Puppet 4 Solaris

Thomas Rübensaal
T-Systems International GmbH
Bamberg

Schlüsselworte
Solaris 11 Deployment, Upgrade Puppet Version 3.x nach 4.x, Installation, Konfiguration, Konfigurationsmanagement

Einleitung
T-Systems nutzt für die Installation und Konfiguration von Solaris 11 Systemen Puppet. Mit den Hersteller-Upgrades der Puppet Version von 3.x nach 4.x ergeben sich zukünftige Herausforderungen, auf die näher eingegangen werden soll.

Puppet – Was ist das?

- Puppet ist ein flexibles, anpassbares Framework, um viele sich wiederholende Tasks, die von einem System Administrator täglich manuell ausgeführt werden müssten, zu automatisieren.
- Es basiert auf Programmiersprache Ruby und ist Open Source.
- Als eine deklarative Modell basierende Vorgehensweise für IT Automation wird ein gewünschter Status definiert.
- Wenn der gewünschte Status einmal definiert ist, installiert Puppet automatisch alle notwendigen Pakete und startet alle zugehörigen Services. Weiterhin wird bei jedem neuen Puppet Run sichergestellt, dass der definierte Zustand vorhanden ist.
- Puppet ist für die meisten UNIX ähnlichen Betriebssysteme (AIX, HP-UX, RedHat, Suse, Solaris, …) und für Microsoft Windows verfügbar.
- Puppet kann sicher und beliebig oft ausgeführt werden, wobei Puppet nur dann Änderungen auf dem Client durchführt, wenn der aktuelle System Status vom definierten System Status abweicht.
- Puppet unterstützt LDAP. Alle Konfigurationsdaten können in einer LDAP Datenbank gespeichert werden. Nur der Puppet Master Server benötigt einen Zugriff auf den LDAP. Von daher benötigen die Puppet Clients keinen zwingenden LDAP Zugriff.

Puppet Historie bei T-Systems

- Erste Tests erfolgten mit Puppet 0.x bzw. 2.x:
  - Test Clients Solaris 10
  - 10 x Puppet Master mit Mongrel Webserver auf Port 1814x (18140, 18141, 18142,…18149)
  - Apache als Load Balancer auf Port 8140
- Mit Start von Solaris 11 wurde Puppet 3.x produktiv eingeführt
  - Apache mit Passenger auf Port 8140
  - Passenger startet und stoppt bei Bedarf (dynamisch) Puppet Master (0 bis beliebig viele Puppet Master können laufen)

T-Systems Solaris 11 Architektur Überblick

- Es existiert eine LDAP Server Infrastruktur mit einem zentralen LDAP Master und lokalen LDAP Master an jeder Lokation (inkl. Master – Master Replikation). Des Weiteren sind LDAP Slave Server für lokale Redundanzen vorhanden. Die LDAP Server enthalten die Solaris Konfigurationsdaten der Solaris Clients.
- Ebenso gibt es einen zentralen Automated Installer, zentrale Solaris 11.x Repositories und einen zentralen Puppet Master in Version 3.6.2.
- Die remote Lokationen sind mit je einem Apache Proxy Cache (für WAN Boot / AI) und Netcat (für Puppet) zur zentralen Lokation angebunden. (Proxy Layer)
- Solaris 11 Clients (an jeder Lokation) verwenden den lokalen Proxy Layer (Client Layer)
- Für den PXE-Boot von x86 Clients werden bestehende lokale Solaris 10 Installserver (DHCP) mit genutzt.
- Für die LDAP Funktionalität der Solaris Clients werden die LDAP Server im jeweiligen RZ genutzt.
- Das Dynamic Toolset (ein eigenentwickeltes zentrales Konfigurationstool mit Weboberfläche) dient zur Erfassung und Verwaltung der Client-Daten im LDAP.

