Zentrales Konfigurationsmanagement mit Puppet

SLAC 2011

WIZARDS OF FOSS
Open Source Schulungen

Martin Alfke
<martin.alfke@wizards-of-foss.de>

Einführung

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

Einführung

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

Konfigurationsmanagement

- ssh-loop
  - for server in db1 db2 db3; do ssh ... done

Konfigurationsmanagement

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

Konfigurationsmanagement

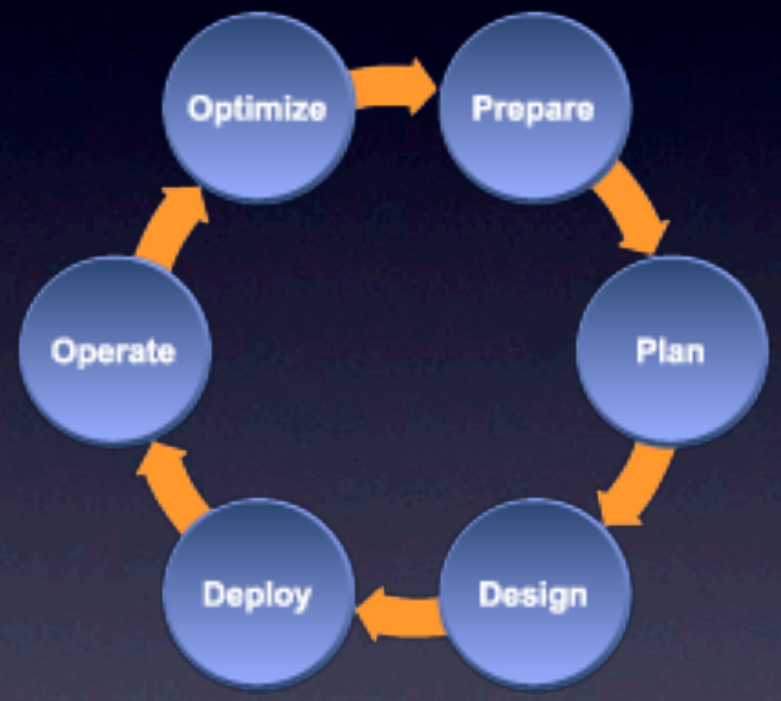
- ssh-loop
  - for server in db1 db2 db3; do ssh ... done
- Installationsimage
- Reboot + Reinstallation

Reproduzierbar? Fehlertolerant?
Verfügbarkeit?

Einführung - Tools - Puppet - Continental

Konfigurationsmanagement

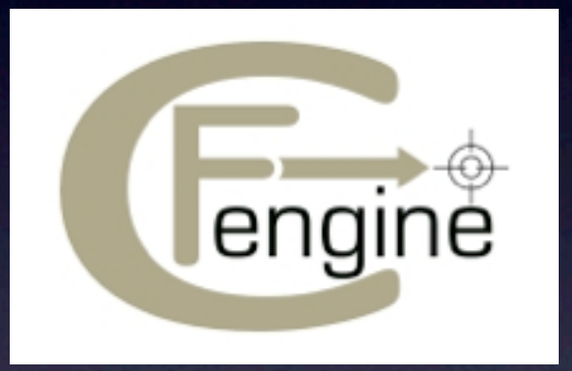
- Vorbereitung
- Planung
- Design
- Initialisierung
- Betrieb
- Verbesserung



