# Projekt: Cluster Management

**High Performance Cloud Computing mit MAAS, Juju und OAR**

Johann Weging

10. Juli 2012

# Inhaltsverzeichnis

# 1 Einleitung

Das Installieren und Warten eines Hochleistungs-Rechen-Clusters ist eine nicht triviale und zeitaufwendige Aufgabe. Daher gibt es Cloudanbieter[1][2], die Rechenleistung in großem Maße anbieten, um die Notwendigkeit eines eigenen Rechenzentrums entgegen zu wirken.

Dieses Projekt versucht durch die Verwendung von Cloudinfrastruktur die Aufgabe der Clusteradministration zu vereinfachen. Hierzu wird eine private Cloud installiert, welche in der Lage ist Ubuntu auf Hardware-Servern zu installieren und Services auf diese Server zu deployen. Ist die Cloud-Infrastruktur konfiguriert, lässt sich die Cloud mit einigen wenigen Befehlen verwalten.

## 1.1 Abstrakt

Dieser Bericht beschreibt das Installieren und Betreiben einer High Performance Compute Cloud unter Ubuntu 12.04 [3]. Verwendet wird die von Ubuntu bereitgestellte Cloudinfrastucktur MAAS (Metal as a Service) [4]. MAAS ist ein Linux Installationsserver, basierend auf cobbler[5], welcher in der Lage ist Server eigenständig mit einem Betriebssystem zu installieren, ohne dass der Benutzer in den Installationsprozess eingreifen muss. Als Cloud-Verwaltungssoftware wird Juju[6] eingesetzt. Mit Hilfe von Juju können Server in der Cloud angefordert und mit Software, sogenannten Services, versehen werden. Als Scheduler wird OAR[7] eingesetzt. Darüber lassen sich Rechenjobs auf den in der Cloud befindlichen Servern starten.

Innerhalb des Projektes ist es gelungen eine Cloud-Infrastruktur für die Rechenknoten des Cluster zu etablieren. Das Projekt beschreibt die Installation und Konfiguration aller oben genannten Softwarekomponenten. Zusätzlich wird der Aufbau der Testumgebung beschrieben. Das Test-Cluster ist ein Verbund aus virtuellen Maschinen auf einem einzigen physikalischen Rechner. Zur Virtualisierung wird KVM[8] eingesetzt. Auf das Installieren von E/A-Servern wird nicht weiter eingegangen. Auch die Benutzerverwaltung ist sehr rudimentär.

## 1.2 Struktur des Berichtes

Das Kapitel 3 ist wie eine Anleitung zu verstehen, die schrittweise die Installation des Cluster beschreibt. Das Kapitel 4 beschreibt die Administration des Clusters und gibt einige Hintergrundinformationen zu den verwendeten Tools. Hier wird unter anderem Hilfe zu einigen Problemen gegeben, die möglicherweise auftreten. Das Kapitel 5 beschreibt die Benutzung des Cluster aus Benutzersicht. Hier geht es um das Submitten von Jobs und anderen Möglichkeiten. Das Kapitel 6 behandelt weiterführende Möglichkeiten um das Cluster zu verbessern.

# 2 Aufbau der Testumgebung

Bei der Testumgebung handelt es sich um einen physikalischen Rechner, der mehrere VMs (Virtual Maschines) bereitstellt. Insgesamt werden vier VMs erstellt. `pcm-master` wird das Verwalten des Cluster übernehmen und als Loginknoten für den Benutzer dienen. `juju` wird zum Verwalten der Cloud-Infrastruktur benötigt. `compute1` und `compute2` werden als Rechenknoten eingesetzt.

# 3 Clusterkonfiguration

Dieses Kapitel beschreibt die Installation und Konfiguration des gesamten Clusters. Diese Anleitung beschreibt die Installation Schritt für Schritt ohne dabei auf die Funktionsweise und Hintergründe einzugehen, dies wird in den Kapiteln 4 und 5 vornommen.

## 3.1 Installieren des VM-Hosts

Bei dem Betriebssystem für den VM-Host handelt es sich um eine Standardinstallation von Ubuntu Server 11.10. Als Rechnername wird `vm-host` gewählt. Die einzigen zusätzlichen Softwarepakete, die installiert werden, sind `OpenSSH server` und `Virtual Machine host` (Abbildung 2).

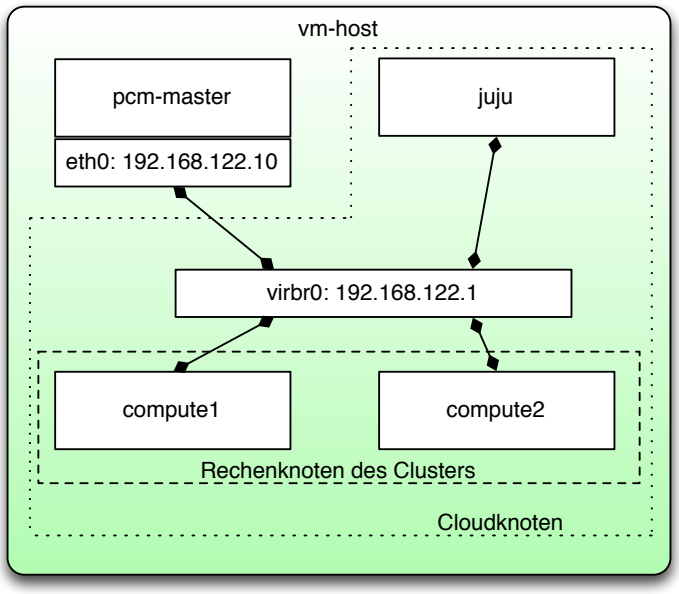


Abbildung 1: Vernetzung der Cluster-Knoten.

Dadurch werden schon einige Einstellungen von dem Installationsprogramm vorgenommen. Nach der Installation werden noch einige weitere Paket benötigt. `acpid` wird benötigt, damit das Powermanagment korrekt funktioniert. `kmv-pxe` ermöglicht es VMs via PXE zu booten, dies ist notwendig, da später die VM zum Installieren mittels PXE gebootet werden. `virtinst`[8.1] stellt das Programm virt-install bereit, das verwendet wird um die VM zu erzeugen. (Listing 1).

Listing 1: Installation zusätzlicher Pakete nach der Installation des VM-Hosts.

```
1 vm-host:~$ sudo aptitude update && sudo aptitude install acpid kvm-pxe virtinst
```

## 3.2 Installieren des Loginknoten

Der Loginknoten übernimmt die Clusterverwaltung, das Bereitstellen der Cloudinfrastucktur, Werkzeuge zum Benutzen und Verwalten des Clusters und die Benutzerverwaltung. Ist die Installation abgeschlossen, kann das Cluster fast ausschließlich von diesem Server aus verwaltet werden.

### 3.2.1 Erzeugen der VM

Das Listing 2 zeigt das Erzeugen einer VM. `--connect` gibt die zu verwendende Domain an. `-name` gibt den Namen der VM an. `-ram` gibt die Menge des Arbeitsspeichers in Megabyte an, der Speicher ist hier auf Grund der zur Verfügung stehenden Hardware knapp bemessen. `-cdrom` gibt den Ort der CD an, von der gebootet wird, hier kann ein CD-Image oder ein CD-Device angegeben werden. Dazu wird das Server-Image von Ubuntu 12.04 verwendet. `-os-type` gibt den Typen des Betriebsystems an, hier kann man z.B. zwischen `linux` und `windows` wählen. Mit `-os-variant` kann die genaue Systemvariante bestimmt werden. Welche Varianten unterstützt werden, kann mit `-os-variant list` angezeigt werden. Listing 3 zeigt ein Auszug aus der Liste. Es wird Ubuntu 11.10 angegeben, da das VM-Hostsystem noch keine neuere Variante unterstützt. `-boot` gibt an, von welchem Medium gebootet werden soll. `-disk` gibt das Medium an, auf dem das System installiert werden soll, hier wird ein Container von dem Installationsprogramm erzeugt. Durch den Parameter `size` wird die Größe des Containers definiert. Die Option `-network` gibt die Art der Verbindung zum Netzwerk an, in diesem Fall wird die VM an die virtuelle Bridge `virbr0` angeschlossen. Diese wurde bei der Installation des Betriebssystems automatisch angelegt. Damit die Installation durchgeführt werden kann, wird mit `-graphics vnc,keymap=de,listen=192.168.111.3` die VM an VNC angebunden. Es ist darauf achten die richtige keymap zu verwenden, da man sonst sein Passwort bei der Installation falsch eingibt. Des Weiteren sollte man darauf achten die richtige Netzwerkschnittstelle mit `listen` anzugeben, falls der Rechner nicht über eth0 angesteuert wird. Zusätzlich kann man mit `password` noch ein Passwort zum Sichern der VNC-Verbindung angeben, dies ist

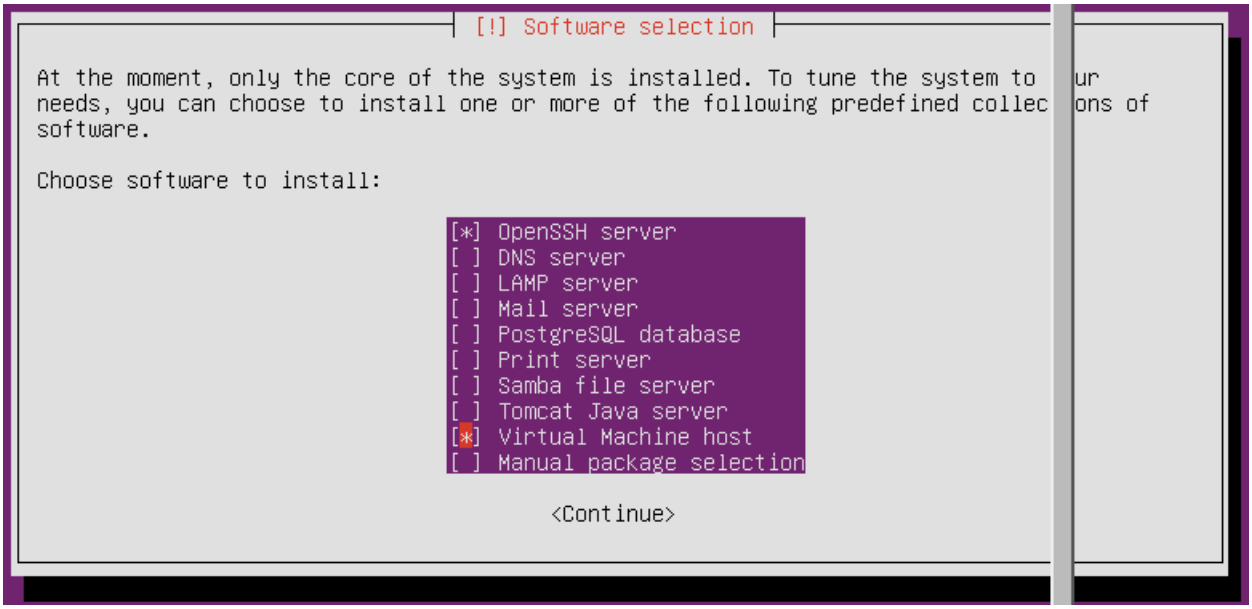


Abbildung 2: Auswählen der zu installierenden Software für den VM-Host.

hier nicht nötig, da die VNC-Verbindung später entfernt wird. Mit `-vcpu` wird die Anzahl der CPUs angeben, auf die die VM Zugriff erhält.

