

# Middleware – Cloud Computing – Übung

Michael Eischer, Laura Lawniczak, Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg  
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)

[www4.cs.fau.de](http://www4.cs.fau.de)

Wintersemester 2019/20



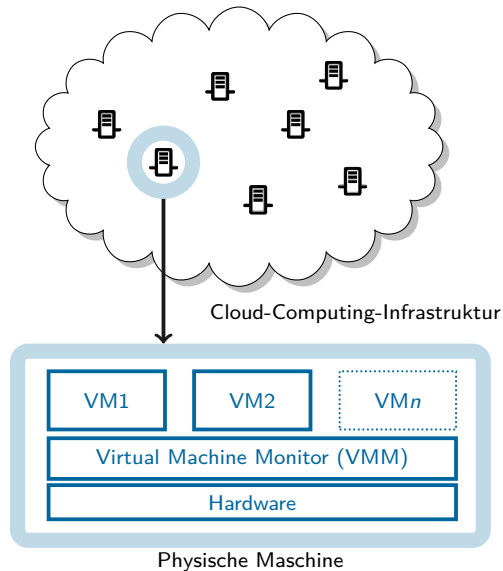
Cloud-Computing-Infrastruktur

Software-Infrastruktur

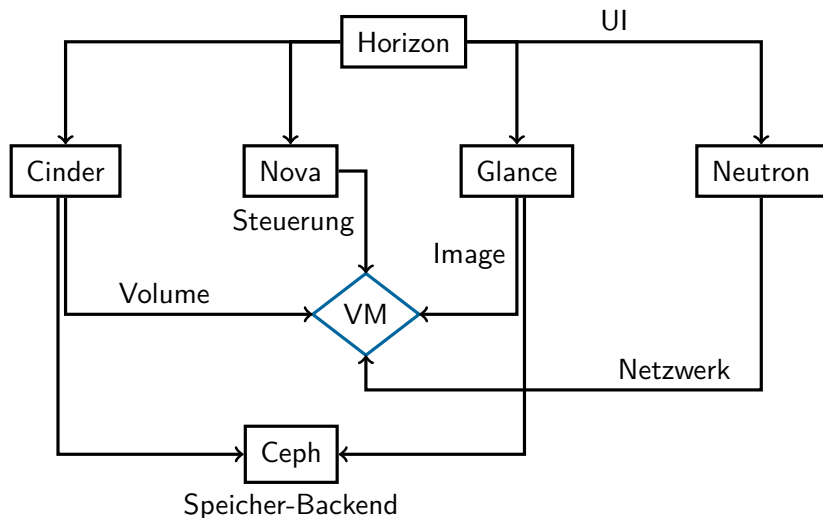
Aufbereiten des Abbilds für OpenStack

Betrieb der virtuellen Maschine





# Software-Infrastruktur am Beispiel von OpenStack



- **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
- **Nova:** Verwaltung virtueller Maschinen
  - Compute: Steuerung von VMs (QEMU/Xen/...) auf Rechnern
  - Scheduler: Verteilung auf verfügbare Hardware
- **Neutron:** Netzwerkmanagement und virtuelle Router
  - Server: Steuerung und Zustandsverwaltung
  - Agents: Helfer für DHCP, Open vSwitch, Metadaten
- **Horizon** (Dashboard): Weboberfläche für Anwender



- API-Dienst je Komponente für REST-Anwenderschnittstelle  
→ Kommandozeilentools
- Kommunikation der Dienste intern via RabbitMQ
  - Gruppenkommunikation über Nachrichtenbus
  - Standardisiertes Protokoll: AMQP
- Speicher-Backend für Glance und Cinder: **Ceph**
  - Block-Storage oder Dateisystem verteilt auf Rechner-Cluster
  - Flexible Konfiguration von Replikationseigenschaften
  - Transaktionen über Paxos-Protokoll



