

Middleware – Cloud Computing – Übung

Cloud-Computing-Infrastruktur

Wintersemester 2025/26

Paul Bergmann, Christian Berger

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Systemsoftware)

<https://sys.cs.fau.de>



Lehrstuhl für Informatik 4
Systemsoftware



Friedrich-Alexander-Universität
Technische Fakultät

Erstellen eines VM-Abbilds in OpenStack

Cloud Computing Software-Infrastruktur

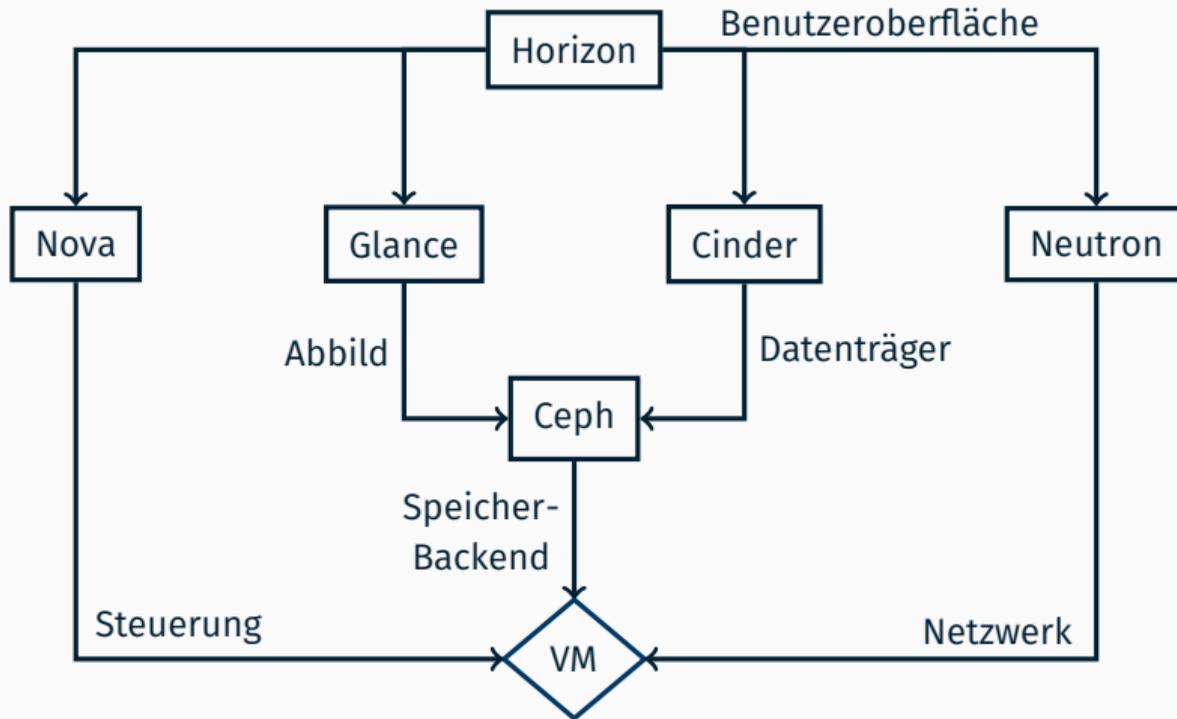
Erstellen des Abbilds für OpenStack

Betrieb der virtuellen Maschine

Erstellen eines VM-Abbilds in OpenStack

Cloud Computing Software-Infrastruktur

Software-Infrastruktur am Beispiel von OpenStack



- **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-2025.05`)
 - 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
```

Grml-Instanz starten

Launch Instance

Details

Please provide the initial hostname for the instances, 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 Total Instances (10 Max) 10/6

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration Count 0 Current Usage 1 Added 9 Remaining

Server Groups

Scheduler Hints

Metadata

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.

Select Root Device

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Allocated

Deploying 1 item

Name	Updated	Size	Type	Visibility
grml-2020-06	10/20/20 5:02 PM	747.25 MB	RAW	Public

Available

Select one

Click here for filters or full text search.

Deploying 1 item

Name	Updated	Size	Type	Visibility
amazon-ec2-ami-2020-06	10/20/20 5:48 PM	2.00 GB	RAW	Public

Cancel Back Next Launch Instance

Launch Instance

Details

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

Source

Flavor

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m5.xlarge	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
m5.xlarge	1	512 MB	3 GB	3 GB	0 GB	Yes
m5.xlarge	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

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Allocated

Select networks from those listed below.

Network	Subnets Associated	Shared	Admin State	Status
internal	internal-pool	Yes	Up	Active

Available

Select at least one network

Click here for filters or full text search.

No available items

Cancel Back Next Launch Instance

Project / Volumes / Volumes

Volumes

Filter 1 + Create Volume ⇌ Accept Transfer 🗑 Delete Volumes

Displaying 1 item

<input type="checkbox"/>	Name	Description	Size	Status	Group	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
<input type="checkbox"/>	my-vol -name	-	2GiB	Available	-	__DEFAULT__		nova	No	No	Edit Volume 3 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>
```

