

Middleware – Cloud Computing – Übung

Aufgabe 2: Hybrid Cloud

Wintersemester 2021/22

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

Hybride Cloud

- Hybride Cloud & Virtualisierung

- Aufgabe 2

Amazon Web Services

- Überblick

- Elastic Compute Cloud (EC2)

- Simple Storage Service (S3)

- Amazon CloudWatch

- Amazon Java SDK

OpenStack

Erstellen eines VM-Abbilds in OpenStack

- Cloud Computing Software-Infrastruktur

- Erstellen des Abbilds für OpenStack

- Betrieb der virtuellen Maschine

Hybride Cloud

Hybride Cloud & Virtualisierung

- Öffentliche Cloud: Cloud-Dienste frei für jeden verfügbar
 - *aaS: „X as a Service“-Gedanke
 - Scheinbar unbegrenzte Ressourcen
- Private Cloud: IT- bzw. Cloud-Dienste werden z. B. von einem Unternehmen oder einer Einrichtung selbst betrieben
 - Interne Nutzung: Datenschutz und IT-Sicherheit
 - Aber auch: Bereitstellung von eigenen Ressourcen für öffentliche Nutzung
- **Hybride Cloud:** Mischform aus privater und öffentlicher Cloud
 - Sicherheitskritische Teile einer Anwendung laufen nur in der privaten Cloud
 - Skalierbarkeit, Ausdehnung auf öffentliche Cloud (z. B. beim Auftreten von Lastspitzen)

■ Notwendige Betriebsmittel

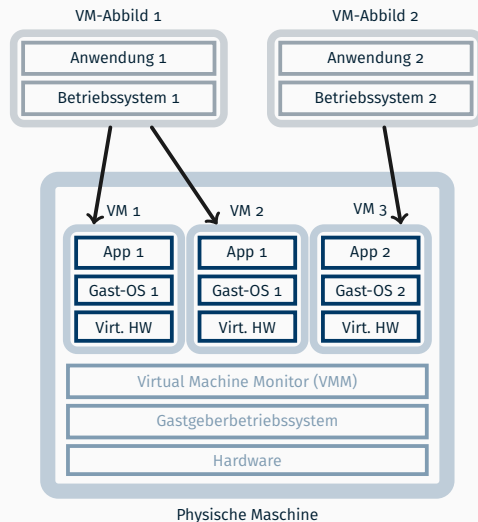
- Physische Maschine und Gastgeberbetriebssystem („Host“)
- Virtualisierungssoftware, die den Virtual Machine Monitor bereitstellt
- **Abbild der virtuellen Maschine**

■ Analogie zur Objektorientierung

- Das statische Abbild einer virtuellen Maschine entspricht einer **Klasse**
- Eine im Betrieb befindliche virtuelle Maschine ist die **Instanz** eines solchen Abbilds

■ Aufbau des Abbilds einer virtuellen Maschine

- **Dateisystem**, beinhaltet für gewöhnlich:
 - Kern des Gastbetriebssystems („Guest“)
 - User-Space-Komponenten des Gastbetriebssystems
 - Anwendung
- Meta-Informationen (VMM-spezifisch)

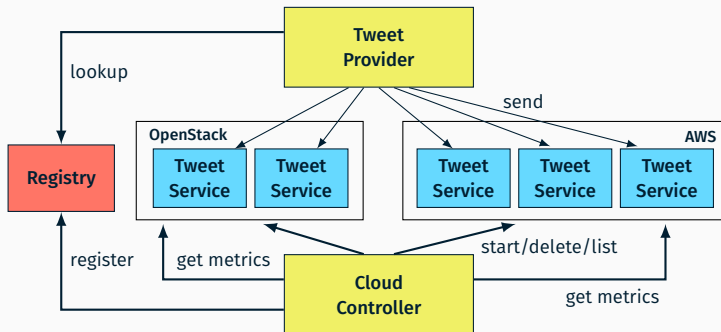


Hybride Cloud

Aufgabe 2

Aufgabe 2: Hybride Cloud

- Bereitgestellten Tweet-Service in hybrider Cloud ausführen
 - Ein bis maximal zwei VMs in privater Cloud
 - Öffentliche Cloud für Lastspitzen
- Teilaufgaben
 - Cloud-Controller für manuelle **Cloud-Ansteuerung** (VMs starten, beenden, auflisten)
 - **Lastverteilung** für Tweet-Anfragen im Provider, VMs per Registry abfragen
 - Erweiterter Cloud-Controller zur **dynamischen Skalierung** der VMs (nur 7.5 ECTS)



Aufgabe 2: Hybride Cloud

- Public Cloud: Amazon Web Services
 - Limitiertes Guthaben: Rund 10 US-Dollar Guthaben pro Gruppe
 - Guthaben kann lediglich für Amazon Web Services verwendet werden
 - Aktuelle AWS-Kosten: <http://aws.amazon.com/pricing/>
- Globaler Systemstatus der Amazon Web Services
 - Bei Störungen können (Teile der) Amazon Web Services ausfallen
 - Aktueller Status: <http://status.aws.amazon.com/>
- Private Cloud: OpenStack-Umgebung des Lehrstuhls
 - Ressourcen der drei Node-Controller sind **beschränkt**
 - Jederzeit auf faire Verwendung achten
- OpenStack-Infrastruktur
 - Bitte sendet bei Problemen oder Ungereimtheiten schnellstmöglichst eine E-Mail an i4mw-owner@lists.cs.fau.de

Achtung!

