Configuration Managment with Cfengine

James Lee
Information Systems & Services
The George Washington University
jameslee@gwu.edu

February 3, 2010

1 Introduction

Configuration management software simplifies the management of large heterogeneous environments such as ours. Currently all of the systems configuration that we do is ad-hoc, either by one-off scripts or by hand. Even in small environments, this inevitably leads to systems which diverge from standards over time, and in growing environments like ours, it is simply not sustanible. I intend to explain how configuration management would improve our current methodology and specifically why Cfengine should replace JASS and manual configuration altogether.

1.1 Why Configuration Management?

Centralization Configuration management software reduces the configuration of hundreds of systems into a few files of well-defined rules. The rules are written in a simple description language defined by the software. Hosts only need to run the software over these rules to apply the configuration changes that they describe. This alone greatly eases scalability concerns. “Write once; apply everywhere.”

Heterogeneous Environment Those of us who have deployed Red Hat and SuSE servers know it is far away from the simplicity of our Jumpstart and JASS setup. We spend hours manually configuring system settings and services on Linux post-installation, trying to emulate what JASS does. JASS is designed to only work on Solaris, but most configuration management software will work on any Unix. It allows you to consolidate configurations which are the same and minimize code for handling special cases. Consider this Cfengine pseudocode which can copy a single `sshd` configuration file, customize it for a class of hosts, and reload the service on any version of Solaris or Linux:

```
copy:
    policyserver:/configs/sshd_config -> /etc/ssh/sshd_config define=reloadssh

editfiles:
    publicservers::
        Replace "AllowUsers .*" with "AllowUsers jameslee" in /etc/ssh/sshd_config

processes:
    reloadssh.solaris.!solaris10::
        "/etc/init.d/sshd restart"
    reloadssh.solaris.solaris10::
        "svcadm refresh ssh"
    reloadssh.linux::
        "service reload sshd"
```

Maintenance Configuration managment software can be used a lot of ways. It can be used like JASS: run it once and forget about it. However, ideally it should be used to maintain systems long after their installation. Whenever you update the central configuration rules or files, hosts should be able to pull the changes and apply them without any human intervention. This way, every server always conforms to the lastest standards all the time.

Looking at the same pseudocode above, if the canonical `sshd_config` file ever changed on the central policy server, Cfengine would notice the difference and reapply the rules on all of the target systems. Conversely, and less intuitively, if the file on the target host was ever modified manually, Cfengine would replace it with the canonical one. I expect if there is going to be any resistance to something like Cfengine, it's this. Used properly, *all* system configuration must be done centrally with Cfengine. It's an extreme departure from what most of us are used to, but it has advantages even beyond centralization and conformity such as:

Auditability The very same rules that Cfengine uses to apply configuration changes can be used to audit a system for standards compliance. JASS, on the other hand, requires separate scripts: one to apply the rules, one to check.

Documentation Configuration management also centralizes documentation. Instead of writing comments in configuration files and then describing what you did on the wiki, documentation can be written with the rules themselves. Furthermore, when paired with a version control system such as Subversion, metadata such as who made a change, when, and why can be preserved forever, even after a change is reverted.

Cfengine can optionally make backups of every target file it replaces or modifies. So in the above example, if `sshd_config` gets overwritten by a newer version, the previous file is preserved as a hidden, timestamped dot-file. Imagine what this could do for the mess of inconsistently named and documented `ipf.conf` files that we have on every host.

Simplicity The languages defined by programs like Cfengine for rules are "description languages." They are not general purpose scripting languages like `sh` which is used by JASS. What this means is you describe *what* you want done, not *how* to do it. The software agent figures out the best way to do what you describe.

Description languages like the one used by Cfengine are very easy to learn and hard to make mistakes in. They are specifically designed for the task of simplifying configuration managment, so often one line will do several things, plus error checking and reporting automatically.

1.2 Why Cfengine?