Listing 2: Erzeugen der VM für den Loginknoten.

```
vm-host:~$: virt-install --connect qemu:///system --name pcm-master --ram 500 --cdrom
  ./ubuntu-12.04-server-amd64.iso --os-type linux --os-variant ubuntuoneiric
  --boot cdrom --disk ./pcm-master.qcow2,size=20 --network bridge=virbr0
  --graphics vnc,keymap=de,listen=192.168.111.3 --vcpu 2
```

Listing 3: Auszug der Liste der OS-Varianten von KVM.

```
vm-host:~$ virt-install --os-variant list
  [...]
  ubuntuoneiric        : Ubuntu 11.10 (Oneiric Ocelot)
  ubuntunatty          : Ubuntu 11.04 (Natty Narwhal)
  ubuntumaverick       : Ubuntu 10.10 (Maverick Meerkat)
  ubuntulucid          : Ubuntu 10.04 (Lucid Lynx)
  ubuntukarmic         : Ubuntu 9.10 (Karmic Koala)
  ubuntujaunty         : Ubuntu 9.04 (Jaunty Jackalope)
  ubuntuintrepid       : Ubuntu 8.10 (Intrepid Ibex)
  ubuntuhardy          : Ubuntu 8.04 LTS (Hardy Heron)
  [...]
```

### 3.2.2 Installieren des Betriebssystems

Nun wird das Betriebssystem in der VM installiert. Falls die VM nicht automatisch gestartet wurde, kann diese mit `virsh`[8.2] (Listing 4) gestartet werden.

Listing 4: Starten einer VM mit Hilfe von virsh.

```
vm-host:~$ virsh list --all
  Id Name                 State
  ----------------------------------
  - pcm-master           shut off

vm-host:~$ virsh start pcm-master
  Domain pcm-master started
```

Da bei dem Erstellen der VM eine VNC Verbindung mit angegeben wurde, kann man sich nun mit einem VNC-Viewer auf die VM verbinden. Hierzu kann man den Port des VNC-Zugangs auslesen, falls noch andere VM's auf dem Host installiert sind (Listing 5).

Listing 5: Auslesen des VNC-Ports einer VM.

```
vm-host:~$ virsh vncdisplay pcm-master
  192.168.111.3:0
```

Nachdem die VM von dem CD-ISO gebootet wurde, kann die Installation von MAAS beginnen (Abbildung 3).

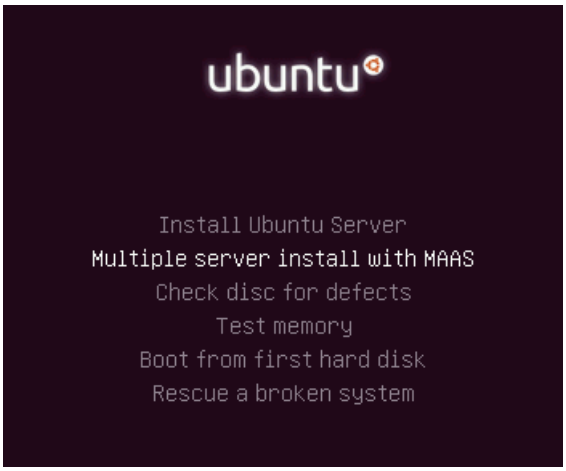


Abbildung 3: Starten der Installation von MAAS.

Zuerst wird der Hostname für den MAAS-Server eingetragen (Abbildung 4).

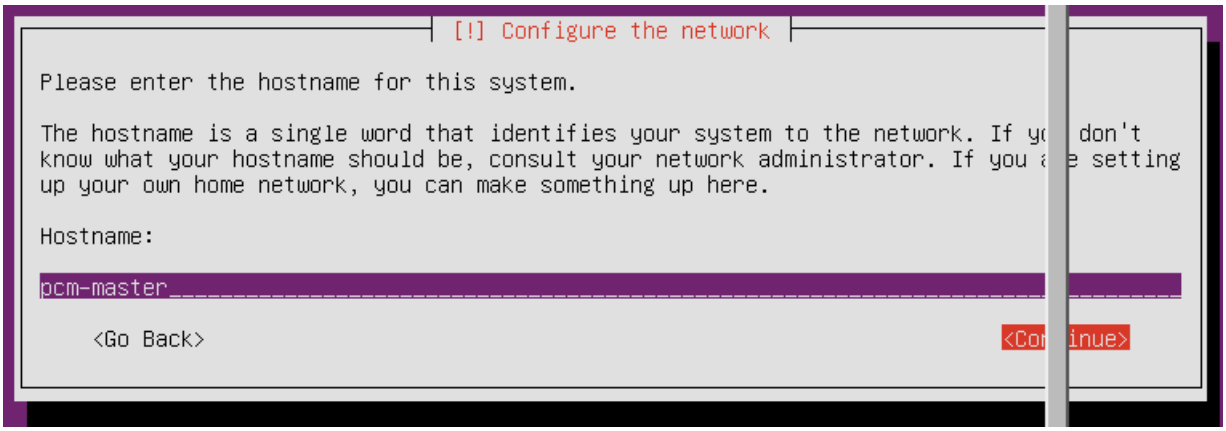


Abbildung 4: Eingeben des Hostnames für den MAAS-Server.

Nachdem man ausgewählt hat einen neuen MAAS-Server auf dem Rechner zu installieren, läuft die weitere Installation wie bei einer herkömmlichen Ubuntu Server Instanz ab (Abbildung 5).

Im Verlauf der Installation richtet MAAS den PXE-Server und den Zugriff auf das Webinterface automatisch ein. Hierzu verwendet das Installationsprogramm die aktuelle IP-Adresse, diese kann nach der Installation mit Hilfe von `dpkg` rekonfiguriert werden (Listing 6).

Listing 6: Rekonfigurieren des PXE-Servers.

```
pcm-master:~$ sudo dpgk-reconfigure maas
```

Nachdem die Installation abgeschlossen ist, wird noch `apcid` installiert und das System wird aktualisiert. Der ACPI-Daemon muss installiert werden, damit die VM auf die Steuersignale von virsh anspricht (Listing 7).

Listing 7: Installieren von acpid.

```
pcm-master:~$ sudo aptitude update && sudo aptitude install acpid -y && sudo aptitude safe-upgrade -y
```

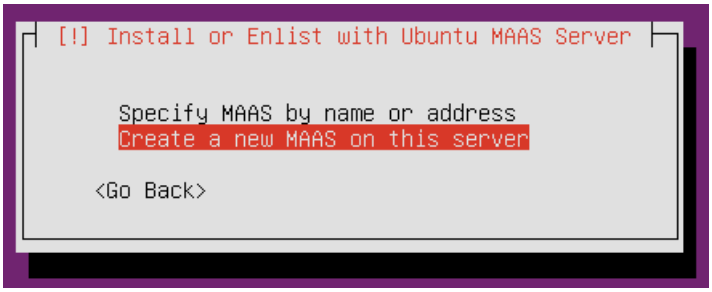


Abbildung 5: Einen neuen MAAS-Server installieren.

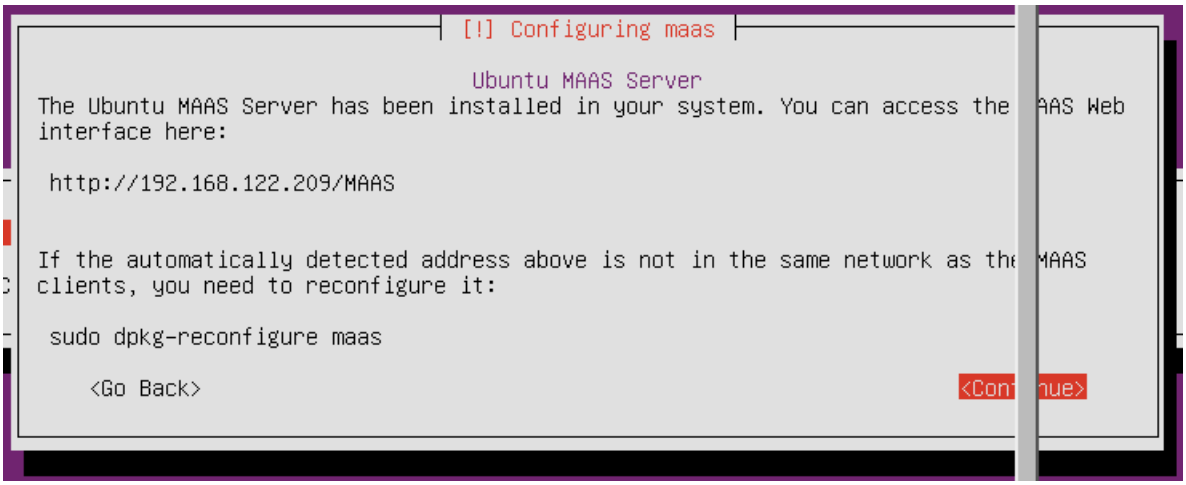


Abbildung 6: Automatisches Einrichten des MAAS-Servers.

Damit der MAAS-Server immer unter derselben IP-Adresse zu erreichen ist, muss die Netzwerkschnittstelle mit einer statischen Adresse versehen werden (Listing 8).

Listing 8: Konfigurieren der Netwerkschnitstelle des Login-Knoten.

```
pcm-master:~$ sudo vim /etc/network/interfaces

  [...]
  auto eth0
    iface eth0 inet static
      address 192.168.122.10
      netmask 255.255.255.0
      network 192.168.122.0
      bcast 192.168.122.255
      gateway 192.168.122.1
  [...]
```

Da der MAAS-Server auch als DHCP- und DNS-Server dient, müssen noch die DNS-Server eingetragen werden. Als primärer DNS-Server wird die eigene Instanz eingetragen. Als sekundärer DNS-Server wird die Netzwerkbrücke eingetragen, an die die VM angebunden ist, damit die Kommunikation nach außen funktioniert. Dazu werden diese in den Kopf der `resolvconf` geschrieben. Diese Variante ist nicht die eleganteste, funktioniert allerdings auch, wenn Zeroconf oder andere Mechanismen die DNS-Auflösung verändern (Listing 9). Dieser Teil der Konfigurationsdatei wird dann von dem System automatisch an den Anfang der Datei `/etc/resolv.conf` geschrieben.

Listing 9: Eintragen der DNS-Server.

```
pcm-master:~$ sudo vim /etc/resolvconf/resolv.conf.d/head

  [...]
  nameserver 192.168.122.10
  nameserver 192.168.122.1
```

```
6   [...]
```

Nachdem alle Einstellungen vorgenommen wurden, wird der MAAS-Server heruntergefahren. Da der ACPI-Daemon auf der VM erst nach einem Neustart aktiv wird und somit noch keine Signale an den VM-Host meldet, wird diese einmal manuell beendet. (Listing 10).

Listing 10: Beenden der VM.

```
1 vm-host:~$ virsh destroy pcm-master
```

Da in Zukunft der MAAS-Server DHCP übernimmt, muss noch der DHCP-Server der Netzwerkbrücke auf eine andere Netzwerkadresse konfiguriert werden, damit dieser nicht mit dem DHCP-Servers des MAAS-Servers konkurriert. Listing 11 zeigt eine Möglichkeit, bei der die DHCP-Range des DHCP-Server der libvirt verändert wird. Hierzu wird die XML-Datei[8.3], die die Netzwerkschnittstelle `virbr0` beschreibt, verändet werden. Bei der Standard Ubuntu Installation ist diese in virsh unter dem Namen `default` gespeichert.