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

Aufgabe 2: Hybride Cloud

- Gemeinsame Schnittstelle: MWCloudPlatform
 - Instanzen starten / beenden / auflisten
 - Metriken der Instanzen abrufen
- MWCloudPlatformAWS: Betrieb des Dienstes in AWS EC2
 - Java 11 & Java-Bibliotheken bereits in vorkonfiguriertem Image (ami-09b7461854552e741) enthalten
 - Passende Konfigurationsparameter userdata übergeben
 - Metriken aus AWS CloudWatch abfragen
- MWCloudPlatformOpenStack: Betrieb des Dienstes in OpenStack Nova
 - Erzeugung und Konfiguration eines eigenen VM-Abbilds
 - Installation des Grundsystems
 - Hinzufügen von Java, Java-Bibliotheken für Dienst
 - ↪ Schritt-für-Schritt Anleitung im Übungsteil „Erstellen eines VM-Abbilds in OpenStack“
 - Metriken aus Gnocchi abfragen
- Hinterlegen des JAR-Archivs des Tweet-Service auf AWS S3

Testen der Dienstlauffähigkeit

- Direkter Zugriff über HTTP-Anfrage (hier: GET-Anfrage)

```
> curl http://<ip-address>:<port>/tweetservice
```

→ Innerhalb der VM unter localhost erreichbar

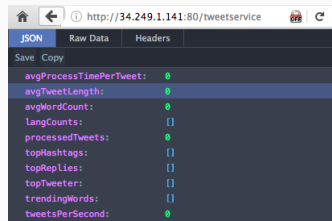
- Direkter Zugriff über den Web-Browser möglich
- Instanz nicht erreichbar?

→ Konfiguration der cloudseitigen Firewall
durch die Security Groups kontrollieren

- Logs per SSH einsehen

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

- Benutzername für AWS: ec2-user, für OpenStack: cloud
- Bei Anmeldeproblemen Benutzernamen und SSH-Key kontrollieren
- Überprüfen, ob Java-Prozess läuft: > ps aux | grep java
oder > sudo systemctl status i4mw-service
- Fehlersuche: Protokolle durchsuchen mit > sudo less /var/log/syslog
oder > sudo journalctl -u i4mw-service

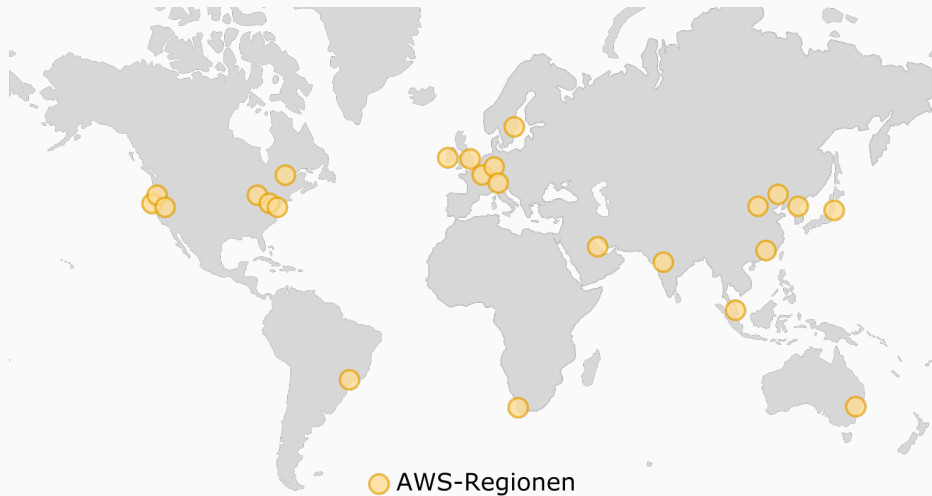


Amazon Web Services

Überblick

- 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)



- 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
```

■ 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)

- 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

■ 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

■ Starten einer Linux-Instanz

- Instanz-Typ: t2.nano
- AMI: ami-05cd35b907b4ffe77 [↔ 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-05cd35b907b4ffe77 \  
    --key gruppe0 --user-data="<user-data>" \  
    --subnet-id <subnet-id> \  
    --security-group-ids <sg-id>
```

- 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`

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

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

- 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)

- 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>`

■ 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
```


- 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

- 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“

■ Metriken

- Enthalten Messwerte mit Zeitstempeln (UTC)
- Gruppieren 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 stellt Java-Bibliotheken für die Verwendung der Amazon Web Services bereit
/proj/i4mw/pub/aufgabe2/aws-java-sdk-2.17.46
→ 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

■ 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
```

