CfAdmin: A User Interface for Cfengine

Charles Beadnall – WR Hambrecht
Andrew Mayhew – Logictier, Inc.

ABSTRACT

CfAdmin provides a relatively easy-to-use web interface for system administrators to package and deploy software using GNU Cfengine. It conceals the intricacies of cfengine configuration and reduces potential errors by storing all configuration data in a relational database.

The interface is divided into administration, inventory, package and deployment sections to follow an administrator's work-flow. In addition to implementing most of the features of cfengine, CfAdmin provides a configuration file template system to allow customization of packages per deployment. This allows an administrator to readily leverage a single package, such as Apache, across multiple installations by modifying templated areas of the configuration file (i.e., httpd.conf ServerRoot).

CfAdmin was created to streamline the deployment of software for a managed service provider, but could be equally at home in other environments with large numbers of applications and servers. It is an update to an existing cfengine installation supporting over 200 servers in various colocation facilities and has the side-benefit of being a self-documenting system.

Motivation

As the number of server hosts and server applications we manage has increased during the Internet years it has become increasingly important to efficiently use human resources to manage those hosts. The available workforce continues to be limited, and non-automated software installation processes must be rigorously proceduralized to maintain the documented uniformity required for reliable operation.

Several frameworks have been either developed or glued together to serve the function of centralized server management. PIKT and cfengine are two examples that automated the administration of hosts. While many other software systems are available, each contains the following components, at a minimum:

- Application and configuration repository
- Network distribution system
- Host management/verification agents

Unfortunately, what many of these systems lack is a user interface beyond the configuration files of the management software. These configuration files, in the cases of Cfengine and PIKT, are written in a unique syntax that can take weeks to learn, and a slight error can have disastrous consequences on a network of managed hosts. Training a large group of administrators to maintain and update these configuration files is difficult since most administrators are busy fighting fires and do not have the free time to learn complex and error-prone command structures.

In an attempt to make the underlying management framework easier to use, we developed CfAdmin – a database-backed web interface to manage the management software. The database and interface for CfAdmin glue various administration tools in our framework together and abstract the arcana of proprietary syntaxes.

Interface and Database

Our administration system is composed of the following major parts, with the latter three comprising the CfAdmin interface:

- CVS (v1.10.7) for version control of configuration files
- Cfengine (v1.6.3) server and agent for host management
- Apache Webserver (v1.3.12)
- Tomcat Java servlet engine (v3.2.1)
- Postgres database server (v7.0.1)

The reasoning for our selection of these software packages primarily related to their free availability and previous use experiences. Budgetary considerations also eliminated the few available commercial packages. As an abstracted design, we believe most of these applications could be replaced with functionally similar software. In our design, Apache serves to authenticate users, passing the user name to Tomcat, which performs the bulk of the work using JSPs for formatting and JavaBeans for database access to Postgres.

The database schema was thought out well in advance of much of the interface development and concurrent to the initial deployments of CVS and cfengine. From a design standpoint, it was easier to have all the possible tables and fields we thought we would need in the beginning, rather than trying to add them later to a running production schema. The information we needed to capture fell into four categories:

- **Administration**: Information on users, groups, domains, vendors and authentication rights for CfAdmin.
- **Inventory**: Hardware asset information to correlate machine architecture, operating system, peripheral hardware, and rack/facility location.

- **Package**: Installation specification for individual versions of an application to create distribution packages (similar to an RPM Spec File).
- **Deployment**: Group relationships of hosts to applications, including configuration file customizations.

In a previous prototype, we found administrators would quickly revert back to manual operation when a function was either counterintuitive or unavailable using a graphical interface. To mitigate this risk, we attempted to divide workflows around the common tasks of host inventory management, application package definition, and package deployment.

Administration

In many respects, this portion of the database and interface contains information which, while needed, does not fit into the other categories. This includes information regarding vendor contacts, domains, timezones, countries, and languages. The primary purpose of the administrative section is to deal with the definition of access rights and controls. Group roles are defined and are assigned access rights. Then users are associated with these groups.