- Ziel: Verlagerung der Übungsaufgabe in eine virtuelle Maschine
- Speicherarten
  - Volume: Änderungen persistent, nur in einer Instanz
  - Image(Abbild): Änderungen flüchtig, Basis für viele Instanzen
- Abbild innerhalb von OpenStack erzeugen
  - Starten einer Grml-Instanz (Live-System)
  - Neues Volume anlegen und einhängen
  - Befüllen mit Daten (Betriebssystem, Anwendung)
  - Anpassen der Konfiguration; Installieren zusätzlicher Softwarepakete
  - Umwandeln in Image
- Abbild starten
  - Öffentlichen Schlüssel für passwortlose Authentifizierung hinterlegen
  - Instanz mit eigenem Image starten
  - Übungsaufgabe in der Cloud laufen lassen



- Web-Frontend
  - Dashboard: <https://i4cloud.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





# Entwicklung eines VM-Abbilds

- 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
  - **In der Übung:** Instanz eines Grml-Abbilds direkt in der Cloud starten
    - ↪ siehe nächste Folien
  - Mit Emulator qemu: siehe Anhang



## Grml-Instanz starten

- Name für Instanz festlegen
- Instanztyp `i4.grml`
  - Kein Swap/Ephemeral-Volume
- Booten vom bereitgestellten Grml-Image (GRML-2018.12-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

Instanz starten

Details

Quelle \*

Variante \*

Netzwerke

Netzwerk-Ports

Sicherheitsgruppen

Schlüsselpaar

Konfiguration

Servergruppen

Schedulerhinweise

Metadaten

Geben Sie bitte den initialen Hostnamen der Instanz an, die Verfügbarkeitszone, in der sie erstellt wird und die Anzahl der Instanzen. Erhöhen Sie die Anzahl, um eine Vielzahl von Instanzen mit den gleichen Einstellungen zu erzeugen.

Instanzname \*

Instanzen gesamt (10 Max)

100%

0 Current Usage  
1 Added  
9 Remaining

Verfügbarkeitszone

Anzahl \*

Abbrechen

< Back

Weiter >

Instanz starten

Instanz starten

Details

Quelle

Variante

Netzwerke \*

Netzwerk-Ports

Sicherheitsgruppen

Schlüsselpaar

Konfiguration

Servergruppen

Schedulerhinweise

Metadaten

Varianten beinhalten die Größen der Compute-, Speicher- und Storage-Kapazität einer Instanz.

Allocated

| Name      | VCPUS | RAM    | Festplatte gesamt | Öffentlich |
|-----------|-------|--------|-------------------|------------|
| > i4.grml | 1     | 768 MB | 1 GB              | Ja         |

Verfügbar

Click here for filters.

| Name        | VCPUS | RAM    | Festplatte gesamt | Öffentlich |
|-------------|-------|--------|-------------------|------------|
| > i4.tiny   | 1     | 512 MB | 3 GB              | Ja         |
| > i4.docker | 1     | 1 GB   | 10 GB             | Ja         |

Abbrechen

< Back

Weiter >

Instanz starten

Instanz starten

Details

Quelle

Variante \*

Netzwerke \*

Netzwerk-Ports

Sicherheitsgruppen

Schlüsselpaar

Konfiguration

Servergruppen

Schedulerhinweise

Metadaten

Die Instanz-Quelle ist die Vorlage, die zur Erstellung der Instanz verwendet wird. Sie können ein Abbild, eine Schattenkopie einer vorhandenen Instanz (Abbild-Schattenkopie), einen Datenträger oder eine Datenträger-Schattenkopie (sofern aktiviert) verwenden. Durch die Erstellung eines neuen Datenträgers können Sie auch die Verwendung von bestandigen Speicher auswählen.

