Infrastruktur als Code (IaC) bereitstellen.
- Das bedeutet, dass wir die Infrastruktur in einer Programmiersprache definieren und diese dann in die Cloud deployen.

Wieso IaC?

- Einfache Wiederherstellung
- Einfache Wartung
- In verbindung mit git ist Versionskontrolle möglich -> GitOps
- Automatisierte Tests

-> Hilft bei **Betriebliche Spitzenleistung** (Operational Excellence) und **Zuverlässigkeit** (Reliability)

Infrastruktur Repository erstellen

- Erstelle einen neuen Ordner und initialisiere ein git Repository
- In diesem Ordner erstelle eine Datei mit dem Namen `my-online-shop.yaml`

```
mkdir my-online-shop
cd my-online-shop
git init
touch my-online-shop.yaml
git add .
git commit -m "Initial commit"
```

- Um die Anwendung nun automatisch via Code zu deployen, verwenden wir AWS CloudFormation. Dadurch können wir unsere Infrastruktur als Yaml Datei definieren.
- Öffne also die simple-ec2.yaml **in diesem Repository** und kopiere den gesamten Inhalt (Strg + A, Strg + C)
- Öffne nun die gerade erstellte Datei `my-online-shop.yaml` mit dem Texteditor deiner Wahl (Ich empfehle VSCode oder vim)

```
# Für vscode
code .

# Für vim
vim my-online-shop.yaml
```

- Füge den kopierten Inhalt nun ein und speichere die Datei
- Jetzt kannst du im Git einen neuen Commit anlegen, dann kann in Zukunft ein Rollback zu diesem Zustand durchgeführt werden

```
git add .
git commit -m 'feat(infra): einfache ec2 instanz erstellt'
```

Initialisierung der AWS CLI

- Um die Infrastruktur nun auch deployen zu können, muss die aws-cli mit dem entsprechenden Access Key und Secret Key konfiguriert werden
- Dafür über den Broweser mit Google bei AWS (SSO) einloggen und eine Sandbox auswählen.

- Dann auf den Reiter 'Command line or programmatic access' klicken -> Access Key und Secret Key sollte sichtbar sein
- Jetzt kann aws cli konfiguriert werden

```
aws configure
# Access Key einfügen und ENTER Taste drücken
# Secret Key einfügen und ENTER Taste drücken
# Als Region eu-central-1 eingeben und ENTER Taste drücken
# Als letztes kann hier noch das bevorzugte Output Format angegeben werden
# ch empfehle 'json' eingeben und ENTER drücken

# Nun sollte die aws-cli konfiguriert sein
# Du kannst es mit folgendem Befehl Testen
aws sts get-caller-identity && echo 'Alles funktioniert' || echo 'Leider nicht
```

Infrastruktur deployen

- Um die Infrastruktur nun zu deployen, muss die CloudFormation Datei als neuer Stack deployed werden

```
aws cloudformation create-stack --stack-name my-online-shop --template-body fi
```

It works!

- Im output des letzten Befehls wird jetzt die öffentliche IP Addresse der EC2 Instanz angezeigt. Kopiere diese und füge sie in deinem Browser ein.
- Es kann sein, dass die Installation des Webservers bis zu einer Minute dauert, falls du eine Fehlermeldung bekommst.

Herzlichen Glückwunsch! Du hast erfolgreich eine Webandwendung via Infrastructure as Code deployed

Zusatzaufgabe

- Alle die noch mehr lernen wollen, sind eingeladen die folgenden Aufgaben auch zu bearbeiten.
- Diese sind allerdings wesentlich anspruchsvoller und viele Informationen müssen eigenständig erarbeitet werden!

Zuverlässigkeit (Reliability)

- Der aktuelle Zustand ist zwar eine Verbesserung im Vergleich zu einer manuel erzeugten EC2 Instanz, ist aber nicht Highly Available -> Wenn ein Fehler in unserer Instanz auftritt, ist die Applikation nicht ereichbar

- Das Ziel ist also das aktuelle CloudFormation Template so umzuschreiben, dass keine einfache EC2 Instanz mehr erstellt wird,sondern eine AutoScaling Group

AutoScaling Group

- Lese dir zum Verständnis folgendes Tutorial durch Tutorial
- Erstelle ein LaunchTemplate nach folgendem Beispiel:

```
MyLaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateName: MyLaunchtemplate
    LaunchTemplateData:
      ImageId: ami-00ad2436e75246bba
      InstanceType: t2.micro
      SecurityGroups:
      - !Ref MySecurityGroup
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash
          yum update -y
          yum install -y httpd
          systemctl start httpd
          systemctl enable httpd
```

- Erstelle eine AutoScaling Group nach folgendem Beispiel:

```
MyAutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    LaunchTemplate:
      LaunchTemplateId: !Ref MyLaunchTemplate
      Version: !GetAtt MyLaunchTemplate.LatestVersionNumber
    MinSize: 1
    MaxSize: 3
    DesiredCapacity: 2
```

Update the stack

- Jetzt kannst du deine Änderungen in git commiten und dann deployen

```
git add .
git commit -m 'feat(infra): autoscaling group hinzugefügt'
aws cloudformation update-stack --stack-name my-online-shop --template-body fi
```

Welche Komponente fehlt noch?

Um wirklich Highly Available und Durable zu sein, fehlt noch ein weiterer service. Welche ist es? Füge ihn in das template ein und deploye erneut

---