Listing 11: Ändern des DHCP-Servers der `libvirt`.

```
1 vm-host:~$ virsh net-edit default
2
3   [...]
4   <dhcp>
5     <range start='192.168.99.2' end='192.168.99.254' />
6   </dhcp>
7   [...]
```

Nun kann der VNC-Zugang aus der VM des Clusters entfernt werden, da dieser ohne ein Passwort eine Sicherheitslücke darstellt und alle weiteren Arbeiten via SSH erledigt werden können. Die Zeile aus dem Listing 12 muss dazu aus der XML-Datei gelöscht werden.

Listing 12: Entfernen des VNC Zugangs.

```
1 vm-host:~$ virsh edit pcm-master
2
3   [...]
4   <graphics type='vnc' port='-1' autoport='yes' listen='192.168.111.3' keymap='de'/>
5   [...]
```

Jetzt wird die Netzwerkwerkbrücke neu gestartet um die Änderungen an den Einstellungen wirksam zu machen. Der Management-Knoten wird neu gebootet.

Listing 13: Neustarten des Login-Knoten und der Netzwerkbrücke.

```
1 vm-host:~$ virsh net-destroy default && virsh net-start default && virsh start pcm-master
```

Nachdem die VM gestartet ist, kann die Installation von MAAS abgeschlossen werden. Da sich die IP-Adresse des Servers geändert hat, muss dieser noch rekonfiguriert werden (Listing 14).

Listing 14: Rekonfigurieren des MAAS PXE-Servers.

```
1 pcm-master:~$ sudo dpkg-reconfigure maas
```

Abbildung 7 zeigt den Dialog, der geöffnet wird, um ggf. die IP-Adresse des PXE-Servers zu ändern. Es ist sicherzustellen, dass die IP-Adresse mit der IP-Adresse aus Listing 8 übereinstimmt.

Da der MAAS-Server auch das Management für DHCP und DNS übernimmt, muss noch das Paket `maas-dhcp` installiert werden (Listing 15).

Listing 15: Installieren des MAAS-DHCP-Servers.

```
1 pcm-master:~$ sudo aptitude install maas-dhcp -y
```

In dem erstem Dialog (Abbildung 8) wird die IP-Range spezifiziert, die für die von MAAS verwalteten Server bereit gestellt wird. Es ist darauf zu achten, dass die Clients in dem gleichen Netzwerk wie der Management-Knoten liegen und die Clients eine Verbindung zum Internet haben.

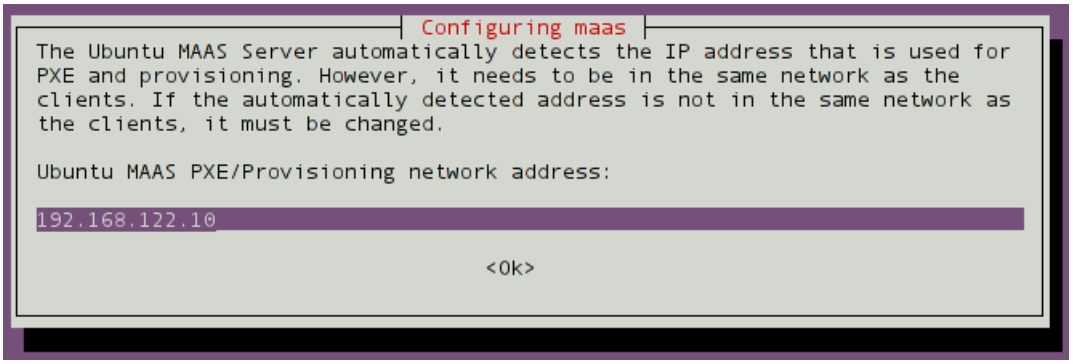


Abbildung 7: Rekonfigurieren des MAAS PXE-Servers.

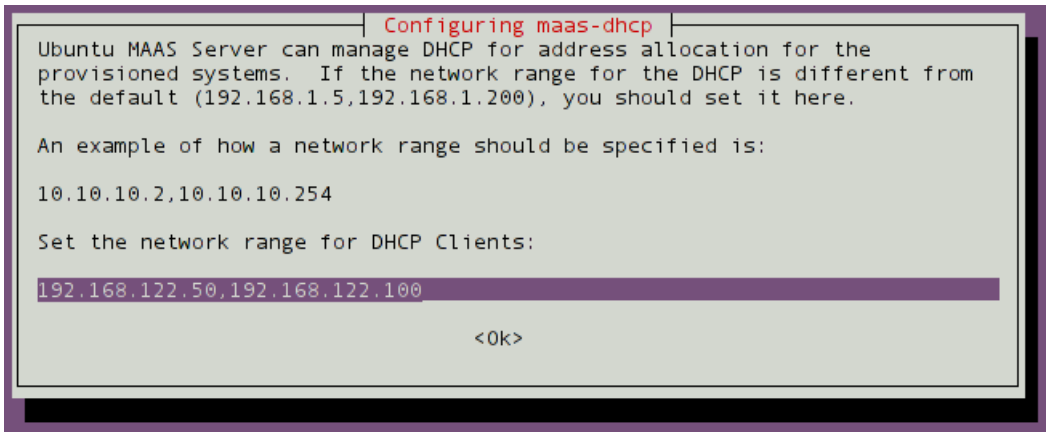


Abbildung 8: Setzen der DHCP-Range für die DHCP-Clients.

Als Gateway wird die Netzwerkbrücke, die die VMs mit dem VM-Host verbindet, gesetzt.

Um die Installation abzuschließen muss noch die Domain der Clients angegeben werden.

Es muss noch ein administrativer Benutzer für MAAS angelegt werden (Listing 16). Die Benutzerdaten werden später in der Installation noch benötigt.

Listing 16: Erstellen eines neuen MAAS Administrators.

```
pcm-master:~$ sudo maas createsuperuser
Username (Leave blank to use 'root'): maas
E-mail address: mail@example.com
Password:
Password (again):
Superuser created successfully.
```

Nun müssen noch die CD-Images heruntergeladen und in MAAS importiert werden. Diese sind Ubuntu Images, die verwendet werden, um das Grundsystem auf den Cloudservern zu installieren (Abbildung 17).

Listing 17: Importieren der Ubuntu-ISOS in MAAS.

```
pcm-master:~$ sudo maas-import-isos
```

## 3.3 Cobbler

MAAS setzt auf dem Linux Installations-Server cobbler auf. Dieser übernimmt auch die Konfiguration von dnsmasq, dem eingesetzten DHCP- und DNS-Server[5.1]. Die Konfiguration findet über Templates statt, die als Grundlage für die Konfiguration dienen. Die Konfiguration von dnsmasq befinde sich in `/etc/cobbler/dnsmasq.template`. Die Clound-Server verwenden den Managment-Knoten als DNS-Server. Damit dieser Anfragen weiterleitet, muss die Netwerkbrücke `virbr0` als weiterer DNS-Server in das Template von dnsmasq eingefügt werden (Listing 18).

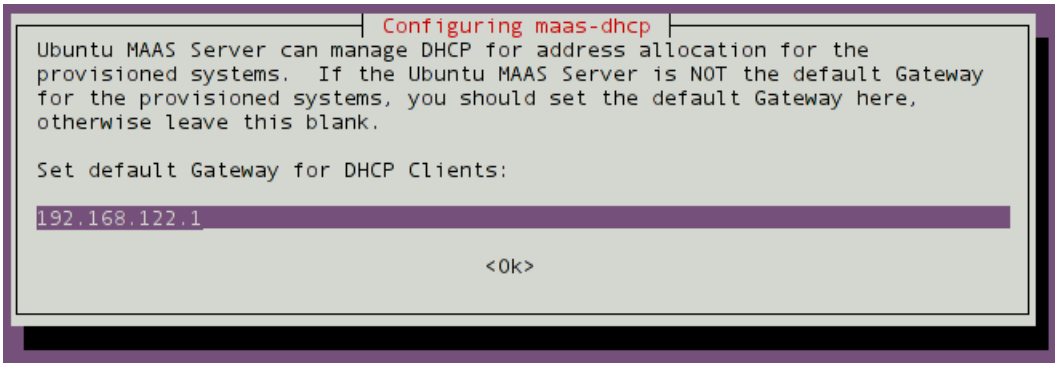


Abbildung 9: Setzten des Gateways für die Cloudserver.

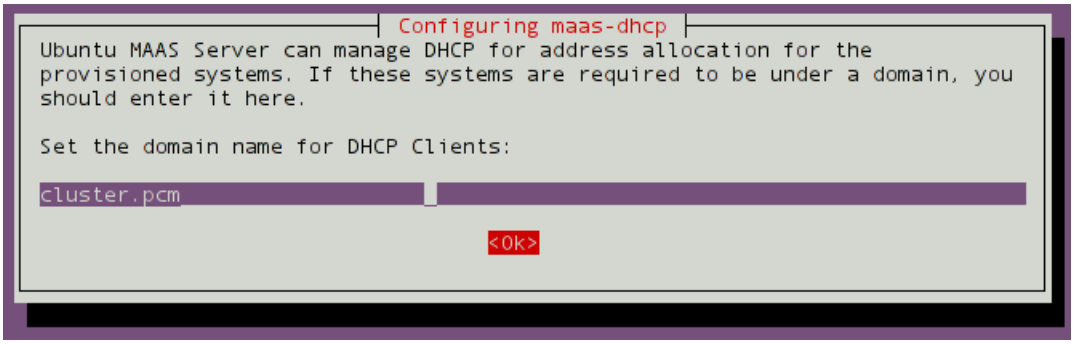


Abbildung 10: Setzten des Domainnamens der DHCP-Clients.

Listing 18: Hinzufügen des DNS-Servers in die DHCP Konfiguration.

```
pcm-master:~$ sudo vim /etc/cobbler/dnsmasq.template

  [...]
  server=192.168.122.1
  [...]
```

Die Cloud-Server werden von MAAS aus mit dem Administrator `ubuntu` und dem Passwort `ubuntu` versehen. Dieses sollte geändert werden. Das Passwort steht als verschlüsselter Hash in `/etc/cobbler/ubntu-server.preseed`.

Listing 19: Zur Instalation verwendeter Passwort-Hash.

```
pcm-master:~$  cat /etc/cobbler/ubuntu-server.preseed | grep user-password-crypted
d-i     passwd/user-password-crypted    password $6$.1eHH0iY$ArGzKX2YeQ3G6U.ml0O3A.NaL22Ewgz8Fi4qqz.
    Ns7EMKjEJRIW2Pm/TikDptZpuu7I92frytmk5YeL.9fRY4.
```

Um das Passwort zu ändern muss man den Passwort-Hash mit dem gewünschten Passwort neu generieren. Listing 20 zeigt ein kurzes Python Programm, das diese Aufgabe übernimmt. [1]. Das Programm verwendete den Salt aus dem Hash aus der Konfigurations Datei von MAAS und das Passwort `ubuntu`. Es wird derselbe Hash wie in der Konfigurationsdatei generiert. Man kann nun das Passwort und/oder den Salt ändern. Ändert man den Salt, sollte man die Formatierung `$6$<random>$` beibehalten. Den so generierten Hash kann man jetzt an der Stelle `d-i passwd/user-password-crypted` in die Konfigurationsdatei eintragen.

