cfengine in der Praxis

11.02.2008
sage@guug Lokalgruppe München

Markus Brylski

Inhalt

- Die Situation
- Der „alte“ Managementansatz
- Werkzeuge im Vergleich
- Was ist cfengine?
- cfengine Syntax; einige Beispiele
- Klassen in cfengine
- Aktionen in cfengine
- Der Aufbau von cfengine
- Trainingseinheit
- Installationsprozeß eines Systems
- Die neue Managementstruktur
- Quellen

Die Situation

11 Millionen Rezepte abrechnen zum Stichtag

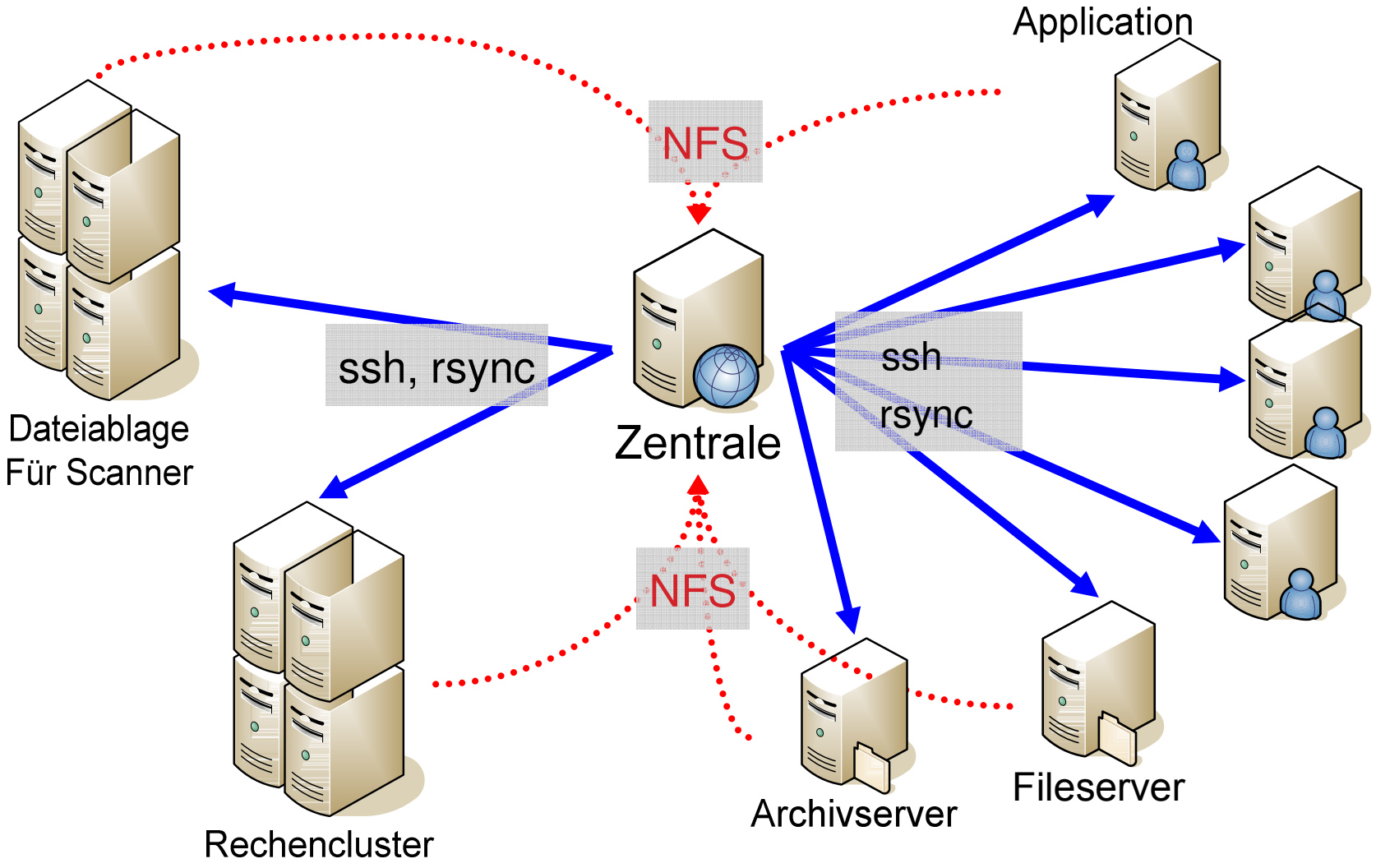
Arbeitsschritte in der Produktion

Rezepte erfassen	⌦ Dialogeingabe, 2 Application Server
Rezepte erkennen	⌦ Scannerstraßen, 30 Application Server
Daten korrigieren	⌦ 70 Arbeitsplätze, 10 Application Server
Rezeptdaten speichern	⌦ Zwei Datenbankserver
Rezeptdaten abrechnen	⌦ 2 Großserver, 4-8 Application Server
Rechnungen schreiben	⌦ 4 Application Server
Datenlieferungen erstellen	⌦ 4 Application Server, CD-Brennstation

