Puppet und Co

Konfigurations-Management

Philipp Grau

sage@GUUG

07 October 2010

- 1993 erste Berührung mit Unix im Studium
- System-Administrator an der ZEDAT (HRZ der FU Berlin)
- Unix-Dinge, die ich nicht mag: `cpio`, `find`, `vi`
- Unix-Dinge, die ich mag: `tar`, `Debian`, `emacs`

Origin2000.jpg

Damals

Origin2000.jpg

Damals

- Ein Server
- Viele Admins

Server-Room.jpg

Heute

Server-Room.jpg

Heute

- Viele Server
- Ein Admin

Aktuelle Fragestunde

- Ruby
- cfengine
- Automatisierte Installation (FAI o.ä.)

Aktuelle Fragestunde

- Ruby
- cfengine
- Automatisierte Installation (FAI o.ä.)

Aktuelle Fragestunde

- Ruby
- cfengine
- Automatisierte Installation (FAI o.ä.)

Aktuelle Fragestunde

- Ruby
- cfengine
- Automatisierte Installation (FAI o.ä.)

Aktuelle Fragestunde

- Ruby
- cfengine
- Automatisierte Installation (FAI o.ä.)
- Puppet

Virtuelle Maschinen, die *Cloud*, Bladecenter und überhaupt

- viele Server
- Statt einer großen viele (kleine|virtuelle) Maschinen
- vApp, Virtuelle Appliances
- Bladecenter (viele CPUs pro HE)

Voraussetzungen?!

- Administrationskonzept vorhanden
- Server zentral verwaltbar

Ziele (Grob)

- Konsistenz der Systemumgebung
- Single Point of Change
- Einstieg für alle beteiligten Admins einfach

Ziele (Grob)

- Konsistenz der Systemumgebung
- Single Point of Change
- Einstieg für alle beteiligten Admins einfach
  Ja, es sind dann doch mehrere!

Ziele (Grob)

- Konsistenz der Systemumgebung
- Single Point of Change
- Einstieg für alle beteiligten Admins einfach
  Ja, es sind dann doch mehrere!
- DRY wird als Prinzip verstanden

Virtues of a programmer

Laziness	Viele gleiche Arbeitsschritte vermeiden
Hubris	Konfiguration vereinheitlichen
Impatience	Schneller Überblick über Infrastruktur

Ziele (feiner)

- Softwaremanagement
- Usermanagement
- Rechtemanagement
- Fileverteilung
- Templates für Dateien

Automatische Installation ist nur die halbe Miete

- Auch hier muss vorbereitet werden
  - Tests, Netzwerkinterfaces, BIOS-Konfiguration
- Wie war der Name doch gleich:

Debian	Pre-Seeding, FAI
RedHat	Kickstart, Cobbler
SuSE	AutoYast
Solaris	JumpStart

Änderungen im Betrieb sind die Regel

- Never change a running system

Änderungen im Betrieb sind die Regel

- Never change a running system, gibt es leider nicht.

Änderungen im Betrieb sind die Regel

- Never change a running system, gibt es leider nicht.
- Irgendwer hat immer irgendwelche Wünsche (Neue Shell, irgendeine Bibliothek, CPAN-Hölle, neue SSH-Keys)

Änderungen im Betrieb sind die Regel

- Never change a running system, gibt es leider nicht.
- Irgendwer hat immer irgendwelche Wünsche (Neue Shell, irgendeine Bibliothek, CPAN-Hölle, neue SSH-Keys)

Und was ist mit der Dokumentation?

Wie verteilt man Änderungen?

- `on-all`, `cluster-ssh`, `rdist`, geshartes `/usr/local`
- was ist mit neu installierten Maschinen, sind da alle Änderungen nachgeführt?
- Änderungen in Dateien?! `sed`
- und lokale Änderungen

Wie verteilt man Änderungen?

- `on-all`, `cluster-ssh`, `rdist`, geshartes `/usr/local`
- was ist mit neu installierten Maschinen, sind da alle Änderungen nachgeführt?
- Änderungen in Dateien?! `sed`
- und lokale Änderungen

Und was ist mit der Dokumentation?