■ Minimal-Beispiel (Fortsetzung)

software.amazon.awssdk.services.ec2.model

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

```
RunInstancesRequest request = RunInstancesRequest.builder()
    .imageId("ami-05cd35b907b4ffe77")
    .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

■ 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

■ 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());
}
```

OpenStack

- Web-Frontend
 - Dashboard: <https://i4cloud1.cs.fau.de>
 - Zugangsdaten: siehe Gruppeneinteilungs-E-Mail
- Kommandozeilen-Client
 - OpenStack-Client-Programm: `openstack`
 - **Vor Verwendung:** `openrc`-Datei sourcen (siehe unten)
- Alle Kommandozeilenbefehle benötigen vorherige Authentifizierung
 - 1) Download der RC-Datei (`<user>-openrc.sh`) über Dashboard:
 - „Projekt“ → „API Access“
 - „Download OpenStack RC File“
 - 2) RC-Datei einlesen und ausführen (sourcen)

```
$ source /path/to/<user>-openrc.sh
```

- Benutzerdaten für Login per OpenStack-Konsole auf einer laufenden Instanz des bereitgestellten Beispielabbilds (`debian-example`):
USER: `cloud` PASSWORD: `cloud`

■ OpenStack4j: Java-API für OpenStack-Dienste

- Bibliotheken: /proj/i4mw/pub/aufgabe2/openstack4j-3.10
- Dokumentation: <https://openstack4j.github.io/learn>

■ Authentifizierung

```
// Package: org.openstack4j.model.common
Identifier userDomainName = Identifier.byName(<user_domain_name>);
Identifier projectIdentifier = Identifier.byId(<project_identifier>);
OSClientV3 client = OSFactory.builderV3() // Packages:
    .endpoint(<os_auth_url>) // org.openstack4j.{api,openstack}
    .credentials(<user>, <pass>, userDomainName)
    .scopeToProject(projectIdentifier)
    .authenticate();
```

- Parameter in OpenStack RC-Datei
 - Benutzer-Domänen-Name (<user_domain_name>): Variable OS_USER_DOMAIN_NAME
 - Projekt-ID (<project_identifier>): Variable OS_PROJECT_ID
 - Endpunkt-Adresse (<os_auth_url>): Variable OS_AUTH_URL
- Benutzername (<user>) und Passwort (<pass>): siehe E-Mail zur Gruppeneinteilung
- OSClientV3 ist an Thread gebunden → Neuen Client für anderen Thread per OSFactory.clientFromToken(client.getToken()) erzeugen

■ Konfiguration (ähnlich zu AWS-API) über ServerCreate-Objekt

```
ServerCreate sc = Builders.server() // org.openstack4j.model.compute.api  
    .<config_option1>  
    .<config_option2>[...].<config_optionN>.build();
```

- Konfigurieren von Instanzname, Instanztyp (Flavor-**ID**), Abbild-**ID**, Keypair, Netzwerk-**ID**, Security-Group, UserData (Kodierung mittels `java.util.Base64`)
- Ersteinrichtung: Siehe Übung zum „Erstellen eines VM-Abbilds für OpenStack“

■ Boot mit Konfiguration (Aufruf blockiert, bis VM aktiv ist)

```
Server server = client.compute().servers()  
    .bootAndWaitActive(sc, <max_wait_time_in_ms>);
```

■ Statusabfrage

`org.openstack4j.model.compute.Server.Status`

```
String serverId = server.getId();  
Status st = client.compute().servers().get(serverId).getStatus();
```

- VM hat initial nur interne IP

→ Zugriff von extern nur mit Floating-IP möglich

- Floating-IP zuweisen

org.openstack4j.model.{compute,common}

```
List<? extends FloatingIP> ips = client.compute().floatingIps().list();
FloatingIP floatingIp = ips.get(0);
// [...] unbenutzte IP mit (floatingIp.getInstanceId() == null) suchen
ActionResponse r = client.compute().floatingIps().addFloatingIP(server,
    floatingIp.getFloatingIpAddress());
```

- Floating-IP abfragen

org.openstack4j.model.{compute,common}

```
String publicIp = "";
List<? extends Address> vmAddresses = server.getAddresses().getAddresses("internal");
for (Address address: vmAddresses) {
    if (address.getType().equals("floating") && address.getVersion() == 4) {
        publicIp = address.getAddr();
        break;
    }
}
```

Zugriff auf Metriken in OpenStack mittels Gnocchi

- Datenabruf per REST-Anfragen
 - Zugriff über WebTarget-Objekt
 - Dokumentation: <https://gnocchi.xyz/rest.html>
- Gnocchi-Endpoint-URL (Servicetyp „Metric“) im Dashboard unter „API Access“ nachschlagen
- Oder Ermitteln der Endpoint-URL mittels der Dienstliste von OpenStack

```
List<? extends Service> catalog = client.identity().tokens().getServiceCatalog(client.getToken().getId())
```

