Konfigurations-Management mit Puppet in einer privaten Oracle DB Cloud

Roman Gächter
Trivadis AG
CH-8152 Glattbrugg

Schlüsselworte
Der Autor hat ein Konfigurationsmanagement mit Puppet Enterprise in einer private Oracle DB Cloud bei einem großen deutschen Automobilhersteller aufgesetzt.
Der Fokus der Session wird sein: Vorstellen des Konzeptes, Erfahrungen mit Puppet Enetrprise

Motivation für ein Konfigurationsmanagement Tool

Wenn man eine größere Umgebung effizient managen und die Aufwände für Implementierungen minimieren will kommt man kaum um ein Konfigurationsmanagement Tool herum. Neben der Effizienz gibt es weitere Bereiche, welche von einem Konfigurationsmanagement Tool profitieren:

- Sicherheit
  - Man kann eine „security baseline“ garantieren
- Qualität
  - Man kann eine identische Konfiguration aller Systeme garantieren
  - Der Faktor „menschliche Fehler“ kann reduziert werden
  - Änderungen können nachvollzogen werden
- Verfügbarkeit
  - Wartungsfenster können kurzgehalten werden
- Standard Tool
  - Puppet Code ist einfacher zu verstehen als Scripts
- Flexibilität
  - Standardisieren von Konfigurationsänderungen
  - Updates einfach implementieren
  - Schnell Konfigurationsänderungen aktivieren
  - Reporting aller Konfigurationsänderungen
  - GUI für Konfigurationsänderungen und Stati
- Konfiguration ist identisch auf allen Systemen (DevOps Ansatz)
  - Durchgängig, Entwicklung / Engineering ➔ ➔ Test ➔ Produktion
- alle Änderungen nachvollziehbar
  - Source Code Verwaltung
  - Konfiguration aller Systeme in zentralem Repository ersichtlich
  - ungewollte Anpassungen werden zurückgesetzt

- „human error“ reduziert

Übersicht Puppet
Puppet ist ein Konfigurationsmanagement Tool, welches den Zustand von Systemen in einem zentralen Repository beschreibt. Puppet ist eine „high level“ Sprache mit vielen „built in“ Ressource Types und basiert auf Ruby. Die gängigen Betriebssysteme werden alle unterstützt, im Linux Bereich ist Puppet der Marktführer. Puppet ist als Open Source oder auch in einer Enterprise Version verfügbar. Die Firma Puppet Labs bietet Support und Schulung an für die Enterprise Version.

Die Abbildung 1 zeigt das 3 Schichten Model von Puppet.
„Configuration Language:
Auf dem Puppet Master befindet sich der Puppet Code in einem zentralen Repository, welches den Zustand der Systeme beschriebt.
„Transaction Layer“:
Die Clients fordern den Puppet Katalog vom Master an, welcher aus dem Repository kompiliert und dann auf dem Client angewendet wird. Der Client sendet danach einen Report über allfällige Änderungen. Ein „Puppet Run“ kann auch vom Puppet Master aus getriggert werden.
„Resource Abstraction Layer“:
Puppet ist ein DevOps Tool, es stehen Mechanismen zur Verfügung um den Puppet Code aus einem Souce Code Controll System auf den Puppet Master zu deployen.

