Middleware – Cloud Computing – Übung

Hybride Cloud: AWS - Öffentliche Cloud

Wintersemester 2020/21

Michael Eischer, Laura Lawniczak, Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
www4.cs.fau.de

Lehrstuhl für Verteilte Systeme und Betriebssysteme

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Überblick

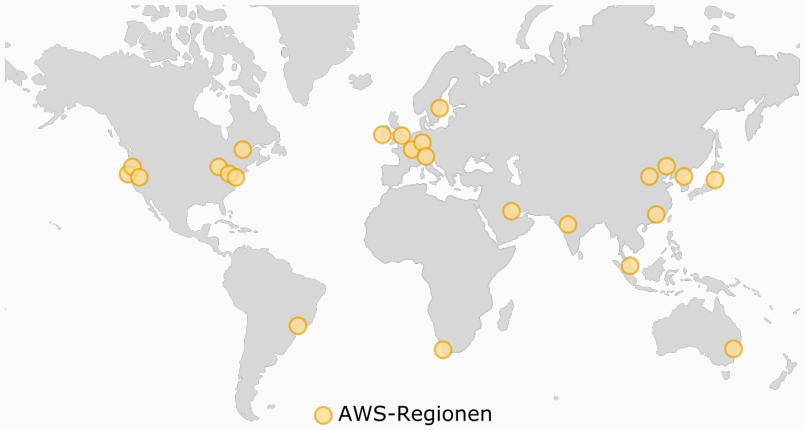
Amazon Web Services

Überblick

Amazon Web Services (AWS)

- Die Amazon Web Services bestehen aus Diensten, die den Aufbau komplexer Systeme in einer Cloud-Infrastruktur ermöglichen
- Dienste (Auszug):
  - Elastic Compute Cloud (EC2) – Betrieb virtueller Maschinen
  - Elastic Block Storage (EBS) – Bereitstellung VM-Abbilder und Datenträger
  - Simple Storage Service (S3) – Netzwerkbasierter Speicher-Dienst
  - CloudWatch – Überwachungsfunktionen für AWS-Dienste
- Die Abrechnung erfolgt nach tatsächlichem Verbrauch **und** Standort
  - Betriebsstunden, Speicherbedarf
  - Transfervolumen, Anzahl verarbeiteter Anfragen
  - Standorte in Nord- und Südamerika, Europa, Südafrika und Asien-Pazifik: `https://infrastructure.aws/`
  - Berechnung der Gesamtbetriebskosten: `https://calculator.aws/`

Amazon Web Services (AWS)



Amazon Web Services: Betriebsumgebung

- Benutzung der Amazon Web Services (u. a.) über Web-Oberfläche
  → `https://i4mw-gruppeXX.signin.aws.amazon.com/console`
  (XX durch eigene Gruppennummer ersetzen)
  ↪ Login-Informationen befinden sich in der Gruppeneinteilungs-E-Mail
  ↪ Immer die Region `eu-west-1` verwenden
- AWS CLI: AWS-Befehlszeilen-Schnittstelle

```
alias aws=/proj/i4mw/pub/aufgabe2/awscli/bin/aws
```

  - Python-Werkzeug zum Zugriff auf sämtliche AWS-Dienste
  - Alias-Befehl am besten in die Datei `~/.profile` eintragen, damit die AWS CLI nach jedem CIP-Pool-Login funktionieren
  - Konfiguration: Setzen der Zugangsdaten und Region. Siehe nächste Folie
- Liste der verfügbaren AWS-Kommandozeilen-Tools

```
> aws help
> aws <service> help
> aws <service> <command> help
```


Amazon Web Services: Betriebsumgebung

- Ablegen der Credentials zum API-Zugriff in Datei `~/.aws/credentials`
  → Automatische Verwendung durch Programme, welche auf die API zugreifen
  1) Anlegen der privaten Konfigurationsdateien `~/.aws/credentials` und `~/.aws/config` mit eingeschränkten Zugriffsrechten

```
> mkdir ~/.aws
> touch ~/.aws/credentials ~/.aws/config
> chmod 600 ~/.aws/*
```

  2) Erstellen von `aws_access_key_id` und `aws_secret_access_key` über die Web-Oberfläche:
  → `https://console.aws.amazon.com/iam/`
  → Menü „Users“, Namen anklicken, Reiter „Security Credentials“, Abschnitt „Access Keys“
  → Eintragen in `~/.aws/credentials`

```
[default]
aws_access_key_id = <schluessel_id>
aws_secret_access_key = <privater_schluessel>
```

  3) Setzen der Region in `~/.aws/config`

```
[default]
region = eu-west-1
```


Amazon Web Services

Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (EC2)