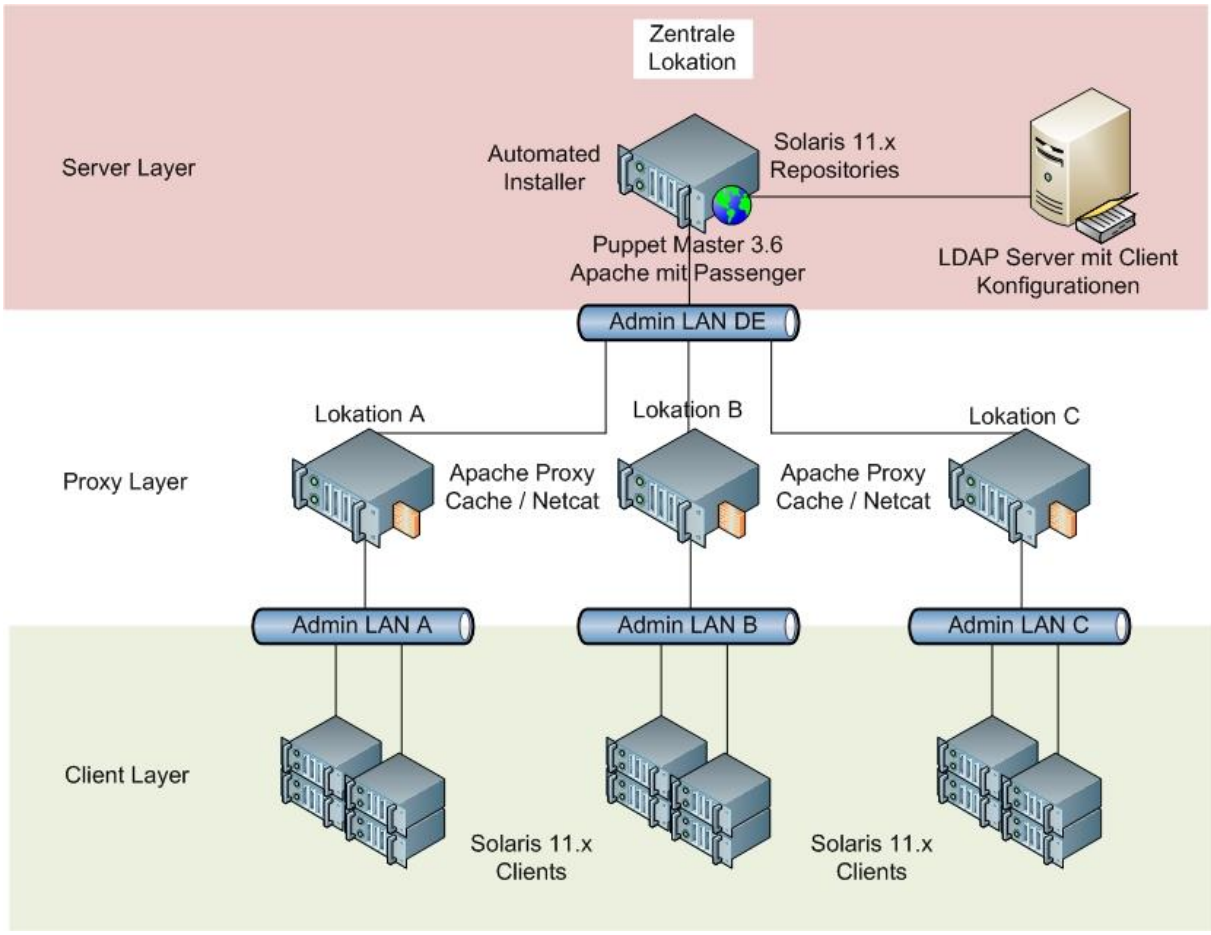

Abb. 1: T-Systems Solaris 11 Infrastruktur

Details zu Puppet 4
Puppet 4 besteht im Gegensatz zu Puppet 3 aus einem All-in-One (AIO) Paket, welches allerdings nur für Linux verfügbar ist. Für Puppet 3und Solaris 11 hatten wir auch ein Solaris IPS Paket erstellt, welches Puppet, Passenger, Ruby, Ruby-LDAP, Ruby-Shadow und Ruby Rack enthält.
Die bei Oracle Solaris 11.3 und Oracle Solaris 12 verfügbaren Puppet Versionen sind leider noch Version 3.x (Solaris 11.3: Puppet 3.6.2; Solaris 12: Puppet 3.8), so dass Original Oracle Solaris Sourcen für Puppet 4 nicht genutzt werden können.
Der Puppet Server ist ein Java Programm mit JRuby, welcher Port 8140 nutzt.
Passenger entfällt mit Puppet 4, da der Puppet Server dynamisch notwendige Puppet Server Prozesse startet und auch wieder stoppt.
Weitere Besonderheit ist ein komplett neugeschriebener Parser und Auswerter.

