

# Zentrales Konfigurationsmanagement mit Puppet

SLAC 2011



WIZARDS OF FOSS  
Open Source Schulungen

Martin Alfke

<[martin.alfke@wizards-of-foss.de](mailto:martin.alfke@wizards-of-foss.de)>

# Einführung

- Wie managed man 600 Linux-Server mit 20 unterschiedlichen Applikationen?



**Einführung - Tools - Puppet - Continental**

# Einführung

- Wie managed man 600 Linux-Server mit 20 unterschiedlichen Applikationen?
- Wieviele Server/Applikationen kann ein Admin verwalten?



**Einführung - Tools - Puppet - Continental**

# Konfigurationsmanagement

- ssh-loop
  - for server in dbl db2 db3; do ssh ... done

# Konfigurationsmanagement

- ssh-loop
  - for server in db1 db2 db3; do ssh ... done
- Installationsimage
- Reboot + Reinstallation

# Konfigurationsmanagement

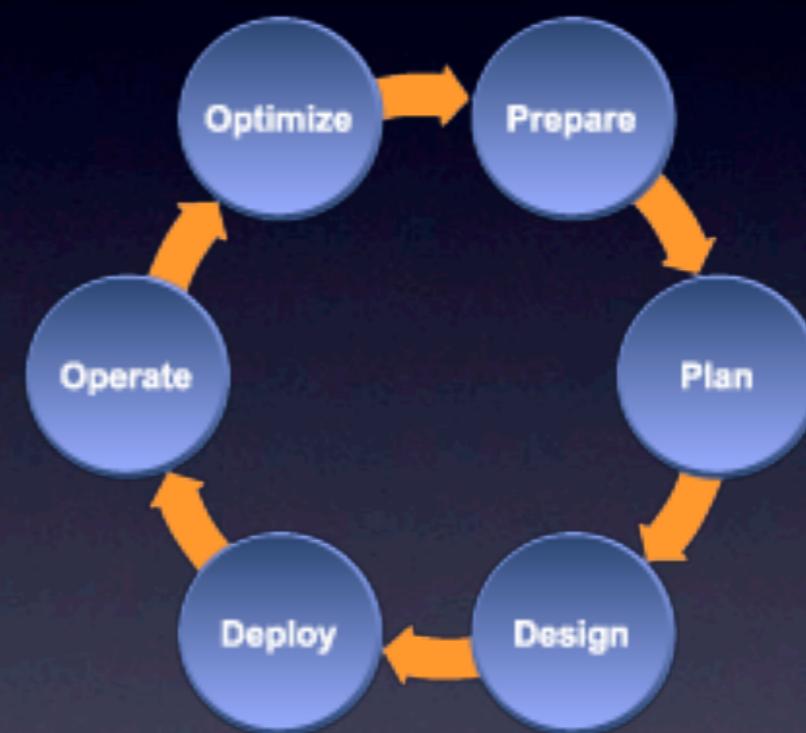
- ssh-loop
  - for server in db1 db2 db3; do ssh ... done
- Installationsimage
  - Reboot + Reinstallation

Reproduzierbar? Fehlertolerant?  
Verfügbarkeit?

