Automatisierte Serververwaltung mit Hilfe von Puppet

Linuxinformationstage Oldenburg

Rene Laakmann <laakmann@bytemine.net>

Fahrplan

- Vorstellung bytemine
- Configuration Management
- Nodes Definition
- Rezepte / Templates / facter
- hiera
- PuppetDB / Puppet-Dashboard
- Schlusswort

bytemine GmbH

- Entwicklungsmanufaktur fuer innovative Loesungen
- Unix- / Linux-Dienstleister
- Sichere, verfuegbare Infrastrukturen
- OpenSource Loesungen
- 2nd und 3rd Level Support
- Housing & Hosting
- Sicherheit und Kryptographie

Eigene Produkte

- boa – bytemine OpenBSD appliance
- cryptorage – Plattform fuer sicheren Datenaustausch
- bytemine-manager – Tool zur Verwaltung von Benutzern fuer OpenVPN

Configuration management

- Verschiedene Tools (puppet, chef, cfengine)
- Warum ueberhaupt configuration management ?
- Vor- und Nachteile
- Viele Erweiterungen fuer puppet
- Client → Server Modell

Configuration Management

- Betriebssystemunabhaengig
- Debian, Ubuntu, CentOS, OpenBSD, ...
- Fuer einige Distributionen gibt es fertige Pakete bei Puppetlabs
- OpenBSD ist ueber Repos und Ports verfuegbar
- Auch fuer Windows ist Puppet verfuegbar
- Puppet bringt ruby als Abhaengigkeit mit

Configuration Management

- Client → Server Modell
- Authentifizierung mittels Zertifikaten
- Puppet nutzt eine eigene DSL (Domain Specific Language)
- Status-Reports per http, mail, log
- Rezepte
- Templates in ruby Syntax
- Facter fuer Systeminformationen

Nodes / Rezepte / Module

- Server/Maschinen/Hosts werden in Nodes definiert
- Nodes enthalten die Definitionen von Resourcen
- Resourcen beschreiben den Zustand eines Systems, den Puppet betriebssystemspezifisch umsetzt
- Es werden keine Aktionen beschrieben

Nodes / Resourcen / Module

- Resourcen koennen in Modulen organisiert werden
- Module verringern Redunanzen
- In den Nodes werden alle benoetigten Module eingebunden
- Aenderung am Modul hat Auswirkung auf alle Nodes, die dieses Modul eingebunden haben

Beispiel fuer Module

```
node  'test.server.local' {
   include ntp
}
```

```
node  'test.server.local' {
  if $::operatingsystem == 'centos'  {
   $service_name = 'ntpd' }
  else { $service_name = 'ntp' }

 package { $package_name:  ensure => installed; }
  service { $service_name:
    ensure     => running,
    enable     => true,
    hasrestart => true,
    hasstatus  => true,
    pattern    => 'ntpd',
    require    => File['/etc/ntp.conf'];
  }
  file { $file_name:
    ensure  => present,
    owner   => 'root',
    group   => 'root',
    mode    => '0644',
    content => template("ntp/ntp}.conf.erb"),
    notify  => Service[$service_name],
    require => Package['ntp'];
  }
}
```


Rezepte / Templates / facter

- Class und define
- Parameter mit und ohne Defaultwerten
- Vererbung
- If / case
- Nutzung von Templates
- Nutzung von statischen Dateien
- Include
- Abhaengigkeiten

Rezepte / Templates / facter

- Rezepte werden nicht sequentiell durchlaufen
- Benoetigte Reihenfolge muss durch Abhaengigkeiten definiert werden
- Bei Aenderung von Konfigurationen koennen Dienste z.B. auch neugestartet werden
- Eigene Aktionen koennen ebenfalls definiert werden ueber exec-Anweisungen

Rezepte / Templates / facter

- Facter enthalten Systeminformationen
- Facter-Informationen koennen in Rezepten genutzt werden
- Informationen z.B. ueber Betriebssystem, Netzwerkinformation, Kernelversion, Uptime, Swap, ...
- Koennen auch eigene facter Informationen definiert werden

Rezepte / Templates / facter

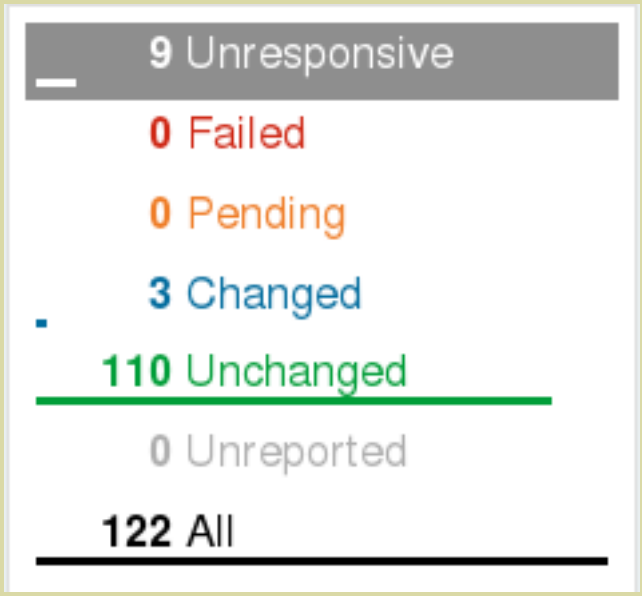
```
class ntp {
  if $::operatingsystem == 'centos' { $service_name = 'ntpd' }
  else { $service_name = 'ntp' }

  package { $package_name:  ensure => installed; }
  service { $service_name:
    ensure     => running,
    enable     => true,
    hasrestart => true,
    hasstatus  => true,
    pattern    => 'ntpd',
    require    => File['/etc/ntp.conf'];
  }

  file { $file_name:
    ensure  => present,
    owner   => 'root',
    group   => 'root',
    mode    => '0644',
    content => template("ntp/ntp}.conf.erb"),
    notify  => Service[$service_name],
    require => Package['ntp'];
  }
}
```


