

Praktische Datenverarbeitung

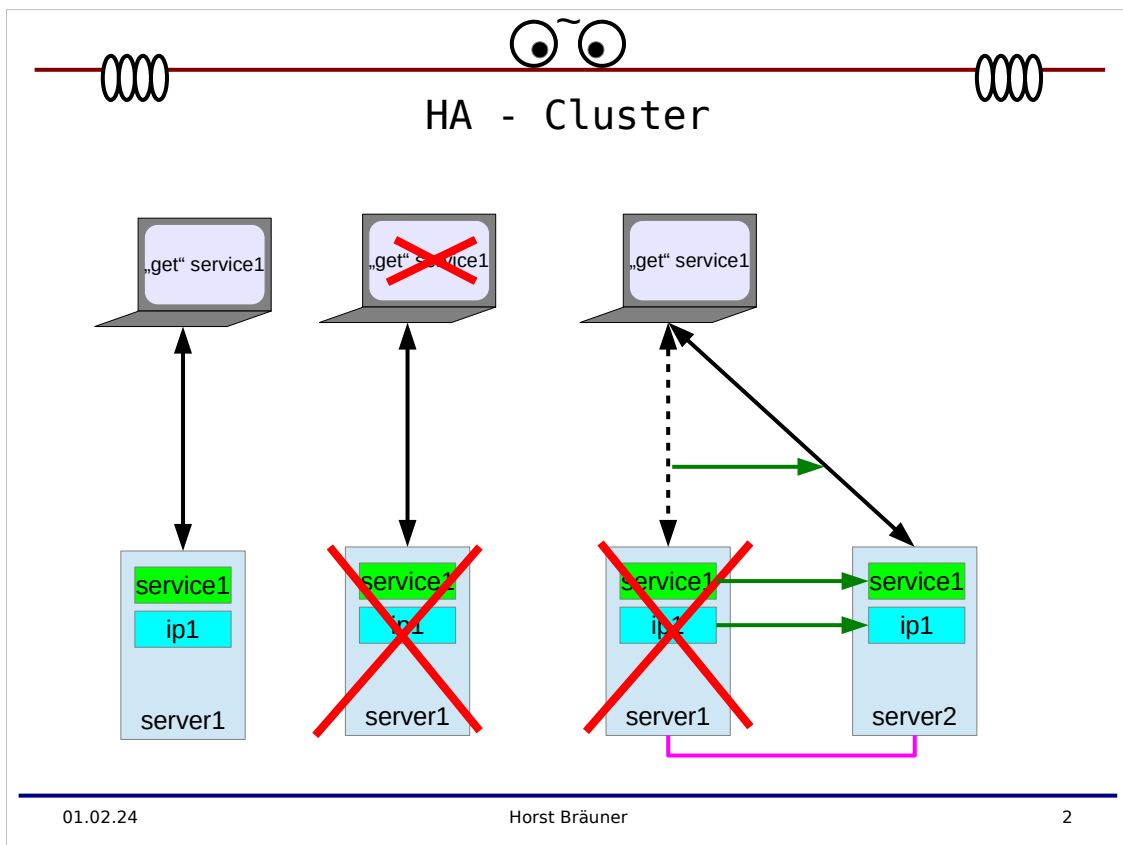
- Cluster
 - HA-Cluster
 - HPC-Cluster
 - HA / HPC
 - HA Setup Beispiel
 - corosync / pacemaker /crm

Cluster unter Linux

Aufwand 2 UE

Ein kurzer Überblick, wie mit Linux hoch verfügbare und/oder hoch performante Cluster konfiguriert werden können.

Als Beispiel dient corosync / pacemaker. Diese Übung ist eigentlich eher für „Hardcore“-Linuxer/innen. Sie können sich gerne daran versuchen. In einer Präsenzvorlesung wird diese Übung an fertig konfigurierten virtuellen Servern demonstriert.



