Puppet Book

Дмитрий Ильин

15 октября 2012 г.

Оглавление

Глава 1

Введение

1.1 Системы автоматизации управления серверами

Работа системного администратора в большинстве случаев связана с многократным выполнением однотипных и повторяющихся задач, например установка программ, создание пользователей, управление демонами и службами. Выполнение и повторное выполнение этих задач вручную очень неэффективно, занимает много времени и часто приводит к ошибкам, особенно при увеличении количества управляемых систем и усложнении задач.

В большинстве случаев системные администраторы начинают создавать свои инструменты, которые могли бы помочь им автоматизировать такие рутинные задачи. Чаще всего это бывают скрипты на Shell, Python, Perl или других языках программирования, которые, хотя и могут быть очень полезными, подходят только для применения их автором и только в его окружении, потому что практически никогда не документируются и не публикуются. Повторное использование таких инструментов тоже крайне затруднено, и при изменении окружения или после прихода новых людей их обычно просто бросают, потому что либо не могут адаптировать их под новые задачи, либо вообще понять как они работают. Но, поскольку появляются те же проблемы, что и раньше, создание таких инструментов начинается снова и снова.

Одним из путей решения проблем автоматизации и стандартизации начальных настроек систем можно назвать различные системы развертывания, такие как Red Hat Kickstart, Solaris Jumpstart, Debian Preseed/FAI, Windows Deployment Services и другие. Обычно они представляют из себя сетевой сервис, позволяющий быстро установить большое количество серверов и рабочих станций с заданными начальными настройками и сразу, без дополнительной работы, ввести их в эксплуатацию. Но после начала использования установленные системы остаются без обслуживания, и, в случае необходимости внесения изменений в конфигурацию уже установленных систем, приходится делать это либо вручную, либо путём переустановки с применением новых настроек, либо при помощи другой системы управления конфигурацией.

Многие крупные компании используют такие системы управления конфигурацией как Microsoft System Center или IBM Tivoli, но их использование влечёт за собой появление ряда новых проблем:

Цена Хотя для крупных компаний это может быть и незначительный фактор, но при увеличении серверного парка расходы могут всё более и более увеличиваться.

Гибкость Коммерческие системы обычно очень хорошо реализуют требуемый набор функционала, но, если требуется решение нестандартной задачи, могут не давать возможностей для этого.

Привязка к поставщику Использование инструментов одного поставщика вынуждает работать только с его решениями, которые могут не взаимодействовать с компонентами другого поставщика или собственными разработками.

Системы управления конфигурацией с открытым исходным кодом зачастую могут решать те же самые задачи, но имеют следующие преимущества:

Бесплатность Инструменты с открытым исходным кодом обычно можно использовать бесплатно и без ограничений, хотя часто могут быть доступны услуги коммерческой поддержки.

Открытость Человека, знакомый с языком программирования, на котором написана данная система, на должном уровне может изучить принципы работы такой системы и участвовать в её разработке.

Расширяемость Большинство подобных систем позволяют описывать решаемые задачи при помощи специального языка и создавать расширения и плагины для более нестандартных задач или задач, решение которых не входит в стандартный функционал системы.

За все время развития IT индустрии было разработано довольно много систем управления конфигурацией, как открытых, так и коммерческих и поддерживающих различные операционные системы и архитектуры. Некоторые более специализированны для решения определённой задачи, например управление кластером, а некоторые более универсальны.

К наиболее известным универсальным системам управления конфигурацией относятся:

1. Cfengine (http://cfengine.com) — старейший проект в этой области, существующий с 1993 года и написанный на C. Система использует клиент-серверную модель, основана на научных теориях и популярна в академической среде и образовательных учреждениях, хотя последняя версия уже имеет весь необходимый функционал для работы в корпоративной среде и коммерческую поддержку, она не получила популярности.

2. LCFG (http://www.lcfg.org/) — тоже старый проект разработанный в Эдинбургском университете и использующий XML для описания конфигурации систем.

3. Bcfg2 (http://trac.mcs.anl.gov/projects/bcfg2) — система управления конфигурацией созданная Аргоннской Национальной лабораторией на языке Python. Использует клиент-серверную архитектуру.

4. Puppet (http://puppetlabs.com) — самая популярная система управления конфигурацией как Unix, так и Windows систем, разрабатываемая с 2005 года Puppet Labs на языке программирования Ruby. Сейчас она используется или поддерживается такими крупными корпорациями как Google, Twitter, eBay, Disney, Citrix, Oracle, VMware и Cisco. Puppet Labs предоставляет как корпоративную версию с коммерческой поддержкой, так и открытую версию для сообщества с открытым исходным кодом под лицензией Apache 2.0. К основным особенностям Puppet можно отнести использование специального проблемно-ориентированного языка для описания конфигурации систем, код на котором может быть использован на всех платформах, декларативный принцип описания конфигурации, возможность работы как в клиент-серверном режиме, так и локально. Кроме специального языка, хорошо подходящего для решения типичных проблем, Puppet может быть легко расширен при помощи плагинов.