- Voraussetzungen für die Instanziierung einer virtuellen Maschine
  - Amazon Machine Image (AMI, Liste: `> aws ec2 describe-images`)
  - EC2-Schlüsselpaar
  - VPC-Netzwerk
- Bei der Instanziierung muss die Größe der virtuellen Maschine festgelegt werden
  - Instanz-Typen variieren in Anzahl der CPU-Kerne, Speichergröße etc.
    → `http://aws.amazon.com/ec2/instance-types/`
  - Für Testzwecke reicht der Betrieb kleiner Instanzen aus
    → API-Name: `t2.nano`
- Optionales Nutzdatenfeld `user-data`
  - Base64-kodierter String
  - Maximal 16 kByte

Amazon EC2: Starten einer Instanz — Vorbereitung

- Einmalig EC2-Schlüsselpaar im Browser generieren
  → `https://console.aws.amazon.com/ec2/home?region=eu-west-1#s=KeyPairs`
  - Schlüsselname wählen (z. B. gruppe0)
  - Privaten Schlüssel unter `~/.aws/gruppe0.pem` speichern
  - Zugriffsrechte mit `chmod` absichern

```
> chmod 600 ~/.aws/gruppe0.pem
```

- VPC-Netzwerk inklusive Subnetz nötig
  → Konfiguration (optional): `https://console.aws.amazon.com/vpc/home?region=eu-west-1`
  - Existiert bereits im zur Verfügung gestellten AWS-Account
- Security-Group für Port-Freigaben einrichten
  → `https://console.aws.amazon.com/ec2/home?region=eu-west-1#SecurityGroups`
  - Basis-Security-Group bereits im AWS-Account vorhanden (Name: `i4mw`)
  - **Achtung:** Erlaubt nur Kommunikation zwischen VMs in AWS

  ↪ Für SSH externe Zugriffe über das TCP-Protokoll mit Port 22 von `0.0.0.0/0` und `::/0` (CIDR-Notation, entspricht weltweitem Zugriff) freigeben!
  - Änderungen möglich während Instanz läuft

Amazon EC2: Starten einer Instanz

- Starten einer Linux-Instanz
  - Instanz-Typ: `t2.nano`
  - AMI: `ami-0bb3fad3c0286ebd5` [↪ Amazon Linux 2 AMI]
  - Schlüsselname (`<key>`): beim Erstellen selbst gewählt (z. B. `gruppe0`)
  - Nutzdatenfeld mit String füllen (`<user-data>`): z. B. `Hello World.`
  - `<subnet-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 --filters Name=group-name,Values=i4mw \
        | grep -i -e groupname -e groupid
```

- Starten über die Kommandozeile

```
> aws ec2 run-instances --instance-type t2.nano \
          --image-id ami-0bb3fad3c0286ebd5 \
          --key gruppe0 --user-data="<user-data>" \
          --subnet-id <subnet-id> \
          --security-group-ids <sg-id>
```


Amazon EC2: Zugriff auf eine Instanz

- Überprüfen des Status der Instanz mit `> aws ec2 describe-instances`

↪ Antwort enthält auch öffentliche IP-Adresse (`PublicIpAddress`)

- Sobald der Boot-Vorgang abgeschlossen ist, erfolgt der Zugriff auf die Instanz mittels SSH

```
> ssh -i ~/.aws/gruppe0.pem \
        ec2-user@ec2-xxx-xxx-xxx-xxx.eu-west-1.compute.amazonaws.com
```

- Bei Konflikten aufgrund erneuter Adressvergabe, alten SSH-Host-Key entfernen:

```
> ssh-keygen -R <server_address>
```

- Bei Zugriffsproblemen: Boot-Meldungen über die Web-Schnittstelle oder mit
  `> aws ec2 get-console-output --instance-id <id> --output text` nach Fehlern durchsuchen

↪ Richtiger Benutzername für SSH verwendet?

- Innerhalb der virtuellen Maschine
  - Abrufen von Meta-Informationen mit `> ec2-metadata`
  - Enthalten Nutzdatenfeld `user-data`

Amazon EC2: Beenden einer Instanz

- Zum Terminieren einer im Betrieb befindlichen Instanz ist die eindeutige Instanz-ID notwendig
- Das Kommando `> aws ec2 describe-instances` listet die `InstanceId` (Format: `i-xxxxxxxx`)
- Unter Kenntnis dieser ID kann die Instanz beendet werden:

