Many Hosts

Mit ssh administrieren

engelbert gruber

30.11.2020
Rev. 0.2

INTRODUCTION

(08:26)

Administrieren von vielen Hosts über ssh.

- clusterssh … öffnet terminals.
- parallel-ssh (pssh)
- ansible
- pdsh

MATERIALS

1. 12 raspberry pi

VORARBEIT

Computer sind zur Übernahme wiederholender Tätigkeiten.

Ich muss den selben Befehl auf vielen Maschinen abschicken … und damit ich auf diese

verbinden kann muss ich das Passwort vielevielemale eintippen … NOT GOOD.

Um das zu umgehen gibt es Schlüssel, Keys. Bei ssh kann mit ssh-keygen ein Schlüssel angelegt werden. Entweder ohne Passwort, wir wollen ja die Passworteingabe am Zielsystem zu vermeiden, oder mit Passwort, wenn mein Ausgangssystem zu unsicher ist.

Wenn man mit passwortgeschütztem ssh-Key arbeitet, gibt es einen Dienst, ssh-Agent, der sich nach einmaliger Eingabe des Passworts für den ssh-key dieses merkt.

ssh-key-gen

ssh-keygen erzeugt zwei Dateien : id_rsa und id_rsa.pub.

id_rsa ist mein **PRIVATE**R key … **NIEMALS NIE NICHT WEITERGEBEN**.

id_rsa.pub ist **PUBLIC**, den kann jeder haben. Wenn jemand diesen in seine "`.ssh/authorized_keys`"-Datei hinzufügt (siehe Handbuch !!!!) kann ich mich weil ich den privaten habe auf diesem System anmelden ohne **dort** ein Passwort

Der id_rsa.pub muss jetzt auf die 12 raspberries.

Dazu gibt es

scp, secure copy:

```
scp .ssh/id_rsa.pub pi@10.10.120.100:
```

Secure-copy die Datei .ssh/id_rsa.pub zum Host 10.10.120.100. Weil hinter der Ip-adresse ein Doppelpunkt ist das der Remotename. Der Username wird mit "pi@" angegeben.

```
scp pi@10.10.120.100:.ssh/authorized_keys ./a
```

Mit diesem Befehl würde die authorized_keys-Datei vom Host zu mir kopiert, der Doppelpunkt ist im ersten/Quellargument.

Oder ssh

ssh ist secure shell.

```
cat .ssh/id_rsa.pub | ssh pi@10.10.120.100 "cat >> .ssh/authorized_keys"
```

- `cat` ist am unix ausgeben einer Datei (am Windows ist das "type").
- `.ssh/id_rsa.pub` ist unser public-Key
- cat schickt den Inhalt an das ssh Kommando

1 many hosts

- ssh führt am Zielsystem "`cat >> .ssh/authorized_keys`" aus. Dieser cat läuft am raspberry liest von der Standardeingabe (weil wir nichts angegeben haben) gibt auf die Standardausgabe aus (weil wir nichts angegeben haben) und die Ausgabe wird mit ">>" umgeleitet, an "`.ssh/authorized_keys`" angehängt.

  ### Aufgaben

  Was macht: `cat > eins.txt`

Wir verteilen den public-Key mit

```
cat .ssh/id_rsa.pub | ssh pi@10.10.120.100 "cat >> .ssh/authorized_keys"
```

auf alle Hosts.

Danach kann man sich mit "`ssh pi@10.10.120.100`" dort anmelden ohne ein Passwort einzugeben. Oder mit "`ssh pi@10.10.120.100 date`" die Uhrzeit dort anschauen.

Wir können noch in .ssh/config den User angeben:

```
Host 10.10.120.*
User pi
```

Für alle Hosts die mit 10.10.120 beginnen wird der Username pi verwendet.

Uhrzeitkontrolle mit Hand

Ein Script:

```
HEAD=10.10.120
for TAIL in 101 100 110 109 105 136  153 135 154 151 102 106 ;
do
    ssh $HEAD.$TAIL date
done
```

Dann sehen wir :

```
Fr Nov 27 09:41:24 CET 2020
Fr Nov 27 09:41:24 CET 2020
Fr Nov 27 09:41:26 CET 2020
Fr Nov 27 09:41:26 CET 2020
Fr Nov 27 09:41:27 CET 2020
Fr Nov 27 09:41:27 CET 2020
```

```
Fr Nov 27 09:41:28 CET 2020
Fr Nov 27 09:41:28 CET 2020
Fr Nov 27 09:41:29 CET 2020
Fr Nov 27 09:41:30 CET 2020
Fr Nov 27 09:41:33 CET 2020
```

Leider ist die Verbindung zum letzten Host schlecht … aber ssh hat ein Timeout … sonst mit Strg-C abbrechen.

