

## Cloud-Computing-Infrastruktur

Physikalische Infrastruktur

Software-Infrastruktur

Aufbereiten des Abbilds für OpenStack

Betrieb der virtuellen Maschine



## Physikalische Infrastruktur

### Standard-Hardware

→ „commodity hardware“

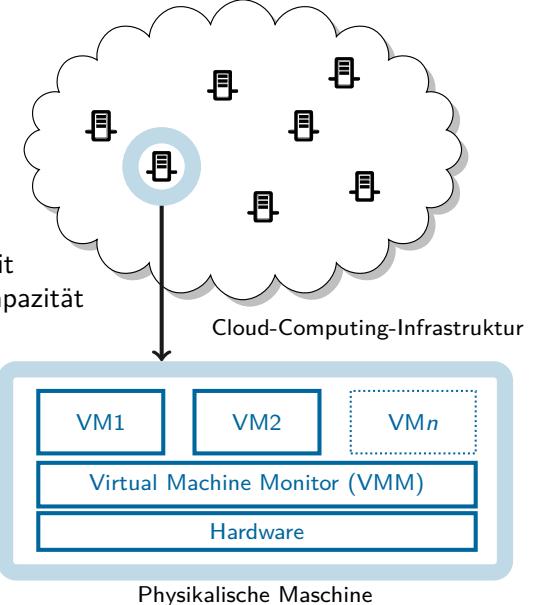
→ billig

→ **fehleranfällig**

### Charakteristik

→ hoher Grad der Verteiltheit

→ „unbegrenzte“ Speicherkapazität



## 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



## Software-Infrastruktur am Beispiel von OpenStack

- 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



## Genereller Ablauf

- 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 passwordlose Authentifizierung hinterlegen
  - Instanz mit eigenem Image starten
  - Übungsaufgabe in der Cloud laufen lassen



## Entwicklung eines VM-Abbilds

`dd(1), truncate(1)`

**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



## Zugriff auf OpenStack

- 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

`qemu(1)`

- 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`, Gerätelpfad `/dev/sr0`) wird automatisch eingehängt, nicht jedoch das leere Abbild (`image.raw`, Gerätelpfad `/dev/sda`)]

- **In der Übung:** Instanz eines Grml-Abbilds direkt in der Cloud starten  
→ siehe nächste Folien



## Grml-Instanz starten

- Name für Instanz festlegen
- Instanztyp i4.grml
  - Kein Swap/Ephemeral-Volume
- Booten vom bereitgestellten Grml-Image (GRML-2017.05-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



MW-Übung (WS18/19)

Cloud-Computing-Infrastruktur – Aufbereiten des Abbilds für OpenStack

4-9

## Grml-Instanz starten

The screenshot shows the OpenStack dashboard with the 'Instances' tab selected. The 'Create Instance' form is open, showing the 'grml-instance' name and 'i4.grml' flavor. The 'Allocated' tab is selected, displaying the server's details: Name (grml), VCPUs (1), RAM (768 MB), and Disk (1 GB). The 'grml' volume is attached. The 'Allocated' tab also lists other resources: a network (internal), a keypair (i4.ssh), and a configuration (i4.docker). The 'Allocated' tab has a 'Details' section with a table showing the server's configuration.

MW-Übung (WS18/19) Cloud-Computing-Infrastruktur – Aufbereiten des Abbilds für OpenStack