**Einführung - Tools - Puppet - Continental**

# Konfigurationsmanagement

- Vorbereitung
- Planung
- Design
- Initialisierung
- Betrieb
- Verbesserung



Einführung - Tools - Puppet - Continental

# Tools

- CFEengine
- seit 1993



Einführung - **Tools** - Puppet - Continental

# Tools

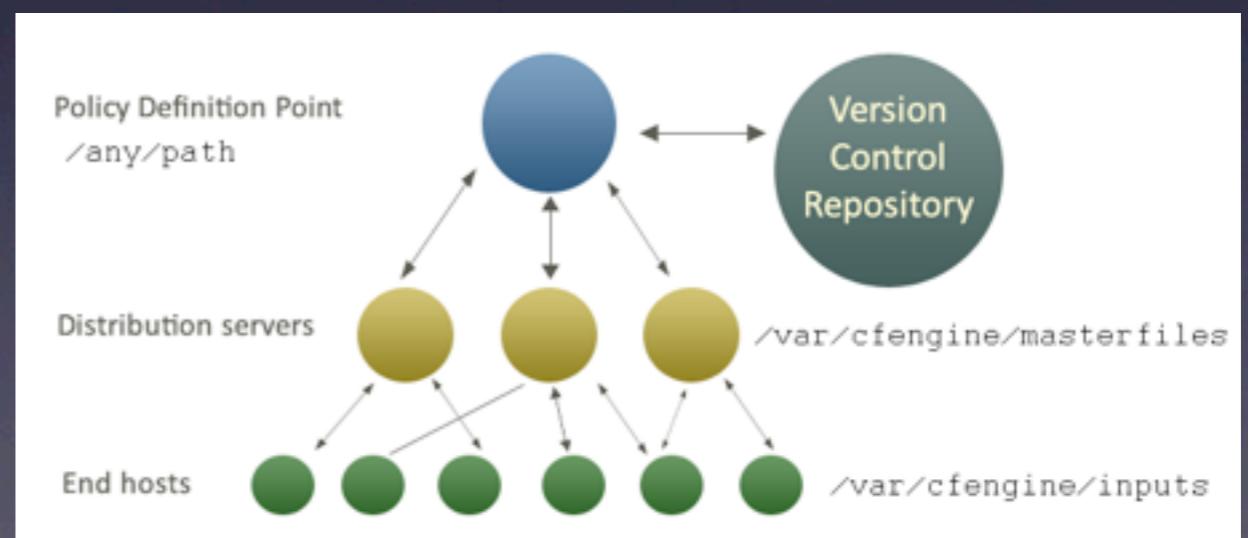
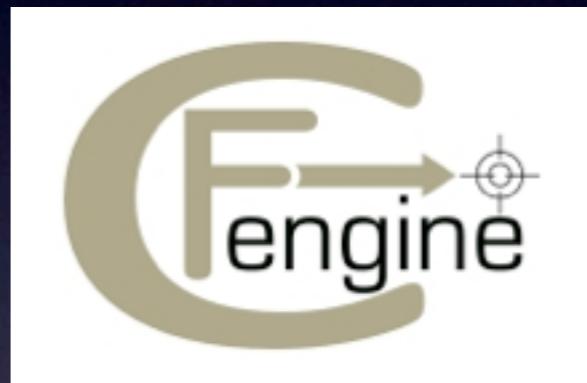
- CFEengine
  - seit 1993
  - atomare Änderungen



Einführung - **Tools** - Puppet - Continental

# Tools

- CFEengine
  - seit 1993
  - atomare Änderungen
  - Master-Client



Einführung - **Tools** - Puppet - Continental

# Tools

- CFEengine
- Bcfg2
- ca. seit 2003



Einführung - **Tools** - Puppet - Continental

# Tools

- CFEengine
- Bcfg2
- ca. seit 2003
- deklarative Beschreibung



Einführung - **Tools** - Puppet - Continental

# Tools

- CFEengine
- Bcfg2
- ca. seit 2003
- deklarative Beschreibung
- Auswertung der Client Antworten



Report Run @ July 24, 2006 10:58 a.m.

BCFG Clients Summary

Enter date or use calendar pop-up:  
2006-07-24 @ 10:58:28 Calendar Go Now

U..d

S	M	T	W	T	F	S
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Summary:  
137 Nodes were included in your report.  
115 nodes are clean.  
22 nodes are bad.