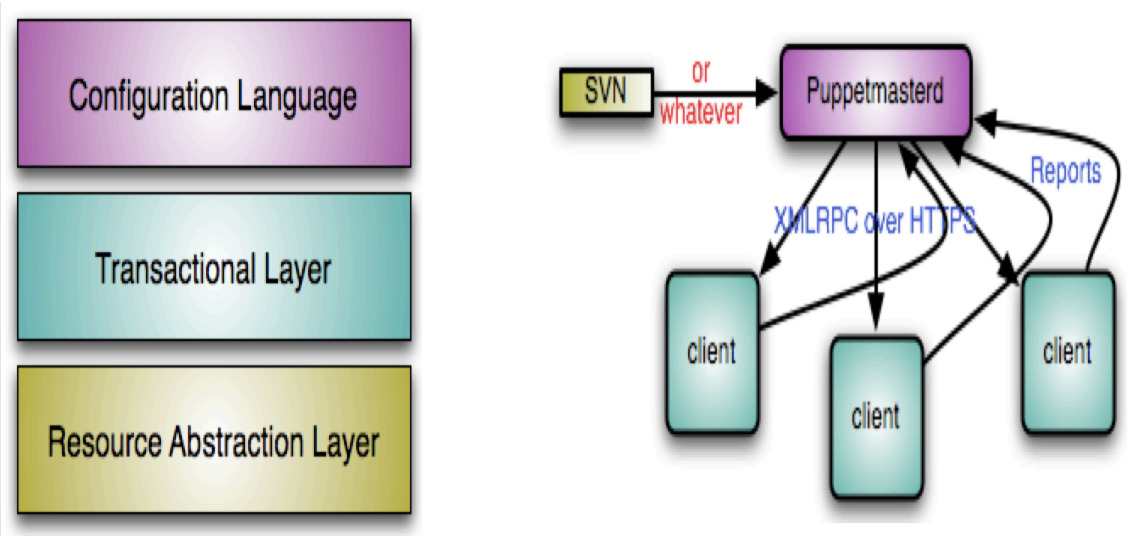

Abb. 1: 3 Schichten Model von Puppet

Puppet Konzept für die DB Cloud
Das Puppet Konzept für die DB-Cloud wurde nach gängiger „best pratice“ ausgearbeitet. Durch die Verwendung von Hiera erreicht man die strikte Trennung von Code und Daten. Wenn man den Code generisch erstellt und die Klassifizierung auslagert, bleibt man portabel. Man muss das Rad nicht immer neu erfinden – durch die Verwendung von „puppet forge“ Modulen erreicht man sein Ziel schneller. Durch eine Trennung von Geschäfts- und Technischer-Logik mit Rollen und Profilen kann man seine Konfiguration so granular wie notwendig ausprägen.
Es macht Sinn mehrere Umgebungen aufzubauen. Eine Testumgebung ist sehr zu empfehlen. In der DB-Cloud Umgebung wird ein neuer Code immer zuerst in der Testumgebung deployed. Erst wenn da alles reibungslos läuft wird er in der Source Code Verwaltung eingecheckt und kann dann 1 zu 1 auf der produktiven Umgebung deployed werden. Für die Source Code Verwaltung wird Git und Gitlab verwendet. Das Deployment von der Test- zur Prod-Umgebung erfolgt über r10k und wird von Hand angestoßen.

Das Trennen von Geschäfts- und technischer Logik wurde zuerst von Craig Dunn an einer Puppet Konferenz vorgestellt. Auf seiner Homepage beschreibt er diese Idee ausführlich: http://www.craigdunn.org/2012/05/239/
Man bildet die Geschäfts-Logik in Rollen ab und es gibt maximal nur eine pro System.
Die Profile beschreiben die technische Logik – eine Rolle kann mehrere Profile haben. Die Profile verwenden Module und Manifeste und beziehen die Daten aus Hiera. Eine Grundbedingung für das Trennen von Code und Daten sind die sogenannten „External Node Classifiers“ auch ENC genannt. Damit bestimmt man, welche Puppet „classes“ (Programme) wo ausgeführt werden unabhängig vom Code. In der DB-Cloud Umgebung lösen wir das über Hostgruppen in der PE Console und definieren eine Hostgruppe pro Node, an welche wir dann den Agenten und die entsprechende Rolle anbinden.

Die Abbildung 2 zeigt das Zusammenspiel von Rollen, Profilen, Modulen, Ressourcen, ENC und Hiera.