4-10

## Entwicklung des VM-Abbilds

Project / Compute / Instances / grml-instance

Create Snapshot

grml-instance

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-00000009)
Send CtrlAltDel
[ OK ] Started Permit User Sessions.
[ OK ] Started D-Bus System Message Bus.
[ OK ] Started Network Configuration.
[ OK ] Started ifup for eth0
[ OK ] Starting rsyncd in background.
[ OK ] Activating language settings:
[ OK ] Listening on Syslog Socket.
[ OK ] Setting up System Logging Service.
[ OK ] Setting up uniconsole Environment.
[ OK ] Started System Logging Service.
[ OK ] Running loadkey for us in background.
[ OK ] Setting font to 'Ubuntu Terminal'.
[ OK ] Using kernel '4.12.0-10-amd64' inside KVM.
[ OK ] CPU(s) featuring virtualization technology detected.
[ OK ] Creating /etc/maddm/maddm.conf for use with maddm.
[ OK ] Just run 'maddm --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 ] Creating dynamic logical volumes (LVM) on your system.
[ OK ] Just run 'Start lvm <volume>' to activate LV or VG 'name' or boot using 'lvm' as bootoption for autostart.
[ OK ] Found CPU: Processor 0 is Westmere E56xx/L56xx (Nehalem-D) 2400MHz, 16384 kB Cache
[ OK ] Trying to set up cpu frequency scaling:
```

- (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



MW-Übung (WS18/19)

Cloud-Computing-Infrastruktur – Aufbereiten des Abbilds für OpenStack

4-11

- Konsole der laufenden Instanz im Dashboard öffnen

- Einrichtung des Betriebssystems und Installation der Java-Anwendung im Verlauf der Übung

MW-Übung (WS18/19)

Cloud-Computing-Infrastruktur – Aufbereiten des Abbilds für OpenStack

4-12

- 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
```
- Das Kommando `mkfs` (make filesystem) 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
```

### 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)`

### Einhängen, Bootstrapping

- Installation der User-Space-Komponenten des zukünftigen Gastbetriebssystems in das neu erzeugte, leere Dateisystem:
  - Einhängen des zuvor erstellten Dateisystems mit `mount`:  

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

**GRML**
  - Kontrolle:  

```
> mount | grep vdb1
```

**GRML**
  - Erstellung der User-Space-Komponenten des Zielsystems mit `debootstrap`:  

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

**GRML**
  - Kontrolle:  

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

**GRML**
  - 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**

### 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'.
```

```
# passwd cloud
```

**CHROOT**

**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> ... <paketc>  
# apt-get clean
```

CHROOT

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

```
openssh-server openjdk-8-jdk-headless screen vim-nox
```

CHROOT



## 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>
```

- Installation benötigter Bibliotheken

```
# mkdir -p /proj/lib  
# uget 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 init(8) die Init-Skripte aus
- Bereitgestelltes Startskript /etc/rc.local
  - Ermittelt öffentliche IP der VM-Instanz
  - Wertet Konfigurationsdaten (user-data) aus; siehe Aufgabenstellung
  - Lädt jar-Datei mit der Anwendung aus S3 herunter
  - 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



## Erstellen eines OpenStack-Abbilds

- Möglichkeiten, ein Abbild zu erzeugen

- Im Dashboard: „Volumes“ → „Upload to Image“

- Imagename eingeben
- Disk format auf raw einstellen

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

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

CIP

- 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



MW-Übung (WS18/19)

Cloud-Computing-Infrastruktur – Aufbereiten des Abbilds für OpenStack

4-21

## Eigenes Abbild als VM starten

- Instanztyp i4.tiny  
→ Erzeugt Swap-Disk und vergößert 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

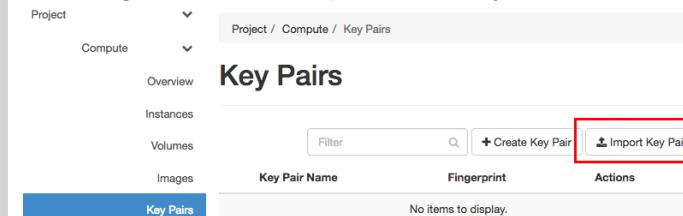
MW-Übung (WS18/19)

Cloud-Computing-Infrastruktur – Aufbereiten des Abbilds für OpenStack

4-23

## SSH-Schlüssel hinzufügen (einmalig)

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



### Key Pairs

- Kommandozeile:

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

CIP



MW-Übung (WS18/19)

Cloud-Computing-Infrastruktur – Aufbereiten des Abbilds für OpenStack

4-22

## Öffentliche IP zuweisen



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

MW-Übung (WS18/19)

Cloud-Computing-Infrastruktur – Aufbereiten des Abbilds für OpenStack

4-24

- 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

## Betrieb der virtuellen Maschine

- 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 herausuchen
$ openstack server delete <instanz id>
```

CIP

- Alte Abbilder/Volumes löschen: Weboberfläche, oder

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

CIP

Ingress = Eingehende Verbindungen, Egress = Ausgehende Verbindungen

## 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`