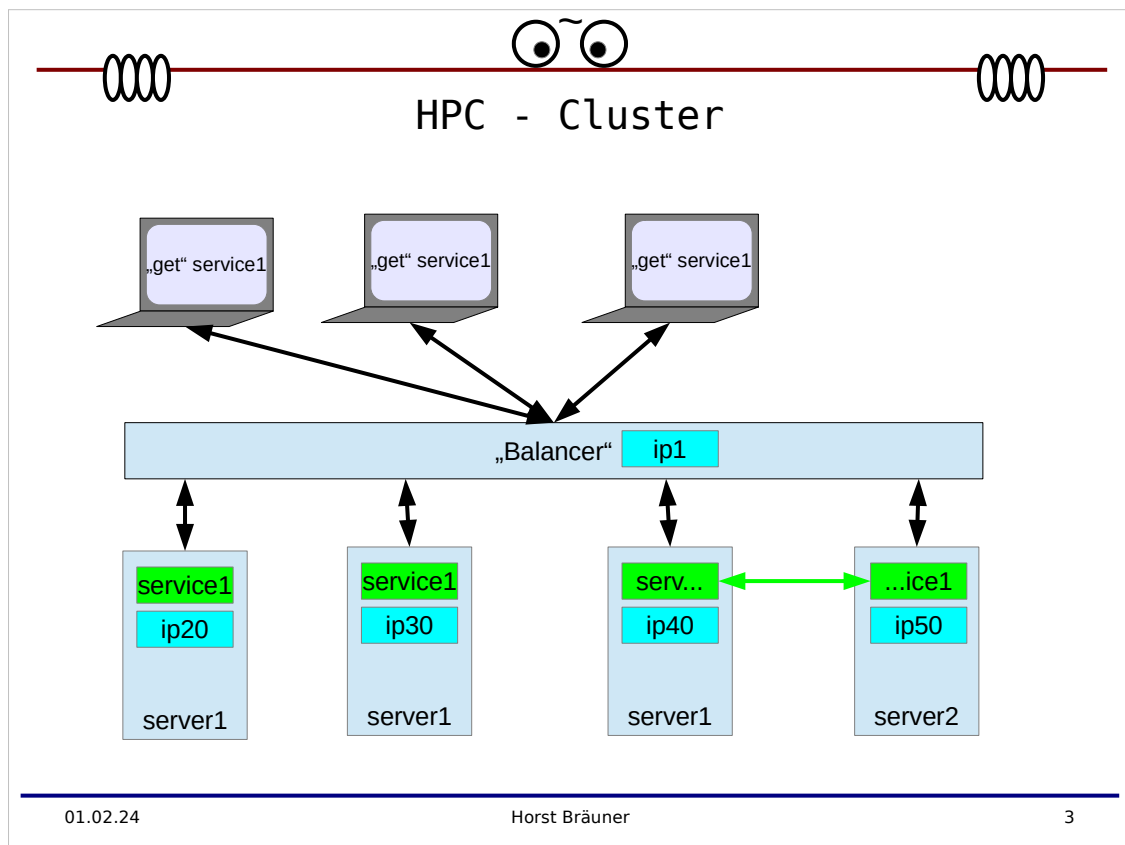
High-Availability Cluster oder kurz HA-Cluster setzen Sie in kritischen Umgebungen ein, um die „Downtime“ für Dienste, die Sie anbieten, so gering wie möglich zu halten.

Stellen Sie sich vor, Sie betreiben einen beliebten Webshop.

Ihre „Kunden“ sollen möglichst immer auf Ihren Shop zugreifen können.

Betreiben Sie Ihren Shop auf einem einzigen physikalischen Server, besteht immer das Risiko, dass die Maschine ausfällt. Außerdem sollten Betriebssystem und Shopsystem gelegentlich gewartet werden. Ihr „Solo“-System ist damit nicht immer verfügbar.

Mit einem HA-Cluster sorgen Sie dafür, dass Ihr Webshop potentiell auf zwei Servern laufen kann. Sollte ein Server ausfallen oder gewartet werden, ziehen Sie den Shop einfach auf eine zweite Maschine um. Dabei kann der Umzug manuell oder, idealerweise, automatisch erfolgen.



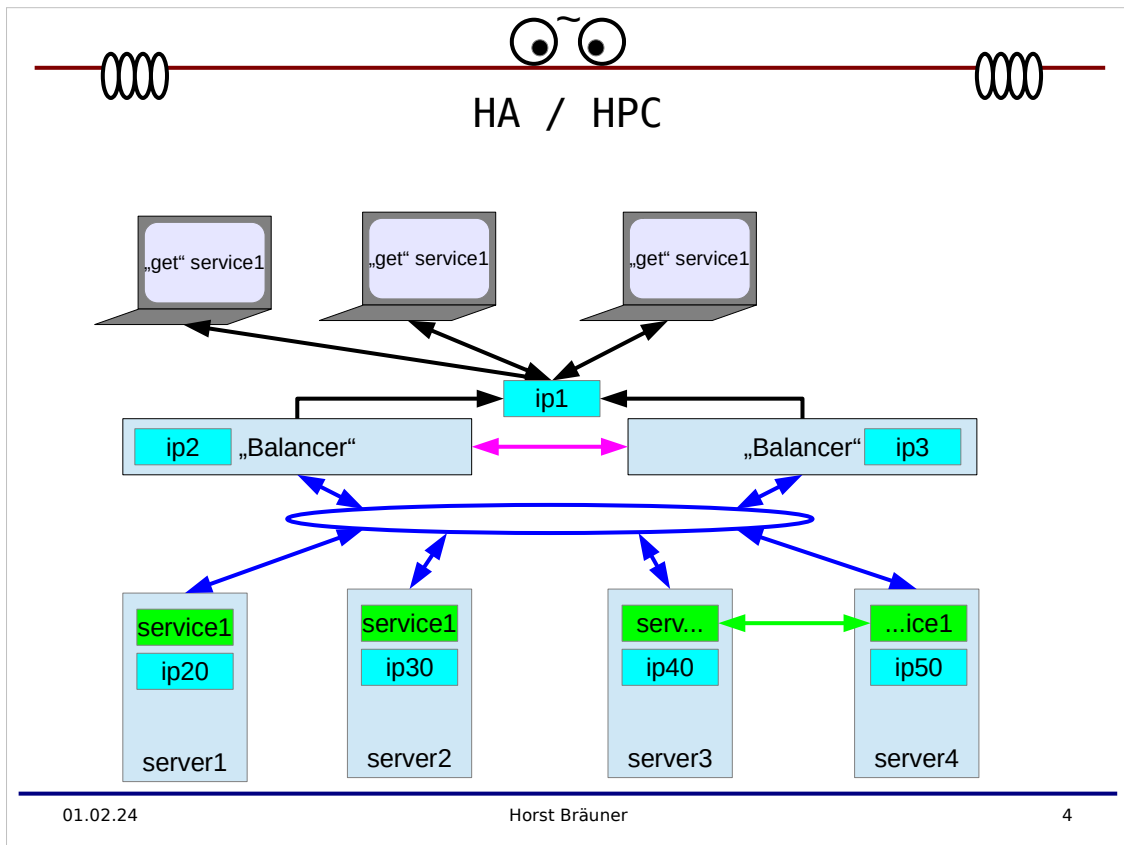
High-Performance Cluster oder kurz HPC-Cluster setzen Sie in kritischen Umgebungen ein, um die „Leistung“ für Dienste, die Sie anbieten, so hoch wie möglich zu halten. Dabei wird die Last auf mehrere Maschinen verteilt.