→ Öffentlichen (Public) Endpoint des Servicetyps „Metric“ verwenden

- Authentifizierung bei Gnocchi-Anfragen erfolgt per HTTP-Header (Schlüssel-Wert-Paare)
 - Für *alle* Anfragen notwendig
 - Schlüssel (<key>): „X-Auth-Token“
 - Wert (<value>): Token von OpenStack anfordern

```
String authToken = client.getToken().getId();
```

- Header-Modifikation bei REST-Anfragen

```
Response r = target.request().header(<key>, <value>).post(Entity.text("test"));
```


- Instanz-spezifische ID einer Metrik (z. B. `cpu`) ermitteln (im Folgenden: `<metric-id>`)

→ GET-Anfrage auf Pfad listet alle Metriken auf:

`<Gnocchi-URL>/v1/resource/instance/<vm-id>`

- Rückgabe der Ergebnisse erfolgt im JSON-Format
- Datentyp: `MWGnocchiInstanceResource`

- Messwerte für eine bestimmte Metrik abfragen

→ GET-Anfrage auf Pfad:

`<Gnocchi-URL>/v1/metric/<metric-id>/measures?start=<time>&granularity=10&aggregation=rate:mean`

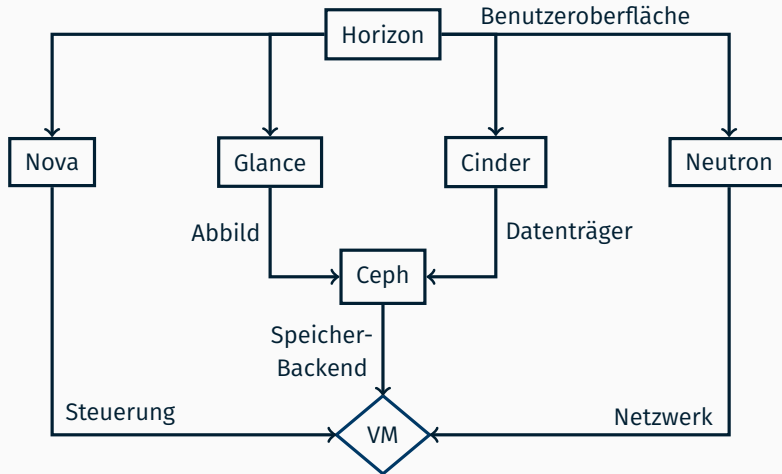
- „`<time>`“: Zeitstempel (analog zu CloudWatch) **oder** relative Zeitangabe, z. B. „`-30seconds`“
- „`granularity=10`“: Jeweils über 10 Sekunden aggregierte Datenpunkte abrufen
 - OpenStack Ceilometer sammelt bei uns alle 10 Sekunden neue Daten
 - Mögliche Aggregationszeiträume: 10 / 60 / 3600 Sekunden
- „`aggregation=rate:mean`“: Durchschnitt über Aggregationszeitraum
- Datentyp: `String[][]`; pro Array-Element: Zeitstempel, Aggregationszeitraum, Wert

- CPU-Metrik gibt akkumulierte Rechenzeit zurück

- CPU-Verbrauch des aktuellen Aggregationszeitraums in Nanosekunden
- CPU-Auslastung: Messwert / Aggregationszeitraum
beispielsweise $7\,000\,000\,000\,ns / 10\,000\,000\,000\,ns = 70\%$

Erstellen eines VM-Abbilds in OpenStack

Cloud Computing Software-Infrastruktur



- **Nova:** Verwaltung virtueller Maschinen
 - Compute: Steuerung von VMs (QEMU/Xen/...) auf Rechnern
 - Scheduler: Verteilung auf verfügbare Hardware
- **Glance:** Bereitstellung von Abbildern
 - Registry: Metadaten für Images
 - API unterstützt verschiedene Speichersysteme
- **Cinder:** Bereitstellung von Volumes
 - Volume-Service: Lokale Datenhaltung
 - Scheduler: Verteilung der Daten(-transfers) auf Rechner
- **Ceph:** Verteiltes Speicher-Backend für Glance und Cinder
- **Neutron:** Netzwerkmanagement und virtuelle Router
 - Server: Steuerung und Zustandsverwaltung
 - Agents: Helfer für DHCP, Open vSwitch, Metadaten
- **Horizon** (Dashboard): Weboberfläche für Anwender
- **REST-Schnittstelle:** API-Dienst je Komponente
→ Kommandozeilentools / SDK
- **RabbitMQ:** Interne Kommunikation der Dienste über Nachrichtenbus