Besser (mit Interpreter line “#!/bin/sh”)

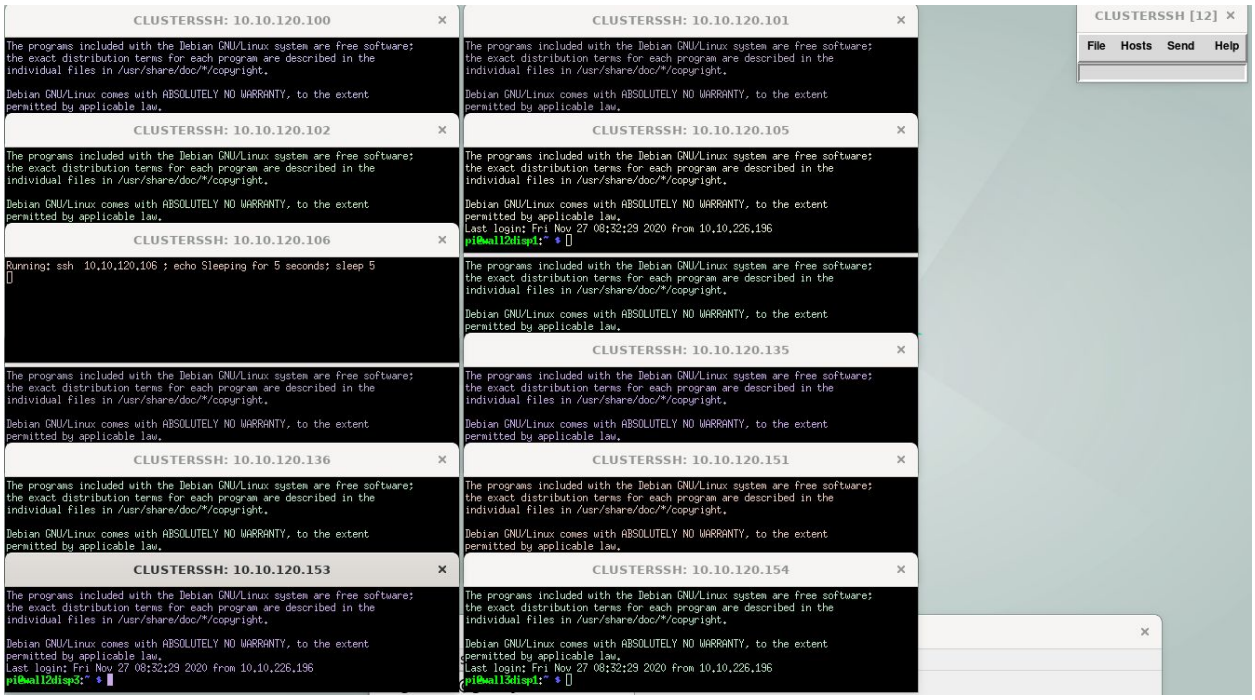
```
#!/bin/sh
HEAD=10.10.120
for TAIL in 101 100 110 109 105 136  153 135 154 151 102 106 ;
do
    ssh $HEAD.$TAIL "hostname; date +'  %Y-%m-%dT%H:%M' "
done
```

Ausgabe:

```
wall1disp1
  2020-11-27T09:48
wall1disp2
  2020-11-27T09:49
wall1disp3
  2020-11-27T09:49
…
```

clusterssh

```
gooey:~/cluster$ clusterssh $(./allips.sh)
Opening to: 10.10.120.101 10.10.120.100 10.10.120.110 10.10.120.109
10.10.120.105 10.10.120.136 10.10.120.153 10.10.120.135 10.10.120.154
10.10.120.151 10.10.120.102 10.10.120.106
```



Wenn man jetzt date eintippt erscheint das in allen Fenstern. Klickt man auf ein Fenster ist man an dem jeweiligen System, klickt man auf das “CLUSTERSS[12]” Fenster tippt man auf allen Systemen.

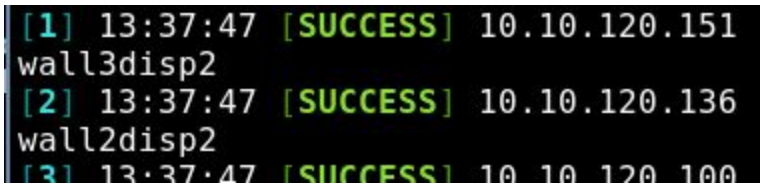
(10:05)

(13:30)

parallel-ssh (pssh)

Man kann die Hosts auf der Kommandozeile angeben:

```
parallel-ssh -H "$(./allips.sh)" -i hostname
```



Das Argument “`-i`” führt dazu, dass die Ausgabe des Befehls, hier “`hostname`”, zwischen den Statuszeilen (SUCCESS oder FAILURE) kommt. ABER die Ausgabe kommt erst wenn der Befehl abgeschlossen ist … die Verbindung abgebaut.

4 many hosts

Die Reihenfolge hängt von der Verbindungs- und Verarbeitungsgeschwindigkeit des Hosts ab.