Stellen Sie sich vor, Sie betreiben einen erfolgreichen Webshop.

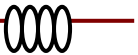


Bei steigender Anzahl von „Kunden“ und steigender Anzahl von Zugriffen soll die Geschwindigkeit / die Leistung / der Durchsatz / ... gegenüber den Kunden immer gleich performant sein oder mitwachsen.

Betreiben Sie Ihren Shop auf einem einzigen physikalischen Server, müssen Sie dafür regelmäßig beispielsweise Speicher nachrüsten oder Ihre Maschine um zusätzliche Platten oder Netzwerkkarten erweitern.

Mit einem HPC-Cluster sorgen Sie dafür, dass Ihr Webshop potentiell auf mehreren Servern verteilt läuft. Sollte zusätzliche Performance notwendig sein, nehmen Sie mehr Server in den Cluster. Je nach Konfiguration erfolgt die Erweiterung manuell oder automatisch.



Es ist auch eine Kombination aus HA- und HPC-Cluster sinnvoll. Dabei setzen Sie beispielsweise einen HA-Cluster für ein hochverfügbares Netzwerk ein und verteilen zusätzlich Ihren Webshop auf mehrere HA-Cluster in einem HPC-Cluster.



Beispiel

- HA-Cluster ist evtl. **nicht** Bestandteil der DEBIAN-Distribution
 - Software installieren
 - apt install pacemaker crmsh corosync at
 - (ggf. apt-get install apache2)
 - Automatischen Start der Cluster-Dienste ausschalten (so lange nicht konfiguriert)
 - systemctl disable pacemaker
 - corosync.conf auf erstem Cluster-Rechner anpassen, authkey generieren („corosync-keygen (-l)“) beide Dateien auf weitere Cluster-Rechner kopieren und dort ggf. anpassen (z.B. IP-Adresse!)
 - z.B. ssh (ohne Passwort) für „stonith“ zwischen den Cluster-Rechnern aktivieren
 - Cluster-Dienste (corosync, pacemaker) starten und mit „crm“ monitoren
 - service corosync start, service pacemaker start, crm status, crm_mon i=1
 - nach erfolgreichem Test: systemctl enable pacemaker
 - mit „crm“ Cluster-Ressourcen definieren

01.02.24

Horst Bräuner

5

Wenn Sie sich zur Übung einen HA-Cluster aufbauen ...

... nehmen Sie 2 neue virtuelle Debian-Server mit minimaler Installation.

Aktivieren Sie zuerst ssh ohne Passwort-Abfrage (= mit „public-key“ Authentisierung) für stonith (kommt noch :-)) zwischen den Servern, siehe Vorlesung „Systemverwaltung“. Tragen Sie die Namen und IP-Adressen der Server in die Datei /etc/hosts ein.

Installieren Sie auf beiden Servern die notwendigen Software-Pakete und schalten Sie Pacemaker erst einmal aus.

```
root@stan:~# apt install pacemaker crmsh corosync at
root@stan:~# systemctl disable pacemaker
```