Erstellen eines VM-Abbilds in OpenStack

Erstellen des Abbilds für OpenStack

- Ziel: Verlagerung der Übungsaufgabe in eine virtuelle Maschine
- Abbild innerhalb von OpenStack erzeugen
 - Starten einer Instanz des Linux-Live-System Grml (<http://grml.org>)
 - Neues Volume anlegen und einhängen
 - Betriebssysteminstallation
 - Anpassen der Konfiguration; Installieren zusätzlicher Softwarepakete
 - Umwandeln in Image
- Abbild starten
 - Öffentlichen SSH-Schlüssel für passwortlose Authentifizierung hinterlegen
 - Instanz mit eigenem Image starten
 - Öffentliche IP konfigurieren
 - Übungsaufgabe in der Cloud laufen lassen
- Speicherarten
 - Volume: Veränderbar, Verwendung nur in einer Instanz
 - Image (Abbild): Nicht veränderbar, Basis für viele Instanzen

■ Web-Frontend

- Dashboard: <https://i4cloud1.cs.fau.de>
- Zugangsdaten: siehe E-Mail mit Zugangsdaten

■ Kommandozeile

- OpenStack-Client-Programm: `openstack`
- **Vor Verwendung:** `openrc`-Datei sourcen (siehe unten)

■ Alle Kommandozeilenbefehle benötigen vorherige Authentifizierung

- 1) Download der RC-Datei (`<user>-openrc.sh`) über Dashboard:
→ „Projekt“ → „API Access“ → „Download OpenStack RC File“
- 2) RC-Datei einlesen und ausführen (sourcen)

```
$ source /path/to/<user>-openrc.sh
```

CIP

- Name für Instanz festlegen
- Instanztyp `i4.grml`
 - Kein Swap/Ephemeral-Volume
- Booten vom bereitgestellten Grml-Image (GRML-2021.07-amd64)
 - Kein zusätzliches Volume erzeugen
- Zugriff auf internes Netzwerk
- Weboberfläche: siehe nächste Folie
- Kommandozeile:

```
$ openstack image list      # --> grml id
$ openstack network list   # --> internal net id
$ openstack server create --flavor i4.grml \
    --image <grml id> \
    --nic net-id=<internal net id> \
    grml-instance
```

CIP

Grml-Instanz starten

Launch Instance

Details

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Source

Instance Name

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Total Instances (10 Max)

10%

0 Current Usage
1 Added
9 Remaining

Availability Zone

eu-central-1

Count

Cancel

Back

Next

Launch Instance

Launch Instance

Details

Instance source is the template used to create an instance. You can use an image, a snapshot of an instance (image snapshot), a volume or a volume snapshot (if enabled). You can also choose to use persistent storage by creating a new volume.

Source

1 Select Root Source

2 Create New Volume

Allocated

Displaying 1 item

Name	Updated	Size	Type	Visibility
> grml-2023-08	10/20/23 9:02 PM	747.25 MB	RAW	Public

Displaying 1 item

Available

Select one

Click here for filters or full text search

Displaying 1 item

Name	Updated	Size	Type	Visibility
> debian-example	10/20/23 5:43 PM	2.05 GB	RAW	Public

Displaying 1 item

Cancel

Back

Next

Launch Instance

Launch Instance

Details

Flavors manage the sizing for the compute, memory and storage capacity of the instance.

Source

Allocated

Flavor

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
> H4.grml	1	768 MB	1 GB	1 GB	0 GB	Yes

Available

Select one

Click here for filters or full text search

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
> H4.tiny	1	512 MB	3 GB	3 GB	0 GB	Yes
> H4.docker	1	1 GB	10 GB	10 GB	0 GB	Yes

Cancel

Back

Next

Launch Instance

Launch Instance

Details

Networks provide the communication channels for instances in the cloud.

Source

Allocated

Flavor

Network	Subnets Associated	Shared	Admin State	Status
> 1 Internal	Internal-pool	Yes	Up	Active

Available

Select at least one network

Click here for filters or full text search

Network	Subnets Associated	Shared	Admin State	Status
No available items				

Cancel

Back

Next

Launch Instance

Volume erzeugen/einhängen

Project / Volumes / Volumes

Volumes

Filter

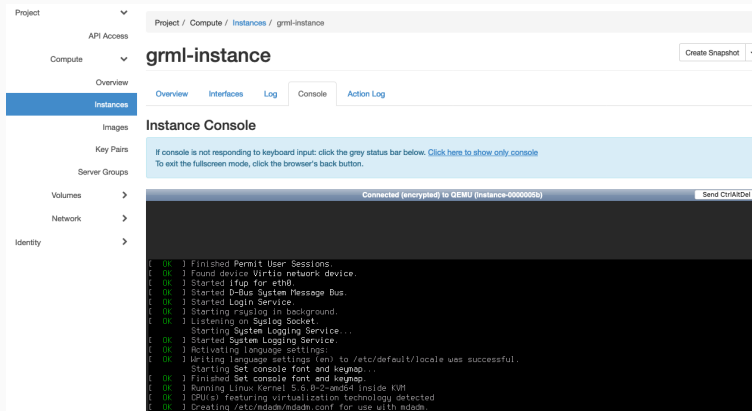
Displaying 1 item

☐	Name	Description	Size	Status	Group	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
☐	my-vol-name	-	2GiB	Available	-	__DEFAULT__		nova	No	No	Edit Volume Extend Volume Manage Attachments Create Snapshot Change Volume Type