In our setup, users are assigned either to facilities, packaging, or operations. People in the facilities group only have access rights to the inventory section of the interface in order to update and maintain facilities-tracking data. Those in the packaging group are the only people allowed to define new packages and to modify existing package definitions. And while operations has read access to all sections, they can only define deployments and make configuration customizations of packages for those deployments. In this way, access controls are maintained.

Inventory

The inventory management section provides a view of systems managed on our network. It captures information related to the location, manufacturer, model, operating system, and network setup of a particular machine. A subset of this hardware information is automatically populated in the database by an audit script when a host is brought up the first time. This automation reduces the amount of time spent on data entry and provides valuable information on the available hardware. CfAdmin provides forms to update this information with anything the audit script missed, including rack elevation and facility location.

Package

The application package section allows the package maintainer to first provide descriptive information about the software package and then to copy the package's files from a CVS repository. This copy process also gathers ancillary information about the file tree, including ownership and permissions. An interface to additional functions such as removing specific files and executing select files on installation is also provided. After the creation of this *gold master* copy, the user can single out configuration files from the imported file system to be modified per deployment. We use a simple templating format to allow the package maintainer to select which areas of the configuration file can be modified per deployment. Beyond these file specifications, the package maintainer can also define actions which need to be done to a package upon deployment. These include making sure certain files contain the proper permissions and ownerships, running pre- and post-install scripts, creating symbolic links so that initialization scripts are run at boot-time, and process checks.

Deployment

The package deployment section provides the ability to create deployment groups, analogous to cfengine Classes, composed of both managed hosts and application packages. The configuration files of packaged applications can also be modified, based on the information provided in the package template.

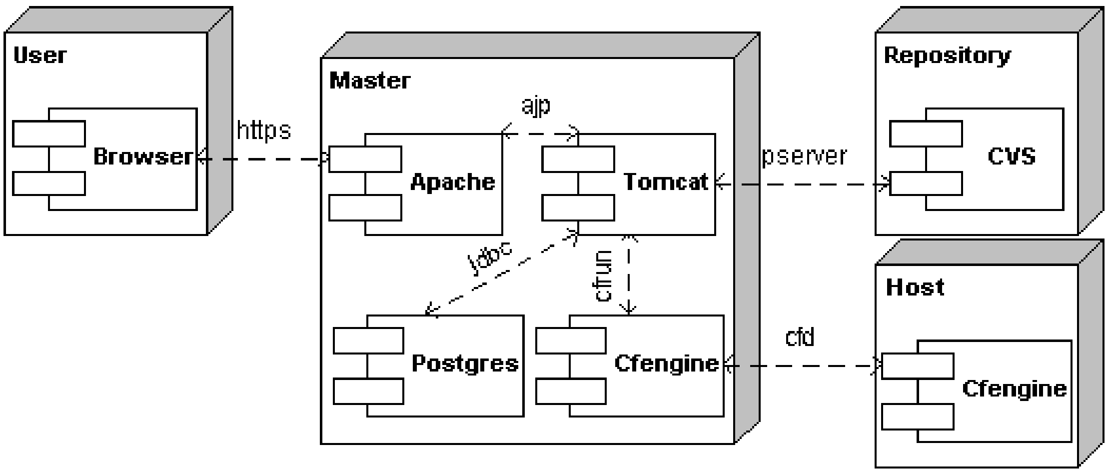

Figure 1: Logical layout of CfAdmin components and communications channels.

In this fashion, an operator can define a deployment group, *Customer 1 Webservers*, which contains hosts *foo* and *bar*. They can then associate the package *Apache 1.3.12 Solaris* to the deployment group. Then they can modify the *httpd.conf* file (based on the configuration template) for the package to listen on port 8080. Once satisfied with the relationship definitions and the configuration customizations, they can just click on the deploy button. This will call a configuration script, which generates the proper cfengine configuration files from the database and then execute the cfengine cfrun agent to force distribution of the application to the newly configured hosts.

Configuration Script

The configuration script's primary function is to act as the glue between CfAdmin and the server-management system. To do this, it takes the information stored in the database regarding packages and deployments, and generates the appropriate configuration files. Our design hope is that analogies between the database schema and the configuration file can be found in any functionally robust server-management system. Admittedly, this is only theory, since we only implemented this glue for one management framework.