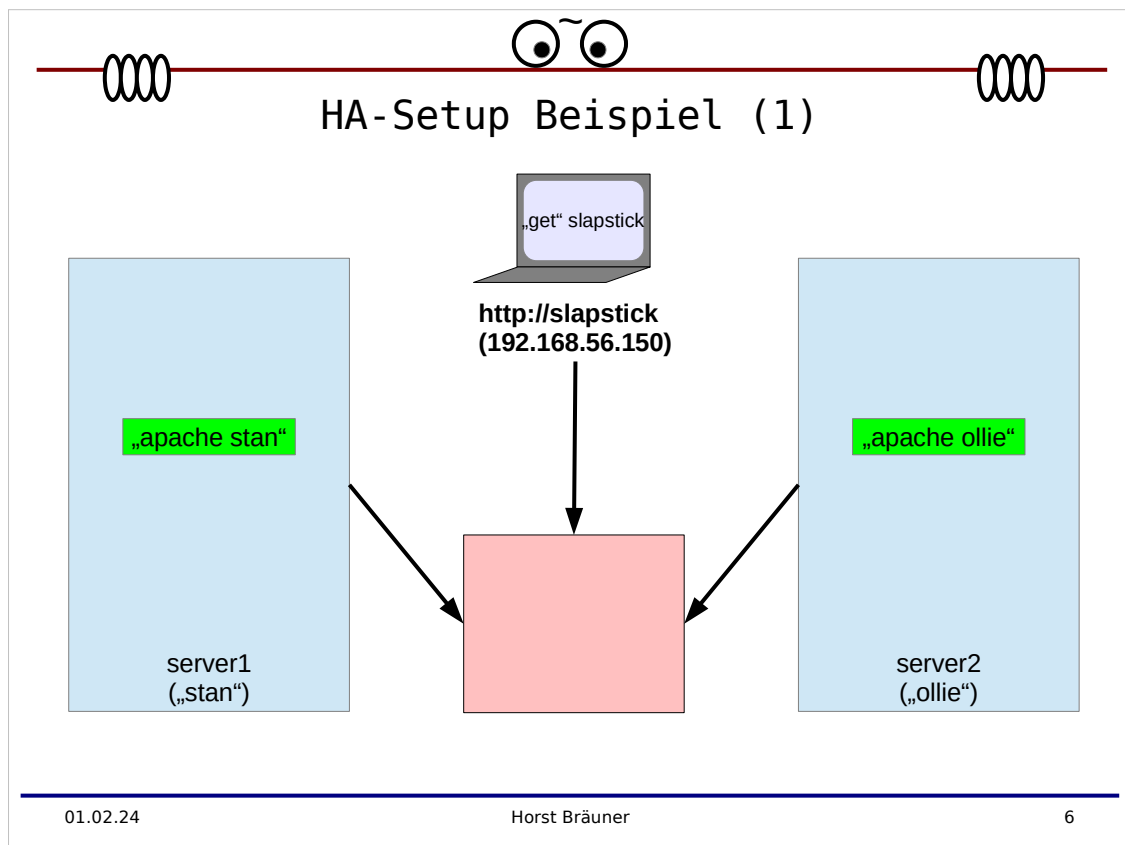
Falls nicht installiert, installieren Sie einen Webserver:

```
root@stan:~# apt install apache2
```

Generieren Sie den corosync authkey und kopieren Sie ihn auf **jeden** Rechner im Cluster.

```
root@stan:~# corosync-keygen
root@stan:~# scp /etc/corosync/authkey ollie:/etc/corosync/
```

Passen Sie auf jedem Server die Datei /etc/corosync/corosync.conf an, siehe Folie 8 und Folie 9.



Zur Übung ein einfaches Setup. Wir betreiben eine Webserver, der hoch verfügbar sein soll. Dafür nehmen wir zwei virtuelle Maschinen, hier: stan und ollie. Diese sollen über eine gemeinsame IP-Adresse einen HA-Cluster „slapstick“ bilden. Mit der IP-Adresse von slapstick wird der „Kunde“ auf den aktuell aktiven Server im Cluster geleitet.

Die Server werden so konfiguriert, dass immer einer der beiden die gemeinsame IP-Adresse besitzt. Fällt einer der beiden Server aus, übernimmt automatisch der andere Server die IP und damit den Service. Der „Kunde“ merkt nichts davon.