Listing 20: Pythonprogramm zur Generierung des Passwort-Hash.

```
python -c 'import crypt; print crypt.crypt("ubuntu","$6$.1eHH0iY$")'
$6$.1eHH0iY$ArGzKX2YeQ3G6U.ml0O3A.NaL22Ewgz8Fi4qqz.Ns7EMKjEJRIW2Pm/TikDptZpuu7I92frytmk5YeL.9fRY4.
```

Damit alle Änderungen in cobbler übernommen werden, muss das `sync` Kommando ausgeführt werden (Listing 21).

[1] Die Generierung mit `openssl` hat nicht funktioniert.

Listing 21: Übernehmen der Änderungen in `cobbler`.

```
pcm-master:~$ sudo cobbler sync
```

## 3.4 Juju

Juju ist ein Cloud-Deployment-System, das dafür verwendet werden kann Software auf einer Cloud Infrastruktur zu depolyen.

Listing 22 zeigt die Installation von Juju[6.1] durch aptitude. Die Konfiguration von Juju befindet sich in der Datei `/.juju/environments.yaml`, im Home-Verzeichniss des Benutzers, der Juju bedienen soll, es werden keine Root-Rechte benötigt. In der Konfigurationsdatei ist auf die Einrückung zu achten. Unter dem Eintrag `environments` können mehrere Umgebungen angeben werden. Der Eintrag `maas` spezifiziert den Namen der Umgebung und kann beliebig geändert werden. Dann wird der Typ der Umgebung angeben. Juju legt seine Daten auf einem WebDAV-Server ab, dieser wird mit `maas-server` angeben. Was bei dem `admin-secret` angegeben wird, ist zu diesem Zeitpunkt egal. Es muss angegeben werden, wird allerdings nicht weiter verwendet. Mit `default-series` wird die auf den Cloud-Knoten zu installierende Ubuntu Version angegeben. Der `maas-api-key` wird zur Authentifizierung von Juju gegen über MAAS verwendet[4.1]. Der Key muss mit Hilfe eines Browser aus MAAS ausgelesen werden. Das Webfrontend ist zu erreichen über `http://192.168.122.10/MAAS` (Abbildung 16), hier meldet man sich mit dem Benutzernamen und Passwort aus Listing 16 an. Oben Rechts klickt man auf `maas` und in dem Popup auf `Prefrences`. Danach kann man den API-Key kopieren (Listing 12).

Listing 22: Installieren von Juju.

```
pcm-master:~$ sudo aptitude install juju

pcm-master:~$ ssh-keygen -t rsa

pcm-master:~$ mkdir -p ~/.juju
pcm-master:~$ vim ~/.juju/environments.yaml

environments:
  maas:
    type: maas
    maas-server: 'http://192.168.122.10:80/MAAS'
    maas-oauth: '${maas-api-key}'
    admin-secret: 'nothing'
    default-series: precise
```

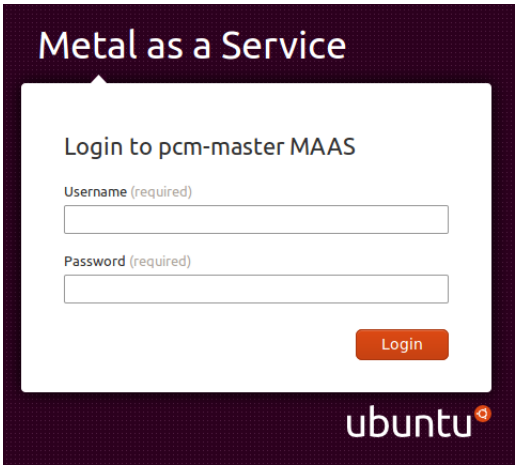


Abbildung 11: Login Bildschirm von MAAS.

## 3.5 Hinzufügen von Servern zu MAAS

Nun werden drei VMs erstellt, die als Cloudserver dienen sollen. Die erste VM mit dem Namen `juju` wird als Juju-Managment-Instanz verwendet und muss daher auch als erster Knoten zu MAAS hinzugefügt werden. Als

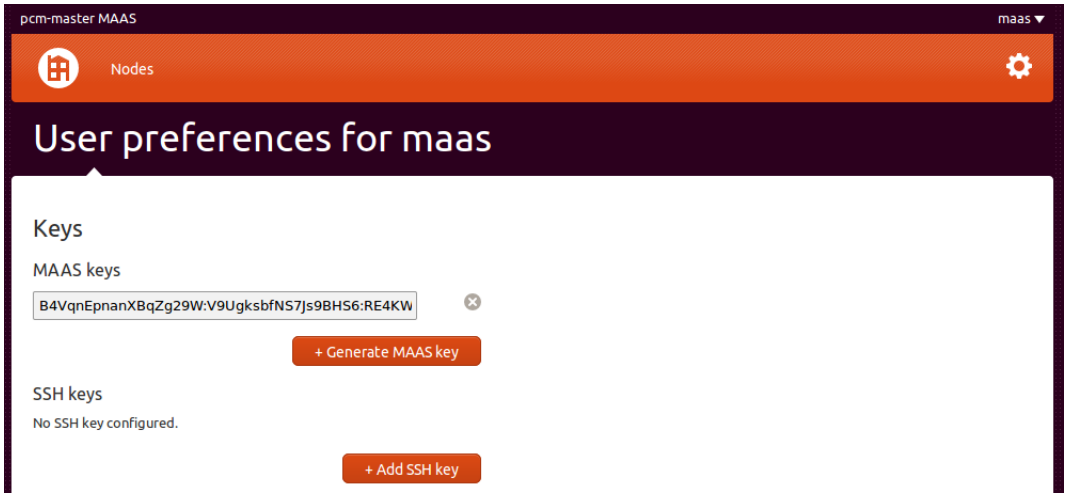


Abbildung 12: Auslesen des MAAS API Keys.

Knoten, auf die Services deployed werden können, werden `compute1` und `compute2` verwendet. Listing 23 zeigt das Erzeugen der VMs für die Cloud-Knoten. `-boot network` und –pxe sorgen dafür, dass die VM via Netzwerk bootet und nach einem PXE-Server sucht. Wurde die VM erzeugt, verlässt man mit der Tastenkombination `ctrl+]` die Konsole. Durch das angefügte `&& virsh destroy <VM-Name>` wird die VM beendet.

Listing 23: Hinzufügen des ersten Rechenknoten.

```
vm-host:~$ virt-install --connect qemu:///system --name juju --ram 500 --os-type linux --os-variant
    ubuntuoneiric --boot network --disk ./juju.qcow2,size=20 --network bridge=virbr0 --vcpu 2 --pxe &&
     virsh destroy juju

  Starting install...
  Creating storage file juju.qcow2                          |  20 GB     00:00
  Creating domain...                                        |    0 B     00:00
  Connected to domain juju
  Escape character is ^]

  Domain installation still in progress. You can reconnect to
  the console to complete the installation process.
  Domain juju destroyed

vm-host:~$ virt-install --connect qemu:///system --name compute1 --ram 500 --os-type linux --os-
    variant ubuntuoneiric --boot network --disk ./comput1.qcow2,size=20 --network bridge=virbr0 --vcpu
     2 --pxe && virsh destroy comput1

vm-host:~$ virt-install --connect qemu:///system --name comput2 --ram 500 --os-type linux --os-variant
     ubuntuoneiric --boot network --disk ./compute2.qcow2,size=20 --network bridge=virbr0 --vcpu 2 --
    pxe && virsh destroy compute2
```

Nachdem alle VMs erzeugt wurden, werden die MAC-Adressen ausgelesen (Listing 24) und die Knoten in MAAS eingetragen. `virsh dumpxml <VM-Name>` gibt die XML-Datei aus, die die VM beschreibt und es wird nach der MAC-Adresse gefiltert.

Listing 24: Auslesen der MAC-Adressen der VMs.

```
vm-host:~$ virsh dumpxml juju virsh dumpxml juju | grep 'mac address'
  <mac address='52:54:00:c2:59:83'/>

vm-host:~$ virsh dumpxml compute1 | grep 'mac address'
  <mac address='52:54:00:ac:9b:57'/>

vm-host:~$ virsh dumpxml compute2 | grep 'mac address'
  <mac address='52:54:00:f4:dd:02'/>
```

Das Hinzufügen der Server zu MAAS geschieht über das Webinterface[4.2] (Abbildung 11). Nachdem Login kommt man über den Menüpunkt `Nodes` zu der Auflistung von Servern innerhalb von MAAS (Abbildung 13).


Abbildung 13: Hinzufügen des Ersten Servers zu MAAS.

Hier klickt man auf den Punkt `+ Add node` und kommt zu dem Menü aus Abbildung 14. Hier trägt man die MAC-Adresse ein, wählt die Prozessorarchitektur aus und trägt einen Hostnamen für das Systems ein.. Es ist wichtig den Server zum Managen der Juju-Umgebung als erstes hinzuzufügen.

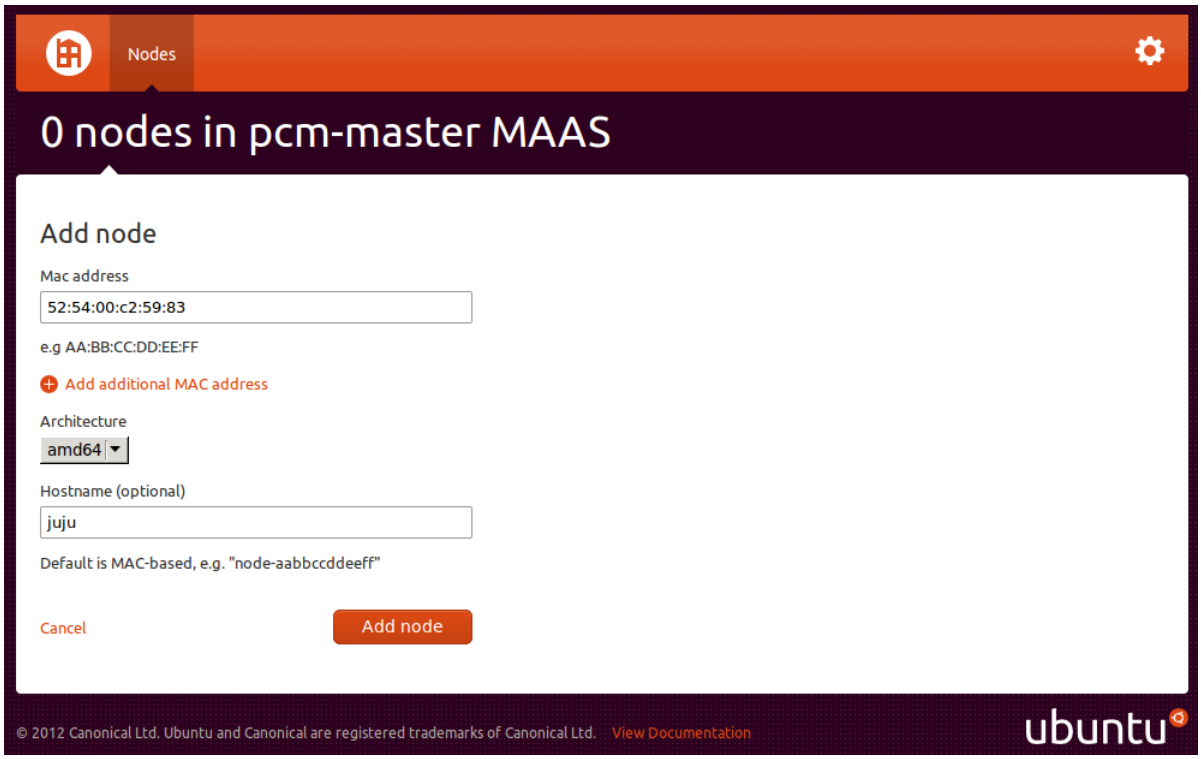


Abbildung 14: Hinzufügen des Cloud-Managment Servers zu MAAS.