Abb. 2: Puppet Logo

Erfahrungen mit Puppet 4

Test: Update von Puppet 3 auf Puppet 4 unter Beibehaltung des Passengers:
Testszenario:

- 1 x Puppet Master 3.8.x
- 2 x Puppet Agent 3.8.x
- Update eines Agenten auf Version 4.x
  ➔ Kann nicht mit dem Puppet Master in Version 3 kommunizieren (erwartetes Ergebnis)
- Update des Puppet Master auf Version 4.x
  ➔ Der Puppet Agent in Version 4 kann mit diesem Puppet Master in Version 4 kommunizieren (erwartetes Ergebnis)

➔ Der Puppet Agent in Version 3 kann nicht mit dem Puppet Master in Version 4 kommunizieren

Die Programmierung der aktuellen Module ist nicht vollständig kompatibel zu Version 4 (Stichwort „Future Parser“). Allerdings funktioniert die LDAP Anbindung auch reibungslos mit Puppet 4.x.
Größtes Problem ist aber, dass die Puppet Agenten in Version 3 nicht mit dem Puppet Master in Version 4 kommunizieren können. Dies verbaut den Update Pfad.

Anpassungsbedarf bei Puppet Modulen bei Migration von Puppet 3.x nach Puppet 4.x:

- Ein String ist nur noch dann ein String, wenn er als solches gekennzeichnet wird:
  Damit ist dies hier in Puppet 4 kein String:

```
case $::operatingsystem {
Solaris: { … }
```

```
default: { ... }
}
```

und musste ersetzt werden durch

```
case $::operatingsystem {
"Solaris": { … }
default: { ... }
}
```

Gleiches gilt für Zahlen die als Strings verwendet werden sollen:
Beim Type „File“ ist dies hier ist somit neuerdings eine Zahl:

```
mode => 0644
```

Daher musste es ersetzt werden durch

```
mode => "0644"
```

damit es wieder ein String ist.

- Für Variablennamen sind nur noch Kleinbuchstaben und Zahlen erlaubt:
  Wir nutzen mit Puppet 3.x aktuell Großbuchstaben bei Variablennamen (z.B. LDAP Variablennamen „OSVersion“ oder „Interface“)
  Mögliche Lösungen:
  - Unsere genutzten LDAP Funktionen müssen so umgebaut werden, dass die erzeugten Variablennamen der neuen Konvention genügen (also ein „Downcase“ für die Variablennamen aus LDAP und Anpassung der betroffenen Puppet Module).
  - Anpassung der Puppet Sourcen, um Ausdruck für die gültigen Variablennamen um Großbuchstaben zu erweitern
  - Da die Anpassung der Variablennamen für uns einen massiven Aufwand bedeutet hätte, haben wir für unsere Tests die Puppet Sourcen erweitert, um Variablen mit Großbuchstaben zu unterstützen (VAR_NAME in lib/ruby/site_ruby/2.1.0/puppet/pops/patterns.rb).
- Ein Define muss nun mit kompletten Klassennamen aufgerufen werden:
  - Bisher war dies nur bei global gültigen Defines nötig
  - Jetzt gilt dies auch wenn es in derselben Klasse definiert wird
  - Dies war bisher zulässig:

    ```
    define modifyentries() {
    ...
    modifyentries{ $entrylist: }
    ```

  - Mit Puppet 4.x ist eine Anpassung erforderlich:

    ```
    define modifyentries() {
    ...
    ipnodes::solaris::modifyentries{ $entrylist: }
    ```

  - Hinweis: Da $entrytlist ein Array ist muss es nicht in Anführungszeichen gesetzt werden.
- Das Klammern von Variablen ist zu vermeiden, sofern es nicht notwendig ist:
  - Somit sind Klammern nur noch bei Hashes zu verwenden
  - Falsch: `$monitoring_client = „${::MonitoringClient}"`
  - Richtig: `$monitoring_client = „$::MonitoringClient"`

Mit diesen Änderungen lief das dann das Deployment mit Puppet Master und Agent in Version 4 reibungslos.

