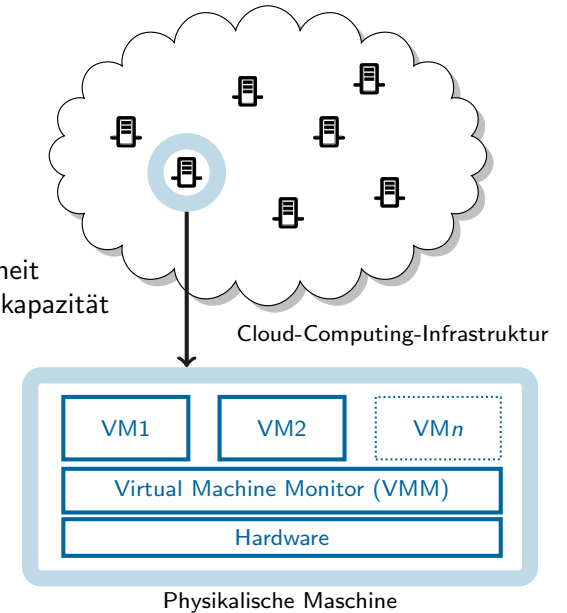


Cloud-Computing-Infrastruktur
Physikalische Infrastruktur
Software-Infrastruktur
Aufbereiten des Abbilds für OpenStack
Betrieb der virtuellen Maschine



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



Zugriff auf OpenStack

- Web-Frontend
 - Dashboard: <http://i4cloud.informatik.uni-erlangen.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“ → „Compute“ → „API Access“ → „Download OpenStack RC File v3“
 - 2) RC-Datei einlesen und ausführen (sourcen)

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



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



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

```
$ 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 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
```



Grml-Instanz starten



Volume erzeugen/einhängen

- (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>
```



Entwicklung des VM-Abbilds

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



Entwicklung eines VM-Abbilds parted(8), mkfs(8)

- 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
```



Entwicklung eines VM-Abbilds mount(8), debootstrap(8)

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
```

Kontrolle:

```
$ mount | grep vdb1
```
 2. Erstellung der User-Space-Komponenten des Zielsystems mit debootstrap:

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

Kontrolle:

```
$ ls -alR /mnt | more
```
 3. Setupskript mittels scp in das Abbild (/mnt) kopieren

```
$ scp <login>@<cip_host>:/proj/i4mw/pub/aufgabe2/post-debootstrap.sh \ /mnt
```



Entwicklung eines VM-Abbilds chroot(2)/(8)

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

```
$ chroot /mnt /bin/bash
```

 - ↪ **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
```



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
```

- 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
```

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

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



VM-Umgebung verlassen

- Shell beenden, um chroot-Umgebung zu verlassen

```
$ exit
```

- Grml-Live-Umgebung herunterfahren

```
$ shutdown now
```

- 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
$ scp -r <login>@<cip_host>:/proj/i4mw/pub/aufgabe1/jaxrs-ri-2.26 \
  <login>@<cip_host>:/proj/i4mw/pub/aufgabe2/tika \
  /proj/lib
```

- Automatisches Starten der Dienste

- Beim Systemstart führt init(8) die Init-Skripte aus
- Bereitgestelltes Startskript /etc/rc.local
 1. Ermittelt öffentliche IP der VM-Instanz
 2. Wertet Konfigurationsdaten (user-data) aus; siehe Aufgabenstellung
 3. Lädt jar-Datei mit der Anwendung aus S3 herunter
 4. 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
```



Erstellen eines OpenStack-Abbilds

- Möglichkeiten, ein Abbild zu erzeugen

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

- Imagenamen 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>
```

- 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.
(...)
```

2. Hinterlegen des **öffentlichen** Schlüssels

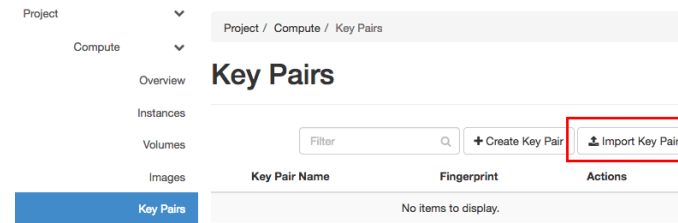
- a) in Openstack
→ siehe nächste Folie
- b) in chroot-Umgebung

```
$ 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 3-27

SSH-Schlüssel hinzufügen (einmalig)

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



- Kommandozeile:

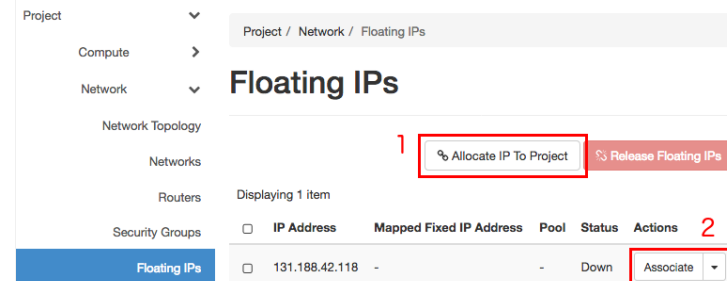
```
$ openstack keypair create --public-key <gruppen_name>.pub \
    <schluessel_name>
```

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
```

Öffentliche IP zuweisen



- (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>
```

- Abfrage innerhalb laufender VM per REST-API:

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

Zugriffsregeln für Netzwerkverbindungen

Project / Network / Security Groups

Security Groups

Filter + Create Security Group Delete Security Groups

Displaying 1 item

Name	Description	Actions
default	Default security group	Manage Rules

- 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
```

Firewall-Zugriffsregeln

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

Betrieb der virtuellen Maschine

- Passwortloser Zugriff mit SSH

```
$ ssh-keygen -R <instanz_ip> # Alten Host-Key entfernen
$ ssh -i <gruppen_name> cloud@<instanz_ip>
```

→ Schlüssel aus letzter Tafelübung, Instanz-IP aus vorheriger Zuweisung
→ Anderes VM-Image unter selber IP erfordert evtl. Zurücksetzen von Host-Key

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

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

- Alte Abbilder/Volumes löschen: Weboberfläche, oder

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

Nachträgliche Anpassungen am Abbild

- Neue GRML-Instanz starten und Volume einhängen (siehe Folie 3-9)
- Partition mit VM-Abbild mounten

```
$ mount /dev/vdb1 /mnt
$ mount -o bind /dev /mnt/dev
$ chroot /mnt /bin/bash
$ mount -t proc proc /proc
$ mount -t sysfs sysfs /sys
$ mount -t devpts devpts /dev/pts
```

- Volume anpassen
- GRML-Instanz ordentlich beenden

```
$ exit
$ shutdown now
```

- Volume aushängen (siehe Folie 3-19)
- Abbild erneut hochladen (siehe Folie 3-20)

- 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