Einführung - Tools - Puppet - Continental

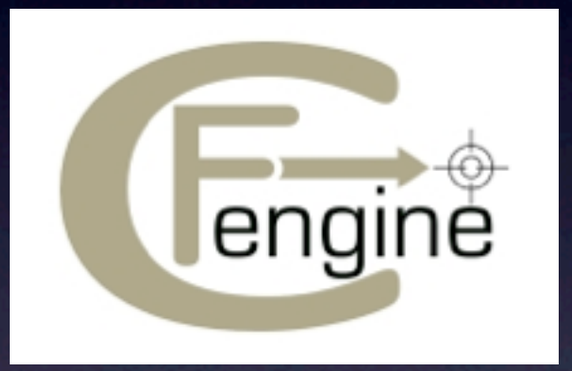
Tools

- CFEngine
  - seit 1993



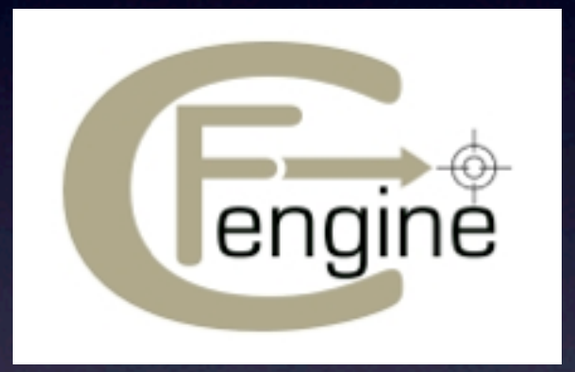
Tools

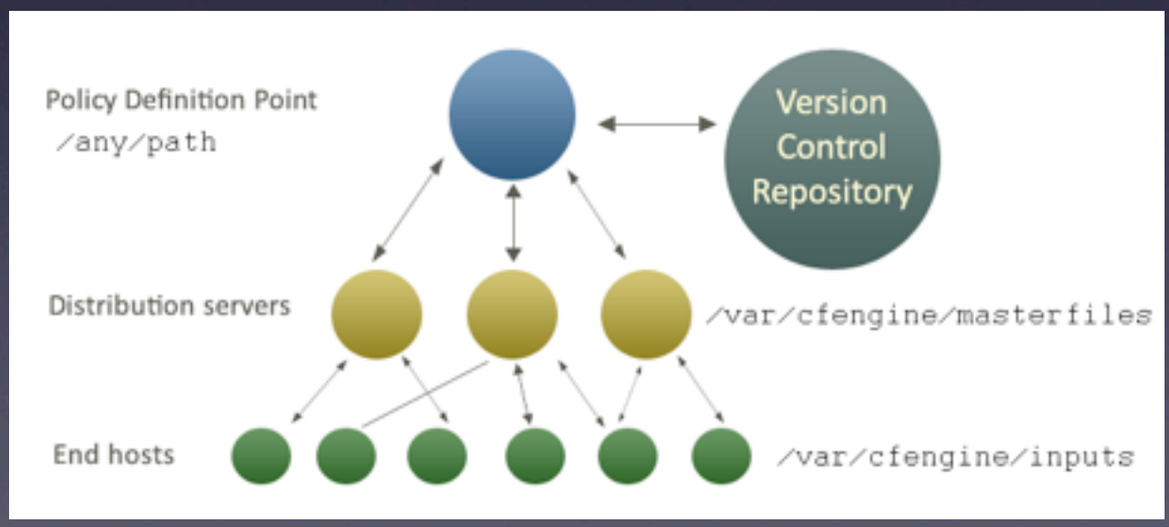
- CFEngine
  - seit 1993
  - atomare Änderungen