For our implementation framework, the script creates cfengine configuration files. From the package section of the database, the script generates a configuration file per package. These configuration files contain the necessary information to install the package files and maintain them. This includes file permission and ownership information along with process-checking and script execution. From the deployment sections of the database, the script creates the classing and control file for cfengine. This file contains all the class definitions and relationships defined by the operations group. In this way, the configuration script is

```
classes:
  midnight = ( Hr00 Min00  )
  Customer1 = ( Customer1_Solaris_webserver  )
  Customer1_Solaris_webserver = ( foo bar  )
  Apache_1_3_12_Solaris_sparc = ( Customer1_Solaris_webserver  )
  Apache_1_3_12_Solaris_sparc_conf = ( Customer1_Solaris_webserver  )

control:
  Customer1_Solaris_webserver::
        Apache_1_3_12_Solaris_sparc_force = ( "true" )
        Apache_1_3_12_Solaris_sparc_class = ( "Customer1_Solaris_webserver" )
        Apache_1_3_12_Solaris_sparc_sched = ( "any" )

import:
  Customer1_Solaris_webserver::  $(CFINPUTS)/packages/cf.Apache_1_3_12_Solaris_sparc
```

Listing 1: A generated cfengine classing and control file.

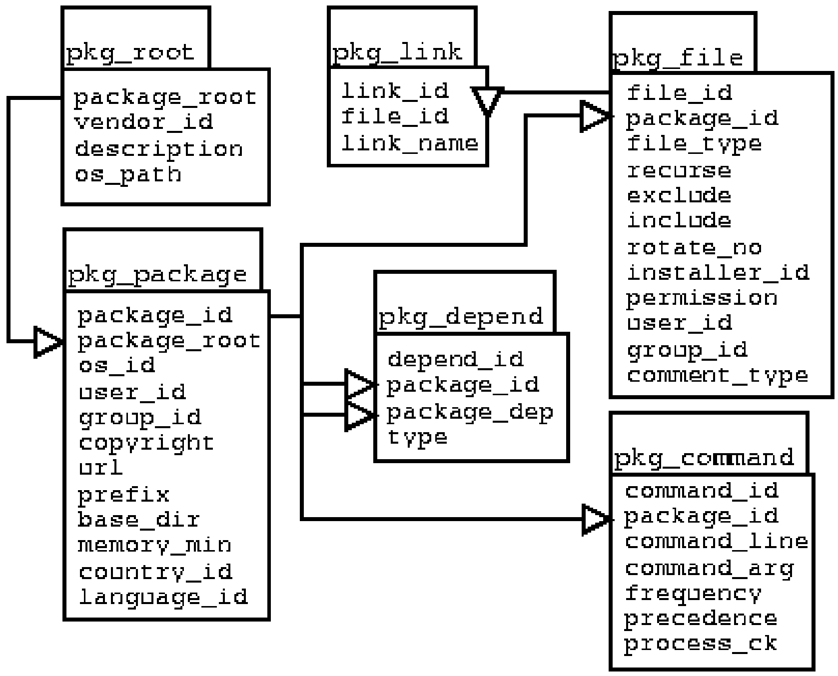

Figure 2: Simplified UML diagram of the packaging section of the database.

able to generate a nearly complete set of cfengine configuration files. The only parts of the cfengine configuration which are not included in the database are implementation-specific bits of data, such as the name of the gold server.

Listing 1 is an example of a generated cfengine classing and control file while Listing 2 shows the cf.Apache_1_3_12_Solaris_sparc package file imported in the previous configuration example.

Future Considerations

In our current implementation, CfAdmin primarily exposes the capabilities of cfengine without expanding on them. A few of the considered expansions include more complete utilization of the CVS repository to provide the ability to tag configurations as "known-good" and to roll back to them in cases of failures. We could also implement an interface to systematically upgrade applications and to schedule deployments at a future date. Scheduled deployments could be useful to plan out long-running maintenance events that span several days. Lastly, CfAdmin by its nature is Unix-centric, and we need to investigate what additional or changed features and functionality the inclusion of Microsoft Windows 2000 management will require.