Displaying 1 item

- (1) Leeres Volume anlegen, benötigt Name und Größe (2 GB)
- (2) Volumegröße kontrollieren
- (3) Volume der laufenden Instanz zuweisen
- Kommandozeile (Volume-Größe: 2 GB):

```
$ openstack volume create --size 2 my-vol-name # --> vol ID
$ openstack server add volume grml-instance <vol id>
```



- Konsole der laufenden Instanz im Dashboard öffnen
- Einrichtung des Betriebssystems und Installation der Java-Laufzeitumgebung im weiteren Verlauf der Übung

- Um als Basis für eine virtuelle Maschine zu dienen, muss das Abbild eine bootbare Partition mit Dateisystem beinhalten
- Mit parted lässt sich eine Partitionstabelle erstellen, was eine der Voraussetzungen ist, um das Abbild später booten zu können:

```
> parted /dev/vdb -s 'mktable msdos' 'mkpart primary 1MiB -1s' print
```

GRML

- Das Kommando mkfs (**m**ake **f**ile**s**ystem) erzeugt Dateisysteme, der Parameter -t spezifiziert dabei den Dateisystemtyp
- Erstellen eines ext4-Dateisystems mit der Bezeichnung „VM-Abbild“ auf dem blockorientierten Gerät (block device) /dev/vdb1:

```
> mkfs -t ext4 -L "VM-Abbild" /dev/vdb1
```

GRML

Installation der User-Space-Komponenten des zukünftigen Gastbetriebssystems in das neu erzeugte, leere Dateisystem:

1. Einhängen des zuvor erstellten Dateisystems mit mount:

```
> mount /dev/vdb1 /mnt
```

GRML

Kontrolle:

```
> mount | grep vdb1
```

GRML

2. Erstellung der User-Space-Komponenten des Zielsystems mit debootstrap:

```
> debootstrap bullseye /mnt/ 'http://ftp.fau.de/debian'
```

GRML

Kontrolle:

```
> ls -alR /mnt | more
```

GRML

3. Setupskript mittels wget herunterladen und ausführbar machen:

```
> wget https://i4mw.cs.fau.de/openstack/post-debootstrap.sh -O /mnt/post-debootstrap.sh  
> chmod +x /mnt/post-debootstrap.sh
```

GRML

- Jeder Linux-Prozess besitzt ein Wurzelverzeichnis (/)
 - Zugriff auf Daten außerhalb des Wurzelverzeichnisses ist **nicht** möglich
 - Kindprozesse erben das Wurzelverzeichnis ihres Elternprozesses (fork(2))
- Beispiel-Code jail.c:

```
int main(int argc, char *argv[])
{
    /* Starte Kindprozess (/bin/bash) nach erfolgreichem
    Wechsel des Wurzelverzeichnisses */
    if (chroot("/mnt/") == 0) {
        execl("/bin/bash", NULL);
    }

    return 0;
}
```

- Die Datei /mnt/bin/bash des Live-Systems entspricht der Datei /bin/bash des Kindprozesses nach Aufruf von chroot(2)

- Weitergeben von `/dev` ins `chroot` (notwendig für die Installation von GRUB (Bootloader) im `post-debootstrap.sh`-Skript)

```
> mount -o bind /dev /mnt/dev
```

GRML

- Wechsel in das von `debootstrap` erstellte System mittels `chroot(8)`

```
> chroot /mnt /bin/bash
```

GRML

↪ **Hinweis:** Sämtliche **Änderungen** an dem von `debootstrap` erstellten System in der `chroot`-Umgebung sind **persistent**

- Aufruf des `post-debootstrap.sh`-Skriptes (siehe Aufgabenstellung) für grundlegende VM-Abbild-Konfiguration in der `chroot`-Umgebung und Setzen des Passworts für User `ccloud`

```
# sh post-debootstrap.sh
Setting up /etc/apt/sources.list
(...)
Please set a password for user 'ccloud'.
```

CHROOT

```
# passwd ccloud
```

CHROOT

- Ergänzen der Software des Grundsystems mittels apt-get
- Aktualisieren der Paketquellen (update) und anschließendes Einspielen potentiell vorhandener Updates (upgrade)

```
# apt-get update  
# apt-get upgrade
```

CHROOT

- Das Kommando apt-get install löst Abhängigkeiten auf und installiert die entsprechenden Pakete, apt-get clean löscht Caches

```
# apt-get install <paket1> <paket2> ... <paketn>  
# apt-get clean
```

CHROOT

- Für die Übung sind noch folgende Pakete nötig oder nützlich:

```
openssh-server openjdk-11-jdk-headless screen vim-nox
```

CHROOT

■ Installation benötigter Bibliotheken

```
# mkdir -p /proj/lib  
# wget https://i4mw.cs.fau.de/openstack/libs.tgz -O libs.tgz  
# tar -xvf libs.tgz -C /proj/lib  
# rm libs.tgz
```