The screenshot shows the OpenStack dashboard interface. On the left is a navigation sidebar with categories like Project, API Access, Compute, Overview, Instances (highlighted), Images, Key Pairs, Server Groups, Volumes, Network, and Identity. The main content area shows the breadcrumb 'Project / Compute / Instances / grml-instance' and the title 'grml-instance' with a 'Create Snapshot' button. Below the title are tabs for Overview, Interfaces, Log, Console (selected), and Action Log. The 'Instance Console' section contains a warning message: '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.' Below this is a terminal window titled 'Connected (encrypted) to QEMU (instance-0000005b)' with a 'Send Ctrl+Alt+Del' button. The terminal output shows system boot logs:

```
[ OK ] Finished Permit User Sessions.
[ OK ] Found device Virtio network device.
[ OK ] Started ifup for eth0.
[ OK ] Started D-Bus System Message Bus.
[ OK ] Started Login Service.
[ OK ] Starting rsyslog in background.
[ OK ] Listening on Syslog Socket.
Starting System Logging Service...
[ OK ] Started System Logging Service.
[ OK ] Activating language settings:
[ OK ] Writing language settings (en) to /etc/default/locale was successful.
Starting Set console font and keymap...
[ OK ] Finished Set console font and keymap.
[ OK ] Running Linux kernel 5.6.0-2-amd64 inside KVM
[ OK ] CPU(s) featuring virtualization technology detected
[ OK ] Creating /etc/mdadm/mdadm.conf for use with mdadm.
```

- 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_umsdos' '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

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 trixie /mnt/ 'http://ftp.fau.de/debian'
```

GRML

Kontrolle:

```
> ls -aR /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 `cloud`

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

CHROOT

```
# passwd cloud
```

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-21-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

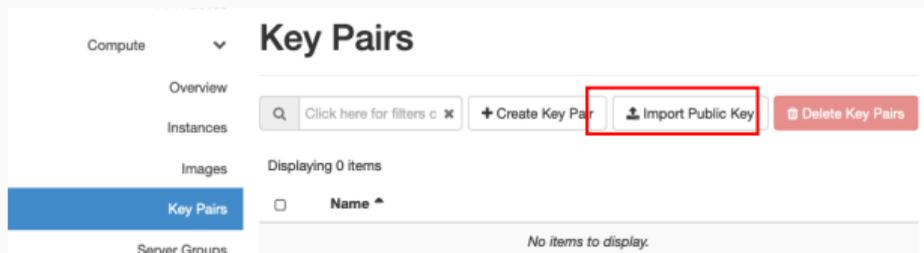
SSH-Schlüssel einrichten (einmalig)

- 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

Project / Network / Floating IPs

Floating IPs

Floating IP Address = Filter

Displaying 1 item

<input type="checkbox"/>	IP Address	Description	Mapped Fixed IP Address	Pool	Status	Actions
<input type="checkbox"/>	131.188.44.15	-	-	i4labnet	Down	<input type="button" value="Associate"/>

Displaying 1 item

- (1) Öffentliche IP aus Pool allokkieren, **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 for managing Security Groups. On the left is a navigation sidebar with categories like Project, API Access, Compute, Volumes, Network, Network Topology, Networks, Routers, Security Groups (highlighted), and Floating IPs. The main content area shows the breadcrumb 'Project / Network / Security Groups' and the title 'Security Groups'. Below the title are buttons for 'Filter', '+ Create Security Group', and 'Delete Security Groups'. A table lists one security group: '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
```

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

→ SSH-Schlüssel siehe Folie 16, 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>
```