Der „alte“ Managementansatz

- Installation der Server von Image
- Zentrale Templates werden von einem Konfigrationsserver verteilt und evtl. per Skript angepasst
- Synchronisationsprogramme werden für jede Aufgabe neu geschrieben
- Zentrale Versionierung von wichtigen Dateien per cvs
- Zentrale Softwareverteilung per NFS
- Zentraler Installationserver für alle eingesetzten SuSE-Versionen

Die „alte“ Managementstruktur



Ein Beispiel

SyncAppsrv

```
#!/bin/bash
export PATH=/sbin:/usr/sbin:/usr/bin:/bin
Host=$1
case "$Host" in
  appsrv1|appsrv2|…)  ;;
  *)        echo "$0 usage: $0 <appsrv>“ ; exit 1
esac
TargetDir=/
BaseDir=/fileservice/AppSrv
rsync -a -e ssh --log-format  "%t_[%p]_${Host}:_%f,_%l_bytes."  \
            ${BaseDir}/    ${Host}:${TargetDir}  \
      | sed 's/_/ /g‘ | logger …
```

Noch ein Beispiel

fanout

```
# fanout " appsrv1 appsrv2 " "grep '^server ntp' /etc/ntp.conf"
Starting appsrv1
Starting appsrv2
Fanout executing "grep '^server ntp' /etc/ntp.conf"
Start time Sat Dec 1 13:28:06 CET 2007 , End time …
==== On appsrv1 ====
  server ntp3
  server ntp4
==== On appsrv2 ====
  server ntp3
  server ntp4
Exiting fanout, cleaning up...done.
```