CHROOT

■ Automatisches Starten der Dienste

- Beim Systemstart führt `systemd(1)` die Init-Skripte aus
- Bereitgestelltes Startskript `/etc/systemd/system/i4mw-service.service`
 1. Wertet Konfigurationsdaten (user-data) aus; siehe Aufgabenstellung
 2. Lädt jar-Datei mit der Anwendung aus S3 herunter
 3. Startet die Anwendung mit den angegebenen Parametern

■ Hilfestellung zum Debugging

- Ausgabe im Log der VM-Instanz beachten (per Dashboard einsehbar)
- Ausgabe ist innerhalb der VM-Instanz im Syslog verfügbar

```
$ sudo less /var/log/syslog
```

VM

- Nur die Ausgaben des Dienstes `i4mw-service` anzeigen

```
$ sudo journalctl -u i4mw-service
```

VM

- Shell beenden, um chroot-Umgebung zu verlassen

```
# exit
```

CHROOT

- Grml-Live-Umgebung herunterfahren

```
> shutdown now
```

GRML

- Eingehängte Dateisysteme werden automatisch ausgehängt
- Stellt sicher, dass alle Änderungen geschrieben wurden

- Volume aushängen

- Per Dashboard: „Volumes“ → „Manage Attachments“ → „Detach Volume“
- Per Kommandozeile:

```
$ openstack server remove volume grml-instance <vol-id>
```

CIP

- Abbild erzeugen

- Per Dashboard: „Volumes“ → „Upload to Image“, Imagenamen eingeben, Disk format auf `raw` setzen
- Per Kommandozeile:

```
$ openstack image create --disk-format raw --volume <volume_id> <image_name>
```

CIP

Erstellen eines VM-Abbilds in OpenStack

Betrieb der virtuellen Maschine

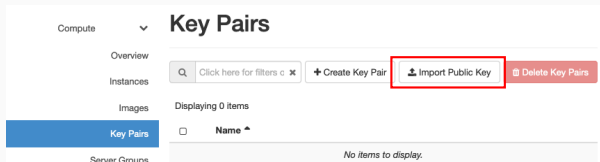
SSH-Schlüssel einrichten (einmalig)

- Privaten und öffentlichen Schlüssel mit `ssh-keygen` auf einem CIP-Pool-Rechner erzeugen

```
$ ssh-keygen -f ~/<gruppen_name> -N ""  
Generating public/private rsa key pair.  
Your identification has been saved in <gruppen_name>.  
Your public key has been saved in <gruppen_name>.pub.  
(...)
```

CIP

- Neu erstellten **öffentlichen Schlüssel** (<gruppen_name>.pub) hinzufügen unter „Compute“ → „Key Pairs“ → „Import Public Key“



- Kommandozeile:

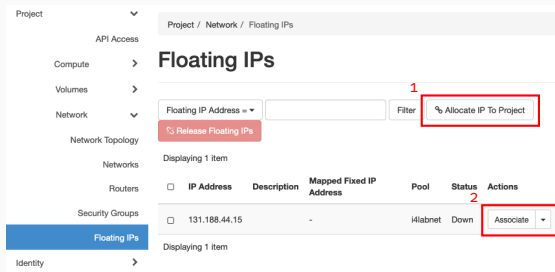
```
$ openstack keypair create --public-key <gruppen_name>.pub <schluessel_name>
```

CIP

- Instanztyp `i4.tiny`
 - Erzeugt Swap-Disk und vergrößert root-Partition
 - Von eigenem Abbild starten
 - SSH-Schlüssel unter „Key Pair“ auswählen
- ↪ Schlüssel wird beim Instanzstart nach `/home/cloud/.ssh/authorized_keys` kopiert
- Kommandozeile: (Schlüsselübergabe mittels Parameter `--key-name`)

```
$ openstack network list # --> internal net id
$ openstack keypair list # --> schluessel_name
$ openstack server create --flavor i4.tiny \
  --image <image name> \
  --nic net-id=<internal net id> \
  --key-name <schluessel_name> \
  my-vm-instance
```

CIP



- (1) Öffentliche IP aus Pool allokieren, **nur einmalig nötig**
- (2) IP-Adresse an laufende Instanz zuweisen
- Kommandozeile:

```
$ openstack floating ip create i4labnet  
$ openstack server add floating ip my-vm-instance <erhaltene IP>
```

CIP

- Abfrage innerhalb laufender VM per REST-API:

```
$ curl http://169.254.169.254/latest/meta-data/public-ipv4
```

VM

The screenshot shows the OpenStack dashboard interface. On the left is a sidebar with navigation links: Project, API Access, Compute, Volumes, Network, Network Topology, Networks, Routers, Security Groups (highlighted in blue), and Floating IPs. The main content area has a breadcrumb 'Project / Network / Security Groups' and a title 'Security Groups'. Below the title are a search filter, a '+ Create Security Group' button, and a 'Delete Security Groups' button. A table lists security groups, with one item displayed: 'default' with ID '085fcad8-b092-42e8-b3d9-084a97927614' and description 'Default security group'. The 'Actions' column for this group contains a 'Manage Rules' button, which is highlighted with a red rectangular box.