Im folgenden Beispiel sind die Server stan und ollie so konfiguriert

stan IP-Adresse 192.168.56.151

ollie IP-Adresse 192.168.56.152

„gemeinsame“ Cluster IP-Adresse slapstick 192.168.56.150

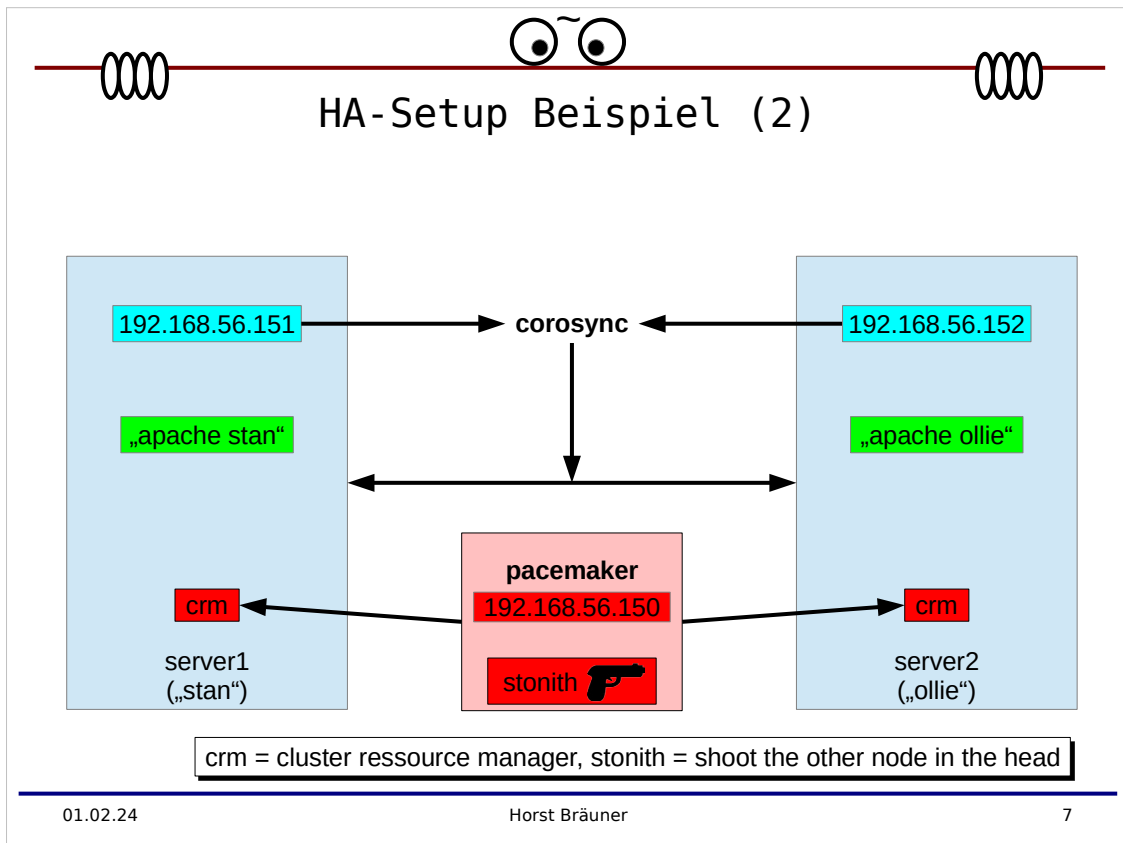
Datei /etc/hosts auf beiden Maschinen **und** auf dem Rechner mit dessen Browser Sie auf „slapstick“ zugreifen enthält zusätzlich diese Einträge

...

192.168.56.150 slapstick

192.168.56.151 stan

192.168.56.152 ollie



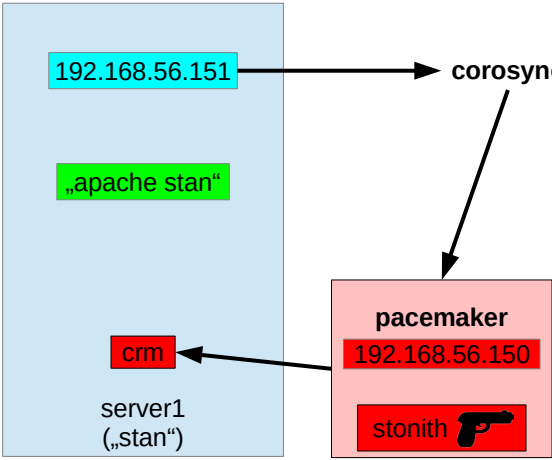
Damit die Übernahme auch reibungslos funktioniert, muss der „ausfallende“ Server die gemeinsame IP-Adresse auch abgeben. Oder anders ausgedrückt, der übernehmende Server muss sicher stellen, dass der andere Server nicht mehr dem „Kunden“ antwortet.

Ein Mechanismus dafür ist STONITH = „shoot the other node in the head“. Für STONITH verbinden Sie beide Server über eine separate Leitung. Das kann ein Cross-Over Kabel zwischen 2 separaten Netzwerkkarten sein, spezielle Hardware-Lösungen, die USB oder serielle Anschlüsse nutzen aber auch beispielsweise eine ssh-Verbindung.

Je weniger komplex die Verbindung der Server für STONITH ist, desto zuverlässiger funktioniert der Mechanismus.

ssh setzt zum Beispiel eine aktive Netzwerkverbindung mit IP-Adressen, funktionierenden Switchen usw. voraus, während ein Serielles Kabel zwischen den Maschinen direkte Befehle hin und her übermitteln kann.

Zur Übung nutzen wir dennoch, der Einfachheit halber, eine ssh Netzwerkverbindung.



The diagram shows a high-availability setup. On the left, a light blue box represents 'server1 („stan“)' with IP 192.168.56.151. Inside this box are 'corosync' and '„apache stan“'. Below it is 'crm'. On the right, a light red box represents 'pacemaker' with IP 192.168.56.150, containing 'stonith' (with a gun icon). Arrows show connections: from the IP box to 'corosync', from 'corosync' to 'pacemaker', and from 'pacemaker' to 'crm'.

```

root@stan:~# cat /etc/corosync/corosync.conf
# Please read the corosync.conf.5 manual page
totem {
    version: 2
    cluster_name: debian
    token: 3000
    token_retransmits_before_loss_const:10
    clear_node_high_bit: yes
    crypto_cipher: none
    crypto_hash: none
    interface {
        ringnumber: 0
        bindnetaddr: 192.168.56.151
        mcastport: 5405
        ttl: 1
    }
}
logging {
    ...
}
quorum {
    provider: corosync_votequorum
    two_node: 1
    expected_votes: 2
}
nodelist {
    node {
        name: stan
        nodeid: 1
        ring0_addr: 192.168.56.151
    }
    node {
        name: ollie
        nodeid: 2
        ring0_addr: 192.168.56.152
    }
    # ...
}
root@stan:~#