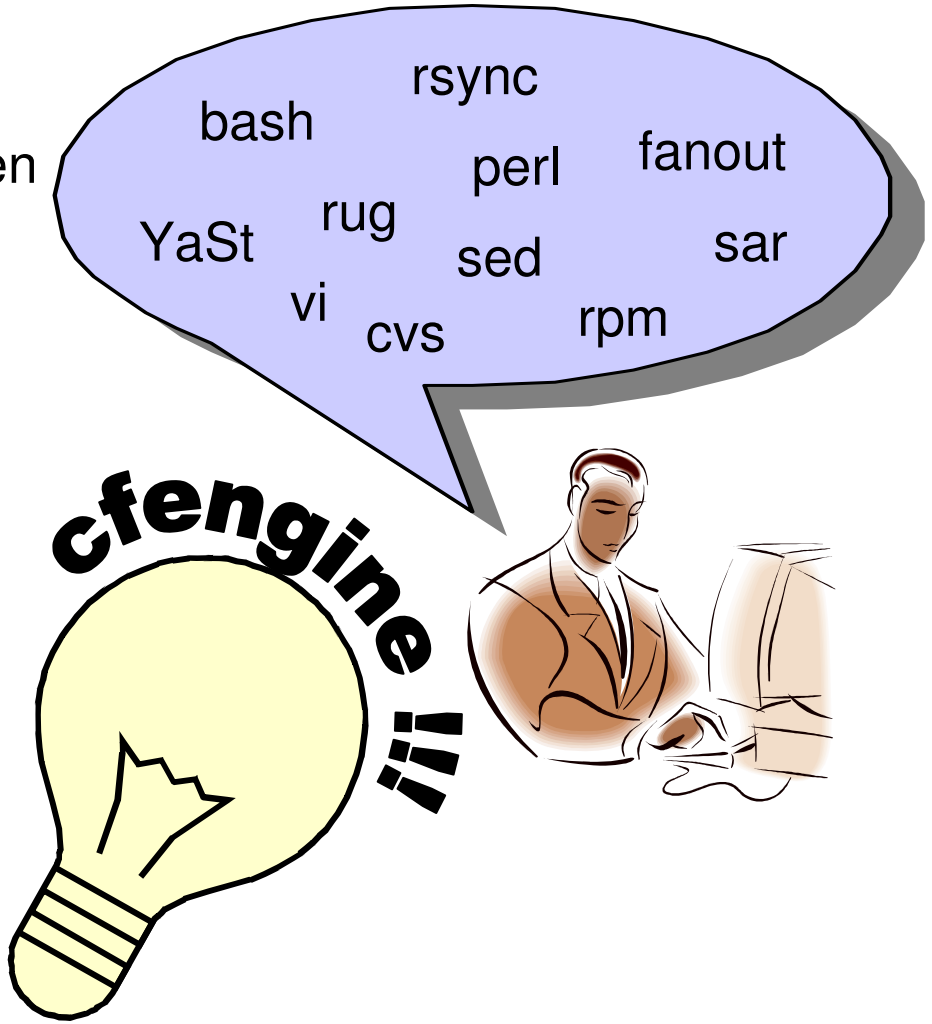
Nachteile der bisherigen Lösung

- Die Images sind zu statisch für neue Hardware
- Die verwendete Imagingsoftware unterstützt das verwendete Filesystem (noch) nicht.
- Zeitaufwendige sich ständig wiederholende Prozesse.
- Inkonsistenzen auf den Systemen durch Flüchtigkeitssfehler.
- Skripten müssen ständig angepasst, getestet und neu ausgerollt werden.
- Konsistenz der verwendeten Skripten nicht gegeben.
- Konsistenz der verwalteten Systeme nicht gesichert.
- Instabilität auf den Produktionsservern durch „NFS-Hänger“, da der zentrale Konfigurationsserver nicht hoch verfügbar ist.

Aufgaben und Werkzeuge

- Schnelle, fehlerfreie Installation von vielen gleichartigen Systemen
- Kontinuierliche Anpassung der Systeme an die Anforderungen
- Nachvollziehbarkeit der Arbeit
- Ständige Verbesserung
- Beratung bei Projekten
- Versionswechsel durchführen
- Überwachung von Systemen
- Früherkennung
- und vieles mehr…

(M)ein Leitwerkzeug ist…



Versprechungen

Auszug aus „Promise You a Rose Garden“ (by Mark Burgess)

Once upon a time there was an arrow. This arrow was reputed to have been tooled in the armoury of Cupid himself. ... But although the arrow was strong and straight and pure, the arrow’s intended target was clad in a special kind of autonomous armour that resisted the thrust of its point, and so, when fired, the arrow was not received, but merely landed close to its target.

Someone close to the target, …, saw the arrow shining in the sun, for arrows from Cupid are made of the finest golden material, pure and universal in their appeal. The competitor was so impressed by the arrow, and appreciative of its value, that it fired a sucker in the direction from which the arrow was aimed, attached to it a message saying ”if you fire such an arrow at me, I shall not only open my armour to receive it, but return an arrow of my own, made from the best I have to offer.”

Versprechungen

Auszug aus „Promise You a Rose Garden“ (by Mark Burgess)

The arrows were fired, not in violence or coercion, but as an offering of riches. Each had its strings attached and hence the two were bound together. And so they lived, each independent of mind and free of choice, but willingly bound in union of mutual value, all in appreciation of the richness of one another’s offers. And they said to one another: ”Let us call this state of mutually beneficial union commerce.“

Versprechungen

Auszug aus „Promise You a Rose Garden“ (by Mark Burgess)

„A promise is like an arrow, ... Each one carries a claim about its originator, to the receiver … Each promise is aimed at a single recipient, but the promiser can fire similar arrows at any number of targets to repeat the promise. Promises are non-transferable and constrain only the promiser, never the promisee. Duplicating a promise to the same recipient has no effect, but repeating a promise with a change added would be a promise broken.

These simple ideas are the basic facts about promises. From these we can describe the world in which system administration takes place.“

- **Mark Burgess**
  - **Author von cfengine und vielen Veröffentlichungen und Büchern zum Thema (theoretische) Systemadministration.**
  - **Professor für Netzwerk und Systemadministration an der Universität Oslo.**

Was ist cfengine?

„Cfengine is an automated suite of programs for configuring and maintaining Unix-like computers. It has been used on computing arrays of between 1 and 20,000 computers since 1993 by a wide range of organizations. Cfengine is supported by active research and was the first autonomic, hands-free management system for Unix-like operating systems. Cfengine is an autonomic maintenance system not merely a change management roll-out tool. Cfengine has a history of security and adaptability.“

(Zitat www.cfengine.org)

Frei übersetzt könnte man sagen:

- cfengine kann sehr viele Systeme verwalten.
- cfengine arbeitet eigenständig

Was ist cfengine?

„Cfengine is designed to let you use configuration files to describe the desired state of a system, rather than describing what should be done to a system to reach that state. You can think of Cfengine configuration files as being in a very high level language - much higher level than Perl or shell scripts. For example, a single cfengine configuration file statement can result in hundreds of links being created, or the permissions of hundreds of files being set.“

(Zitat www.cfwiki.org/cfwiki/index.php/Overview)

Frei übersetzt könnte man sagen:

- cfengine ist eine Programmiersprache, die den Wunschzustand des Systems beschreibt und nicht den Weg dahin.
- Im Gegensatz zu selbstgeschriebenen Skripten ist man nicht mehr mit der Umsetzung der Methode befasst, sondern mit der Beschreibung des Zieles.

Was kann cfengine?