Tools

- CFEngine
  - seit 1993
  - atomare Änderungen
  - Master-Client





Tools

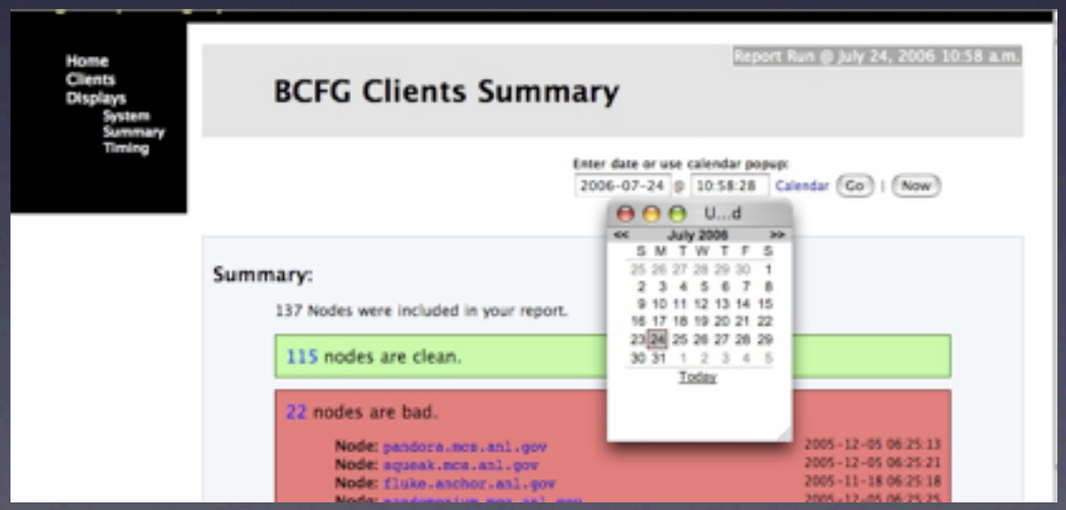
- CFEngine
- Bcfg2
  - ca. seit 2003

Tools

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

Tools

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



Tools

- CFEngine
- Bcfg2
- Puppet
- seit 2005

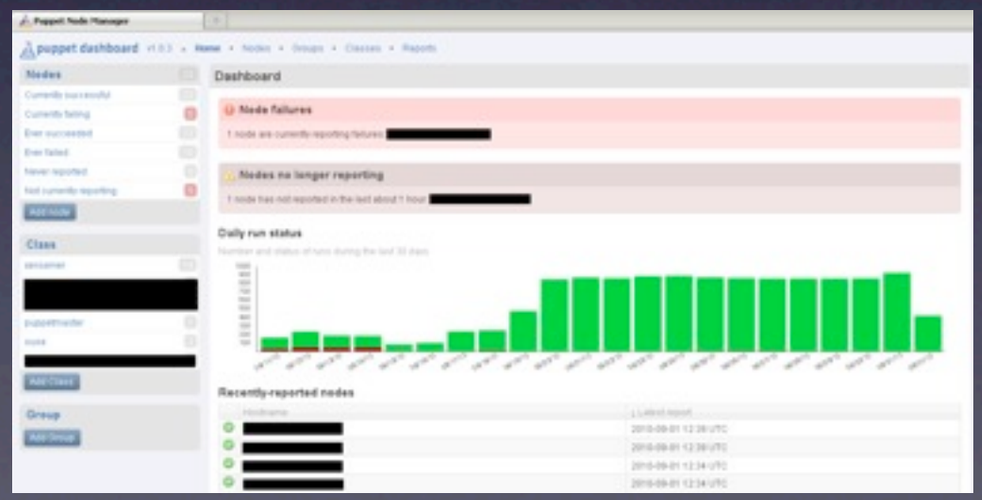
Tools

- CFEngine
- Bcfg2
- Puppet
  - seit 2005
  - Ruby Anwendung

Tools

- CFEngine
- Bcfg2
- Puppet

- seit 2005
- Ruby Anwendung
- Reporting



Tools

- CFEngine
- Bcfg2
- Puppet

- Open Source

Tools

- CFEngine
- Bcfg2
- Puppet

- Open Source
- Fixpunkt Beschreibung

Tools

- CFEngine
- Bcfg2
- Puppet

- Open Source
- Fixpunkt Beschreibung
- Client-Master Setup

Puppet

- Master
  - Manifeste
  - Dateien
  - Vorlagen

Puppet

- Master
  - Manifeste
  - Dateien