Node: pandora.mcs.anl.gov  
Node: squeak.mcs.anl.gov  
Node: fluke.anchor.anl.gov  
Node: athena.mcs.anl.gov

2005-12-05 06:25:13  
2005-12-05 06:25:21  
2005-11-18 06:25:18  
2005-12-05 06:25:25

Einführung - **Tools** - Puppet - Continental

# Tools

- CFEengine
- Bcfg2
- Puppet
- seit 2005



Einführung - **Tools** - Puppet - Continental

# Tools

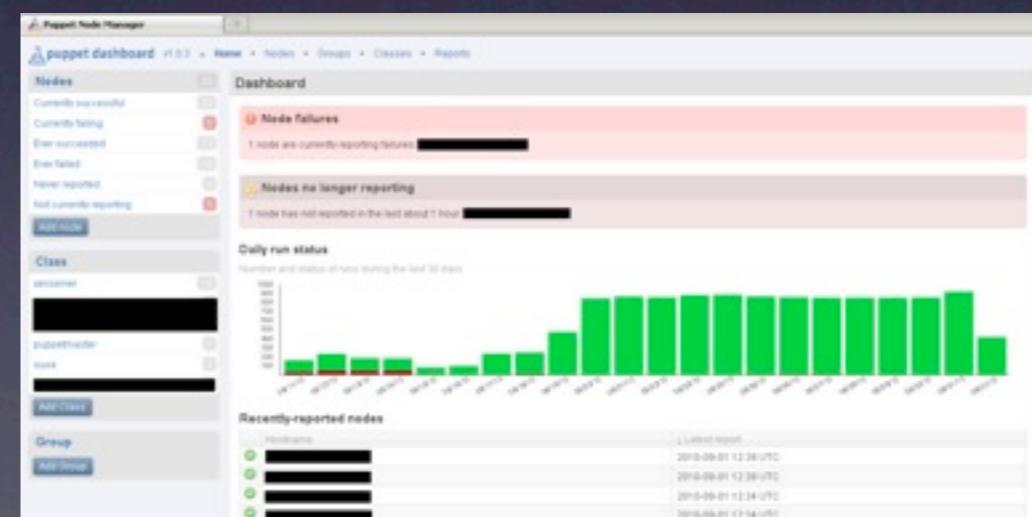
- CFEengine
- Bcfg2
- Puppet
  - seit 2005
  - Ruby Anwendung



Einführung - **Tools** - Puppet - Continental

# Tools

- CFEengine
- Bcfg2
- Puppet
- seit 2005
- Ruby Anwendung
- Reporting



Einführung - **Tools** - Puppet - Continental

# Tools

- CFEengine
- Bcfg2
- Puppet
- Open Source

# Tools

- CFEengine
- Bcfg2
- Puppet
- Open Source
- Fixpunkt Beschreibung

# Tools

- CFEengine
- Bcfg2
- Puppet
- Open Source
- Fixpunkt Beschreibung
- Client-Master Setup