- Konfigurationen von Servern lokal und weltweit standardisieren.
- Rechte und Besitz von wichtigen Dateien prüfen und protokollieren.
- Schablonen wichtiger Konfigurationsdateien verteilen, anpassen und deren Besitzer bzw. Zugriffsrechte prüfen.
- sehr einfach von einem Punkt aus „batch jobs“ und selbst geschriebene Skripten steuern.
- versionierte Softwarepakete nachinstallieren bzw. überprüfen.
- Prüfen ob wichtige Prozesse auf einem System laufen und diese bei Bedarf nachstarten.
- Plattenplatz überwachen und rechtzeitig eine Warnung ausgeben.
- Dateiänderungen an Hand von Prüfsummen erkennen.
- Menschliche Fehler erkennen und evtl. korrigieren.
- Change Management wirkungsvoll unterstützen.

(Quelle: http://www.cfengine.org/cfstart.html)

cfengine Syntax

Einfache Zuweisungen

perl

```
$Variable = 123;
$Array = ("one", "two", "three");
```

cfengine

```
groups:
    Variable = ( 123 )
    Liste = ( one:two:three )
```

cfengine Syntax

Klassifizierung eines Systems

perl

```
if ( `uname` == „Linux“ ) {  <Aktion>  }
If ( `lsb_release` =~ /SuSE Linux Enterprise Server 10/ ) {
      <Aktion>;
}
```

cfengine

```
<Aktion>:
      linux::
            … # Linuxsysteme allgemein
      SLES10::
            … # Enterprise Server 10 im Besonderen
```

cfengine Syntax

Erstellen eines Verzeichnisses

bash

```
mkdir –v –p /opt/my_software/install
chmod –c 0755 /opt/my_software/install
chown –c root:wheel /opt/my_software/install
```

cfengine

```
directories:
      /opt/my_software/install owner=root group=wheel
            mode=0755 inform=true
```

cfengine Syntax

- **Korrigieren von Dateirechten** *

bash

```
chmod –c 0750 /var/log/messages
chown –c 0:1984 /var/log/messages
```

- **Aktion wird beschrieben.**

cfengine

```
files:
      /var/log/messages owner=0 group=1984
            mode=0750 action=fixall inform=true define=FI_Msgs
```

- **Definition des Zielzustandes.**

cfengine Syntax

- **Einfache Dateiänderung**

bash

```
grep 192.168.127.127 /etc/hosts > /dev/null || {
  echo "192.168.127.127 fileserver1" >> /etc/hosts;
};
```

cfengine

```
editfiles:
{ /etc/hosts
      AppendIfNoSuchLine "192.168.127.127 fileserver1"
}
```

cfengine Syntax

- **Prozessüberwachung**

bash

```
checkproc /usr/bin/automount || {
        /etc/init.d/autofs start;
        logger –t monitor –p daemon.notice - -  \
          "Restarted automount";
};
```

cfengine

```
processes:
        „automount“ restart="/etc/init.d/autofs start"
                syslog=true define=RestartedAutomount
```

cfengine Syntax

- **Überwachen von Dateiänderungen** *

bash

```
md5sum –c /usr/local/etc/hosts.md5
```

- **md5sum /etc/hosts > /usr/local/etc/hosts.md5 nicht vergessen!!!**
- **Bei Änderung der /etc/hosts Checksumme aktualisieren!!!**

cfengine

```
files:
        /etc/hosts checksum=md5 inform=true syslog=true
```

- **Checksummen werden automatisch verwaltet.**

cfengine Syntax

- **Komplexe Dateiänderung auf einer Gruppe von Maschinen**

bash

```
case $HOST in
   host1|host2|host3)
      grep –v "192.168.128.127 /etc/hosts" > /etc/hosts.neu
      mv –v /etc/hosts.neu /etc/hosts;
      grep "192.168.127.127" /etc/hosts || {
         echo "192.168.127.127 fileserver1" >> /etc/hosts;
      }
  …
esac
```

cfengine Syntax

- **Komplexe Dateiänderung auf einer Gruppe von Maschinen cfengine**

```
groups:
  ErkennerHosts = ( host1 host2 host3 )

editfiles:
  ErkennerHosts::
      { /etc/hosts
            DeleteLinesMatching "192.168.128.127 fileserver1"
            AppendIfNoSuchLine "192.168.127.127 fileserver1"
      }
```

cfengine Klassen

Sehen wir genauer hin:

```
groups:
  ErkennerHosts = ( host1 host2 host3 )
```

- Es wird eine Klasse namens „ErkennerHosts“ definiert und deren Mitglieder festgelegt.
- Die Aktion erfolgt genau dann, wenn die gewünschte Aktion in der Klasse „enthalten“ ist.
- Die Beispielklasse fasst Hosts zu einer Gruppe zusammen und wird daher als Gruppe bezeichnet.

