Automatisiertes Konfigurationsmanagement mit Puppet

Daniel Schulz
Opitz Consulting
Gummersbach

Schlüsselworte:

Puppet, Konfigurationsmanagement, Cross-Plattform, Compliance, Server Life-Cycle, Change Management

Einleitung

Das von der Firma PuppetLabs (www.puppetlabs.com) unter der Apache 2.0 Open Source Lizenz veröffentlichte Konfigurationsmanagement-Framework Puppet ist darauf ausgerichtet, die Verwaltung komplexer, sich verändernder heterogener Systemlandschaften zu automatisieren und vereinfachen. Änderungen sind dabei besser nachvollziehbar und dokumentiert als bei einer manuellen Verwaltung. Puppet ermöglicht dadurch eine bessere Einhaltung von Firmenstandards, Sicherheitsrichtlinien und Compliance Vorgaben.
Ist die Systemlandschaft noch klein und übersichtlich, können viele Dinge noch manuell oder mit selbst erstellten Shell-Skripten wie „SSH-Aufruf in einer for Schleife“ erledigt werden. Werden die Umgebung und die Aufgaben jedoch größer und komplexer und Konfigurationsänderungen häufiger, dann stößt dieser Ansatz schnell an Grenzen. Ein spezialisiertes Werkzeug zur Konfigurationsverwaltung wie Puppet kann eine wertvolle Hilfe sein, wenn sich die eigene Skriptsammlung mittelfristig als nur schwer zu warten erweist und das Rad immer wieder neu erfunden werden muss.
Neben Puppet gibt es noch weitere Tools, die eine ähnliche Zielsetzung haben: Chef, Cfengine und Bcfg2. Ein Vergleich dieser Tools geht jedoch über den Rahmen des Vortrages hinaus.

Architektur und unterstützte Plattformen

Puppet besteht aus mehreren Komponenten und einer eigenen Sprache, die es Systemadministratoren ermöglicht eine gewünschte Konfiguration eines bis vieler Systeme in einem „Manifest“ zu beschreiben. Im Manifest werden „Ressourcen“ beschrieben, dabei kann eine Ressource im Puppet Kontext vieles sein: Ein Useraccount, eine Datei, ein Dienst, ein Softwarepaket und vieles mehr. Eine Ressource hat einen Namen und diverse Attribute mit entsprechenden Werten. Jeder Useraccount hat einen Namen, eine UID und eine Gruppe, jede Datei hat einen Namen, Eigentümer, Berechtigungen und einen Pfad. Gleichartige Ressourcen können nach Typen gruppiert werden. Manifeste können als Templates erstellt werden, die konkreten Werte werden beim Ausrollen der Konfiguration nach zuvor festgelegten Regeln gesetzt.
Damit Puppet lauffähig ist benötigt es eine Ruby Umgebung und das ebenfalls von PuppetLabs entwickelte „Facter“ Tool. Facter ermittelt „Facts“ wie Servername, IP Adresse oder Plattform, die dann von Puppet als Variablen referenziert verwendet werden können. Unterstützt werden diverse Linux Distributionen, Solaris, AIX, HP-UX, Open und NetBSD. Eine Version für Windows ist in einem frühen Stadium.
In einer typischen Puppet Umgebung gibt es einen zentralen Server, den Puppetmaster, und diverse Clients, die vom Master ihre Anweisungen erhalten. Die Kommunikation zwischen Puppetmaster und Client ist dabei per SSL abgesichert. Ein Puppet-Client kann eine Workstation sein ebenso wie ein Clusternode in einem Verbund von Servern. Der Master gibt die Zielkonfiguration vor und von dort aus werden Änderungen umgesetzt. Auf einem einzelnen Rechner kann Puppet in der Kommandozeile

in einem „serverless-mode“ ausgeführt werden, doch ist diese Variante eher bei den ersten Schritten mit Puppet interessant, um sich mit den Konzepten vertraut zu machen. Seine Stärken spielt Puppet dann aus, wenn die Anzahl der Server größer wird und die Konfigurationseinstellungen einem kontinuierlichen Veränderungsprozess unterliegen.
Je nach Bedarf kann die Konfiguration durch expliziten Aufruf von Puppet oder automatisch an die Puppet Clients ausgerollt werden. Puppet erkennt, ob sich etwas geändert hat und wird auch nur in diesem Fall die betroffenen Ressourcen verändern. Ist bereits der Zielzustand vorhanden, so unterbleiben jegliche Änderungen. Die Konfiguration kann also gefahrlos beliebig oft ausgerollt werden. Vor einer tatsächlichen Änderung kann auch in einem „dry-run“ oder „no-op mode“ geprüft werden, welche Änderungen offen sind ohne diese jedoch umzusetzen. Dies kann natürlich auch für Reportingzwecke angewendet werden, beispielsweise um sicherzustellen, dass es keine lokale Manipulation an der Konfiguration gegeben hat.

