Configuration Management Summit

June 24, 2010
Boston, MA

Summarized by Aleksey Tsalolikhin
(aleksey.tsalolikhin@gmail.com)

On Thursday, June 24, USENIX hosted the first Configuration Management Summit, on automating system administration using open source configuration management tools. The summit brought together developers, power users, and new adopters. There are over a dozen different CM tools actively used in production, and so many choices can bewilder sysadmins. The workshop had presentations of four tools, a panel, and a mini-BarCamp. This summary covers the four tool presentations and includes some brief notes on the BarCamp.

- ***Bcfg2***

Narayan Desai thinks of configuration management as an API for programming your configuration. Bcfg2's job is to be configuration management "plumbing"—it just works.

Centralized and lightweight on the client node, each server can easily handle 1000 nodes.

Bcfg2, pronounced be-config-two, uses a complete model of each node's configuration, both desired and current. Models can be compared (with extensive reporting on differences), or you can designate one node as *exemplar* and its configuration will be imposed on other nodes.

To facilitate learning, the Bcfg2 client can be run in dry-run (no changes, print only), interactive (are you sure you want to do this?), and non-interactive modes.

Bcfg2 supports extensive configuration debugging to help the sysadmin get to the bottom of things quickly, with full system introspection capability (why is Bcfg2 making the decisions that it is?).

Strengths: Reporting system. Debugging.

Weaknesses: Documentation (new set of documentation is coming out now, but still weak in examples). Sharing

policies between sites is not easy; group names need to be standardized first.

- ***Cfengine***

Mark Burgess explained the underlying philosophies of Cfengine:

- Promise theory: Files promise to be there, packages promise to be installed, processes promise to be running—or not, etc. Cfengine is the promise engine to fulfill those promises.
- Convergence: Describe an ideal state and Cfengine will get you there, as opposed to a roadmap/log of system changes necessary to bring a system to a configured state from a known starting state—Cfengine will get you to an ideal state from a known or unknown state.
- Self-healing: Assume the environment is hostile and entropy exists, and take measures to continuously check and restore the integrity of the system.
- Pragmatism: Environments can be participated in but not controlled; constrain rather than control the environment; cooperation will take you further than enforcement.

Strengths: Highly multi-platform: runs on very old and very new systems, the full gamut—underwater unmanned vehicles, Nokia cell phones, and supercomputer clusters; lightweight (1.9 MB footprint). The only prerequisites are Berkeley DB library and crypto-library. Cfengine has the largest user base—more companies using it than all the other tools combined. Resilient—able to continue operating under degraded conditions (e.g., if the network is down and server is unreachable, node agents will used the cached policy; Chef and Puppet run the same way). Secure—Cfengine has a healthy paranoid streak (assume they're out to get you) and an impressive security record (only three serious vulnerabilities in 17 years). Commercial version addresses knowledge management—ISO standard topic maps, etc.

Weaknesses: Hard to get started because there is a lot to learn.

- ***Chef***

Aaron Peterson, Opscode Technical Evangelist, presented Chef, primarily a configuration management *library* system and *system integration* platform (helps integrate new systems into existing platforms).

Chef is data-driven. Configuration is just data. Enable infrastructure as code to benefit from software engineering practices such as agile methodologies, code sharing through github, release management, etc. You manage configuration as *resources* (files, packages, processes, file systems, users, etc.), put them together in *recipes* (lists of resources), and track it like source code to configure your servers. *Cookbooks* are packages of recipes. Chef has been out since 2009.

Chef grew out of dissatisfaction with Puppet's non-deterministic (graph-based) ordering. Sequence of execution in Chef is tightly ordered.

Strengths: Cloud integration (automating provisioning and configuration of new instances). Multi-node orchestration. Reusable policy cookbooks and highest degree of recipe reuse between sites (compared to the other three tools).

Weaknesses: Attributes have nine different levels of precedence (role, node, etc.) and this can be daunting.

- ***Puppet***

Michael DeHaan explained that Puppet grew out of dissatisfaction with Cfengine 2. Puppet has a centralized model: a server detects deltas from the desired configuration and instructs the node agent to correct them. Chef works the same way.

Puppet's internal logic is graph-based. It uses decision trees and reports on what it was able to do and on what failed (and everything after it). Manual ordering is very important, as decision trees will be based on it. Ordering is very fine-grained.

The Puppet language is a datacenter modeling language representing the desired state. The Puppet language is designed to be very simple and human readable. This prevents you from inserting Ruby code but it also makes it safer (prevents you from shooting yourself in the foot). However, you can still call external (shell) scripts. Also, an upcoming version (2.6) will support programming in a Ruby DSL.

The server gets the client to tell the server about itself. These are *facts* in Puppet. The configuration policies are the *manifests*. The server compares the facts to the manifests and, if necessary, creates instructions for the clients on the managed nodes to move from what is to what should be. These instructions are encoded as a JSON catalog.

Strengths: Large community of users (over 2000 users on the Puppet mailing list).

Weaknesses: The Puppet server right now is a potential bottleneck, which is solved by going to multiple servers. Execution ordering can be non-deterministic but reports will always tell you what succeeded and what failed, and order can be mandated.

- ***BarCamp***

A BarCamp is an informal colloquium where the audience members take turns presenting to the audience (http://en.wikipedia.org/wiki/BarCamp).

There were a total of five 15-minute presentations from the audience during the final part of the summit. Matt Richards presented "Converting an Ad-Hoc Site to CM: The Story," narrating a successful Cfengine deployment with resulting increase in stability and uptime. Aaron Peterson gave a Chef demo. Michael DeHaan presented "Cobbler: Automated OS Installs," a Linux installation server. David Pullman presented "Cfengine: Complexities of Configuring Different Operating Systems." Finally, Michael DeHaan presented "Func—Attack!!!—Your Systems!!!"—Func is a distributed one-time command or query tool for Red Hat systems.

You can find a much more detailed report at http://www.verticalsysadmin.com/config2010/.