```

01.02.24
Horst Bräuner
8

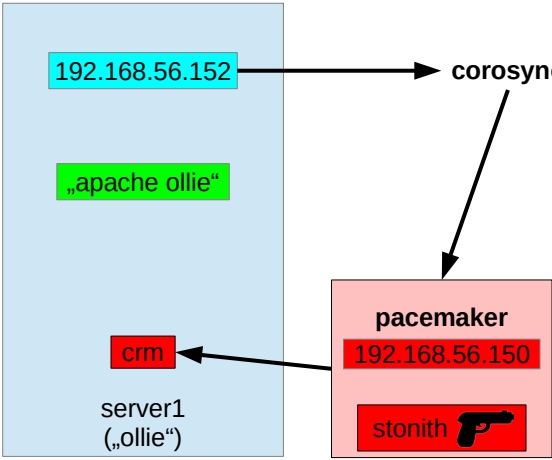
Wir konfigurieren auf „stan“ den corosync-Dienst mit **stans** und **ollies** IP-Adresse.

Ergänzen Sie in der Datei /etc/corosync/corosync.conf die nodelist.

```

totem {
    siehe oben
}
nodelist {
    node {
        nodeid: 1
        ring0_addr: 192.168.56.151
        name: stan
    }
    node {
        nodeid: 2
        ring0_addr: 192.168.56.152
        name: ollie
    }
}
logging {
    siehe oben
}
quorum {
    siehe oben
}

```

The diagram shows a high-availability setup. On the left, a light blue box represents 'server1 („ollie“)' with IP 192.168.56.152. Inside this box are 'apache ollie' and 'crm'. On the right, a pink box represents 'pacemaker' with IP 192.168.56.150, containing 'stonith' (with a gun icon). Arrows show 'corosync' connecting to both the 'server1' box and the 'pacemaker' box.

```

root@ollie:~# cat /etc/corosync/corosync.conf
# Please read the corosync.conf.5 manual page
totem {
    version: 2
    cluster_name: debian
    token: 3000
    token_retransmits_before_loss_const:10
    clear_node_high_bit: yes
    crypto_cipher: none
    crypto_hash: none
    interface {
        ringnumber: 0
        bindnetaddr: 192.168.56.152
        mcastport: 5405
        ttl: 1
    }
}
logging {
    ...
}
quorum {
    provider: corosync_votequorum
    two_node: 1
    expected_votes: 2
}
nodelist {
    node {
        name: stan
        nodeid: 1
        ring0_addr: 192.168.56.151
    }
    node {
        name: ollie
        nodeid: 2
        ring0_addr: 192.168.56.152
    }
    # ...
}
root@ollie:~#

```

01.02.24
Horst Bräuner
9

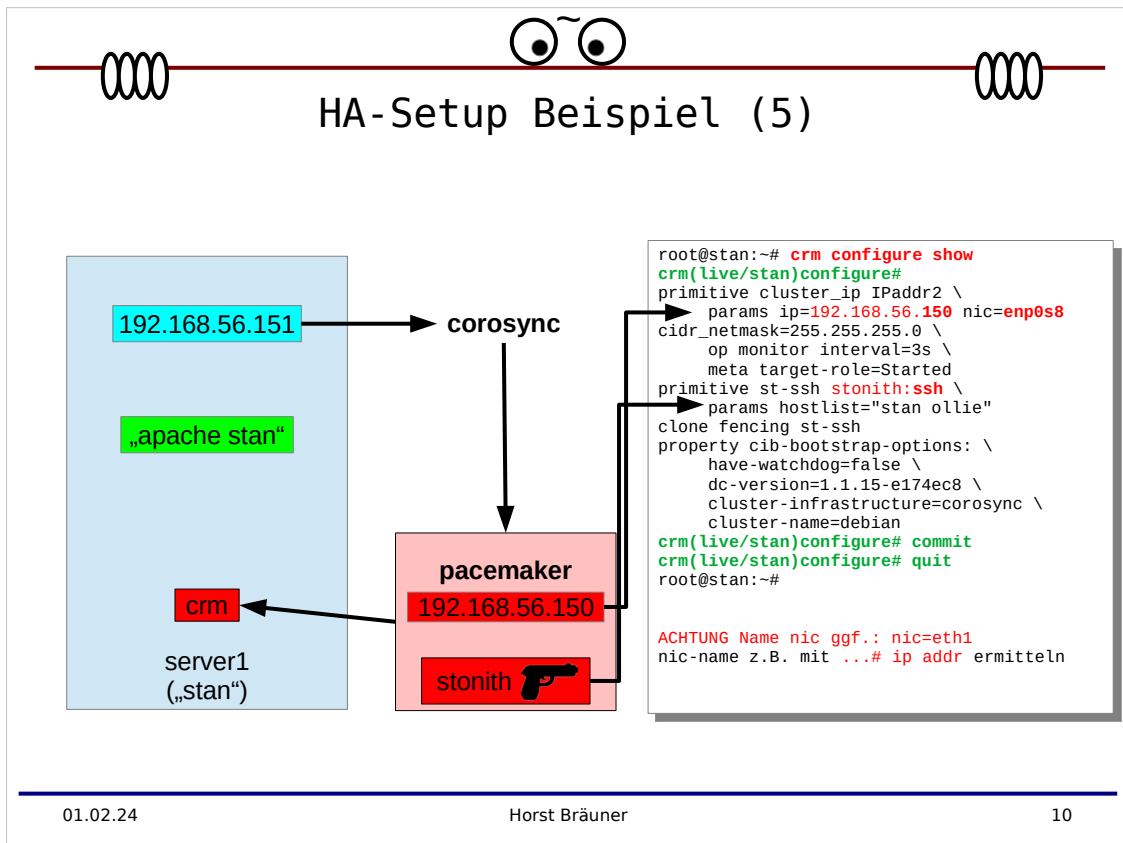
Wir konfigurieren auf „ollie“ den corosync-Dienst mit **stans** und **ollies** IP-Adresse.

