Use of *Cfengine* for deploying LCG/gLite middle-ware

Colin Morey

E-mail: Colin.Morey@manchester.ac.uk

Abstract. *Cfengine* is a middle to high level policy language and autonomous agent for building expert systems to administrate and configure large computer clusters. It is ideal for large-scale cluster management and is highly portable across varying computer platforms, allowing the management of multiple architectures and node types within the same farm. After brief introduction to *Cfengine*, the rest of this paper will focus on deployment of common middleware components utilizing various parts of *Cfengine*.

1. Introduction

A number of sites within the GridPP [1] collaboration and across the wider EGEE project are using *Cfengine* for the deployment and maintenance of their clusters and LCG/gLite middleware installations. This paper will collect together the experiences of both the initial setup and the ongoing maintenance of *Cfengine* installations. This paper will also discuss some of the common implementation scenarios that may be encountered. This paper will also provide some solutions to issues that the various sites have faced.

2. What is *Cfengine*?

Started in 1993 by Professor Mark Burgess as a research project at the Oslo University College into network unix configuration [2], *Cfengine* is a powerful configuration manager capable of scaling from a single laptop to a large heterogeneous deployment across company-wide IT infrastructures and runs on a wide-range of operating systems and dialects from AIX through to Microsoft Windows via cygwin. *Cfengine* is under active development and in use at a number of sites around the world and across the EGEE collaboration. Utilizing a flexible class system, coupled with a simple language for describing actions, *Cfengine* operates on an ideal-state basis, rather than a pre-described list of tasks to be undertaken. For example, to ensure that all compute nodes should have a specified minimal version of a package, you would say, "local-site.pkg version=2.4 install" rather than "rpm -Uvh local-site.pkg-2.4.rpm". A number of actions are predefined within *Cfengine*, from tidying directories, mounting remote filesystems, to an inbuilt alerting mechanism for system anomalies.

3. Classes within *Cfengine*

Much of *cfengine*'s power and flexibility comes from its use of classes. Classes can be broken down into 2 main types: hard classes, which typically do not change between execution runs, for example, node OS type or release, and soft classes, which are defined at the start of an execution run, and can be modified during execution, e.g. service type, package list, memory sockets, disk usage etc; Compound classes, e.g. sunos.nameserver, can be used to restrict actions to certain machines, allowing the re-use of one configuration file for many systems, and the inclusion or exclusion of specific tasks for other hosts.

4. Components of *Cfengine*

There are 4 main components to a *Cfengine* installation: *cfagent* - the workhorse of *Cfengine*; *cfexecd* - the automated wrapper around *cfagent*; *cfservd* - the fileserving daemon; and *cfenvd* - the anomaly monitor. *cfagent* is normally executed either from 'cron' or by an existing *cfexecd* daemon, on a schedule determined by cfagent.conf. Communication between *cfagent* and *cfservd* is via 2-way public key verification, both client and server verifying their identities with each other. File transfers can be done either in the clear for trivial files or via SSL encrypted connections for more sensitive data. The remaining persistent process, *cfenvd* builds up a profile of a host, and can define classes within *cfagent* when certain metrics increase dramatically over their norm, for example, load, network throughput or number of active sessions to a network socket.

5. Typical site Installation

Usually the *Cfengine* configuration files will be stored within a local site CVS or SVN repository, which the primary *cfservd(s)* will extract from on a periodic basis. Clients will then update their local copies from the *cfservd* before continuing on with their execution run. Larger sites may have more *cfservd* instances, with some dedicated to serving different classes of client. Simple balancing of load between *cfservd* instances can be obtained through the use of 'schedules' and weightings.

6. An Overview of the LCG/gLite Middleware deployment

The most common scenario for installing the gLite Middleware within GridPP [1], is via YAIM [3] on top of Scientific Linux [4], so will be the focus of this paper. After the installation of the base OS, a mirror of the glite repository needs to be added to the *yum* [5] configuration file. As this file is relatively simple, a copy of a pre-modified file from a CVS repository using *Cfengine*'s *copy*directive will suffice. Then the installation of YAIM [3], along with the relevant meta-rpm to pull in all the dependancies for the node-type can be accomplished using the *packages* directive. Before the invocation of the *configure_node* command, a copy of the *site_info.def* file needs to be copied over from CVS via *cfservd.* A simple *shellcommands* statement is then all that is sufficient to run *configure_node* and complete the installation. As there is no human interaction after the initial boot stage, this method is reproducible and highly scalable.