Beispiele für Klassen

- Vom System vorgegebene Klassen
  - Der Name eines Betriebssystems „linux“, „ultrix“, …
  - Der Name eines Hosts „foo“, „bar“, …
  - Ein Wochentag „Monday“, „Tuesday“, …
  - Stunden- oder Minutenbezeichnung „Hr00“, „Hr01“, „Min00“, …
  - Ein Fünfminutenintervall der Form „Min10_15“
  - Ein Datumsbestandteil „Year1997“, „January“, …
  - Eine IP-Adresse der Form „192_168_100_127“, „192_168_100“, …
- Benutzerklassen
  - Eine vom Benutzer festgelegte Bezeichnung („ErkennerHosts“)
  - Gruppen werden vom Benutzer festgelegt

Verknüpfung von Klassen

- Logische Undverknüpfung von Klassen mit „.“ oder „&“
  - „linux.suse10.ErkennerHosts“
  - „ErkennerHosts&Monday&Hr00“
- Logische Oderverknüpfung von Klassen durch „|“
  - „solaris|irix“
- Klassen können mit „!“ auch negiert werden
  - „linux.!fileserver“ – Linuxhosts aber keine Dateiserver.

Programmstruktur in cfengine

```
<Aktion>:
   <Klasse>::
        <Tätigkeitsbeschreibung>
  <Klasse2>::
        <Tätigkeitsbeschreibung>
```

- Mögliche Aktionen sind:
  - groups, control, copy, homeservers, binservers, mailserver, mountables, import, broadcast, resolve, defaultroute, directories, miscmounts, files, ignore, tidy, required, links, disable, shellcommands, strategies, editfiles, processes

Häufige Aktionen

- control
  - Ablaufsteuerung, Definition von wichtigen Vorgaben („actionsequence“)
- groups
  - Gruppieren von Hosts
- copy
  - Kopieren von Dateien ähnlich „rsync“
- directories, files
  - Kontrolle von Rechten
  - Überwachen von Dateiänderungen
- editfiles
  - Ändern von Dateien
- processes
  - Prozesse überwachen und steuern

Die wichtigsten Konfigurationsdateien

- cservd.conf
  - Konfigurationsdatei für den Serverprozeß „cfservd“
  - Hier legt man fest, welche Maschinen auf „cfservd“ zugreifen können
  - Insbesondere werden hier gezielt Verzeichnisse frei gegeben von denen sich die Clients bedienen dürfen.
- update.conf
  - Diese Datei wird in jedem Fall zuerst von cfagent verarbeitet .
  - Sie dient der Verteilung der Konfigurationsdateien und Binaries von cfengine auf die Clientmaschinen.
  - Fehler in anderen Dateien können zentral behoben werden nicht aber Fehler in der update.conf.
  - Jeder Fehler in dieser Datei kann das Gesamtsystem von cfengine zum Zusammenbruch bringen.
  - **Ergo: Keep it simple!**

Die wichtigsten Konfigurationsdateien

- cfagent.conf
  - Die „Innereien“ des Konfigurationssystems finden sich in dieser Datei.
  - Gruppendefinitionen werden hier festgelegt
    - Nach Funktion des Clients
    - Nach Betriebssystem
  - Aktionen beschreiben den Zielzustand der Clientsysteme
    - Erzeugen von Verzeichnissen und Links
    - Kopieren von Dateien Editieren
    - Überwachen von Eigentümer bzw. Zugriffsrechten
  - Modularisierung erreicht man durch die Sektion „import“
    - Aufgliedern der „cfagent.conf“ in übersichtliche Teilbereiche
    - Wartungsfreundlichkeit und Risikominimierung

Die grundlegenden Kommandos

- cfagent
  - Der Konfigurationsagent (Agent) führt die eigentlichen Änderungen am zu konfigurierenden System (Client) durch.
- cfservd
  - Dieser Daemon dient als Server für Filetransfers oder zur Steuerung von Konfigurationsagenten auf Clients (Pushbetrieb)
- cfexecd
  - Der Sklaventreiber von cfagent dient der lokalen Ausführung des Agenten via cron (Pullbetrieb).
- cfkey
  - Dies ist der Schlüsselgenerator zur Bereitstellung der Public und Private keys für jeden Host.
- cfrun
  - Fernsteuerung des Konfigurationsagenten auf dem Client über cfservd.