Ergänzen Sie in der Datei /etc/corosync/corosync.conf die nodelist.

```

totem {
    siehe oben
}
nodelist {
    node {
        nodeid: 1
        ring0_addr: 192.168.56.151
        name: stan
    }
    node {
        nodeid: 2
        ring0_addr: 192.168.56.152
        name: ollie
    }
}
logging {
    siehe oben
}
quorum {
    siehe oben
}

```



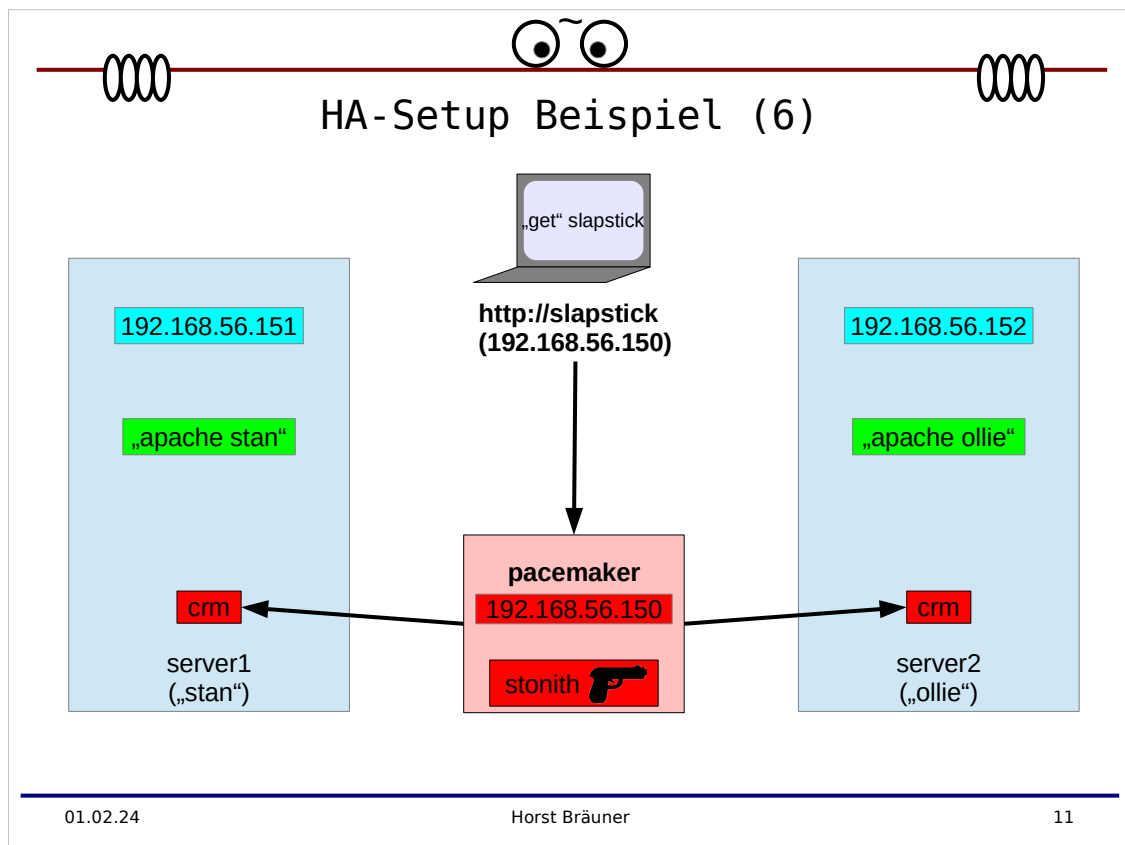
Dann konfigurieren wir auf **beiden** Server die gemeinsame IP-Adresse und das STONITH via ssh. Die Konfiguration geben Sie direkt am prompt der crm-"shell" ein. **nic**="Name des Interfaces" z.B. **enp0s8**

```

root@stan:~# crm
crm(live/stan)# configure
crm(live/stan)configure# primitive cluster_ip IPAddr2 \
  params ip=192.168.56.150 nic=eth1 \
  cidr_netmask=255.255.255.0 \
  op monitor interval=3s \
  meta target-role=Started
crm(live/stan)configure# primitive st-ssh stonith:ssh \
  params hostlist="stan ollie"
  clone fencing st-ssh
  location cli-prefer-cluster_ip cluster_ip role=Started inf:
ollie
crm(live/stan)configure# property cib-bootstrap-options: \
  have-watchdog=false \
  dc-version=2.0.1-9e909a5bdd \
  cluster-infrastructure=corosync \
  cluster-name=debian
crm(live/stan)configure# commit

```

siehe <https://crmsh.github.io/>






Auf dem „Client“ sollten wir noch einen /etc/hosts - Eintrag für die gemeinsame IP-Adresse des Clusters machen (siehe Vorlesung „Webserver“).