Author Information

Charles Beadnall is currently responsible for auction operations at investment bank WR Hambrecht in San Francisco. He previously worked for managed service provider Logictier responsible for infrastructure management and for CNN in Atlanta responsible for web site operations. His prior experience includes architecting and implementing Turner Broadcasting's enterprise Internet security systems, managing web development projects at computer manufacturer NCR and managing application support for German Internet service provider Connect GmbH Informations System.

```
#################################################################
## Apache from ASF package classes.
#################################################################
copy:
  Apache_1_3_12_Solaris_sparc::
        /master/package/Apache_1_3_12_Solaris_sparc/apps/apache
                dest=/apps/apache
                server=$(cfd_master)
                recurse=inf
                type=checksum
                force=$(Apache_1_3_12_Solaris_sparc_force)

  Apache_1_3_12_Solaris_sparc_configs::
        /master/configure/$(Apache_1_3_12_Solaris_sparc_class)\
/Apache_1_3_12_Solaris_sparc/apps/conf/httpd.conf
                dest=/apps/conf/httpd.conf
                server=$(cfd_master)
                recurse=inf
                type=checksum
                force=$(Apache_1_3_12_Solaris_sparc_force)

files:
  Apache_1_3_12_Solaris_sparc::
        /etc/init.d/apachectl
                mode=554
                owner=httpd
                group=httpd
                recurse=inf

links:
  Apache_1_3_12_Solaris_sparc::

shellcommands:
Apache_1_3_12_Solaris_sparc.after::
        "apachectl start"

editfiles:
  Apache_1_3_12_Solaris_sparc::
        { /apps/etc/package.lst
                AutoCreate
                SetLine "Apache_1_3_12_Solaris_sparc    $(date)"
                AppendIfNoLineMatching "^Apache_1_3_12_Solaris_sparc.*"
        }
```

Listing 2: The cf.Apache_1_3_12_Solaris_sparc package file.

Andrew Mayhew has a BS-CS from the University of Central Florida in 1997. He stayed in Orlando to work at various ISP's until moving to California to work for Netscape Communication. He left Netscape after the AOL purchase to work as a Senior Internet Infrastructure Engineer for Logictier, Inc. until their closing this September. He continues to live in Sunnyvale, CA and can be reached electronically at amayhew@icewire.com.

References

[apache] "Apache HTTP Server Project," http://httpd.apache.org/ .

[burgess] Burgess, Mark, "Cfengine: A Site Configuration Engine," *USENIX Computing Systems*, Vol. 8, No. 3, http://www.iu.hio.no/˜mark/papers/cf1.ps, 1995.

[burgess] Burgess, Mark, "Strategies for Distributed Resource Administration Using Cfengine," http://www.iu.hio.no/ ˜mark/papers/cf2.ps, *Software-Practice and Experience,* Vol. 27, p. 1083, 1997.

[burgess] Burgess, Mark, *Computer Immunology*, Proceedings: Twelfth Systems Administration Conference (LISA 1998), http://www.iu.hio.no/˜mark/research/AIdrift/AIdrift.html, USENIX, 1998.

[cfadmin] "CfAdmin," http://www.icewire.com/cfadmin/ .

[cfengine] "GNU Cfengine," http://www.cfengine.org/ .

[cvs] "Concurrent Versions Systems," http://www.cyclic.com/cvs/info.html .

[harlander] Harlander, Dr. Magnus, "Central System Administration in a Heterogeneous Unix Environment: GeNUAdmin," *USENIX Proceedings: Eighth Systems Administration Conference (LISA 1994)*, http://www.usenix.org/publications/library/proceedings/lisa94/harlander.html, 1994

[huddleston] Huddleston, Joel, and Steve Traugott, "Bootstrapping an Infrastructure," *USENIX Proceedings: Twelfth Systems Administration Conference (LISA 1998)*, http://www.infrastructures.org/papers/bootstrap/bootstrap.html, 1998.

[jakarta] "Tomcat Java Servlet Engine," http://jakarta.apache.org/ .

[pikt] "PIKT, A System Monitor (And Much More)," PIKT's primary task is to warn of problems, but to fix those problems when needed, http://www-gsb.uchicago.edu/pikt/ .

[postgres] "PostgreSQL Database Server," http://www.postgresql.org/ .

[rpm] "Redhat Package Management," http://www.rpm.org/ .

[webmin] "Webmin," http://www.webmin.com/webmin/ .