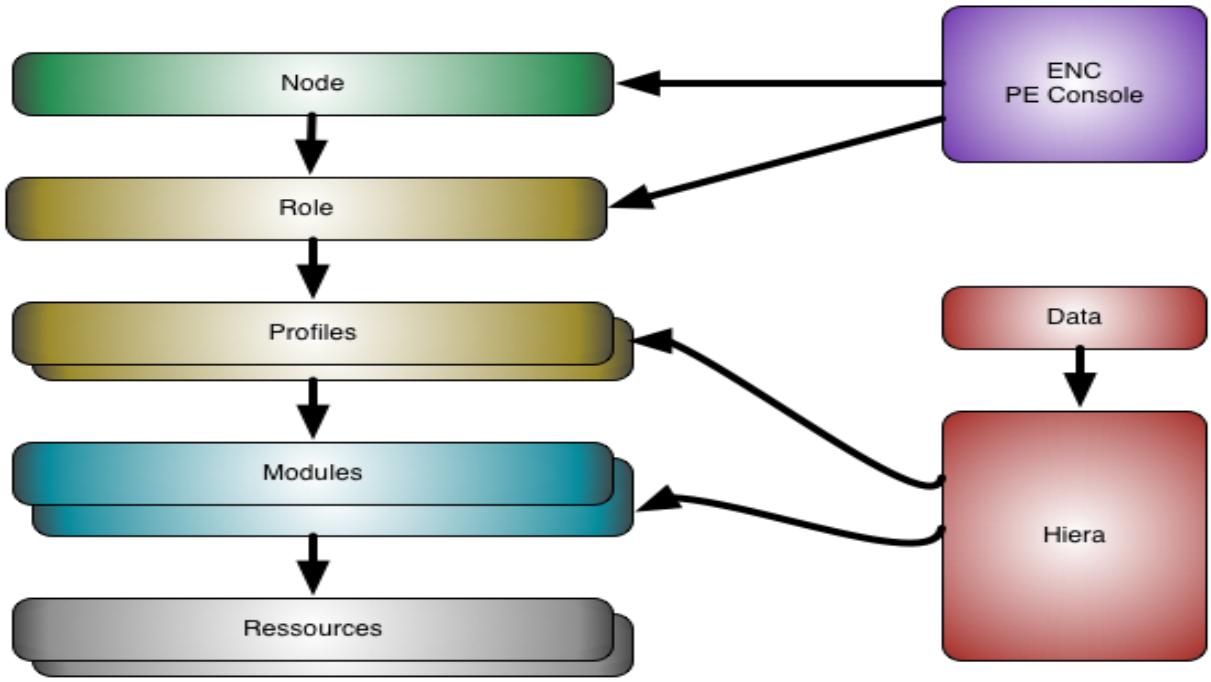

Abb. 2: Zusammenspiel von Rollen, Profilen, ENC und Hiera

Betrieb
Puppet Enterpeise Console:
Das zentrale Element für den Betrieb ist die Puppet Enterprise Console, welche die 3 Funktionen “Configuration Management”, “Verwaltung der Nodes” und “Access Control” bereitstellt. Hier erhält man eine Status-Übersicht und kann sich Events sowie Reports ansehen. Es gibt ein Inventory der Nodes mit einer CA sowie die Klassifizierungskonfiguration. Im “Acces Control” kann man die Benutzer Authentisierung und Autorisierung verwalten.

Hiera:
Der Begriff wurde wahrscheinlich vom Wort Hierarchie abgeleitet. In einer flexibel zu definierenden Hierarchie kann man gesteuert über Facts die Daten für Puppet strukturieren.
Bei Konfigurationsänderungen muss man in der Regel nur die Daten im Hiera anpassen – der Code bleibt unverändert. Zum Beispiel einen neuen Linux User definieren, clusterspezifische Host Einträge definieren, NTP Konfigurationen ändern, Anpassungen in der Rsyslog Konfiguration vornehmen und vieles mehr.