Warum Puppet?

In vielen Umgebungen ist es nicht ungewöhnlich, dass es Gruppen von Servern gibt, also Maschinen oder VMs, die nahezu identisch aufgesetzt sind. Ein neuer Web- oder Datenbankserver ist weitgehend identisch zu bestehenden Systemen mit der gleichen Funktionalität, unterscheidet sich allerdings in einigen wenigen, aber wesentlichen Details. Auch bei einer Aufteilung nach Entwicklungs-, Test- und Produktionsservern sind weitgehend identische Maschinen vorzufinden. Bei einer auch noch so kleinen Konfigurationsänderung sind immer wieder identische Schritte notwendig auf allen betroffenen Umgebungen – eine mitunter zeitaufwändige und fehlerträchtige Aufgabe und somit eine Gefahr für den sicheren Betrieb. Mit Puppet kann ein für alle betroffenen Maschinen gültiges Template verwendet werden. Die individuellen Details werden außerhalb des Templates verwaltet. Dadurch können die Templates ohne Anpassung von Entwicklungsservern und produktiven Systemen verwendet werden. Es entfällt die oftmals fehlerträchtige Anpassung der Konfigurationsdateien, wenn eine neue Version aus der Entwicklung an Test und Produktion geliefert wird. Dies erleichtert natürlich auch den Umzug von einer Anwendung auf einen neuen Server.
Puppet abstrahiert die spezifischen Eigenschaften der Zielplattform, Systemadministratoren beschreiben deklarativ, „was“ zu tun ist: Einen Benutzeraccount anlegen, einen Dienst starten, ein Softwarepaket installieren, eine Datei mit einem bestimmten Inhalt anlegen, ssh-keys verteilen und vieles mehr. Das „wie“ übernimmt Puppet, indem es über sogenannte Provider die für die jeweilige Plattform notwendigen Schritte ausführen lässt. Ein Mindestmaß an Standardisierung was Serverbenennung, Verzeichnisstrukturen, Betriebssystem und dergleichen angeht erleichtert die Arbeit mit Puppet jedoch deutlich. Sind die Server zu verschieden, dann wird die Erstellung von Manifesten sehr zeitaufwändig.
Anfänglich macht Puppet die Arbeit eher etwas aufwändiger und es erfordert etwas Disziplin, nicht ‚mal eben‘ schnell außerhalb von Puppet eine lokale Änderung vorzunehmen. Mittelfristig macht sich die Umgewöhnung bezahlt, was durchaus wörtlich genommen werden kann, denn der administrative Aufwand pro administrierter Maschine sinkt gegenüber der manuellen Verwaltung. Darüber hinaus hilft Puppet auch bei der Minimierung von Fehlkonfigurationen und Ausfällen.

Zusammenspiel mit einer Versionsverwaltung

Puppet kann und sollte zusammen mit einem Versionierungssystem wie Subversion oder Git verwendet werden. Diese Systeme sind fester Bestandteil in der Softwareentwicklung und viele der Vorteile einer zentralen Versionsverwaltung lassen sich auch auf die Systemadministration übertragen. Nur eine Puppet Konfiguration, die auf dem aktuellen Versionsstand ist, also mit dem Repository synchron ist, sollte ausgerollt werden. Durch die Versionierung sind Änderungen leichter

nachvollziehbar und mit wichtigen Metadaten dokumentiert: Wer hat die Änderung vorgenommen? Wann wurde die Änderung gemacht? Warum wurde etwas geändert? Diese Informationen sind bei der Fehlersuche sehr hilfreich. Neue Konfigurationseinstellungen können zunächst in einem Entwicklungszweig (sogenannter „Branch“ in Subversion) unsichtbar für die Produktion eingepflegt werden und nach erfolgreichem Test in die Hauptlinie („Trunk“), die Produktionskonfiguration übernommen werden. Über entsprechende „Tags“ können wichtige Meilensteine der Konfiguration markiert werden, wie eine neue Version der eigenen Anwendung. Im Notfall erleichtert dies das Zurückgehen auf die letzte funktionierende Konfiguration.

Kontaktadresse:

Daniel Schulz
Opitz Consulting Gummersbach GmbH
Kirchstraße 6
D-51647 Gummersbach

Telefon: +49 (0) 2261 6001 0
Fax: +49 (0) 2261 6001 4200
E-Mail daniel.schulz@opitz-consulting.com
Internet: www.opitz-consulting.com