# Puppet

- Master
  - Manifeste
  - Dateien
  - Vorlagen

# Puppet

- Master
  - Manifeste
  - Dateien
  - Vorlagen
- Client

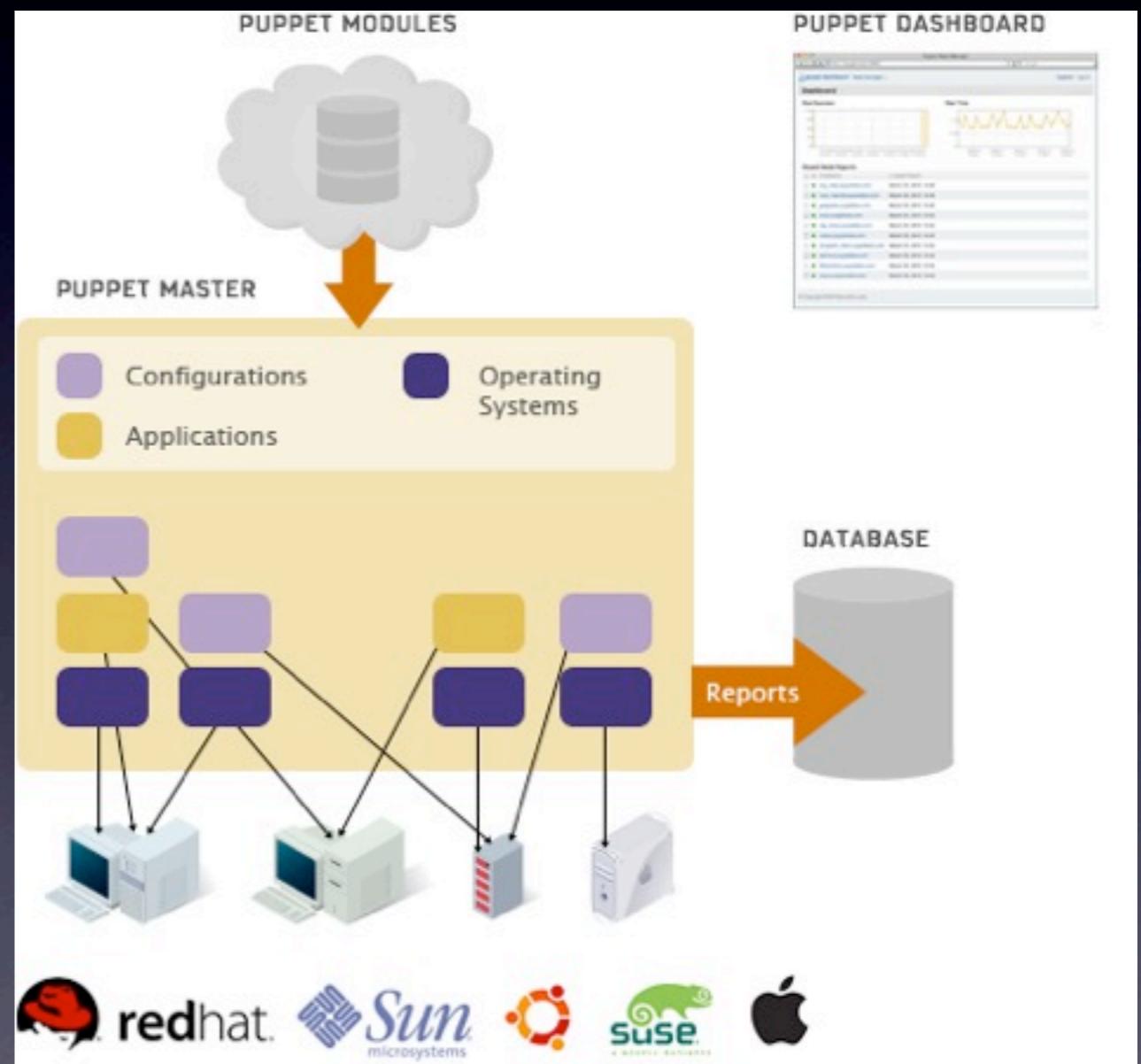
Einführung - Tools - **Puppet** - Continental

# Puppet

- Master
  - Manifeste
  - Dateien
  - Vorlagen
- Client
  - kein Provisioning!
  - FAI
  - Cobbler

# Puppet Aufbau

- Master
  - Beschreibung der Clients
- Client
  - Erstellt Report
  - Dashboard
  - Report Auswertung



# Puppet Ressourcen

## Syntax:

```
<resource-type> { "<namevar>":  
    <key1> => '<value1>',  
    <key2> => '<value2>',  
    <key3> => [ '<value3>', '<value4>' ],  
}
```

# Puppet Ressourcen

- user
- group

Beispiel:

```
group { "tux":  
    gid      => '1050',  
    ensure   => 'present',  
}  
  
user { "tux":  
    uid      => '1050',  
    gid      => '1050',  
    home    => '/home/tux',  
    shell   => '/bin/tcsh',  
    ensure   => 'present',  
}
```