Nachdem die drei Server hinzugefügt wurden, sollte die Übersichtsseite von MAAS wie in Abbildung 15 aussehen.

Damit alle Server von MAAS verwendet werden, müssen alle Server einmal gebootet werden, damit diese von MAAS in den Status `ready` versetzt werden. Das Booten muss von Hand geschehen, da KVM kein Wake on Lan unterstützt. Nachdem die Server von MAAS initialisiert wurden, werden diese automatisch abgeschaltet und sind bereit verwendet zu werden. Nachdem alle Server initialisiert wurden, sollte die Übersicht wie in Abbildung 16 aussehen.

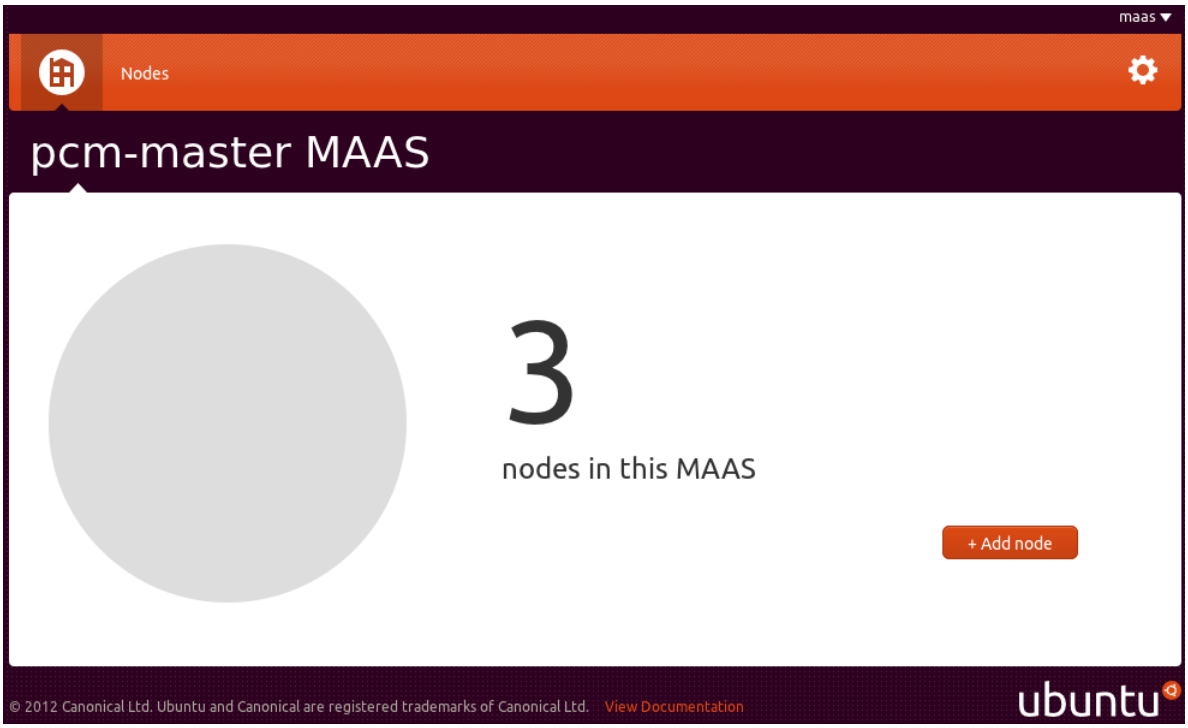


Abbildung 15: MAAS-Übersicht nach dem Hinzufügen der ersten drei Server.

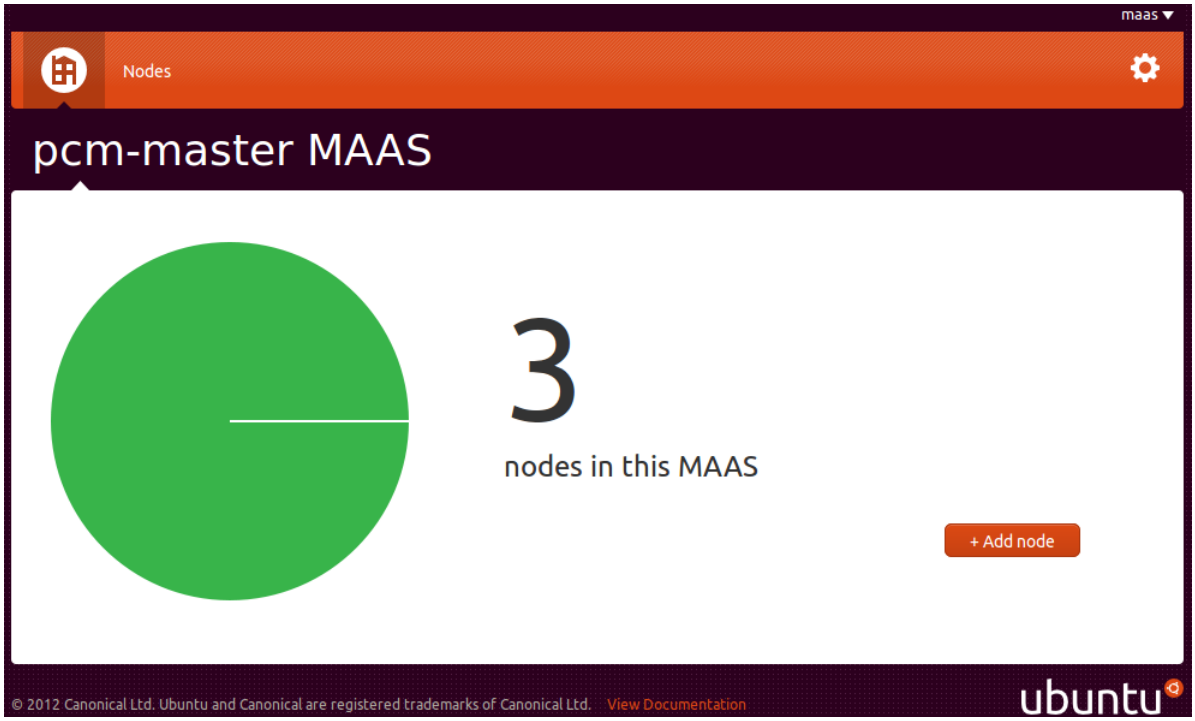


Abbildung 16: Übersicht von MAAS nach dem Initialisieren der Server.

## 3.6 Installieren von OAR

OAR[7] ist ein Batch-Scheduler. OAR besteht aus mehreren Modulen, die über eine Datenbank miteinander kommunizieren. Hierzu gehört der OAR-Server, der die Datenbank bereitstellt. Die Frontend-Node fungiert als Login-Server für die Benutzer des Clusters und stellt die Programme bereit um Jobs auf dem Cluster zu starten. Installiert wird OAR innerhalb des Cluster-Managment-Servers.

### 3.6.1 Installieren des OAR-Servers

Das Entwickler-Team von OAR stellt DEB-Pakete[7.1] bereit, die eigentlich für Debian gedacht sind, allerdings funktionieren diese auch unter Ubuntu. Listing 25 zeigt das Hinzufügen der Quellen zu apt und das Installieren der benötigten Pakete. Zuerst wird die Datei `oar.list` erzeugt, in die die Paketquelle geschreiben wird. Dann wird noch mit Hilfe von wget der Schlüssel zu apt hinzugefügt.

Listing 25: Hinzufügen der OAR-Paketquellen.

```
1 pcm-master:~$ echo "deb http://oar-ftp.imag.fr/oar/2.5/debian squeeze main" | sudo tee /etc/apt/
      sources.list.d/oar.list
2 pcm-master:~$ wget -O - http://oar-ftp.imag.fr/oar/oarmaster.asc | sudo apt-key add -
3
4 pcm-master:~$ sudo aptitude update
5 pcm-master:~$ sudo aptitude install oar-server oar-server-pgsql postgresql-9.1 taktuk oar-admin oar-
      restful oar-restful-api ruby-dbd-pg
```

Die Konfiguration von OAR ist in der Listing 26 zu sehen. Als Datenbank wird Postgresql verwendet. Da die Datenbank lokal auf dem Server liegt, kann als Hostname `localhost` eingetragen werden. Als Datenbank-Port wird der Standartport der Installation verwendet. Namen und Passwörter sind frei wählbar, diese werden bei der Einrichtung von OAR gesetzt. Das `OPENSSH_CMD` gibt an, wie sich der OAR-Server auf die OAR Rechenknoten verbindet. Die Rechenknoten starten einen SSH-Dienst auf dem Port `6667`, über den sich der OAR-Server via Public-Key verbinden kann. Um Befehle auf den Rechenknoten auszuführen wird TAKTUK verwendet. TAKTUK führt die Befehle auf allen Rechenknoten aus. Um festzustellen ob ein Rechnerknoten läuft wird NMAP verwendet. Da ein Rechnerknoten einen SSH-Server auf Port `6667` startet, wird via NMAP geprüft, ob der Rechnerknoten über diesen Port zu erreichen ist.

Listing 26: Anpassen der OAR-Konfiguration.

```
pcm-master:~$ vim /etc/oar/oar.conf

  # Database type ("mysql" or "Pg")
  DB_TYPE="Pg"

  # DataBase hostname
  DB_HOSTNAME="localhost"

  # DataBase port
  DB_PORT="5432"

  # Database base name

  DB_BASE_NAME="oar"

  # DataBase user name
  DB_BASE_LOGIN="oar"

  # DataBase user password
  DB_BASE_PASSWD="oarpwd"

  # DataBase read only user name
  DB_BASE_LOGIN_RO="oar_ro"

  # DataBase read only user password
  DB_BASE_PASSWD_RO="oar_ropwd"

  [...]

  # Command to use to connect to other nodes (default is "ssh" in the PATH)
  OPENSSH_CMD="/usr/bin/ssh -p 6667"

  [...]

  # If you have installed taktuk and want to use it to manage remote
  # admnistration commands then give the full command path
  # (with your options except "-m" and "-o").
  # You don t also have to give any taktuk command.
  # (taktuk version must be >= 3.6)
  TAKTUK_CMD="/usr/bin/taktuk -t 30 -s"

  [...]

  # nmap
  # nmap may be used instead of ping to check aliveness of nodes.
  # uncomment next line to use nmap. Give the complete command path.
  # It will test to connect on the ssh port (22)
  PINGCHECKER_NMAP_COMMAND="/usr/bin/nmap -p 6667 -n -T5"

  [...]
```

Ist die Konfiguration von OAR abgeschlossen, wird die Datenbank erzeugt werden. Dabei werden die in der Konfiguration angegebene Datenbank und die Benutzer erzeugt und die nötigen Berechtigungen gesetzt. (Listing 27).