Die Dateisystemstruktur am Client

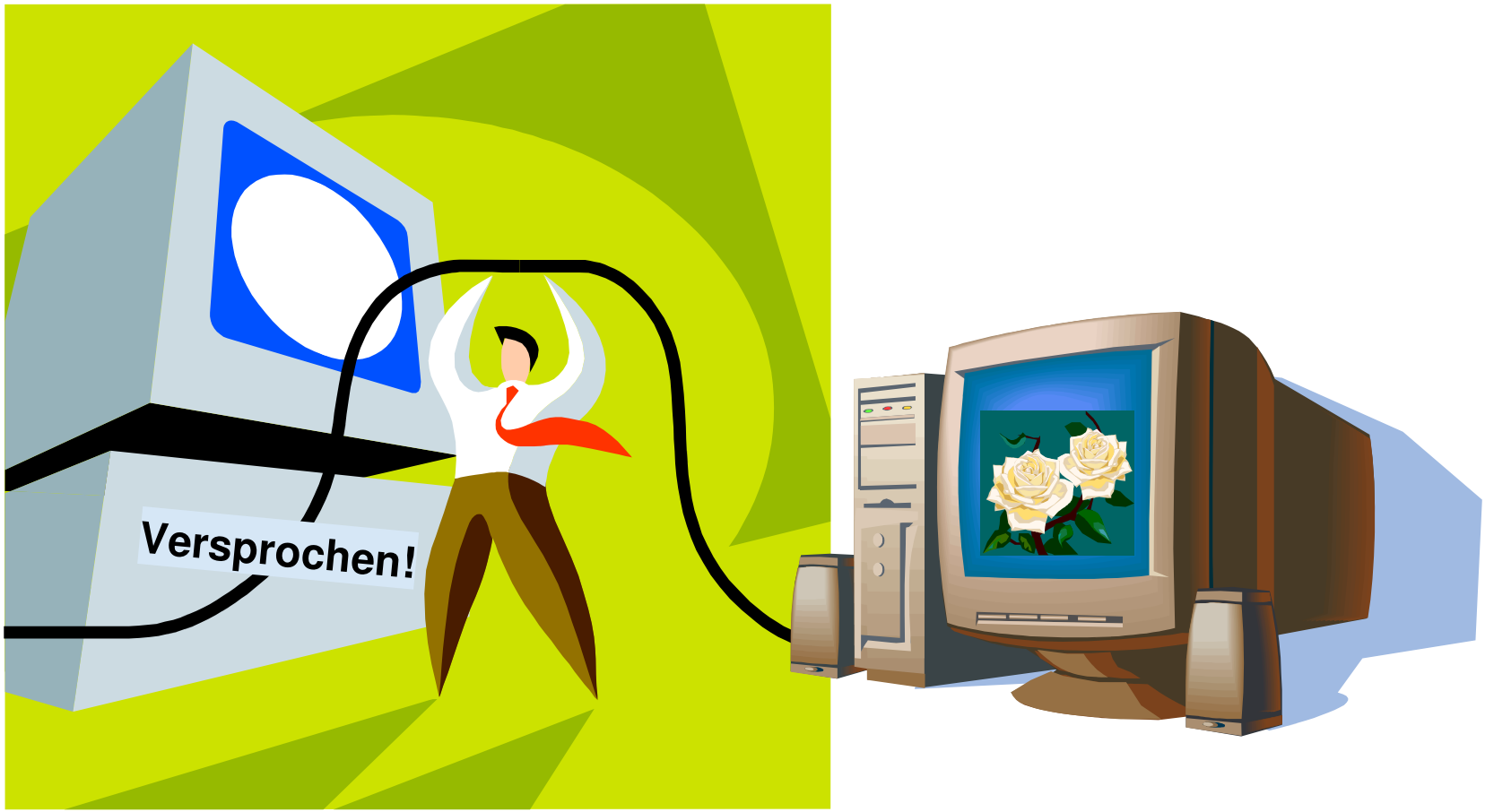
- /var/cfengine/bin
  - Die wichtigen Kommandos (cfgent, cfexecd, …) werden auf jedem Client lokal abgelegt um von NFS unabhängig zu sein.
- /var/cfengine/inputs
  - Beschreibungsdateien für den Systemzustand finden sich hier
- /var/cfengine/outputs
  - Ausgaben der Konfigurationsläufe
- /var/cfengine/ppkeys
  - Ablage für Public und Private Key des Clients bzw. Servers
- /var/cfengine/repository
  - Altdateien werden von uns hier für einige Zeit aufgehoben um eine Rückfallposition zu haben.