# Puppet Ressourcen

- user
- group
- file

Beispiel:

```
file { "/etc/inet/inetd.conf":  
    ensure  => '/etc/inetd.conf',  
}
```

```
file { "/home/tux":  
    owner   => 'tux',  
    group   => 'tux',  
    mode    => '0700'  
    ensure  => 'directory',  
}
```

# Puppet Ressourcen

- user
- group
- file
- package
- service
- cron

## Beispiel:

```
package { "apache2":  
    ensure  => 'installed',  
}
```

```
service { "apache2":  
    ensure  => 'running',  
}
```

# Puppet Facts

- OS und System spezifische Parameter
- Variablen und Werte sind innerhalb von Puppet verwendbar

# Puppet Facts

*hardwaremodel => i386*

*hostname => client01*

*ipaddress => 192.168.45.114*

*kernel => Linux*

*kernelmajversion => 2.6*

*operatingsystem => Debian*

*operatingsystemrelease => 6.0*

*timezone => CET*

# Puppet Facts

- OS und System spezifische Parameter
- Variablen und Werte sind innerhalb von Puppet verwendbar

```
case $operatingsystem {  
  'Solaris': { ...  
  }  
  'Debian': { ...  
  }  
  default: { ...  
  }  
}
```

# Puppet Facts

- OS und System spezifische Parameter
- Variablen und Werte sind innerhalb von Puppet verwendbar
- Eigene Facts

# Puppet Facts

- OS und System spezifische Parameter
- Variablen und Werte sind innerhalb von Puppet verwendbar
- Eigene Facts

```
require 'facter'  
Facter.add("PUPPET_FUNCTION") do  
  setcode do  
    %x{/bin/cat /etc/function}.chomp  
  end  
end
```

# Puppet Vorlagen

- Host-spezifische Konfigurationsdateien
- Wiederverwendbare Konfigurationsdateien

# Puppet Vorlagen

- Host-spezifische Konfigurationsdateien
- Wiederverwendbare Konfigurationsdateien

```
$document_root    = '/srv/wiki'  
$vhost_port      = '80'  
$vhost_name       = 'wiki.domain.local'  
file { '/etc/apache2/apache2.conf'  
  content        => template ('apache2.erb'),  
}
```

# Puppet Vorlagen

- Host-spezifische Konfigurationsdateien
- Wiederverwendbare Konfigurationsdateien

```
$document_root      = '/srv/wiki'  
$vhost_port        = '80'  
$vhost_name         = 'wiki.domain.local'  
file { '/etc/apache2/apache2.conf'  
    content          => template ('apache2.erb'),  
}
```

```
# apache2 config  
DocumentRoot = <% document_root %>  
NameVirtualHost = <% vhost_name -%>:<%vhost_port %>
```

# Puppet Klassen

- Gruppieren logisch zusammengehörender Definitionen

# Puppet Klassen

- Gruppieren logisch zusammengehörender Definitionen

```
class apache {  
    package{'apache2': ensure => installed }  
    service{'apache2': ensure => running }  
    file{'/etc/apache2/apache2.conf':  
        mode     => 644,  
        owner    => root,  
        group   => root,  
    }  
}
```

# Puppet Module

- Sammlung von Klassendefinitionen mit Dateien und Vorlagen innerhalb einer Verzeichnisstruktur

Einführung - Tools - **Puppet** - Continental

# Puppet Module

- Sammlung von Klassendefinitionen mit Dateien und Vorlagen innerhalb einer Verzeichnisstruktur

```
modules/
  <modulename>/
    manifests/
      init.pp          <- erforderlich
      <classname>.pp <- optional (Klassen)
    templates/        <- optional (Vorlagen)
    files/            <- optional (Dateien)
    lib/              <- optional (Facts)
```