Werkzeugkoffer

Es gibt Programme, Tools, Frameworks, die es einem ermöglichen sollen, einzelne Hosts, Cluster oder Serverfarmen verwalten.
Hier eine kleine Auswahl:

Werkzeugkoffer

Es gibt Programme, Tools, Frameworks, die es einem ermöglichen sollen, einzelne Hosts, Cluster oder Serverfarmen verwalten.
Hier eine kleine Auswahl:

- One-Shot-Installer

Werkzeugkoffer

Es gibt Programme, Tools, Frameworks, die es einem ermöglichen sollen, einzelne Hosts, Cluster oder Serverfarmen verwalten.
Hier eine kleine Auswahl:

- One-Shot-Installer
- Idempotente Systeme

Idempotenz von Operationen

Aus der Informatik:

- Undo- Redo-Operationen müssen bei mehrfacher Ausführung gleiche Resultate zeigen
- Editieren in Konfigurations-Dateien (`append-if-no-such-line` vs. `echo "option=on" >> file`)
- Lesen ist idempotent
- Schreibende Vorgänge können idempotent gemacht werden. (`einzahlung(100)` vs. `neuerKontostand(600)`)

[1]

- Nix wie weg hier!

[1] Tagcould: wordle.net

Werkzeuge (Teil 1)

- automateit (http://automateit.org/)
  - Ruby, kein Server und keine DSL
- chef (http://www.opscode.com/chef/)
  - Ruby and CouchDB (und OpenID)
- puppet (http://puppet.reductivelabs.com/)
  - läuft überall (ex. MS)
  - Ruby mit DSL

Werkzeuge (Teil 2)

- FAI mit Softupdates (http://www.informatik.uni-koeln.de/fai/)
  - Mit alles und scharf
- bcfg2 (http://trac.mcs.anl.gov/projects/bcfg2)
  - Uni-Entwicklung, läuft überall (ex. MS), XML
- cfengine (http://www.cfengine.org/)
  - Mark Burgess (Theorie des Konfigurations-Management)

Werkzeuge (Teil 3)

- Capistrano (http://www.capify.org/)

  Capistrano is an open source tool for running scripts on multiple servers; its main use is deploying web applications. It automates the process of making a new version of an application available on one or more web servers, including supporting tasks such as changing databases.

Werkzeuge (Teil 4)

- Sprinkle (http://github.com/crafterm/sprinkle)
  - Lightwight Ansatz
    http://www.agileweboperations.com/sprinkle-automated-infrastructure-for-the-rest-of-us/

    Sprinkle is a software provisioning tool you can use to build remote servers with, after the base operating system has been installed. For example, to install a Rails or Merb stack on a brand new slice directly after its been created.

Big Player (OpenSource)