hiera

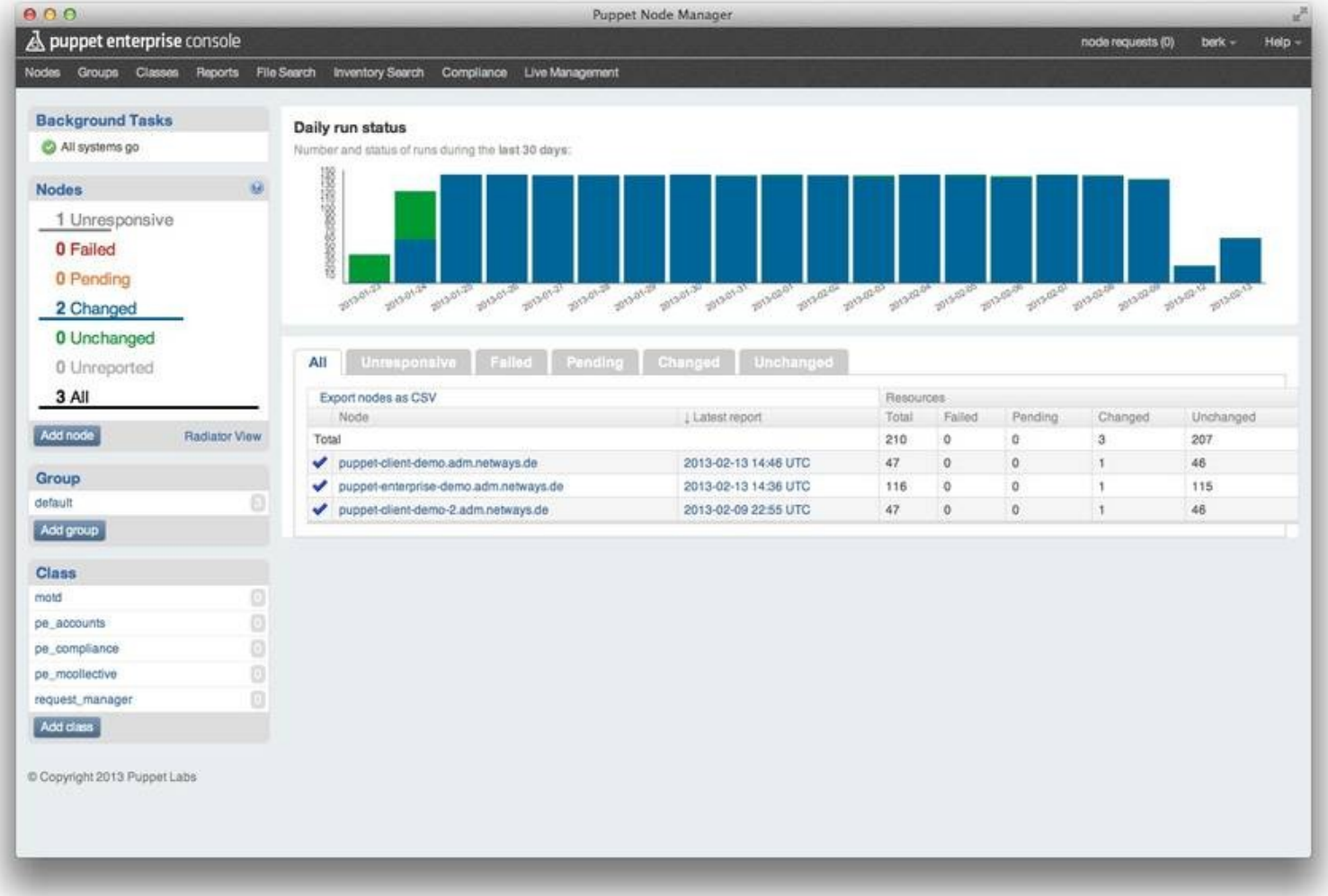
- Key / value lookup tool
- Zur Vermeidung von Wiederholungen
- Defaults fuer Nodes koennen definiert werden
- Konfiguration in Yaml Dateien
- Puppet Rezept kann direkt zugreifen

Puppet-Dashboard

- Visualisierung mittels Puppet-Dashboard
- Leider keine Weiter-entwicklung
- Alternativen sind noch nicht so weit
- Reports koennen einge-sehen werden
- Uebersicht ueber den Status



Puppet-Dashboard



PuppetDB

- PuppetDB sammelt Daten ueber Maschinen, Resourcen und Reports
- Java-Applikation
- Kann auch fuer andere Tools nuetzlich sein (z. B. Inventarisierung)

Best practices

- Getrennte Umgebungen fuer Entwicklung und Produktivbetrieb
- Unterstuetzung mit git
- Commits per Mail
- Anwendung von puppet-lint ueber git commit hooks
- Puppet-Reports im Fehlerfall per Mail

Vielen Dank für die Aufmerksamkeit!

bytemine GmbH

Marie-Curie-Str. 1
26129 Oldenburg

info@bytemine.net
http://www.bytemine.net
+49-441-3091970