```
> aws ec2 describe-instances
(...)
> aws ec2 terminate-instances --instance-ids i-xxxxxxxx
```

- Kontrolle: `https://console.aws.amazon.com/ec2/home`

Achtung!

Bitte stets sicherstellen,
dass **alle unbenutzten** Instanzen beendet (gelöscht) werden!

Amazon Web Services

Simple Storage Service (S3)

Amazon Simple Storage Service (S3)

- Der Simple Storage Service (S3) ist ein Netzwerk-Dateisystem
  - Einfache API
  - REST-Schnittstelle
  - Zugriffskontrolle mittels Zugriffskontrolllisten (Access Control Lists, ACLs)
- Eindeutige Identifikation von Dateien durch Bucket (Kübel) und Dateiname:
  `s3://<bucket>/<dateiname>`
- Kein hierarchischer Namensraum
  - Dateinamen mit Separator / möglich

  → Web-Konsole zeigt dies als Ordner an
- Übersetzung der S3-Adressrepräsentation in eine URL
  - S3: `s3://<bucket>/<dateiname>`
  - URL: `http://<bucket>.s3.amazonaws.com/<dateiname>`

Amazon S3: Zugriff auf Daten

- Zugriff auf Daten in S3 im CIP-Pool via
  `> aws s3 <befehl>`
  - `cp` / `rm` / `mv`
  - `mb` / `rb`
  - `ls`
  - …
- Erstellen eines Bucket:

```
> aws s3 mb s3://gruppe0-bucket
make_bucket: gruppe0-bucket
```

- Speichern einer *öffentlichen* Datei im Bucket `gruppe0-bucket`:

```
> echo "Hello World." > foo.bar
> aws s3 cp --acl public-read foo.bar s3://gruppe0-bucket/foo.bar
upload: foo.bar to s3://gruppe0-bucket/foo.bar
```

Amazon S3: Zugriff auf Daten

- Laden der Datei `foo.bar` aus dem Bucket `gruppe0-bucket`:

```
> aws s3 cp s3://gruppe0-bucket/foo.bar foo.bar.copy
download: s3://gruppe0-bucket/foo.bar to foo.bar.copy
```

- Löschen der Datei `foo.bar` aus dem Bucket `gruppe0-bucket`:

```
> aws s3 rm s3://gruppe0-bucket/foo.bar
delete: s3://gruppe0-bucket/foo.bar
```

- Ausführliche Liste mit Beschreibungen der `s3`-Befehle:

```
> aws s3 help
```

- Alternative Zugriffsmethoden:
  - Browser (Amazon Web Services Console, `https://console.aws.amazon.com/s3/home`)
  - Einhängen als Dateisystem (`s3fs`, FUSE-basiert)

Amazon Web Services

Amazon CloudWatch

Amazon CloudWatch

- Umfangreiche Überwachungsfunktionen für viele AWS-Dienste
- Protokollierung und lange Speicherung der Daten

- Beispiele
  - Amazon EC2: CPU-Auslastung, gesendete/empfangene Netzwerkpakete
  - Amazon EBS: Lese- und Schreiblatenz

- Metriken: Messwerte über Zeit
  - Metriken abfragen aber auch eigene Metriken einpflegbar
  - Minutengranularität möglich
  - Ältere Daten werden aggregiert und ausgedünnt
- Alarme: Automatische Reaktion bei auffälligen Veränderungen
- Visualisierung: Darstellung der Daten in einem Dashboard möglich
  `https://eu-west-1.console.aws.amazon.com/cloudwatch` → „Metriken“

Amazon CloudWatch: Metriken

- Metriken
  - Enthalten Messwerte mit Zeitstempeln (UTC)
  - Gruppiert in *Namensräume* wie `AWS/EC2`, `AWS/EBS`, `AWS/S3`, ...
  - *Dimensionen* zum Zuordnen von Datensätzen, z. B. per Instanz-ID

- Metriken für EC2 Instanzen
  - Grundlegende Überwachung (5 Minutenintervalle), kostenlos
  - Detaillierte Überwachung (1 Minutenintervalle), zusätzliche Kosten
  - Benutzerdefinierte Metriken: aus Anwendung heraus, selbst definierbar

- Abruf
  - Benötigt Start- und Endzeitpunkt sowie Aggregationszeitraum
  - Aggregation innerhalb eines Zeitraums (Period) per Minimum / Maximum / Durchschnitt / ...
  - Zeitraum muss gleich oder ein Vielfaches des Erzeugungsintervall sein
  - Möglicherweise verzögert verfügbare Daten

Amazon Web Services

Amazon Java SDK

Amazon Java SDK

- Amazon stellt Java-Bibliotheken für die Verwendung der Amazon Web Services bereit
  `/proj/i4mw/pub/aufgabe2/aws-java-sdk-2.15.11`
  → Dokumentation: `https://sdk.amazonaws.com/java/api/latest/`

- Java-Packages für den Betrieb virtueller Maschinen in Amazon EC2 und Amazon CloudWatch
  - `software.amazon.awssdk.services.ec2`
  - `software.amazon.awssdk.services.cloudwatch`

- Grundlegende Verwendung des SDK
  1. Initial: Client-Objekt (z. B. Typ `Ec2Client`) erstellen und gegenüber AWS authentifizieren
  2. Anfrageparameter in Anfrageobjekt (z. B. Typ `RunInstancesRequest`) setzen
  ↪ Objekte nicht modifizierbar, Erzeugung per Builder-Pattern
  3. Anfrage über Client-Objekt abschicken
  4. Gibt Ergebnisobjekt (z. B. Typ `RunInstancesResponse`) zurück, das Ergebnis der Anfrage enthält
  ↪ Ergebnisobjekt spiegelt *Zustand zum Zeitpunkt der Antwort* wider

Amazon Java SDK: Instanziierung einer VM

- Minimal-Beispiel (analog Kommandozeilen-Beispiel)
  Beachte: Vor dem Aufruf am `Ec2Client.Builder` müssen in der Konfigurationsdatei `~/.aws/credentials` die Optionen `aws_access_key_id` und `aws_secret_access_key` gesetzt sein.

- Initialisierung `software.amazon.awssdk.services.ec2, software.amazon.awssdk.regions`