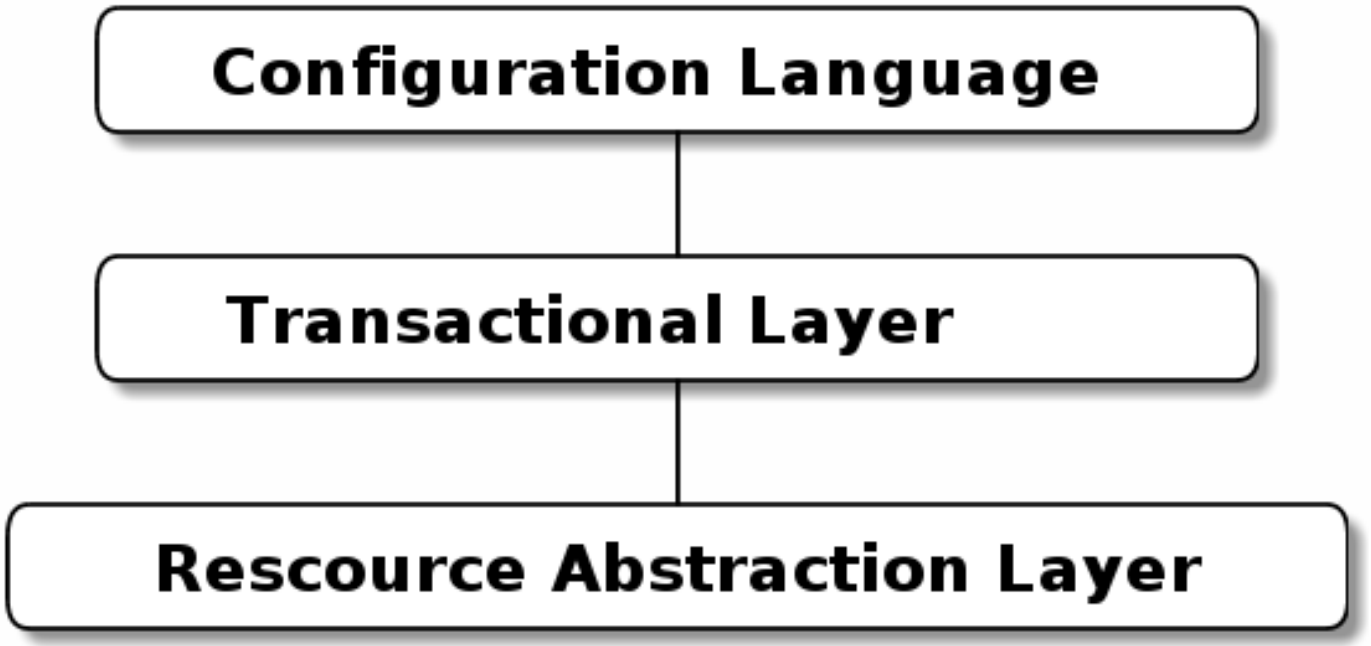
- cfengine
- puppet
- chef
- bcfg2

Warum Puppet

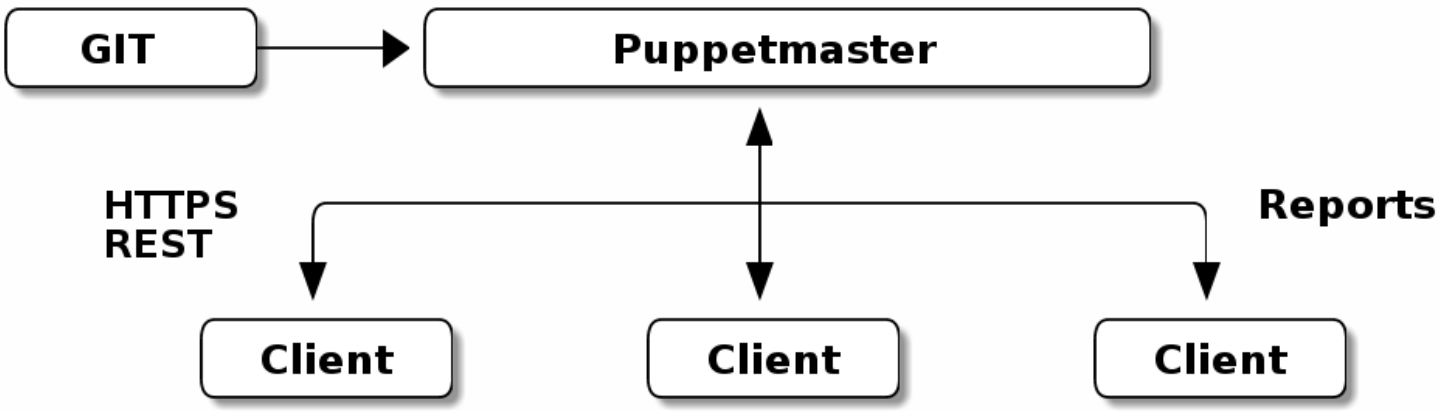
- Gute Dokumentation
- Gewisse Sympathien wg. Ruby
- Kein XML zur Konfiguration (wie bcfg2)
- Es ist nicht cfengine

RALSH

So sieht's aus:



Kommunikation



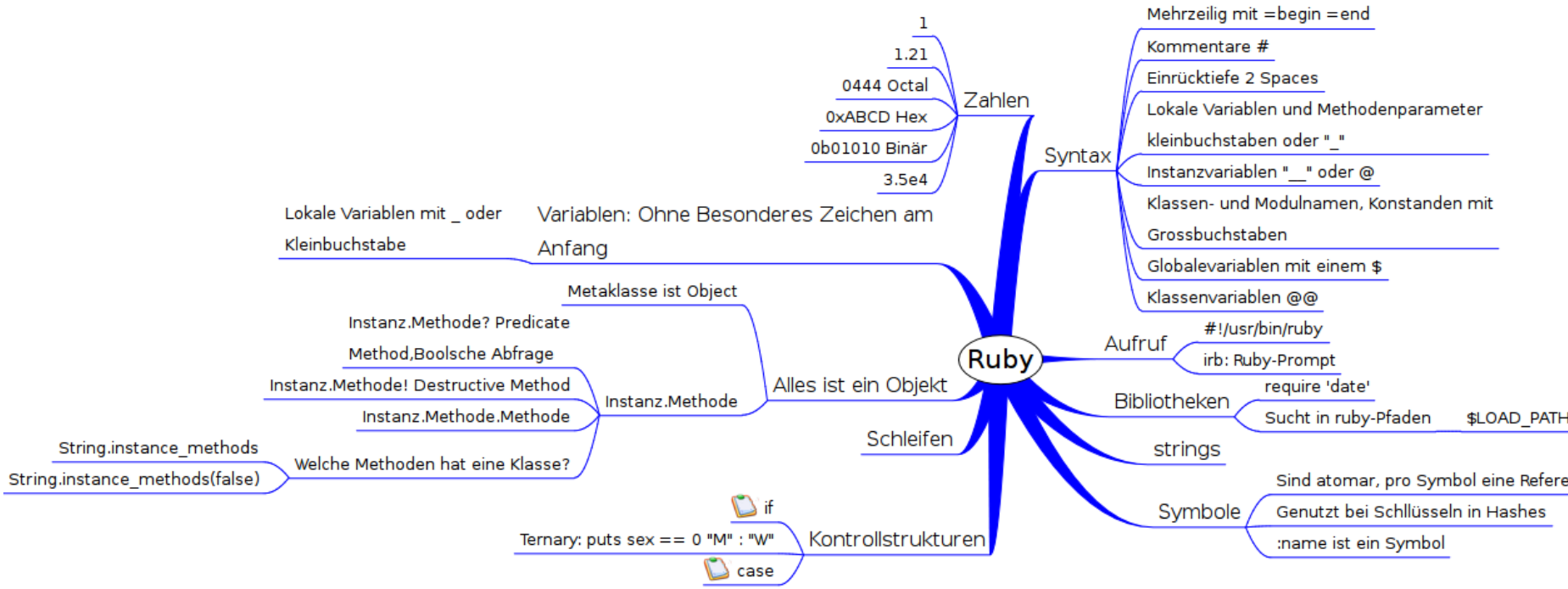
Voraussetzung

- Ruby (Version 1.8.2+, 1.9 wird z.Z. nicht empfohlen)
- plus einige Pakete aus der Distributions-Tüte (Installations-Guide)
- SSL-Zertifikate (eigene CA)
- Bei größeren Installationen (> 20 Hosts) sollte eine anderer Webserver als der eingebaute WebBrick benutzt werden (Mongrel oder Passenger)

Exkurs Ruby

- ▸ Nix wie weg hier!

Mindmap Ruby



Ruby

- Skriptsprache
- http://www.ruby-lang.org/
- 21. Dezember 1995 die erste Version von Ruby, 0.95
- Aktuell 1.9.2

Die Ideale des Ruby-Erfinders

Ruby ist eine Sprache der Balance. Ihr Schöpfer Yukihiro ‘matz’ Matsumoto vermischte Teile seiner Lieblingssprachen (Perl, Smalltalk, Eiffel, Ada und Lisp) und formte daraus eine neue Programmiersprache, in der funktionale und imperative Programmierung ausbalanciert sind.

Beispiele

```
puts "Hello World!"
```

- Alles ist ein Objekt

```
"Hello World!".methods

puts "\n\object.methods : "+
  "Hello World".methods.sort.join("\n").to_s+"\n\n"

5.times { puts "Wir * lieben * Ruby!" }
```

```
search_engines =
   %w[Google Yahoo Bing].map do |engine|
    "http://www." + engine.downcase + ".com"
end
```

Templates mit erb

```
require 'erb'

x = 42
template = ERB.new <<-EOF
    The value of x is: <%= x %>
EOF
puts template.result(binding)
```

The value of x is: 42

(Templates gibt es in vielen Geschmacksrichtungen[1])

[1] http://www.hokstad.com/mini-reviews-of-19-ruby-template-engines.html

Was kann es:

Puppet

1. Fileverteilung
2. Packetinstallation
3. Usermanagement
4. Templates für Dateien
5. Vererbung
6. Erweiter- und anpassbar
7. Konfigurationssprache fast Ruby

Betriebsdetails

puppetmaster Daemon auf Server, Zertifikat-basierte Authentifizierung

puppetd Daemon auf Client, wird in Intervallen aktiv

puppetrun Triggerprogramm auf dem Server

SSL und Zertifikate

- SSL-Zertifikats-Basierte Kommunikation
- Client erzeugt Zertifikat
- Client schickt Zertifikat an Server
- Weiter geht es erst wenn Zertifikat auf dem Server signiert wurde (`puppetca`)

Facter

Sammelt Informationen über das System ein und stellt diese als Variablen bereit

$$architecture => i386$$
$$domain => campus.fu\text{-}berlin.de$$
$$facterversion => 1.5.4$$
$$fqdn => Arwaut.campus.fu\text{-}berlin.de$$
$$hardwareisa => unknown$$
$$hardwaremodel => i686$$
$$hostname => Arwaut$$