Kontrolle der Auslastung

```
parallel-ssh -P -H "$(./allips.sh)" "while true ; do cat /proc/loadavg ;
sleep 10 ; done"
```

- Auf einer Zeile eingeben.
- "`-P`" führt dazu, dass die Daten ausgegeben werden wenn sie empfangen werden, nicht erst nach dem Ende der Verbindung.
- "`cat /proc/loadavg`" gibt die derzeitige Prozessorauslastung an, siehe Handbuch.

Weil die Ausgabe nicht geblockt nach Host ist, wird der Hostname davor geschrieben.

```
10.10.120.110: b'0.13 0.03 0.01 1/196 7995\n'
10.10.120.136: b'0.08 0.04 0.00 1/196 1449\n'
10.10.120.109: b'0.00 0.00 0.00 1/193 8664\n'
10.10.120.105: b'0.00 0.00 0.00 2/195 31587\n'
10.10.120.135: b'0.10 0.03 0.01 1/195 25608\n'
10.10.120.151: b'0.09 0.06 0.01 2/200 19548\n'
10.10.120.154: b'0.00 0.02 0.00 1/195 2335\n'
10.10.120.153: b'0.04 0.06 0.01 1/193 9150\n'
```

Liste der installierten Pakete

```
parallel-ssh -o out  -H "$(./allips.sh)" "dpkg -l"
```

Im Verzeichnis "`out`" steht für jeden Host eine Datei die die Liste der dort installierten Pakete enthält.

und mit

```
cat * | cut -c5-40|sort|uniq -c|sort -nr
```

sieht man welche Pakete wie oft installiert sind.

ansible

ansible ist von redhat aber Opensource.

Wir benötigen die Hosts in einer Konfigurationsdatei, ini-Format oder yaml.

Man gruppiert die Hosts.

```
[walls]
10.10.120.101
10.10.120.100
10.10.120.110
10.10.120.109
```

oder in yaml, Datei: inventory.yaml

```
all:
  hosts:
    w1d1:
      ansible_host: 10.10.120.101
    w1d2:
      ansible_host: 10.10.120.100
    ...
  children:
    wall1:
     hosts:
       w1d1:
       w1d2:
       w1d3:
       w1d4:
     wall2:
        ...
```

Dann kann man mit

```
ansible -i inventory.yaml wall1 -m ping
```

“`wall1`” gibt die Gruppe von Servern an.

“`-m ping`” das zu verwendende Modul

```
ansible -i inventory.yaml wall1
   -m package -a "name=tcpdump state=present" -b
```

Auf den Servern der Gruppe wall1 das Modul “`package`” anweisen, den Zustand “`tcpdump present`” herzustellen, falls er nicht schon ist.

Für windows ?

Siehe https://docs.ansible.com/ansible/latest/user_guide/windows.html

Via WinRM mit powershell, vielleicht auch ssh.

BEISPIELE

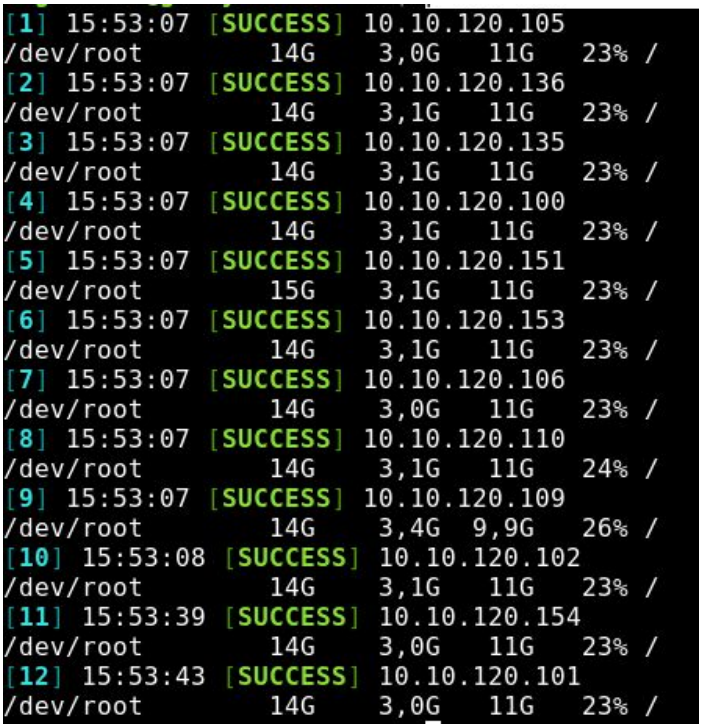
Resize Raspberry Partition

raspi-config ist eine GUI. Wir starten clusterssh und verbinden zu allen Systemen.

Geben sudo raspi-config ein.

Mit den Pfeiltasten zum Advanced Eintrag Enter, dann steht die Auswahl schon am Resize Partition ... Enter Escape und so weiter.

Mit "`parallel-ssh -i -H "$(./allips.sh)" "df -h|grep root"`" kann man kontrollieren ob root jetzt größer ist.



REFERENCES

REVISION HISTORY

- 0.1 27.11.2020 - manuell, clusterssh, ansible
- 0.2 30.11.2020 - resize sd