Bootquelle auswählen

Neuen Datenträger erstellen

Abbild 1

Ja

Nein 2

Allocated

| Name   | Aktualisiert    | Größe     | Typ | Sichtbarkeit |
|--------|-----------------|-----------|-----|--------------|
| > grml | 8/22/17 3:57 PM | 590.00 MB | iso | Öffentlich   |

Verfügbar

Click here for filters.

| Name                       | Aktualisiert | Größe | Typ | Sichtbarkeit |
|----------------------------|--------------|-------|-----|--------------|
| Keine verfügbaren Einträge |              |       |     |              |

Abbrechen

< Back

Weiter >

Instanz starten

Instanz starten

Details

Quelle

Variante

Netzwerke

Netzwerk-Ports

Sicherheitsgruppen

Schlüsselpaar

Konfiguration

Servergruppen

Schedulerhinweise

Metadaten

Netzwerke stellen die Kommunikationskanäle für die Instanzen in der Cloud zur Verfügung.

Allocated

Wählen Sie aus den unten stehenden Netzwerken aus.

| Netzwerk | Gemeinsam verwendet | Admin-Status | Status |
|----------|---------------------|--------------|--------|
| > 1      | Internal            | Ja           | Up     |

Verfügbar

Click here for filters.

| Netzwerk                   | Gemeinsam verwendet | Admin-Status | Status |
|----------------------------|---------------------|--------------|--------|
| Keine verfügbaren Einträge |                     |              |        |

Abbrechen

< Back

Weiter >

Instanz starten



# Volume erzeugen/einhängen

Project / Compute / Volumes

## Volumes

Volumes Volume Snapshots

Filter  **+ Create Volume**

Displaying 1 item

| <input type="checkbox"/> | Name        | Description | Size | Status    | Type | Attached To | Availability Zone | Bootable | Encrypted | Actions                                    |
|--------------------------|-------------|-------------|------|-----------|------|-------------|-------------------|----------|-----------|--|
| <input type="checkbox"/> | my-vol-name | -           | 2GiB | Available | -    |             | nova              | No       | No        | <input type="button" value="Edit Volume"/> |

Displaying 1 item

**2**

- Extend Volume
- Manage Attachments**
- Create Snapshot
- Change Volume Type

- (1) Leeres Volume anlegen, benötigt Name und Größe (2 GB)
- (2) 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>
```

CIP



# Entwicklung des VM-Abbilds

Project / Compute / Instances / grml-instance

## grml-instance

Create Snapshot ▾

Overview Log Console Action Log

### Instance Console

If console is not responding to keyboard input: click the grey status bar below. [Click here to show only console](#)  
To exit the fullscreen mode, click the browser's back button.

```
Connected (encrypted) to: QEMU (instance-0000003f) Send CtrlAltDel
[ OK ] Started Permit User Sessions.
[ OK ] Started D-Bus System Message Bus.
[ OK ] Started Login Service.
[ OK ] Started ifup for eth0.
[ OK ] Starting rsyslog in background.
[ OK ] Activating language settings:
[ OK ] Listening on Syslog Socket.
[ OK ] Starting System Logging Service...
[ OK ] Writing language settings (en) to /etc/default/locale was successful.
[ OK ] Setting up unicode environment.
[ OK ] Started System Logging Service.
[ OK ] Running loadkeys for us in background
[ OK ] Setting font to Uni3-Terminus16
[ OK ] Running Linux Kernel 4.9.0-1-grml-amd64 inside KVM
[ OK ] CPU(s) featuring virtualization technology detected
[ OK ] Creating /etc/mdadm/mdadm.conf for use with mdadm.
[ OK ] Just run 'mdadm --assemble --scan' to assemble md arrays or boot using 'suraid' as bootoption for autostart.
[ OK ] Searching for any present dmraid sets:
[ OK ] No block devices found
[ OK ] You seem to have logical volumes (LVM) on your system.
[ OK ] Just run 'Start lvm2-pvscan@name' to activate LV or VG 'name' or boot using 'lvm' as bootoption for autostart.
[ OK ] Found CPU: Processor 0 is Westmere E56xx/X56xx/X56xx (Nehalem-D) 2400MHz, 16384 KB Cache
[ OK ] Trying to set up cpu frequency scaling:
```

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



- Um als Basis für eine virtuelle Maschine zu dienen, muss die Abbild-Datei (z. B. `image.raw`) 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**ilesystem) 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



## Einhängen, Bootstrapping

- 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 buster /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



### Exkurs: Wechsel des Wurzelverzeichnisses

- 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)





# Entwicklung eines VM-Abbilds

## Systemkonfiguration

- 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 `cloud`

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

CHROOT