Verzeichnisstruktur der Konfigurations-Dateien

```
/etc/puppet
      +-- /files
      +-- /manifests
            +-- /classes
      +-- /templates
      +-- /modules
```

`files` anders als FAI nicht hierarchisch aufgebaut

`templates` Dateivorlagen

`manifests` Beschreibung der Aufgaben

`modules` Eigene Typen

Beispiele

```
# /etc/puppet/manifests/site.pp

import "classes/[a-z]*.pp"

node default {
  notice('Hello world!')
  include operator
  include packages
  include filedist
  file { "/etc/sudoers":
    owner => root, group => root, mode => 440
  }
}
...
```

User einrichten

```
class operator {
  group { "operator":
    ensure => present,
    gid    => 37,
  }
  user { 'operator':
    ensure  => 'present',
    uid     => '14195',
    comment => 'ZEDAT FS',
    gid     => "operator",
    home    => '/home/operator',
    shell   => '/bin/bash',
  }
}
```

Pakete installieren

```
class packages {

  define zedat-gold() {
    package { "${name}": ensure => installed }
  }

  zedat-gold { ["sipcalc", "htop", "multitail",
                "lftp", "ack-grep"]: }

  package { "oclock":      ensure => absent }
  package { "traceroute": ensure => absent }

}
```

Dateien verteilen

```
class filedist {

  file {"/etc/puppet/zedat":
    owner  => root,
    group  => root,
    mode   => 644,
    source =>
          "puppet://puppet/files/etc/puppet/zedat"
  }
}
```

Datein löschen

```
class absentfiles {

  define debabsent() {
    file { "${name}": ensure => absent }
  }

  debabsent { ["/etc/skel/.bash_logout",
               "/etc/skel/.bashrc",
               "/etc/skel/.profile"]:
  }
}
```

Templates für angepasste Dateien

Also die Anpassung von Konfigurations-Dateien in Abhängigkeit von IP-Nummer, Hostname etc.

Class

```
class www {

  file { "index":
    path    => "/var/www/index.html",
    owner   => root,
    group   => root,
    mode    => 644,
    content => template("index.erb")
  }
}
```

Template

```
<html>
<body>
<% if has_variable?("hostname") then %>
<h1><%= hostname %></h1>
<% end %>
<h2>It works!</h2>
</body>
</html>
```

Was gibt es?

Eine Auswahl der definierten Typen:

cron	mount	zone
exec	nagios	zpool
file	package	macauthorization
group	service	ssh
host	user	augeas
selinux		

Weitere Möglichkeiten

- Dienst in Abhängigkeit von Änderungen an Variablen neustarten
- Abhängigkeit von Variablen (Facter) für Fallentscheidungen nutzen

Abhängigkeiten I

```
file { "sshdconfig":
  name => $operatingsystem ? {
    solaris => "/usr/local/etc/ssh/sshd_config",
    default => "/etc/ssh/sshd_config",
  },
  owner => root,
  group => root,
  mode  => 644,
}
```

Abhänigkeit II

```
service { "sshd":
    subscribe => File[sshdconfig],
    ensure    => running,
}
```

- Dashboard:

Probleme

- Testbetrieb: Virtuelle Maschinen, wechselnde Netzwerkumgebungen machen mit Zertifikaten keinen Spass
- Größe Paket-Listen könnten für Puppet unhandlich sein (apt-get install für jedes Paket) evtl. ein Meta-Packet bauen oder eine der vielen Rezepte für diesen Zweck
- Ruby und Konsorten sind *fast moving targets*, Cfengine wirkt dagegen wie ein Eisberg

Danke!

Danke!

Fragen?

Buch

- James Turnbull: Pulling Strings with Puppet

URL-Sammlung I

- Vergleich Puppet Chef 1
- Vergleich Puppet Chef 2
- Vergleich Puppet Chef 3
- Vergleich Puppet Chef 4
- www.infrastructures.org/papers/turing/turing.html
- Vergleich Bcfg2 Puppet
- AutomateIt und der Rest der Welt
- Interview mit James Kanies (Puppet-Entwickler)
- Interview mit James Turnbull (Puppet-Buch-Schreiber)

URL-Sammlung II

Blog-Postings:

- Splitting puppetd from puppetmaster
- Howto: Einstieg in Puppet
- Effizientes Linux Systems Management
- Puppet errors explained
- Puppet resources