Listing 27: Erstellen der Datenbank für OAR.

```
pcm-master:~$ sudo oar-database --create --db-is-local
```

### 3.6.2 Installieren des NFS-Server

Damit allen Rechenknoten auf das Home-Verzeichnis des Benutzers zugegriffen können, muss noch ein NFS-Server[9] installiert werden (Listing 28).

Listing 28: Installiern des NFS-Servers.

```
pcm-master:~$ sudo aptitude install sudo aptitude install nfs-kernel-server
```

Nachdem der NFS-Server installiert wurde, wird noch ein Verzeichnis für die Clusterbenutzer erstellt, das dann über NFS für die Rechenknoten freigeben wird. Der Benutzer, der mit OAR arbeiten soll, wird erstellt. Die Zeile aus Listing 29 muss dann noch in `/etc/exports` ergänzt werden. Der Eintrag `anonuid`, ist die ID des Benutzers, mit der die NFS-Clients auf dem Server arbeiten. Die ID des OAR-Bentuzers lässt sich mit dem Kommando `ID` herausfinden.

Listing 29: Einrichten der Freigabe für die Rechenknoten.

```
pcm-master:~$ sudo mkdir /home/clusterusers/
pcm-master:~$ sudo useradd --base-dir /home/clusterusers -m -p oaruser --shell /bin/bash oaruser
pcm-master:~$ id oaruser
  uid=1001(oaruser) gid=1001(oaruser) groups=1001(oaruser)

pcm-master:~$ sudo vim /etc/exports

[...]
/home/clusterusers/       192.168.122.0/255.255.255.0(rw,no_root_squash,async,all_squash,anonuid=1001)
[...]
```

Damit sich der OAR-Scheduler gegenüber den Rechenknoten authentifizieren kann, wird noch ein SHH-Key erzeugt und dieser in die Datei `authorized_keys` kopiert.

Listing 30: Anlegen des OAR-Benutzers.

```
pcm-master:~$ sudo su oaruser
oaruser@pcm-master:~$ ssh-keygen -t rsa
  Generating public/private rsa key pair.
  Enter file in which to save the key (/home/clusterusers/oaruser/.ssh/id_rsa):
  Enter passphrase (empty for no passphrase):
  Enter same passphrase again:

oaruser@pcm-master:~$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

## 3.7 Installieren der Rechnerknoten

Die Rechenknoten werden mit Hilfe von Juju deployed. Juju deployed sogenannte Charms[6.2]. Ein Charm beinhaltet die Installationsanweisungen um die Programme auf den Knoten zu installieren und einzurichten. Es gibt noch mehrere Möglichkeiten, z.B. Relationen zwischen Prozessen zu definieren, allerdings werden diese Features hier nicht benötigt und daher wird nicht weiter darauf eingegangen.

Das Modul für die Rechenknoten von OAR nennt sich `oar-node`. Dieses Modul muss nun mit Hilfe von Juju auf den Cloudknoten installiert und eingerichtet werden.

## 3.8 Erstellen des Charms

Um einen Charm zu erstellen[6.3] muss zuerst die benötigte Ordnerstruktur erzeugt werden, in der Form `repository/<ubuntu-release>/charm/` (Listing 31). Danach kann der Charm erstellt werden. Der Charm bekommt den gleichen Namen wie das zu installierende Modul.

Listing 31: Erstellen des Charms für Juju.

```
pcm-master:~$ mkdir ~/charms/precise
pcm-master:~$ charm create ~/charms/precise/oar-node
```

Der Charm, der hier erstellt wird, repräsentiert einen Rechenknoten als ganzes, da es zu diesem Zeitpunkt nur möglich ist einen Charm pro Server zu deployen. Innerhalb dieses Charms müssen also alle Einstellungen für den Rechenknoten vorgenommen werden. Innerhalb dieses Charms wird `oar-node`, `mpich2` und `nfs-common` installiert. OAR benötigt dieselbe Konfiguration und denselben Benutzer mit identischen SSH-Keys auf allen Servern.

Zuerst wird die nötige Ordnerstruktur innerhalb des Charms erzeugt. In `/charms/precise/oar-node/etc/oar` befinden sich die zu kopierenden Konfigurationsdateien für OAR. In `/charms/precise/oar-node/var/lib` wird das Home-Verzeichniss des Benutzers `oar` kopiert und dann mit Hilfe des Charms auf die Rechenknoten verteilt. Symbolische Links sind leider nicht möglich.

Listing 32: Kopieren der benötigten Daten in den Charm.

```
pcm-master:~$ mkdir -p ~/charms/precise/oar-node/etc/oar
pcm-master:~$ mkdir -p ~/charms/precise/oar-node/var/lib
pcm-master:~$ sudo cp /etc/oar/oar.conf ~/charms/precise/oar-node/etc/oar/
pcm-master:~$ sudo cp -r /var/lib/oar ~/charms/precise/oar-node/var/lib/
pcm-master:~$ sudo chown weging:weging ~/charms/precise/oar-node/etc/oar/oar.conf
pcm-master:~$ sudo chown -R weging:weging ~/charms/precise/oar-node/var/lib/oar
```

Der Ordner `hooks`, innerhalb des Charms, enthält alle Skripte, die von Juju ausgeführt werden können. Das erste Skript, das von Interesse ist, ist das `install` Skript. Hier werden alle Befehle definiert, die zur Installation benötigt werden. Listing 33 zeigt das fertige Installationsskript für eine OAR-Node[7.1]. Zu erst muss der HTTP-Proxy von apt abgeschaltet werden, da sonst die Installation aus den OAR-Paketquellen fehlschlägt. Eine elegantere Methode wäre den Proxy so zu konfigurieren, dass dieser die fremden Quellen zulässt. Der Proxy ist der MAAS-Server. Danach werden wie bei dem OAR-Server die Paketquellen in apt eingetragen. Nun wird `oar-node`, `mpich2` und `nfs-common` installiert. Nun werden die Daten von OAR kopiert. Das Home-Verzeichnis für den Cluster Benutzer wird erzeugt und via NFS gemountet und der Benutzer wird angelegt.

Listing 33: Instalationsskript OAR-Node.

```
pcm-master:~$ vim ~/charms/precise/oar-node/hooks/install

  #!/bin/bash
  # Here do anything needed to install the service
  #i.e. apt-get install -y foo  or  bzr branch http://myserver/mycode /srv/webroot

  echo "" > /etc/apt/apt.conf
  echo "deb http://oar-ftp.imag.fr/oar/2.5/debian squeeze main" | sudo tee /etc/apt/sources.list.d/oar
      .list

  wget -O - http://oar-ftp.imag.fr/oar/oarmaster.asc | sudo apt-key add -

  apt-get -y update
  apt-get install -y oar-node
  apt-get install -y mpich2
  apt-get install -y nfs-common

  cp etc/oar/oar.conf /etc/oar/oar.conf
  chown root:root /etc/oar/oar.conf

  cp -r var/lib/oar /var/lib
  chown -R oar:oar /var/lib/oar

  mkdir -p /home/clusterusers/oaruser

  mount -orw,hard,intr 192.168.122.10:/home/clusterusers /home/clusterusers

  useradd --base-dir /home/clusterusers --no-create-home --shell /bin/bash oaruser
```

Um den Service auf der OAR-Node zu starten oder zu stoppen werden die Skripte `start` und `stop` ausgeführt (Listing 34).

Listing 34: Start- und Stop Skript des Charms.

```
pcm-master:~$ vim ~/charms/precise/oar-node/hooks/start

  #!/bin/bash
  # Here put anything that is needed to start the service.
  # Note that currently this is run directly after install
  # i.e. 'service apache2 start'
  mount -orw,hard,intr 192.168.122.10:/home/clusterusers /home/clusterusers
```

```
  service oar-node start

pcm-master:~$ vim ~/charms/precise/oar-node/hooks/stop

  #!/bin/bash
  # This will be run when the service is being torn down, allowing you to disable
  # it in various ways..
  # For example, if your web app uses a text file to signal to the load balancer
  # that it is live... you could remove it and sleep for a bit to allow the load
  # balancer to stop sending traffic.
  # rm /srv/webroot/server-live.txt && sleep 30
  umount /home/clusterusers
  service oar-node stop
```

Werden Einstellungen an dem OAR-Server vorgenommen, müssen auch diese an die Cloudknoten propagiert werden. Dazu wird das Skript `upgrade-charm` ausgeführt. Listing 35 zeigt das Upgrade-Skript. Es wird das Installationsskript erneut ausgeführt und die OAR-Node neu gestartet.

Listing 35: Das Upgrade-Skript des Charms.

```
pcm-master:~$ vim ~/charms/precise/oar-node/hooks/upgrade-charm

  umount /home/clusterusers/oaruser
  ./install
```

Damit sind alle nötigen Einstellungen für Juju vorgenommen.

## 3.9 Hinzufügen der Ressourcen zu OAR.

Zuletzt müssen noch die Rechenknoten zu OAR hinzugefügt werden[7.1].

Listing 36: Hinzufügen von Ressourcen zu OAR.

```
pcm-master~$ oarnodesetting -a -h compute1.cluster.pcm
pcm-master~$ oarnodesetting -a -h compute2.cluster.pcm
```

# 4 Benutzung des Clusters als Administrator

Die Konfiguration des Cluster ist damit abgeschlossen. Nun kann die Cloud-Infrastruktur in Betrieb genommen werden. Dieses Kapitel wird zuerst die Funktionsweise eines Tools beleuchten und dann den Umgang anhand einiger Beispiele zeigen. Dabei wird versucht auf mögliche Fehlerquellen und Fallgruben hinzuweisen.

## 4.1 Juju

### 4.1.1 Funktionsweise von Juju

Alle Server, die in MAAS eingebunden werden, sind nicht installiert. Erst wenn ein Service in die Cloud deployed wird, wird der benötigte Server gestartet, das Betriebssystem installiert und dann der Service installiert. Dabei werden die Server in der Reihenfolge verwendet, wie sie in MAAS hinzugefügt wurden. Die erste Instanz, die von Juju automatisch installiert wird, ist eine reine Utility-Instanz. Diese verwaltet die weiteren Server, die der Cloud-Umgebung zur Verfügung gestellt werden. Außerdem werden hier alle Daten, die zum Installieren der Services benötigt werden, gespeichert. Da diese Utility-Instanz nicht viel Rechenleistung benötigt, sollte hier kein kompletter Server verwendet werden, es reicht eine VM. Da auf dieser Instanz Zookeeper zum Verwalten der anderen Knoten läuft, der in Java geschrieben ist, sollte die VM über mindestens 1 GB RAM verfügen.

Juju arbeitet mit Charms um einen Service zu deployen. Alle Daten innerhalb des Verzeichnisses des Charms werden auf den Cloud-Management-Server [2] kopiert und werden von da aus auf die Cloudknoten verteilt. Der einfachste Weg um die benötigten Dateien von OAR auf die Cloudknoten zu deployen ist diese in den Charm zu

[2] In diesem Fall Juju.

kopieren und diese von dort aus auf die Cloudknoten zu kopieren. Zuerst müssen also die nötigen Daten in den Ordner des Charms kopiert werden (Listing 32). Symbolische Links auf die Dateien funktionieren leider nicht.

Juju verwendet den Public-Key des Benutzers, der Juju bedient, um sich gegenüber der Juju-Utility-Instanz `juju` zu authentifizieren. Verbindet sich Juju das erste Mal zu der Utility-Instanz, wird der SSH-Fingerprint in `/.ssh/know_hosts` eingetragen. Erstellt man die Umgebung von Juju ein zweites Mal, muss man den veralteten Eintrag aus dieser Datei löschen. Des Weiteren darf das SSH-Schlüsselpaar des Juju Benutzers nicht geändert werden. Dies führt dazu, dass die Umgebung neu erzeugt werden muss.

### 4.1.2 Erzeugen einer Juju-Umgebung

Nachdem alle nötigen Installationen und Einstellungen vorgenommen wurden, kann das Cluster in Betrieb genommen werden. Hierzu muss erst die Juju-Umgebung erzeugt werden (Listing 37)[6.4]. Dadurch fordert Juju den ersten Server von MAAS an um den Juju-Managment-Knoten zu installieren. Dieser muss von Hand gestartet werden. Mit Hilfe des `status` Kommandos aus Listing 42 lässt sich überprüfen, ob die Umgebung fertig installiert ist. Wird der Fehler angezeigt, dass der Server noch nicht zu erreichen ist, ist dieser noch nicht fertig installiert. Wird bei dem Erzeugen der Umgebung nicht der Juju-Managment-Knoten von Juju angefordert, muss die Umgebung so lange zerstört und wieder neu erzeugt werden, bis Juju den richtigen Server verwendet (Listing 39).

Listing 37: Starten der Juju-Umgebung.