```
Ec2Client ec2 = Ec2Client.builder()
    .region(Region.EU_WEST_1)
    .build();
```

- Setzen des Namens einer VM-Instanz `software.amazon.awssdk.services.ec2.model`

```
Tag tag = Tag.builder().key("Name").value("MyVMName").build();

TagSpecification spec = TagSpecification.builder()
    .tags(tag)
    .resourceType("instance")
    .build();

[...] // Fortsetzung auf der nächsten Folie
```


Amazon Java SDK: Instanziierung einer VM

- Minimal-Beispiel (Fortsetzung) `software.amazon.awssdk.services.ec2.model`

```
String userData = "Hello world.";
byte[] userDataBytes = userData.getBytes();
```

```
RunInstancesRequest request = RunInstancesRequest.builder()
    .imageId("ami-0bb3fad3c0286ebd5")
    .tagSpecifications(spec)
    .instanceType("t2.nano")
    .minCount(1)
    .maxCount(1)
    .keyName("gruppe0-key")
    .userData(Base64.getEncoder().encodeToString(userDataBytes)) // java.util.Base64
    // optional, detailliertere Metriken aktivieren
    .monitoring(RunInstancesMonitoringEnabled.builder().enabled(true).build())
    .securityGroupIds("sg-989f5ce3") // z.B. im Web-Interface erstellen
    .subnetId("subnet-0eab946a")     // (VPC muss Security-Group vorab zugeordnet werden)
    .build();
RunInstancesResponse response = ec2.runInstances(request);
```

- Hinweise:
  - Mittels des Objektes `response` die Instanz-ID in Erfahrung bringen
  - Auf die eigentliche Instanziierung prüfen (`DescribeInstancesRequest`)
  - Zwischen zwei Abfragen des Instanzstatus kurz warten

Amazon Java SDK: CloudWatch

- Initialisierung (ähnlich wie bei EC2) `software.amazon.awssdk.services.cloudwatch`

```
CloudWatchClient cw = CloudWatchClient.builder()
    .region(Region.EU_WEST_1).build();
```

- Metrik abrufen: Zeitintervall und Dimension festlegen
  - Erwartetes Zeitformat: ISO 8601, UTC (z. B. 2020-11-25T09:00:00Z)
  - Beispielhaftes Definieren von Anfangs- und Endzeitpunkt

```
// Packages: java.time.Clock, java.time.Instant
Instant endTime = Clock.systemUTC().instant();
Instant startTime = endTime.minusSeconds(120); // Datenpunkte ueber 2-Min.-Intervall

// Package: software.amazon.awssdk.services.cloudwatch.model.Dimension
Dimension dimension = Dimension.builder()
    .name("InstanceId")
    .value("i-xxxxxxx")
    .build();
```

- Weiterführende Links
  - https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_concepts.html
  - https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/API_GetMetricStatistics.html
  - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/viewing_metrics_with_cloudwatch.html

Amazon Java SDK: CloudWatch

- Metrik abrufen (Fortsetzung) `software.amazon.awssdk.services.cloudwatch.model`

```
// Request zum Holen der Werte einer Metrik zusammensetzen und absenden
// festlegen, dass nur Durchschnittswerte abgefragt werden
GetMetricStatisticsRequest req = GetMetricStatisticsRequest.builder()
    .statistics(Statistic.AVERAGE)
    .metricName("NetworkIn")
    .dimensions(dimension)
    .namespace("AWS/EC2")
    .period(60)
    .startTime(startTime)
    .endTime(endTime)
    .build();
GetMetricStatisticsResponse res = cw.getMetricStatistics(req);

// Zeitstempel und Durchschnittswerte ausgeben
for (Datapoint dp : res.datapoints()) {
    System.out.printf("%s: %s\n", dp.timestamp(), dp.average());
}
```