Die Tabelle 1 zeigt ein Beispiel für die Konfiguration von NTP in Hiera mit dem NTP „puppet forge“ Module von Puppet Labs.

```
ntp::restrict:
  - default ignore
  - 127.0.0.1
  - time.meinefirma.de mask 255.255.255.255 nomodify notrap noquery
ntp::driftfile: /var/lib/ntp/drift
ntp::fudge:
  - 127.127.1.0 stratum 10
ntp::autoupdate: false
ntp::enable: true
ntp::servers:
  - 127.127.1.0
  - time.meinefirma.de
```

Tabelle. 1: NTP Konfiguration in Hiera

Code Verwaltung
Für die Source Code Verwaltung wird in der DB Cloud GIT und GITLAB verwendet. Alle Module sind ebenfalls im GIT. Das Deployment von Code und Daten auf die Puppet Master Server erfolgt über r10k.

Die Abbildung 3 zeigt die Übersicht der Code Verwaltung.

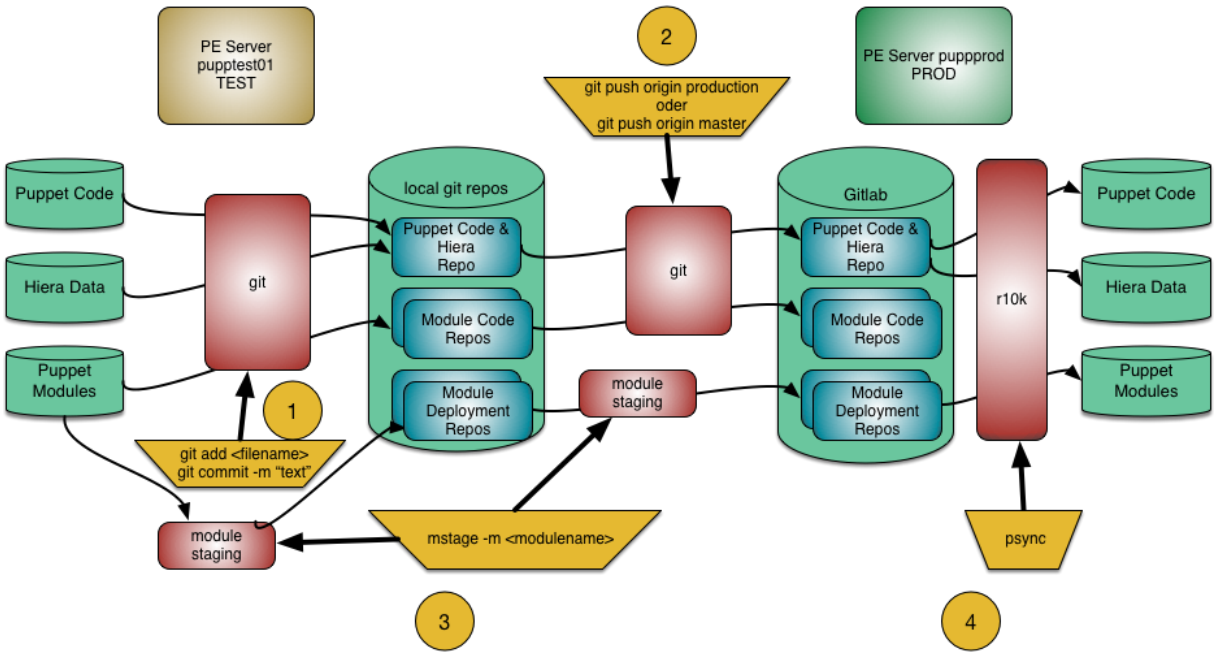

Abb. 3: Übersicht der Codeverwaltung

Deployment

Puppet Agents werden initial von Kickstart installiert und konfiguriert. Beim Start erfolgt ein automatischer „Certificate Signing Request“ vom Agent. In der Puppet Konsole muss der Request manuell bestätigt werden. Danach ist der neue Agent bereits in die Puppet Infrastruktur eingebunden.

Kontaktadresse:

Roman Gächter
Trivadis AG
Europastrasse 5 Tel: +41-44-8087020
CH-8152 Glattbrugg Fax: +41-44-8087021