- TCP-Ports müssen für öffentlichen Zugriff freigegeben werden
- Kommandozeile, z. B. für TCP-Port 22 (SSH):

```
$ openstack security group rule create default \  
  --ingress --src-ip 0.0.0.0/0 \  
  --protocol tcp --dst-port 22
```

CIP

Add Rule

Rule ^{*}

Custom TCP Rule

Description [?]

SSH

Direction

Ingress

Open Port ^{*}

Port

Port ^{*} [?]

22

Remote ^{*} [?]

CIDR

CIDR ^{*} [?]

0.0.0.0/0

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Cancel

Add

Ingress = Eingehende Verbindungen, Egress = Ausgehende Verbindungen

- Passwortloser Zugriff mit SSH

```
$ ssh -i <gruppen_name> cloud@<instanz_ip>
```

CIP

→ Schlüssel aus der Rechnerübung, Instanz-IP aus vorheriger Zuweisung

- Wechsel von VM-Images erfordert evtl. Zurücksetzen von Host-Key

```
$ ssh-keygen -R <instanz_ip>    # Alten Host-Key entfernen
```

CIP

- Instanzen beenden: „Terminate“ auf der Weboberfläche, oder

```
$ openstack server list    # id heraussuchen  
$ openstack server delete <instanz id>
```

CIP

- Alte Abbilder/Volumes löschen: Weboberfläche, oder

```
$ openstack volume delete <volume id>  
$ openstack image delete <image id>
```

CIP

- Neue GRML-Instanz starten und Volume einhängen (siehe Folie 5)
- Partition mit VM-Abbild mounten

```
> mount /dev/vdb1 /mnt  
> mount -o bind /dev /mnt/dev  
> chroot /mnt /bin/bash
```

GRML

```
# mount -t proc proc /proc  
# mount -t sysfs sysfs /sys  
# mount -t devpts devpts /dev/pts
```

CHROOT

- Volume anpassen
- GRML-Instanz ordentlich beenden

```
# exit
```

CHROOT

```
> shutdown now
```

GRML

- Volume aushängen und Abbild erneut hochladen (siehe Folie 15)

- Modifikationen des VM-Abbilds über Grml-Instanz
 - Installation weiterer Softwarepakete
 - Anpassung der Startskripte
 - Systemkonfiguration
- Limitationen der Cloud-Umgebung des Lehrstuhls
 - Ressourcen der drei Node-Controller sind **beschränkt**
 - Beenden von nicht (mehr) benötigten Instanzen
 - Jederzeit auf faire Verwendung achten
- Infrastruktur
 - Bitte sendet bei Problemen oder Ungereimtheiten schnellstmöglichst eine E-Mail an `i4mw-owner@lists.cs.fau.de`

Anhang

Hinweis: Im Folgenden gezeigte (Code-)Beispiele dienen als zusätzliche Information und sind für das Lösen der Übungsaufgabe nicht vonnöten.

■ Gebräuchliche Abbild-Typen für virtuelle Maschinen (VM)

- Kopie eines Datenträgers (z. B. ISO-Image einer CD oder DVD):

```
$ dd if=/dev/sdb of=./cd-image.iso  
$ file -b ./cd-image.iso  
ISO 9660 CD-ROM filesystem data (bootable)
```

- Erzeugen einer leeren Abbild-Datei:

```
$ truncate -s 100M image.raw  
$ ls -lh image.raw  
-rw-r--r-- 1 thoenig users 100M  4. Nov 12:11 image.raw  
$ du image.raw  
0  
$ file -b image.raw  
data
```

- Alternativ ist es möglich, einen physischen Datenträger als Basis für eine virtuelle Maschine zu verwenden

- Die Erstellung und Aufbereitung des Abbilds der virtuellen Maschine benötigt erweiterte Privilegien (Root-Rechte)
- Die Aufbereitung des Abbilds geschieht daher isoliert in der Betriebsumgebung einer virtuellen Maschine („Live-System“)
 - ↪ In der Übung: Linux-Live-System Grml (<http://grml.org>)
- Varianten, dieses Live-System zu verwenden

- Mit Emulator qemu:

```
$ qemu -drive file=grml.iso,index=0,media=cdrom \
      -drive file=image.raw,index=1,media=disk
```

[root-Dateisystem (Teil von `grml.iso`, Gerätepfad `/dev/sr0`) wird automatisch eingehängt, nicht jedoch das leere Abbild (`image.raw`, Gerätepfad `/dev/sda`)]

- **In der Übung:** Instanz eines Grml-Abbilds direkt in der Cloud starten ↪ siehe vorangegangene Folien
- Eingerichtetes Abbild kann in Cloud hochgeladen werden

```
$ openstack image create --disk-format qcow2 \
  --file <image_file (e.g., image.raw)> <image_name>
```