```
pcm-master:~$ juju bootstrap

2012-06-12 02:48:42,709 INFO Bootstrapping environment 'cluster' (origin: distro type: maas)...
2012-06-12 02:48:47,008 INFO 'bootstrap' command finished successfully
```

Listing 38: Statusabfrage der Juju-Umgebung.

```
pcm-master:~$ juju status
  2012-06-12 02:47:14,658 INFO Connecting to environment...
  2012-06-12 02:47:21,267 INFO Connected to environment.
  machines:
    0:
      agent-state: running
      dns-name: juju
      instance-id: /MAAS/api/1.0/nodes/node-f3f87412-a81a-11e1-b097-52540008fc88/
      instance-state: unknown
  services: {}
  2012-06-12 02:47:21,410 INFO 'status' command finished successfully
```

Listing 39: Zerstören der Juju-Umgebung.

```
pcm-master:~$ juju destroy-environment
WARNING: this command will destroy the 'cluster' environment (type: maas).
This includes all machines, services, data, and other resources. Continue [y/N]y
2012-06-12 02:52:46,616 INFO Destroying environment 'cluster' (type: maas)...
2012-06-12 02:52:54,526 INFO 'destroy_environment' command finished successfully
```

### 4.1.3 Deployen von Services

Ist die Umgebung korrekt erzeugt, kann eine OAR-Node deployed werden (Listing 40). Mit `-repository` wird der Pfad zu den Charms angegeben. Das `local:` vor dem Namen des Charms gibt an, dass sich der Charm lokal auf der Platte befindet und nicht aus dem Internet herunter geladen werden muss. Nun muss der angeforderte Server wieder von Hand gestartet werden. Mit Hilfe des Status-Kommandos wird überwacht, wann die Installation abgeschlossen ist.

Listing 40: Deployen der OAR-Node.

```
juju deploy --repository ~/charms local:oar-node
  2012-06-14 16:26:01,161 INFO Searching for charm local:precise/oar-node in local charm repository: /
      home/weging/charms
```

```
  2012-06-14 16:26:01,220 INFO Connecting to environment...
  2012-06-14 16:26:07,668 INFO Connected to environment.
  2012-06-14 16:26:08,699 INFO Charm deployed as service: 'oar-node'
  2012-06-14 16:26:08,709 INFO 'deploy' command finished successfully
```

Möchte man einen weiteren Server als OAR-Node deployen, kann dies mit Hilfe des `add-unit` Befehls geschehen, zusätzlich kann man noch die Anzahl spezifizieren, wie viele Server angefordert werden. Die angeforderten Server müssen wieder von Hand gestartet werden.

Listing 41: Hinzufügen von Einheiten zu dem OAR-Node Service.

```
pcm-master:~$ juju add-unit --num-units 1 oar-node
  2012-06-15 22:32:30,148 INFO Connecting to environment...
  2012-06-15 22:32:37,982 INFO Connected to environment.
  2012-06-15 22:32:38,629 INFO Unit 'oar-node/1' added to service 'oar-node'
  2012-06-15 22:32:38,633 INFO 'add_unit' command finished successfully
```

Sind alle Server fertig installiert, sieht die Statusabfrage von Juju wie in Listing 42 aus.

Listing 42: Juju-Status nach dem Deployen der Server.

```
pcm-master:~$ juju status
  2012-06-18 18:07:15,566 INFO Connecting to environment...
  2012-06-18 18:07:22,106 INFO Connected to environment.
  machines:
    0:
      agent-state: running
      dns-name: juju
      instance-id: /MAAS/api/1.0/nodes/node-f3f87412-a81a-11e1-b097-52540008fc88/
      instance-state: unknown
    1:
      agent-state: running
      dns-name: compute2
      instance-id: /MAAS/api/1.0/nodes/node-333788ce-b1a4-11e1-a33d-52540008fc88/
      instance-state: unknown
    2:
      agent-state: running
      dns-name: compute1
      instance-id: /MAAS/api/1.0/nodes/node-106efec6-b1a4-11e1-8389-52540008fc88/
      instance-state: unknown
  services:
    oar-node:
      charm: local:precise/oar-node-1
      relations:
        relation-name:
        - oar-node
      units:
        oar-node/0:
          agent-state: started
          machine: 1
          public-address: compute2.localdomain
        oar-node/2:
          agent-state: started
          machine: 2
          public-address: compute1.localdomain
  2012-06-18 18:07:22,660 INFO 'status' command finished successfully
```

Um Server aus der Umgebung zu entfernen sind zwei Schritte notwendig. Zuerst muss der Service von dem gewünschtem Server entfernt werden. Dies geschieht mit Hilfe von `remove-unit` (Listing 43). Hierdurch wird der Service von dem Server entfernt, der Server bleibt allerdings präsent. Als Parameter wird der Servicename inklusive seiner Unit-Nummer angegeben. Möchte man den Server Compute2 entfernen, muss die Unit `oar-node/0` entfernt werden.

Listing 43: Entfernen eines Service von einem Server.

```
pcm-master:~$ juju remove-unit oar-node/0
  2012-06-18 18:33:38,693 INFO Connecting to environment...
  2012-06-18 18:33:45,111 INFO Connected to environment.
  2012-06-18 18:33:45,305 INFO Unit 'oar-node/0' removed from service 'oar-node'
  2012-06-18 18:33:45,310 INFO 'remove_unit' command finished successfully
```

Nun kann der Server aus der Cloud entfernt werden. Dies geschieht durch den Befehl `terminate-maschine`. Hierzu gibt man die ID des Servers an, die man von dem Statuskommando erhält (Listing 44).

Listing 44: Entfernen eines Servers aus der Umgebung.

```
pcm-master:~$ juju terminate-machine 1
  2012-06-18 18:35:11,727 INFO Connecting to environment...
  2012-06-18 18:35:18,252 INFO Connected to environment.
  2012-06-18 18:35:18,343 INFO Machines terminated: 1
  2012-06-18 18:35:18,345 INFO 'terminate_machine' command finished successfully
```

### 4.1.4 Upgrades und Debugging

Möchte man einen Charm updaten[6.5], geschieht dies über das Kommando `upgrade-charm` (Listing 45). Ein Upgrade ist notwendig, wenn man etwas an den Einstellungen oder den Daten des Charms geändert hat.

Listing 45: Upgrade eines Charms.

```
pcm-master:~$ juju upgrade-charm --repository ~/charms/ oar-node
  2012-07-02 22:59:22,353 INFO Connecting to environment...
  2012-07-02 22:59:28,794 INFO Connected to environment.
  2012-07-02 22:59:28,865 INFO Setting /home/weging/charms/precise/oar-node to revision 22
  2012-07-02 22:59:29,435 INFO 'upgrade_charm' command finished successfully
```

Es ist möglich, dass ein Charm fehlerhaft ist. Um ein Kommando von Juju zu überwachen kann man das `debug-log` (Listing 46) verwenden. Hierzu öffnet man zuerst eine Konsole, in der man das Debug-Log anzeigen möchte, und dann eine zweite Konsole in der man das Juju-Kommando ausführt. Die Ausgabe Cloudknoten ist nun in dem Log zu sehen. In diesem Log wird die Ausgabe aller angezeigt. Es ist daher ratsam ein Kommando nur auf einem Cloudknoten auszuführen.

Listing 46: Das Debug-Log von Juju.

```
pcm-master:~$ juju debug-log
  2012-07-06 14:15:03,955 INFO Connecting to environment...
  2012-07-06 14:15:10,364 INFO Connected to environment.
  2012-07-06 14:15:10,365 INFO Enabling distributed debug log.
  2012-07-06 14:15:10,375 INFO Tailing logs - Ctrl-C to stop.
  [...]
```

### 4.1.5 Fehlerbehebung

Schlägt das Upgraden oder das Installieren eines Charms fehl, werden die Cloudknoten in einen Fehlerzustand versetzt. Nach einem fehlgeschlagenen Upgrade ist dieser: `charm-upgrade-error` (Listing 47).

Listing 47: Juju Statusausgabe nach einem fehlgeschlagenen Upgrade.

```
pcm-master:~$ juju status
  2012-07-06 14:24:39,089 INFO Connecting to environment...
  2012-07-06 14:24:46,028 INFO Connected to environment.
  machines:
    0:
      agent-state: not-started
      dns-name: juju
      instance-id: /MAAS/api/1.0/nodes/node-f3f87412-a81a-11e1-b097-52540008fc88/
      instance-state: unknown
    2:
      agent-state: running
```

```
      dns-name: compute1
      instance-id: /MAAS/api/1.0/nodes/node-106efec6-b1a4-11e1-8389-52540008fc88/
      instance-state: unknown
    4:
      agent-state: running
      dns-name: compute2
      instance-id: /MAAS/api/1.0/nodes/node-333788ce-b1a4-11e1-a33d-52540008fc88/
      instance-state: unknown
  services:
    oar-node:
      charm: local:precise/oar-node-23
      relations:
        relation-name:
        - oar-node
      units:
        oar-node/2:
          agent-state: charm-upgrade-error
          machine: 2
          public-address: compute1.localdomain
        oar-node/3:
          agent-state: charm-upgrade-error
          machine: 4
          public-address: compute2.localdomain
  2012-07-06 14:24:46,646 INFO 'status' command finished successfully
```

Hat man den Fehler in dem Charm gefunden und diesen behoben, kann man mit dem Befehl `juju resolved` (Listing 48) den Fehler als behoben markieren und Juju führt das Upgrade automatisch erneut aus.

Listing 48: Den Upgradefehler als behoben markieren.