5. Chef (http://www.opscode.com/chef/) — самая молодая из систем управления конфигурацией промышленного уровня. Chef разрабатывается с 2009 года компанией Opscode на языке Ruby. Идеи, положенные в основу этой системы, во многом схожи с Puppet и Chef не уступает ей по функционалу и тоже имеет как версию для сообщества под лицензией Apache 2.0, так и коммерческую поддержку. Из основных отличий от Puppet можно назвать более близкий к чистому Ruby и более свободный проблемно-ориентированный язык, который, хоть и не ограничивает разработчика в нестандартных ситуациях, но и не настолько удобен для решения типичных задач, и императивный стиль описания конфигурации, а не декларативный, как у Puppet.

Хотя все эти системы и работают по схожим принципам, позволяя как распространять файлы по управляемым системам, так и устанавливать программы и выполнять другие задачи, только Puppet и Chef позволяют системным администраторам описывать конфигурацию систем при помощи специального языка, который одинаково работает на всех платформах. Управление разными операционными операционными системами отличается только именами устанавливаемых пакетов и расположением конфигурационных файлов, которые могут быть выбраны автоматически по заданным правилам. Обе эти системы обладают отличной расширяемостью, как при помощи плагинов и дополнительных источников информации о целевой системе, так и путём использования шаблонов, которые позволяют адаптировать конфигурационные файлы и распространяемые скрипты в соответствии с требованиями операционной системы или задачи сервера.

Мы выбираем Puppet а не Chef по следующим причинам:

- Puppet более зрелый проект
- Puppet поддерживается и используется большим количеством крупных корпораций
- Puppet имеет большее сообщество разработчиков и пользователей
- Puppet лучше документирован
- Декларативный язык Puppet позволяет как проще решать типичные задачи, так и нетипичные, при помощи расширений.

1.2 Архитектура Puppet

Чаще всего развернутая система Puppet представляет из себя один управляющий сервер puppetmaster (кукловод) и много управляемых систем, которые называют puppets или nodes (куклы или узлы). Конфигурация хранится на управляющем сервере в виде текстовых файлов, связанны, которые называются manifests или recipes (манифесты или рецепты) и которые могут быть организованы при помощи специальной структуры каталогов в модули — группы, собранные по какому-либо признаку.

Управляемые сервера подключаются к управляющему, который компилирует все элементы конфигурации, которые должны быть применены к подключившемуся серверу в каталог, который передается управляемому серверу. Получив каталог, управляемый сервер начинает сравнивать своё текущее состояние с состоянием, описанным в каталоге, и принимает решения о том, какие действия нужно выполнить для приведения текущего состояния к описанному. При выполнении каталога учитывается, что некоторые действия необходимо выполнить раньше или позже других, что позволяет администратору не продумывать самому точный порядок выполнения задач, а только указать зависимости, если они необходимы.

Таким образов процесс применения конфигурации Puppet обладает следующими свойствами:

Идемпотентность Многократное применение одного и того же каталога никак не изменяет состояния системы и приводит к тому же результату, что и однократное применение. При внесении изменений в каталог произойдет их применение и привод текущего состояния системы к изменённому.

Декларативность Системному администратору требуется описать не порядок действий, который нужно выполнить для решения поставленной задачи, а то, каким должен быть результат работы.

Отказоустойчивость В случае невозможность связаться с управляющим сервером управляемая система воспользуется последним полученным от сервера каталогом и не предпримет никаких действий. В случае недоступности управляемой системы она получит произошедшие за время её отсутствия изменения позже, когда станет доступной, без вмешательства администратора.

На управляемых системах должна быть установлена клиентская часть Puppet — puppetd, которая выполняет как получение данных от сервера, так и применение изменений. Клиентская программа может запускаться следующими способами:

1. Демон. Программа будет работать постоянно, обращаясь к серверу за новым каталогом через определённый период времени (по умолчанию 30 минут) и сравнивая его с текущим состоянием.

2. Cron. Программа может запускаться при помощи cron получать новый каталог, выполнять его и завершать работу.

3. Вручную. Программа может запускаться вручную администратором после внесения им изменений в конфигурацию систем на управляющем сервере.

4. Локально. Клиентская программа может работать локально без участия управляющего сервера, исполняя манифест, который был помещён на управляемый сервер каким-либо другим способом, что может быть полезно при необходимости создания полностью независимых систем, которыми все же можно управлять, обновив вручную хранящийся на них манифест.

Подключение клиентской части к серверной происходит по протоколу HTTPS с использованием подписанного сертификата для каждого клиента, что обеспечивает как конфиденциальность передаваемой информации, так и принудительную аутентификацию каждого подключающегося клиента.