# Puppet Module

- Beispiel

```
modules/  
  apache/  
    manifests/  
      init.pp  
    templates/  
      vhost.erb  
  files/  
    apache2.conf
```

# Puppet Module

- Beispiel

```
modules/
apache/
  manifests/
    init.pp
```

```
class apache {
  package { 'apache2': ensure => present }
  file { '/etc/apache2/apache2.conf':
    source => 'puppet:///modules/apache/apache2.conf',
  }
}
```

# Puppet Performance

- Standard: Ruby Webrick
  - bekannt für schlechte Performance
  - einfache Implementierung

# Puppet Performance

- Standard: Ruby Webrick
  - bekannt für schlechte Performance
  - einfache Implementierung
- vorgeschalteter Apache/NGINX/Pound
- Apache mit mod\_passenger

# Puppet Reporting

- Client sendet Bericht an Master

# Puppet Reporting

- Client sendet Bericht an Master
- Master speichert Bericht im Dateisystem oder in MySQL-Datenbank

# Puppet Reporting

- Client sendet Bericht an Master
- Master speichert Bericht im Dateisystem oder in MySQL-Datenbank
- Puppet Dashboard liest Berichte aus Datenbank

# Puppet Reporting

- Client sendet Bericht an Master
- Master speichert Bericht im Dateisystem oder in MySQL-Datenbank
- Puppet Dashboard liest Berichte aus Datenbank
- Dashboard erzeugt strukturierte Darstellung

# Puppet Reporting

Screenshot of the Puppet Node Manager dashboard (v1.0.3) showing reporting status and run history.

**Nodes** (23 total):

- Currently successful: 22
- Currently failing: 1
- Ever succeeded: 23
- Ever failed: 15
- Never reported: 0
- Not currently reporting: 1

**Add node**

**Class** (3 total):

- xenserver (14 runs)
- puppetmaster (1 run)
- suse (2 runs)

**Add Class**

**Group**

**Add Group**

**Dashboard**

**Node failures**: 1 node are currently reporting failures: [REDACTED]

**Nodes no longer reporting**: 1 node has not reported in the last about 1 hour: [REDACTED]

**Daily run status**: Number and status of runs during the last 30 days.

Date	Runs
2010-08-11	~150
2010-08-12	~200
2010-08-13	~150
2010-08-14	~150
2010-08-15	~100
2010-08-16	~100
2010-08-17	~200
2010-08-18	~250
2010-08-19	~400
2010-08-20	~800
2010-08-21	~850
2010-08-22	~850
2010-08-23	~850
2010-08-24	~850
2010-08-25	~850
2010-08-26	~850
2010-08-27	~850
2010-08-28	~850
2010-08-29	~850
2010-08-30	~850
2010-08-31	~150
2010-09-01	~200

**Recently-reported nodes**:

Hostname	Latest report
[REDACTED]	2010-09-01 12:39 UTC
[REDACTED]	2010-09-01 12:39 UTC
[REDACTED]	2010-09-01 12:34 UTC
[REDACTED]	2010-09-01 12:34 UTC

Einführung - Tools - **Puppet** - Continental

# Puppet @ **Continental**®

- Allgemein
- UNIX Abteilung

Einführung - Tools - Puppet - **Continental**

# Puppet @ Continental

- Allgemein
  - UNIX Abteilung
  - Applikations Abteilung

Einführung - Tools - Puppet - **Continental**

# Puppet @ **Continental**®

- Zustand 2009

Einführung - Tools - Puppet - **Continental**

# Puppet @

- Zustand 2009
  - vorhandenes Konfigurationsmanagement
  - kommerzielle Software
  - Lizenzpflichtig
  - gewachsenes Setup (Standards mit Ausnahmen)

