Deploying with confidence

Self-Service

Combining Puppet, Foreman, Pulp, Candlepin, and Katello lets you deploy software and servers automatically and securely. By Kurt Seifried

In last month's column, I talked about finding and remediating security problems; this month, I will continue with administrative automation. Security remediation is great, but at some point, you need the servers to do useful things like run software, process data, and so on. So, how do you efficiently control hundreds or thousands of systems, deploy and distribute software, and make configuration changes as needed? Of course, you can't just deploy stuff straight to production and hope for the best, so you'll probably want to support separate development, testing, and production groups at a minimum.

Puppet

Puppet [1] is a great tool, and it's written in Ruby[2], so it runs on pretty much every modern OS, and although not all vendors include Puppet, packages are available for most systems [3]. In a nutshell, Puppet allows you to manage your servers securely. You can define configurations and tests for servers, groups of servers, types of servers, applications, and so on, and then apply these tests and configurations to your systems. For example, you can create tests to check filesystem permissions and apply them to your systems, ensuring that files in `/etc/` are properly configured or locked down. You can also define an application to be installed, such as MySQL, and a specific version, say 5.5.20. And, because Puppet understands various systems (e.g., Red Hat Enterprise Linux, Debian, FreeBSD) it will execute the appropriate commands (i.e., `yum`, `apt-get`, and so on) for that system.

Puppet also supports inheritance, allowing you to override values. For example, Red Hat Enterprise Linux ships MySQL 5.1.61 but backports the relevant security updates, so for the MySQL example, you might require MySQL 5.5.20 on all systems except Red Hat Enterprise Linux, for which the 5.1.61 version with backports (`5.1.61-1.el5_2.1` at the time of this writing) is up to date.

Even better, Puppet modules are available to manage specific services such as Exim, Bacula, and Postfix, to name a few. The majority of modules seem to be hosted by various authors on GitHub, so going to GitHub and searching for "Puppet *something*" should return a Puppet module (assuming one exists); otherwise, Google is a good bet. The Exim module, for example, allows you to check that graylisting is present and enabled. These modules really help with a lot of the heavy lifting and application-specific issues that admins face.

Another aspect of Puppet is secure and authenticated communications, Puppet use SSL encryption with certificate-based authentication, so clients and Puppet servers are authenticated against each other, making it difficult to fall to man-in-the-middle attacks (in theory, it

KURT SEIFRIED

Kurt Seifried is an Information Security Consultant specializing in Linux and networks since 1996. He often wonders how it is that technology works on a large scale but often fails on a small scale.

shouldn't be possible). This is a critical aspect when managing servers distributed across a large network or even around the globe (an increasingly common occurrence with cloud computing). A final aspect, critical in large-scale deployments, is the reporting provided by Puppet. Making configuration changes is great, but you need to know what state the servers are in. (Did the changes take effect? Has someone undone them accidentally or otherwise?)

Foreman

With everything that Puppet does, why would you need Foreman [4]? For one thing, Foreman adds a really nice web-based front end to Puppet, especially when it comes to reporting and seeing the current status of systems. Foreman also supports provisioning servers using KickStart, JumpStart, Preseed, and AutoYaST, which means the majority of Linux systems are covered. Katello (discussed later) also supports DNS/DHCP and LDAP, so you can easily integrate the systems you are creating and deploying into your existing infrastructure.

Pulp

Simple question: Where do you plan to get your software updates and custom packages? Chances are, if you have any number of systems, you want to run your own repositories so you can control which software gets deployed to which systems (to update the system safely, knowing that every updated package available has been tested) and so you can place repositories close to your systems (e.g., one at each cloud provider you use). Pulp [5] lets you define package groups, systems groups, and so on. This approach allows you to have development, testing, and production repositories, and, by pointing the appropriate systems at them, you can easily control software updates. A package can start in development, be pushed to testing, and, once tested, pushed to production, or even a specific production group. This way, all your systems will be able to update. This process allows you to update systems selectively and in stages. For example, you can push a library update to production servers that have no closed third-party software installed and delay the update of this library on your servers with closed source software until it has been tested; or, you could even split your servers with the closed source software in half, update those, and if nothing breaks, then update the other half. Pulp also allows you to customize software easily; pushing your own packages into your repositories, having your own SELinux configuration packages (one for each application group), and deploying modified versions of software become very easy with Pulp.

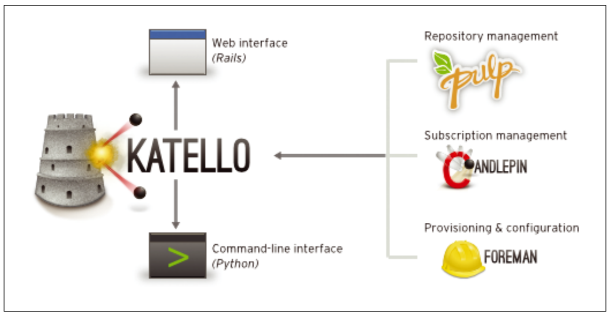

Figure 1: Katello and components (reproduced from katello.org).

Candlepin

One aspect of modern virtualized systems, and cloud computing especially, is that it is really easy to create and deploy servers. In the past, deploying a server usually involved purchase orders, waiting for hardware to be delivered, then burning in, setting up, and deploying. With modern systems, you can deploy a hundred servers in minutes with only a few clicks. So, if you are using software for which you have a finite number of licenses or are providing services that are licensed, you need some form of entitlement management.

Candlepin [6] provides just this, allowing you to create subscriptions and manage them, both internally (e.g., defining that you have up to 10 instances of a commercial product) and externally (e.g., allowing a client to deploy up to 20 instances of a server). Of course, Candlepin supports more than just arbitrary limits. For example, you can use certificate-based authentication or usernames and password, so the customer can go to a self-service portal and pay to receive a product or service. Business rules are also supported; you could share a pool of licenses among various business units based on the time of day (e.g., allowing more servers during working hours and rotating them around the globe to follow the sun, as it were).

I should note that licensing has become increasingly difficult with rapid increases and decreases in utilization; a crude example would be a customer who purchases 100 licenses of a product but actually wants to use 300 licenses at the same time, but for only 8 hours a day. Providing this kind of licensing structure will become common, I suspect, and providers who can accommodate it will be popular with their customers' accountants.

All Together Now – Katello

Now you face a classic open source problem: You have a great collection of tools and utilities, but you need to tie them all together into one central management system. This is exactly what Katello [7] does. Katello is, quite literally, a set of glue programs and web front end that ties Foreman, Pulp, and Candlepin together (Figure 1). Additionally, you get a great web-based administrative interface and a Python command line (you can automate management at a very high level using these tools).

Finally, because Katello supports different user roles and permissions, you can delegate administration securely, allowing developers to handle their own systems and make limited updates to test servers, for example, and you can restrict heavy lifting for management of production systems to a few trusted administrators. Thus, different departments and business units can share a single central management system, which allows IT to keep an eye on all the systems all the time. ■■■

INFO

[1] Puppet Labs: *http://puppetlabs.com/*
[2] Ruby language: *http://www.ruby-lang.org/en/*
[3] Downloading Puppet: *http://projects.puppetlabs.com/projects/1/wiki/Downloading_Puppet*
[4] Foreman: *http://www.theforeman.org/*
[5] Pulp: *http://www.pulpproject.org/*
[6] Candlepin: *http://www.candlepinproject.org/*
[7] Katello: *http://katello.org/*