Die Dateisystemstruktur auf dem Beispielserver

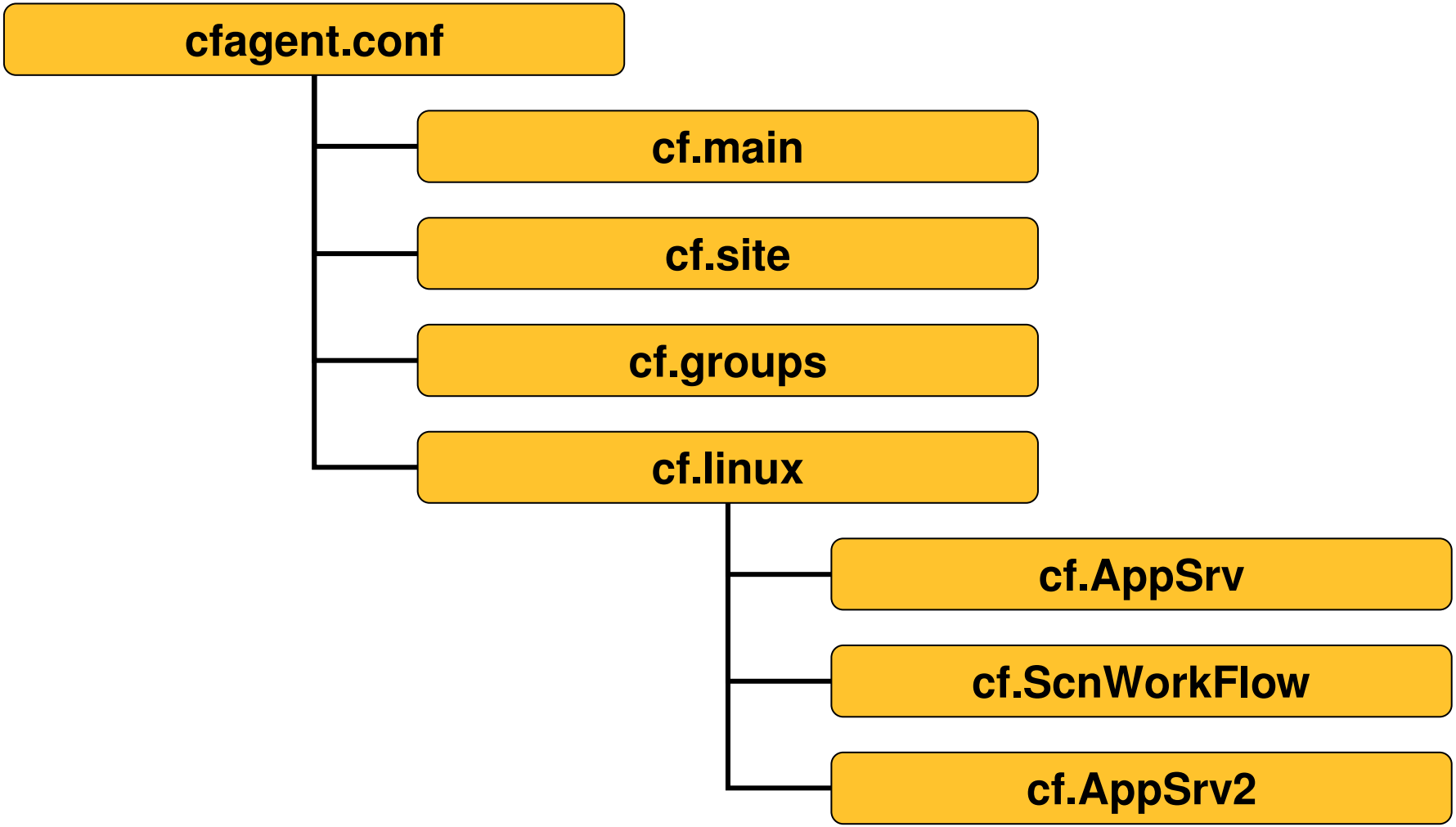
- /usr/local/vsa/cfengine
  - Basisverzeichnis auf dem Server.
  - Hier ist das Installationsskript von cfengine abgelegt
- /usr/local/vsa/cfengine/inputs
  - Beschreibungsdateien für den Systemzustand werden hier zur Verteilung eingestellt.
- /srv/www/SuSE
  - Das Filerepository ist als Datenquelle http://central/SuSE verfügbar.
- /var/cfengine
  - Basisverzeichnis für Dateien des Masters "cfservd".
- /work/src/cfengine
  - Strukturiertes Verzeichnis für Dateiverteilung per cfengine auf die Clients

Trainingseinheit

- Vorführung gewünscht ?



Ein mögliches Modularisierungskonzept



Installationsprozeß eines Systems

- Erstinstallation per autoYaST
  - Nur Basisbetriebssystem wird installiert
  - Flexibilität senkt Hardwarekosten
- Integration in die Systemumgebung erfolgt manuell
  - Netzwerkkonfiguration
  - Aufspielen von cfengine
- Anpassung die Aufgabe des Systems durch cfengine
  - Zusätzliche Software oder Patches installieren
  - Konfigurationen anpassen und aktualisieren
- Abnahme durch QS
  - cfengine liefert reproduzierbaren Systemstand