Auf den Servern können Sie als Webseiten je eine „default“-Site anlegen und in der index.html Datei beispielsweise den jeweiligen Servernamen anzeigen (SSI, siehe „Webserver“) und sich alle paar Sekunden neu laden („refresh“-Statement).

Beispiel dieser HTML-Seite auf ollie

```
root@ollie:/var/www/html# cat index.shtml
<html>
  <head>
    <META HTTP-EQUIV="refresh" CONTENT="3">
    <title>Webservice SLAPSTICK</title>
  </head>
  <body>
    <H1>Slapstick</H1>
    <h2>
      Slapstick made by <font color="#0000FF">OLLIE</font>
    </h2>
    <!--#config timefmt="%H:%M:%S" -->
    <h2><font color="#FF0000"><!--#echo var="DATE_LOCAL"
--></font></h2>
  </body>
</html>
```



Cluster – commands

- Die wichtigsten Kommandos
 - Dienste
 - service **corosync** start / stop / restart
 - service **pacemaker** start / stop / restart
 - Anzeigen
 - **crm status**
 - crm resource list
 - Clusterdienst anhalten / neu starten
 - **crm node standby**
 - crm node online
 - Ressourcen verschieben
 - crm resource migrate ZIELHOST
 - crm resource move ZIELHOST
 - z.B. **root@host# crm resource migrate/move cluster_ip ZIELHOST**
 - Monitoring
 - **crm_mon -i 1**

01.02.24Horst Bräuner12

Zum Abschluss die wichtigsten Kommandos zum managen des Clusters.

Mit **crm** können Sie auch interaktiv Kommandos ausführen:

```
root@ollie:~#
root@ollie:~# crm
crm(live/ollie)# status
...
crm(live/ollie)# quit
root@ollie:~#
```

Sie können die Kommandos auch direkt übergeben.

```
root@ollie:~#
root@ollie:~# crm status
root@ollie:~#
```

Mit **crm_mon** können Sie den Status beobachten. „-i“ definiert die Zeit in Sekunden für die Aktualisierung.

Beispiel (1)

```
root@ollie:~# crm status
Stack: corosync
Current DC: ollie (version 2.0.1-9e909a5bdd) - partition with quorum
Last updated: Thu May 14 09:19:58 2020
Last change: Thu May 14 09:06:35 2020 by root via crm_attribute on ollie

2 nodes configured
3 resources configured

Online: [ ollie stan ]

Full list of resources:

   cluster_ip   (ocf::heartbeat:IPaddr2): Started ollie
Clone Set: fencing [st-ssh]
   Started: [ ollie stan ]

root@ollie:~#
```



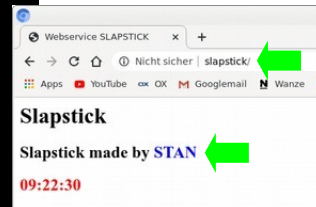
Cluster online auf den Servern stan und ollie
cluster-ip hat ollie
heartbeat über 2te IP-Adresse
„fencing“ via ssh

Hier das Beispiel für einen aktiven Cluster.

Aktuell hat **ollie** die gemeinsame cluster-IP Adresse für slapstick.

Beispiel (2)

```
root@ollie:~#  
root@ollie:~# crm node standby  
root@ollie:~# crm status  
Stack: corosync  
Current DC: ollie (version 2.0.1-9e909a5bdd) - partition with quorum  
Last updated: Thu May 14 09:22:11 2020  
Last change: Thu May 14 09:22:06 2020 by root via crm_attribute on ollie  
  
2 nodes configured  
3 resources configured  
  
Node ollie: standby  
Online: [ stan ]  
  
Full list of resources:  
  
cluster_ip (ocf::heartbeat:IPaddr2):  
Clone Set: fencing [st-ssh]  
Started: [ stan ]  
Stopped: [ ollie ]  
  
root@ollie:~#
```



Cluster online auf den Servern stan und ollie
ollie geht in „standby“
cluster-ip hat automatisch stan erhalten
ressource Webserver läuft jetzt auf stan

Hier das Beispiel für einen Ausfall eines Servers im Cluster.

Ollie wurde in den standby versetzt. Damit hat stan automatisch die Cluster-IP Adresse übernommen und der Browser des „Kunden“ zeigt nach wie vor slapstick als URL an. Sie sehen den Wechsel live im Browser, wenn Sie die bei Folie 11 beschriebene HTML-Seite auf den jeweiligen Servern angelegt haben.