Einführung - Tools - Puppet - **Continental**

# Puppet @ Continental

- Definierte Aufgaben von Puppet

Einführung - Tools - Puppet - **Continental**

# Puppet @

- Definierte Aufgaben von Puppet
  - Anlegen/Löschen von Usern
  - Anlegen/Löschen/Verwalten von Konfigurationsdateien
  - Starten/Stoppen/Deaktivieren/Aktivieren von Diensten
  - Installation/Deinstallation von Paketen

Einführung - Tools - Puppet - **Continental**

# Puppet @ Continental

- Puppet Setup

Einführung - Tools - Puppet - **Continental**

# Puppet @

- Puppet Setup
  - zwei Module mit Unterklassen
    - Location
    - Function
  - Puppet Beschreibungen und Dateien in GIT
  - Dashboard

Einführung - Tools - Puppet - **Continental**

# Puppet @ **Continental**®

- Puppet Client Setup

Einführung - Tools - Puppet - **Continental**

# Puppet @ **Continental**®

- Puppet Client Setup
- Basis-Installation via OVS oder Kickstart/  
Jumpstart
- `puppet.conf` ist Bestandteil der Basis-  
Installation
- `/etc/puppet_function` wird während der  
Basis-Installation erzeugt

# Puppet @

- Puppet Master Setup

Einführung - Tools - Puppet - **Continental**

# Puppet @

- Puppet Master Setup
- Software Komponenten
  - RHEL 5.2
  - ruby 1.8.7p334 als Backport
  - Puppet 2.6.9
  - Puppet Dashboard 1.2.0

Einführung - Tools - Puppet - **Continental**

# Puppet @ Continental

- Puppet Master Setup
  - Software Komponenten
  - Location Modul
  - Auslesen des Domain facts und includen der notwendigen Sub-Klassen

Einführung - Tools - Puppet - **Continental**

# Puppet @

- Puppet Master Setup
  - Software Komponenten
  - Location Modul
  - Function Modul
  - Auslesen von `/etc/puppet_function` und includen der Sub-Klassen

Einführung - Tools - Puppet - **Continental**

# Puppet @

- Puppet Master Setup

- `puppet_function` fact

```
require 'facter'

Facter.add("PUPPET_FUNCTION") do
  setcode do
    %x{/bin/egrep "^\$PUPPET_FUNCTION=" /etc/
    puppet_function | sed -e 's/.*=//'}.chomp
  end
end
```

# Puppet @

- Puppet Master Setup
  - `puppet_function` fact

`PUPPET_FUNCTION=HPC-INFINIBAND`

# Puppet @ **Continental**®

- Puppet Dual Client

Einführung - Tools - Puppet - **Continental**

# Puppet @ **Continental**®

- Puppet Dual Client
  - Zwei Abteilungen (UNIX + Application)
  - Beide wollen eigenen Puppet Master
  - UNIX Puppet Daemon im root-Kontext
  - Applikations Puppet im Kontext des Applikations-Users

Einführung - Tools - Puppet - **Continental**

# Puppet @

- Puppet Dual Client
  - UNIX Puppet (root-User) verwaltet die Konfigurationsdateien und init-Skripte für den zweiten Puppet Client
  - Applikation Puppet (Funktions-User) verwendet eigene Konfgurationsdateien, eigenes libdir, vardir, logdir, ssldir, server und report\_server

# Puppet @ **Continental**®

- Projekt Verlauf
  - Start August 2010
  - Stand November 2011
    - Puppet 2.6.9
    - Dashboard 1.2.0
    - ca. 550 Server in Puppet

Einführung - Tools - Puppet - **Continental**

# Zentrales Konfigurationsmanagement mit Puppet

Demo

Martin Alfke  
[`<martin.alfke@wizards-of-foss.de>`](mailto:<martin.alfke@wizards-of-foss.de>)

# Zentrales Konfigurationsmanagement mit Puppet

Fragen??

Martin Alfke  
[`<martin.alfke@wizards-of-foss.de>`](mailto:<martin.alfke@wizards-of-foss.de>)