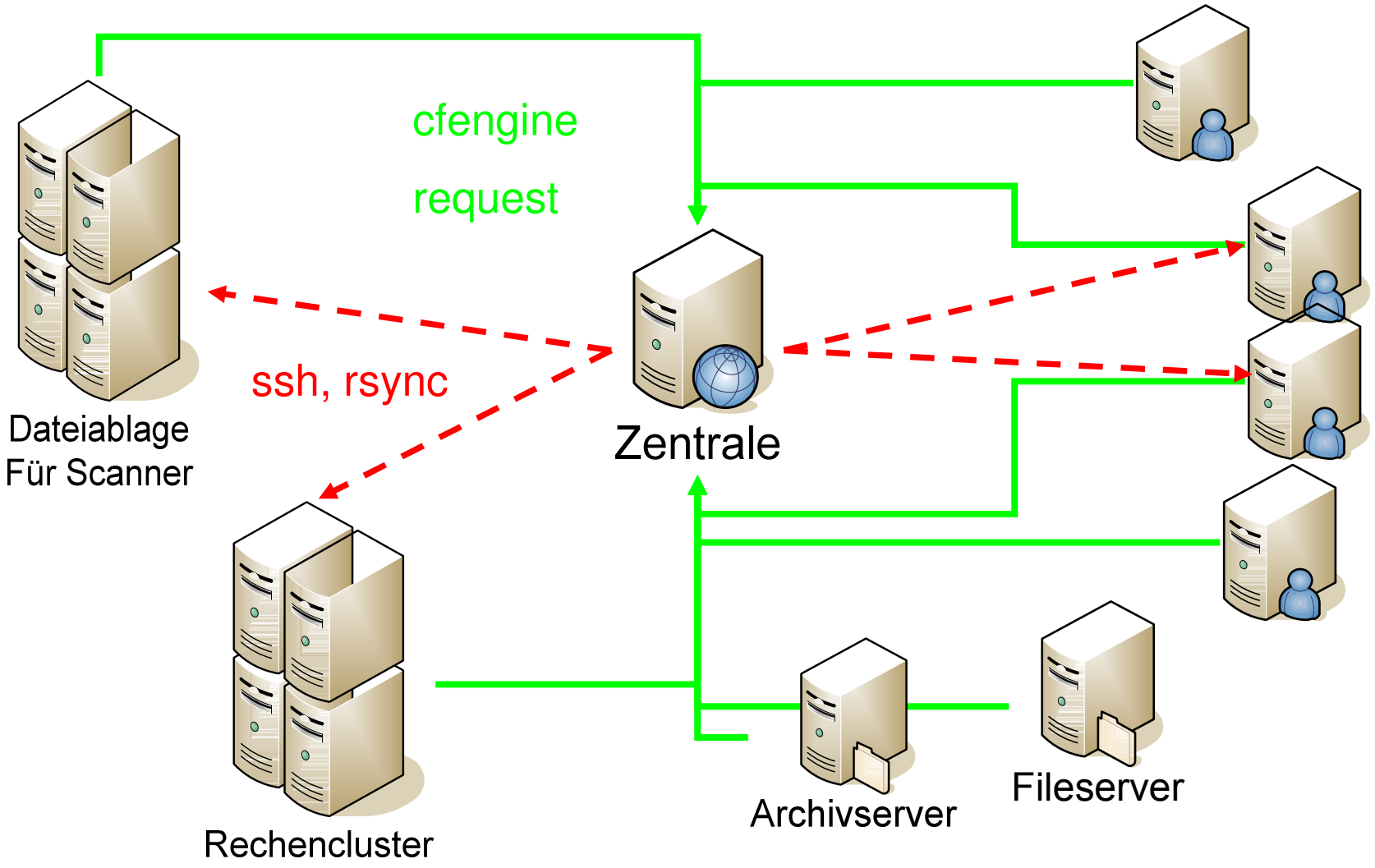
Revisionssicherheit

- Überwachen des Systems mittels cfengine
  - Ungewollte Änderungen erkennen (Prüfsummen)
  - Wiederherstellen des korrekten Zustandes
  - Alarmierung der Systemverantwortlichen
- Aktive Unterstützung beim Änderungsprozeß
  - Änderungen werden geplant und an vielen Systemen gleichzeitig korrekt durchgeführt

Der „neue“ Managementansatz

- Installation der Server per autoYaST
- PXE-Bootserver zur zentralen, einfachen Installation
- Clientsysteme werden per cfengine konfiguriert
- Zentrale Versionierung von wichtigen Dateien per cvs
- Zentraler Installationserver für alle eingesetzten SuSE-Versionen
- Eigene Softwarerepositories für Zusatzsoftware
- Desaster Recovery global integrieren

Die neue Managementstruktur



Quellen

- http://www.cfengine.org, die offizielle Homepage
  - Downloads
  - Links zu guter Dokumentation
  - Gute Papers unter http://www.cfengine.org/papers.html
- http://eternity.iu.hio.no/papers/rosegarden.pdf
- http://www.cfwiki.org
  - Das Wiki zu cfengine
- Zusätzlich interessant sind:
  - Kirk Bauer, „Automating UNIX and Linux Administration“, a! Apress, 2003
  - Erik Keller, „Unix/Linux Survival Guide. Der Leitfaden zur plattformunabhängigen Systemverwaltung.“, Addison-Wesley
  - Thomas Lange, FAI, http://www.informatik.uni-koeln.de/fai/

Vielen Dank für Ihre Aufmerksamkeit!

Fragen? Anmerkungen?

Diskussionsrunde!