```
juju resolved oar-node/2
  2012-07-06 14:36:00,991 INFO Connecting to environment...
  2012-07-06 14:36:07,420 INFO Connected to environment.
  2012-07-06 14:36:07,495 INFO Marked unit 'oar-node/2' as resolved
  2012-07-06 14:36:07,497 INFO 'resolved' command finished successfully
pcm-master:~$ juju resolved oar-node/3
  2012-07-06 14:36:24,713 INFO Connecting to environment...
  2012-07-06 14:36:31,066 INFO Connected to environment.
  2012-07-06 14:36:31,126 INFO Marked unit 'oar-node/3' as resolved
  2012-07-06 14:36:31,128 INFO 'resolved' command finished successfully
```

Wenn der Fehler behoben wurde und Juju das Upgrade erneut ausgeführt hat, sieht die Statusabfrage wieder wie gewohnt aus (Listing 49).

Listing 49: Die Status abfrage von Juju nach dem Beheben des Fehlers.

```
pcm-master:~$ juju status
  2012-07-06 14:36:37,245 INFO Connecting to environment...
  2012-07-06 14:36:43,793 INFO Connected to environment.
  machines:
    0:
      agent-state: not-started
      dns-name: juju
      instance-id: /MAAS/api/1.0/nodes/node-f3f87412-a81a-11e1-b097-52540008fc88/
      instance-state: unknown
    2:
      agent-state: running
      dns-name: compute1
      instance-id: /MAAS/api/1.0/nodes/node-106efec6-b1a4-11e1-8389-52540008fc88/
      instance-state: unknown
    4:
      agent-state: running
      dns-name: compute2
      instance-id: /MAAS/api/1.0/nodes/node-333788ce-b1a4-11e1-a33d-52540008fc88/
      instance-state: unknown
  services:
```

```
    oar-node:
      charm: local:precise/oar-node-23
      relations:
        relation-name:
        - oar-node
      units:
        oar-node/2:
          agent-state: started
          machine: 2
          public-address: compute1.localdomain
        oar-node/3:
          agent-state: started
          machine: 4
          public-address: compute2.localdomain
  2012-07-06 14:36:44,444 INFO 'status' command finished successfully
```

## 4.2 OAR

### 4.2.1 Funktionsweise von OAR

OAR startet auf jedem Rechenknoten einen SSH-Server, auf den sich der Scheduler verbindet, um die Jobs zu starten. Um zu prüfen ob, ein Rechenknoten zur Verfügung steht, wird NMAP verwendet. Dabei wird geprüft, ob die in die OAR-Datenbank eingetragenen Knoten auf dem Port des von OAR gestartetem SSH-Servers zu erreichen sind.

OAR verwaltet die Rechenknoten über Propperties, die in der Datenbank von OAR gespeichert werden. Es ist somit möglich die Anzahl der CPUs eines Rechenknotens in der Datenbank zu speichern und später bei dem Submitten eines Jobs als Voraussetzung an zugeben.

### 4.2.2 Verwalten von Rechenknoten

Um Rechenknoten komfortabler zu verwalten wird das Programm oaradmin verwendet. Installiert wird es über das Paket `oar-admin`[7.2]. Mit diesem Tool ist es möglich mehrere Rrechenknoten auf einmal zu generieren. Listing 50 fügt zehn Rechenknoten in die Datenbank ein, `compute1.cluster.pcm` bis `compute10.cluster.pcm`. Jeder Rechenknoten verfügt über zwei CPUs und jede CPU über sechs Cores.

Listing 50: Mehrere Rechenknoten auf einmal hinzufügen.

```
pcm-master:~$ sudo oaradmin resources -a "/node=compute{10}.cluster.pcm/cpu={2}/core={6}" | sudo bash
```

Die Properties `cpu` und `cores` sind nicht standardmäßig in der OAR Datenbank vorhanden. Diese müssen erst wie in Listing 51 in der Datenbank angelegt werden. Es ist somit möglich Rechenknoten beliebige Eigenschaften zu zusprechen.

Listing 51: Hinzufügen von Properties in die Datenbank von OAR.

```
pcm-master:~$ sudo oarproperty -a cpu
pcm-master:~$ sudo oarproperty -a core
```

Es sollte auf jeden Fall ein Blick auf die Seite der möglichen Anpassungen von OAR geworfen werden[7.2]. Hier ist unter anderem beschrieben wie man abgestürzte Rechenknoten ńeu starten kann.

# 5 Benutzung des Clusters aus Benutzersicht

Es wird an einem einfachen Beispiel gezeigt, wie man einen Job auf dem Cluster startet und dessen Status überprüft[7.3]. Zuerst muss eine kleine Veränderung an der SSH-Konfiguration vorgenommen werden, damit die SSH-Fingerprints automatisch in die `known_hosts` geschrieben werden (Listing 52).

Listing 52: Anpassen der SSH Konfiguration.

```
oaruser@pcm-master:~$ echo "StrictHostKeyChecking no" > .ssh/config
```

Danach wird überprüft ob die Rechenknoten zur Verfügung stehen (Listing 53).

Listing 53: Status der Rechenknoten überprüfen.

```
oaruser@pcm-master:~$ oarnodes -s
compute1.cluster.pcm
    1 : Alive
compute2.cluster.pcm
    2 : Alive
```

Listing 54 zeigt eine Jobdatei mit dem nötigen `mpiexec`.

Listing 54: Ausführen des Testprogrammes mit `mpiexec`.

```
oaruser@pcm-master:~$ vim job.sh
  mpiexec -hosts compute1.cluster.pcm,compute2.cluster.pcm -n 2 ./hostname.sh

oaruser@pcm-master:~$ chmod +x job.sh
```

Die Datei `hostname.sh` ist das auszuführende Programm, in diesem Fall wird einfach nur der Hostname des Rechenknoten ausgegeben (Listing 55).

Listing 55: Programm zum Ausgeben des Hostnamens.

```
oaruser@pcm-master:~$ vim hostname.sh
  #!/bin/bash
  echo ‘hostname‘

oaruser@pcm-master:~$ chmod +x hostname.sh
```

Das Job-Skript kann nun submittet werden. Dies geschieht mithilfe von `oarsub` (Listing 56). Mit dem Parameter `-l` können Properties angeben werden, die dem Job zugesprochen werden. in diesem Fall soll der Job auf zwei Rechenknoten gestartet werden.

Listing 56: Submitten eines Job mit OAR.

```
oaruser@pcm-master:~$ oarsub -l nodes=2 ./job.sh
  [ADMISSION RULE] Set default walltime to 7200.
  [ADMISSION RULE] Modify resource description with type constraints
  OAR_JOB_ID=26
```

Der Job wurde mit der ID 26 gestartet. Der Status des Jobs lässt sich mit `oarstat` abfragen. Mit dem Parameter `-f` für `full` lässt sich der Status im Detail betrachten (Listing 57).

Listing 57: Statusabfrage eines Jobs.

```
oaruser@pcm-master:~$ oarstat
  Job id     Name           User           Submission Date     S Queue
  ---------- -------------- -------------- ------------------- - ----------
  26                        oaruser        2012-07-09 00:06:56 L default

oaruser@pcm-master:~$ oarstat -f
  Job_Id: 26
      job_array_id = 26
      job_array_index = 1
      name =
      project = default
      owner = oaruser
      state = Launching
      wanted_resources = -l "{type = 'default'}/network_address=2,walltime=2:0:0"
      types =
      dependencies =
      assigned_resources = 1+2
      assigned_hostnames = compute1.cluster.pcm+compute2.cluster.pcm
      queue = default
      command = ./job.sh
      launchingDirectory = /home/clusterusers/oaruser
```

```
22      stdout_file = OAR.26.stdout
23      stderr_file = OAR.26.stderr
24      jobType = PASSIVE
25      properties = desktop_computing = 'NO'
26      reservation = None
27      walltime = 2:0:0
28      submissionTime = 2012-07-09 00:06:56
29      startTime = 2012-07-09 00:06:57
30      cpuset_name = oaruser_26
31      initial_request = oarsub -l nodes=2 ./job.sh
32      message = Karma = 2.770
33      scheduledStart = 2012-07-09 00:06:57
34      resubmit_job_id = 0
35      events =
```

Die Ausgabe befindet sich jetzt in den Dateien `OAR.26.stdout` und `OAR.26.stderr` (Listing 58).

Listing 58: Die Ausgabe des beendeten Jobs.

```
1 oaruser@pcm-master:~$ cat OAR.26.stdout
2   compute1
3   compute2
```

# 6 Verbesserungen und Weiterführende Arbeit

Dieses Projekt ist ein erster Schritt zu einem benutzbaren HPC-Cluster. Es gibt noch einige Bereiche in denen Verbesserungen nötig sind. Man kann dabei zwischen zwei Arten unterscheiden. Die einen sind jene, die durch weitere Konfiguration hinzugefügt werden können, und auf der anderen Seite jene, die mit der verwendeten Software nur schwer umzusetzen sind.

Zu den Verbesserungen, die durch weitere Konfiguration hinzugefügt werden können, gehört die Benutzerverwaltung mit Hilfe von LDAP. Der Scheduler OAR bietet viele weitere Anpassungsmöglichkeiten. Auf das Verwenden von Taktuk[7.4] wurde nicht weiter eingegangen.

Ein Problem, das sich nicht so einfach lösen lässt, ist das Installiern von E/A-Servern. Hierzu müsste ein zweiter MAAS-Server inklusive Juju Umgebung installiert werden. Das eigentliche Problem ist die Partitionierung. Das Betriebssystem wird auf der ersten gefundenen Platte installiert. Um die Partitionierung zu ändern muss man sich weiter in cobbler einarbeiten. Des weiteren ist es bisher nur möglich einen Service pro Cloudknoten zu installieren. Juju ist zu diesem Zeitpunkt nicht in der Lage mehrere Services auf einem Cloudknoten zu installieren, dies soll sich aber noch ändern.

# 7 Quellen


1) http://aws.amazon.com/ec2/ (2012-07-06)
2) http://www-05.ibm.com/de/cloud/ (2012-07-06)
3) http://www.ubuntu.com/download/server (2012-07-06)
4) https://wiki.ubuntu.com/ServerTeam/MAAS/ (2012-07-06)
   4.1) https://wiki.ubuntu.com/ServerTeam/MAAS/Juju (2012-07-06)
   4.2) https://wiki.ubuntu.com/ServerTeam/MAAS/AddNodes (2012-07-06)
5) http://cobbler.github.com/ (2012-07-06)
   5.1) https://github.com/cobbler/cobbler/wiki/DHCP%20management
6) https://juju.ubuntu.com/ (2012-07-06)
   6.1) https://juju.ubuntu.com/docs/getting-started.html (2012-07-06)
   6.2) https://juju.ubuntu.com/docs/charm.html (2012-07-06)
   6.3) https://juju.ubuntu.com/docs/write-charm.html (2012-07-06)
   6.4) https://juju.ubuntu.com/docs/user-tutorial.html (2012-07-06)
   6.5) https://juju.ubuntu.com/docs/charm-upgrades.html (2012-07-06)
7) http://oar.imag.fr/ (2012-07-06)
   7.1) http://oar.imag.fr/installation/2.5/#index1h1 (2012-07-06)
   7.2) http://wiki-oar.imag.fr/index.php/Configuration_tips (2012-07-06)
   7.3) http://oar.imag.fr/user-quickstart/
8) http://www.linux-kvm.org/page/Main_Page (2012-07-06)
   8.1) http://virt-manager.org/ (2012-07-06)
   8.2) http://libvirt.org/virshcmdref.html (2012-07-06)
   8.3) http://libvirt.org/format.html (2012-07-06)
9) http://nfs.sourceforge.net/nfs-howto/