  - Vorlagen
- Client

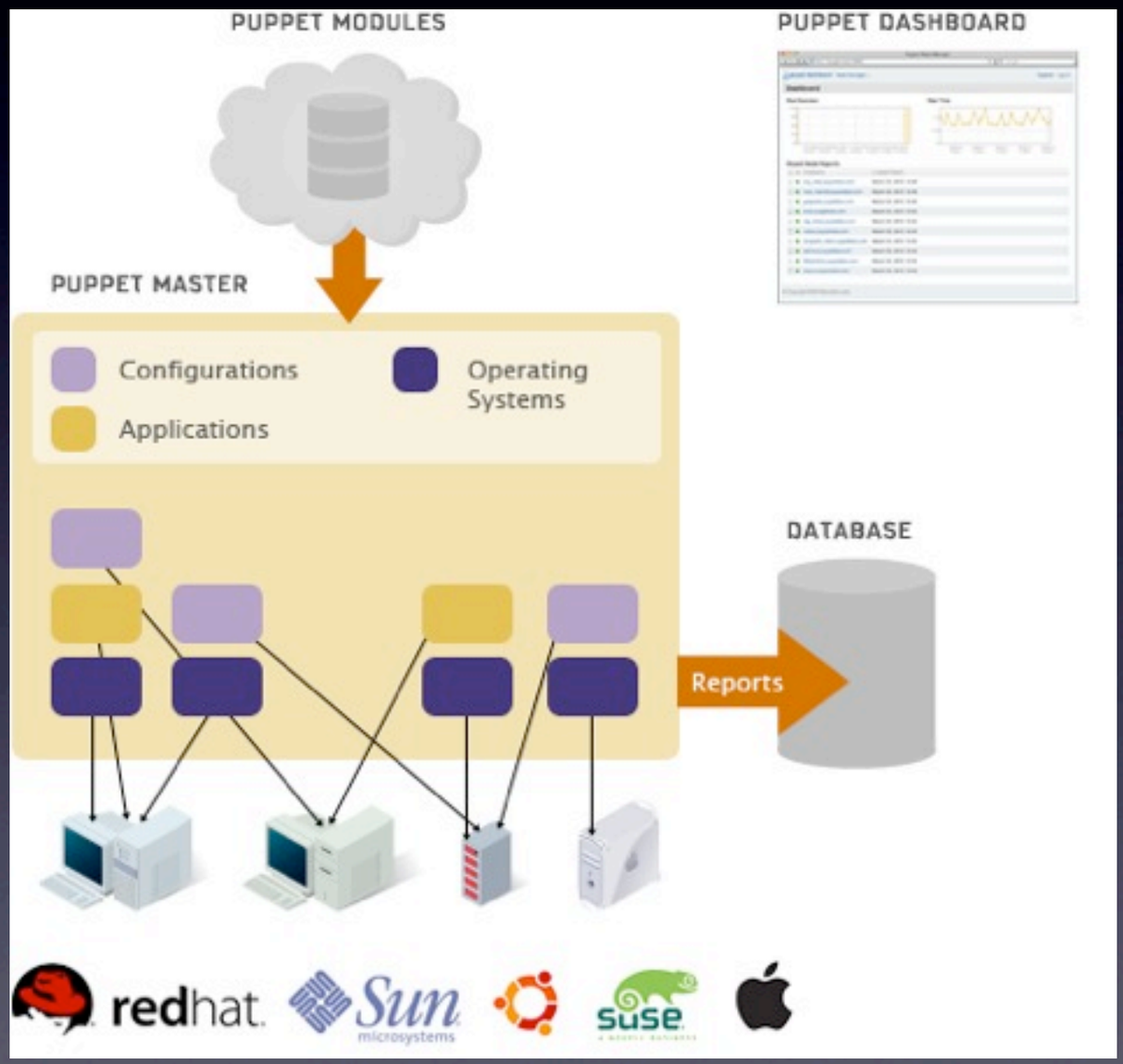
Puppet

- 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

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

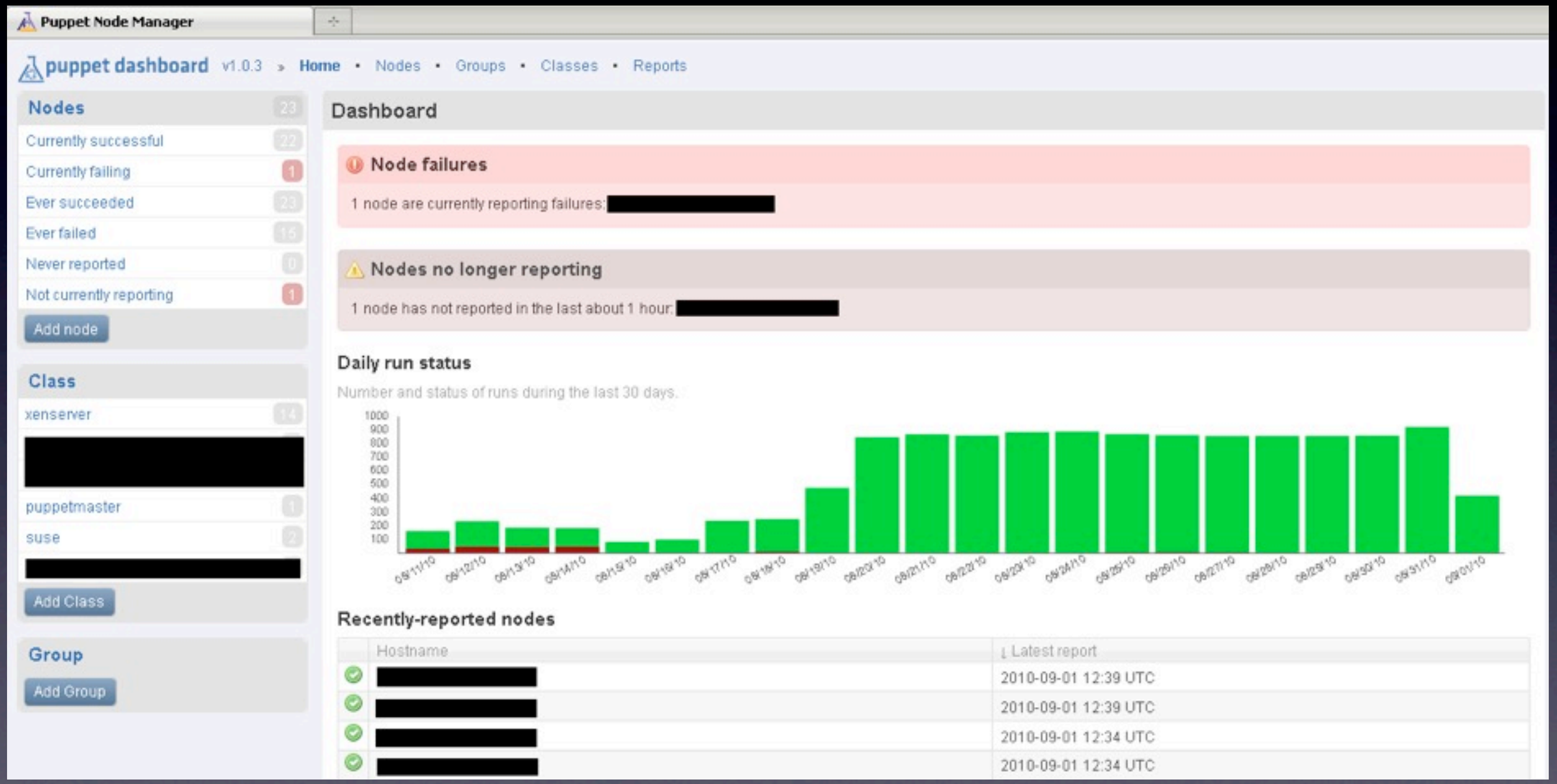
Puppet Reporting

- 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



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 @ Continental

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

Puppet @ Continental

- Definierte Aufgaben von Puppet

Einführung - Tools - Puppet - **Continental**

Puppet @ Continental

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

Puppet @ Continental

- Puppet Setup

Einführung - Tools - Puppet - **Continental**

Puppet @ Continental

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

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 @ Continental

- Puppet Master Setup

Einführung - Tools - Puppet - **Continental**

Puppet @ Continental

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

Puppet @ Continental

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

Puppet @ Continental

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

Puppet @ Continental

- 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 @ Continental

- Puppet Master Setup
  - puppet_function fact

    PUPPET_FUNCTION=HPC-INFINIBAND

Einführung - Tools - Puppet - **Continental**

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

Puppet @ Continental

- 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

Zentrales Konfigurationsmanagement mit Puppet

Demo

Martin Alfke
<martin.alfke@wizards-of-foss.de>

Zentrales Konfigurationsmanagement mit Puppet

Fragen??

Martin Alfke
<martin.alfke@wizards-of-foss.de>