

Bereitstellung des Tweet-Dienstes in der Public-Cloud

- Allgemeine Vorbereitungen
 - Zugriff auf die Web-Oberfläche:
`https://console.aws.amazon.com`
→ Zugangsdaten per E-Mail erhalten
 - AWS-Credentials (`AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`) anlegen
→ `http://tinyurl.com/access-keys`, Reiter „Access Keys“
→ Nur einmal pro Gruppe anlegen
 - AWS-CLI einrichten
→ `export PATH=$PATH:/proj/i4mw/pub/aufgabe2/aws`
 - Key-Pair für SSH-Zugriff auf Instanz erzeugen
→ `https://console.aws.amazon.com/ec2/home?region=eu-west-1#s=KeyPairs`
- Tweet-Dienst in der Public-Cloud laufen lassen
 - JAR-Datei des Dienstes auf S3 hochladen
 - JAR-Datei: `/proj/i4mw/pub/aufgabe2/MWTweetService.jar`
 - Bucket anlegen und JAR-Datei hineinkopieren (über AWS-CLI: siehe F. 3-17)
 - Security-Group einrichten → Folie 4-2
 - Instanz (und Dienst) starten → Folie 4-3
 - Dienstlauffähigkeit testen → Folie 4-4

MW-Übung (WS17/18)

4-1

Anpassen einer Security-Group

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	:::0	

- Obligatorische Felder
 - Type: spezifiziert Standard-Port(-Range), z. B. SSH → 22, IMAP → 143
 - Protocol: z. B. TCP, UDP, ICMP
 - Port Range: abhängig vom Dienst / den Diensten in der Instanz (z. B. 80)
 - Source: IP-Adresse(n) in CIDR-Notation oder andere Security-Group
→ Obiger Screenshot zeigt Erweiterung der Security-Group `i4mw`, falls der (Tweet-)Dienst auf Port 80 laufen soll und von überall außerhalb (`0.0.0.0/0` bzw. `:::0`) darauf zugegriffen werden darf

MW-Übung (WS17/18)

4-2

Erstellen/Starten einer Instanz

- Über die Kommandozeile (hier mit exemplarischem Bucket `gruppe0-bucket`)

```
> aws ec2 run-instances --instance-type <inst-type> --image-id <img-id> \  
--key <key> \  
--subnet-id <net-id> \  
--security-group-ids <sg-id> \  
--user-data="group=gruppe0-bucket;jar=MWTweetService.jar;parameters=\  
mw.hybridcloud.MWTweetService http://\${I4MW_ADDRESS}:<port>/tweetservice"
```

 - Escapen von „\$“ vor `I4MW_ADDRESS` nötig, da diese Variable innerhalb der gestarteten Instanz und nicht bei obigem Aufruf ausgewertet werden soll
→ Beim Starten über Java-API oder die Web-Oberfläche nicht nötig
 - `<port>` (z. B. 80): Muss in der Security-Group freigegeben sein (siehe F. 4-2)
- Instanztyp (`<inst-type>`): `t2.nano`; Image-ID (`<img-id>`): `ami-9ca607e5`
- Schlüsselname (`<key>`): beim Erstellen selbst gewählt (z. B. `gruppe0-key`)
- `<net-id>`: Ermitteln der ID (SubnetID) eines VPC-Subnetzes z. B. über

```
> aws ec2 describe-subnets | grep -i subnetid
```
- `<sg-id>`: Ermitteln der ID (GroupID) der Security-Group `i4mw` z. B. über

```
> aws ec2 describe-security-groups | grep -B3 -i groupid
```

MW-Übung (WS17/18)

4-3

Testen der Dienstlauffähigkeit

- Einloggen per SSH

```
> ssh -i <private_key (e.g., gruppe0-key.pem)> ec2-user@<ip_address>
```

 - IP-Adresse z. B. über `aws ec2 describe-instances` ermitteln
 - Überprüfen, ob Java-Prozess läuft: `> ps axu | grep java`
 - Fehlersuche: Durchsuchen von `/var/log/i4mw-java.log` (nicht existent, falls Java-Prozess nicht gestartet werden konnte) und `/var/log/syslog`
- Direkter Zugriff über HTTP-Anfrage (hier: GET-Anfrage)

```
> curl http://<ip-address>:<port>/tweetservice
```
- Direkter Zugriff über den Web-Browser (hier mit ermittelter IP-Adresse `34.249.1.141` und Port 80):

```
{  
  "avgProcessTimePerTweet": 0,  
  "avgTweetLength": 0,  
  "avgWordCount": 0,  
  "langCounts": {},  
  "processedTweets": 0,  
  "topHashtags": [],  
  "topRepLies": [],  
  "topTweeter": [],  
  "trendingWords": [],  
  "tweetsPerSecond": 0  
}
```

MW-Übung (WS17/18)

4-4