Um auch die Kompatibilität mit Puppet Agenten in Version 3 zu erreichen ist ein weiterer Schritt erforderlich: Der Einsatz des Puppet Servers 2.1 anstatt von Passenger. Insbesondere auch da der Passenger bzw. Rack Code als auslaufend markiert sind.

<u>Test: Installation Puppet Server 2.x:</u>

Die Installation des Puppet Server 2.x unter Solaris 11 gestaltete sich aufwendig. Wir haben zunächst darauf verzichtet unsere angepassten Pfade zu verwenden und den Puppet Server im Standard Layout eingerichtet:

- Konfiguration: /etc/puppetlabs
- Binaries: /opt/puppetlabs

Ergebnis:

- Anbindung von Puppet Agenten in Version 3.6 und neuer sowie Version 4.x funktionieren
- Anbindung von Puppet Version 3.0 funktionierte nicht - was aber auch am Ruby 1.8 gelegen haben könnte

Mit der Integration unserer üblichen Konfiguration musste dann auch die LDAP Integration eingerichtet werden.
Die erste Hürde scheint lösbar zu sein: Ersetzen des Paketes „ruby-ldap“ für das native Ruby durch „jruby-ldap“ für den JRuby Interpreter des Puppet Servers.
Aber leider ist es uns trotz diverser Versuche nicht gelungen eine erfolgreiche LDAP Authentifizierung zu erreichen.

Fazit

- Eine Umgebung Apache / Passenger mit Puppet in Version 4 wäre einsatzbereit
- Großer Nachteil ist die fehlende Kompatibilität mit Puppet Agenten in Version 3
- Der Puppet Server 2.1 und neuer löst das Problem der fehlenden Kompatibilität für Puppet 3 Agenten, dafür fehlt aber derzeit die LDAP Integration
- Die LDAP Integration ist für uns essentiell - diese muss mit dem Puppet Server möglich sein oder wir haben massiven Aufwand eine Alternative zu implementieren

Für einen schnellen Start mit Puppet 4 sind allerdings auch einige Arbeiten wegen der Inkompatibilität zwischen Puppet Master in Version 4 und Puppet Agenten in Version 3 notwendig:

- Ein Update aller Clients scheidet wohl aus. Dies müsste ja mit einem Big Bang gemacht werden. Sobald der zentrale Puppet Master auf Version 4 ist, sind alle älteren Clients tot. Oder so ein Zustand wird akzeptiert und mit dem Update des OS auf ein Puppet 4 basiertes Image geht dann die Automation wieder.
- Wir behalten den Port 8140 für Puppet 4 bei, legen aber einen neuen virtuellen Hostnamen (zum Beispiel „puppet4“) an. Allerdings müssen wir dann das Puppet Master Zertifikat abändern weil „puppet4“ derzeit kein gültiger Alias ist. Somit brauchen alle Clients irgendwann ein neues Zertifikat – entweder mit einem Big Bang oder beim Update von einem Puppet 3 auf ein Puppet 4 basierendes Image.
- Wir behalten den virtuellen Hostnamen „puppet“ bei und wählen einen anderen Port für Puppet 4. Zum Beispiel könnte man Port 8141 dann für Puppet 4 verwenden, Port 8140 bleibt für Puppet 3. Hiermit müssten allerdings auch die Default Port Freischaltungen auf den Firewalls erweitert werden.

Ausblick

Mit der zukünftigen Einführung von Solaris 12 haben wir die Nutzung von Puppet 4 vorgesehen. Allerdings müssen wir uns noch um eine funktionierende LDAP Anbindung an den Puppet Server kümmern. Dies könnte auch bedeuten, dass die LDAP Anbindung außerhalb von Puppet realisiert werden muss.

Kontaktadresse:
Thomas Rübensaal
T-Systems International GmbH
Gutenbergstraße 13
D-96050 Bamberg

Telefon: +49 (0) 951 13365427
Fax: +49 (0) 391 580228475
E-Mail: Thomas.Ruebensaal@t-systems.com
Internet: www.t-systems.com