```
# passwd cloud
```

CHROOT



### Software-Installation

- Ergänzen der Software des Grundsystems mittels apt-get
- Aktualisieren der Paketquellen (update) und anschließendes Einspielen potentieller 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



# VM-Umgebung verlassen

- 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



# Erstellen eines OpenStack-Abbilds

## ■ Möglichkeiten, ein Abbild zu erzeugen

a) Im Dashboard: „Volumes“ → „Upload to Image“

- Image name eingeben
- Disk format auf **raw** einstellen

b) Abbild aus Volume erzeugen (z. B. bei Volume-Erstellung über Weboberfläche)

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

CIP

c) Datei als Abbild hochladen

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



# SSH-Authentifizierung mit Schlüsseln

## ■ SSH-Authentifizierung mit einem Schlüsselpaar **ohne** Passwort

### 1. 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

### 2. Hinterlegen des **öffentlichen** Schlüssels

#### a) in Openstack

→ siehe nächste Folie

#### b) in chroot-Umgebung

```
$ su - cloud    # Zum cloud Benutzer wechseln  
$ mkdir .ssh  
$ scp <user>@<cip_pool_host>:~/<gruppen_name>.pub \  
    .ssh/authorized_keys  
$ exit         # Shell beenden - Rueckkehr vom Benutzerwechsel
```

### 3. Zugriff auf VM siehe Folie 4-27



# SSH-Schlüssel hinzufügen (einmalig)

- Neu erstellten **öffentlichen Schlüssel** (<gruppen\_name>.pub) hinzufügen unter „Compute“ → „Key Pairs“ → „Import Key Pair“

The screenshot shows the OpenStack dashboard interface. On the left, there is a sidebar with navigation links: Project, Compute, Overview, Instances, Volumes, Images, and a highlighted 'Key Pairs' button. The main content area is titled 'Key Pairs' and shows a breadcrumb 'Project / Compute / Key Pairs'. Below the title, there is a filter input field and two buttons: '+ Create Key Pair' and 'Import Key Pair' (which is highlighted with a red box). Below these buttons is a table with columns 'Key Pair Name', 'Fingerprint', and 'Actions'. The table currently displays 'No items to display.'

- Kommandozeile:

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

CIP



# Eigenes Abbild als VM starten

- Instanztyp `i4.tiny`  
→ Erzeugt Swap-Disk und vergrößert `root`-Partition
- Von eigenem Abbild starten
- SSH-Schlüssel unter „Key Pair“ auswählen
- 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





# Öffentliche IP zuweisen

Project / Network / Floating IPs

## Floating IPs

1 [Allocate IP To Project](#) [Release Floating IPs](#)

Displaying 1 item

| IP Address     | Mapped Fixed IP Address | Pool | Status | Actions   |
|----------------|-------------------------|------|--------|-----------|
| 131.188.42.118 | -                       | -    | Down   | Associate |

- (1) Öffentliche IP aus Pool allokatieren, **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



# Zugriffsregeln für Netzwerkverbindungen

Project / Network / Security Groups

## Security Groups

Filter  [+ Create Security Group](#) [Delete Security Groups](#)

Displaying 1 item

| <input type="checkbox"/> | Name    | Description            | Actions                      |
|--------------------------|---------|------------------------|------------------------------|
| <input type="checkbox"/> | default | Default security group | <a href="#">Manage Rules</a> |

- 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

**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



# Nachträgliche Anpassungen am Abbild

- Neue GRML-Instanz starten und Volume einhängen (siehe Folie 4-9)

- 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 (siehe Folie 4-19)
- Abbild erneut hochladen (siehe Folie 4-20)



- Modifikationen des VM-Abbilds über Grml-Instanz
  - Installation weiterer Software-Pakete
  - 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 `mw@i4.informatik.uni-erlangen.de`



Anhang



**Hinweis:** Im Folgenden grau unterlegte (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