7. *Cfengine* at the Lancaster University HEP Grid farm

Lancaster University runs *Cfengine* across its Compute and 200 worker node farm, along with the dcache[8] Storage Elements and a few of their departmental machines. As a site, the biggest benefit of running *Cfengine* is the reassurance that all of their worker nodes are running the same software releases with the same configuration, cutting down administrative overheads. Recently they have been migrating their cluster from SL3[4] to SL4[4], and in the process identified a number of extra packages that needed to be added to support glite3.1[6] running on SL4/64[4]. Due to the simple language that *Cfengine* configuration files are written in, this work has been trivial to share and to integrate within the configurations for other sites, even though the layout may be drastically different.

8. *Cfengine* at the Glasgow University Tier2

Glasgow University's *Cfengine* install spans 140 Worker nodes, with 10 DPM[7] based Storage Elements and a range of 8 service machines from a Compute Element through a Resource Broker to a local User Interface. Mostly simple tasks are managed with *Cfengine*, from the distribution and installation of new RPMs across the cluster, e.g. lcg-CA, to the maintenance of symbolic links to the latest java installation. Ownership and permissions of the DPM filesystems within the Storage Elements is also handled from within *Cfengine*. Another common task performed is the centralization of the user databases, i.e. /etc/passwd, /etc/shadow and /etc/group, maintaining a consistent view across the cluster to all users, and avoiding the overhead of managing a site-wide NIS or ldap authentication system.

9. *Cfengine* at the Manchester University HEP Tier2 Facility

Due to the size of the Manchester cluster (900 worker nodes split evenly across 2 clusters), they have 2 cfservd instances serving both logical clusters and their associated servers, from d-cache [8] head nodes running Scientific Linux 3[4], service machines running Gentoo Linux[9] and an SL4[4]/glite3.1 testbed. Currently *Cfengine* is used to augment the site nagios[10] network and host monitoring system by configuring alerts when free space falls below an acceptable threshold. Also local daemon processes for example, pbs_mom, and ntpd, are monitored, restarting as required. Package management is a now handled almost exclusively by *Cfengine*, enabling the fixing of software at specific releases, and also to ensure daemons and configuration files remain in place after middleware upgrades. Coupled with the kickstart architecture, re-installation and configuration of a node or an entire cluster is possible within 20-40 mins, and ensures that all nodes are at the same state and configuration. VO (Virtual Organisation) administration has also been made significantly smoother by centralizing the configuration, and having *Cfengine* invoke YAIM[3] to setup the node whenever a new VO is added, or an existing VO configuration changes. Other uses of *Cfengine* have included the upgrade of both dcache[8] Storage Elements from 1.6 to 1.7, and the automated rollout of an XrootD[11] instance for local users.

10. Problems Common to all Sites

All of the sites have expressed concerns about the helpfulness of the error messages, with the two most common ones being SIGPIPE issues, mainly caused by load on the server, fixable with an increased 'splay_time'. And the 'Host authentication failed. did you forget the domain?' message often due to keys being out-of-align. Most of the helpful information is hidden in the debug messages beyond the -d2 level, requiring wading through a lot of extra information.

11. Conclusion

The benefits of *Cfengine* on large clusters are mainly due to offloading a lot of the trivial day-to-day maintenance work of rpm deployment, file distribution and service monitoring, freeing time up for system development and user support. The amount of time saved is dependent on the size of cluster and in part to how much of the system is under *Cfengine* control, at Manchester *Cfengine* saves in order of a few hours of work a week, more when systems need to be re-built following hardware replacements. Without *Cfengine*, or any other scripts, a typical system administrator may be able manage 10s of machines on a day-to-day basis, with scripts and automated file distribution tools like dsh[12], the number of manageable systems rises to the low 100s. Success stories include AMD[13] with a company-wide install across 9 sites, with the three largest having over 10,000 CPU cores and over 500 workstations[14] each.

12. References

[1] http://www.gridpp.ac.uk
[2] http://www.iu.hio.no/ mark/papers/cfengine_history.pdf
[3] http://en.wikipedia.org/wiki/Yaim
[4] https://www.scientificlinux.org/
[5] http://en.wikipedia.org/wiki/Yellow_dog_Updater
[6] http://glite.web.cern.ch/glite/packages/R3.1/
[7] https://twiki.cern.ch/twiki/bin/view/LCG/DpmAdminGuide
[8] http://www.dcache.org/
[9] http://www.gentoo.org/
[10] http://www.nagios.org
[11] http://xrootd.slac.stanford.edu/
[12] http://www.netfort.gr.jp/ dancer/software/dsh.html.en
[13] http://www.amd.com/
[14] https://cfengine.org/pipermail/help-cfengine/2007-December/002563.html