Automating the world's datacenters

Cfengine, the world technology leader in datacenter automation, based on state-of-the-art research and development, is used by more than 5000 companies on millions of machines world-wide. Versatile and lightweight, Cfengine is the preferred solution for the most exacting system administrators.

WHY CFENGINE?

DOWNLOAD

SUPPORT

THOUGHT-LEADER SINCE 1993

Computer immunology - Convergence

Promise theory - Knowledge Map

Events

May 08-12	Interop - Las Vegas
May 16-17	Computer World OSBC - San Francisco, CA
June 14-16	O'Reilly Velocity - Santa Clara, CA
July 19-21	FOSE - Washington DC
Aug 29-Sept 1	VMworld - Las Vegas
Dec 4-9	LISA - Boston

Latest news

We are hiring!

Cfengine announced as finalist at Network Products Guide 2011

Limited time and resources? Pick Cfengine.

Upgrade from Cfengine 2 › 3 and beyond

Linux Network Services:

Configuration Management: CFengine

This material is based on work supported by the National Science Foundation under Grant No. 0802551

National Science Foundation
WHERE DISCOVERIES BEGIN

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author (s) and do not necessarily reflect the views of the National Science Foundation

Lesson Overview

System administrators are constantly challenged when managing large enterprise computer systems using Linux-based operating systems. These challenges may lead to inefficient operations and additional financial burdens. Administrators are required to know a variety of command line differentiations, dependency variations, support options and a host of other challenges.

CFengine was developed to help administrators manage large enterprise systems without the heavy reliance on shell-scripting. CFengine offers a free, reliable, platform independent option for remote enterprise management.

This lesson will introduce you to the CFengine administrative tool and will provide a basic overview of its use and configuration. Lab activities, assignments, and forum discussions have been designed to introduce you to the CFengine application and increase your familiarity with this reliable tool.

Objective

You should know what will be expected of you when you complete this lesson. These expectations are presented as objectives. Objectives are short statements of expectations that tell you what you must be able to do, perform, learn, or adjust after reviewing the lesson.

Lesson Objective:

Given five computers that need to be configured, students will evaluate the shortcomings of shell scripting that gave rise to configuration management tools such as CFengine and will illustrate the use of one configuration tool for maintenance as per industry standards.

Lesson Outline

In this lesson, you will explore:

- Introductory Notes
- CFengine Overview
  - What is CFengine?
  - Network Admin Tools
  - Linux-based Config Tools
  - Benefits and Use of CFengine
- Installation & Configuration
  - System Management
  - Installation & Prerequisites
  - Authentication & Syntax
  - Log Files

Notes to Students

This lesson is written to provide basic information about CFengine. The links and videos will provide essential and detailed information that you will need to complete labs and activities. Be sure to review the videos and links, even if they require some patience or might be longer than usual.

Review the links on this page before starting this lesson to increase your familiarity with available resources on CFengine.

Recommended Links: CFengine

- ❖ Lecture on CFengine (long, but helpful)
- ❖ Reference manual (Helpful for lab activities)
- ❖ CFengine Solutions (Helpful for lab activities)

Helpful Links: CFengine

- Intro to CFengine
- White papers
- Guide to CFengine
- Getting started
- Crash Course
- CFengine examples
- Complete configuration

CFengine

CFengine is a suite of programs for integrated autonomic management of either individual or networked computers. It has existed as a software suite since 1993 and is published under the GNU Public License (GPL v3) and a Commercial Open Source License (COSL).

CFengine was designed to be an easy to use, automated remote configuration tool. It can be used to generate reports, monitor system changes in remote computers, add and remove users and more. Basically, CFengine can allow one administrator the ability to remotely manage thousands of computer configurations all over the world.

CFengine's main attractive feature is it's price. It is free to the general public. With over 17 years in the IT age, CFengine is proudly used by hundreds of corporations all of the world as well as numerous Fortune 500 companies.

Demo Videos: CFengine

- Installation
- Webserver
- DNS Resolver
- Change detection
- Process Kill Restart

Versions of CFengine

There are two different versions of CFengine currently in use. CFengine3 is the choice of most administrators because it is based on promise theory and is much easier to use than the previous version—CFengine 2.

CFengine2 is still used today because it works well and is trusted by those who have used it for some time.

There is a conversion utility for those who wish to convert from CFengine2 to CFengine3.

Select **PLAY** below to review the conversion process from CFengine 2 to CFengine 3.

View Video
VideoLesson7Cfengine2to3(C4L7S7).mp4

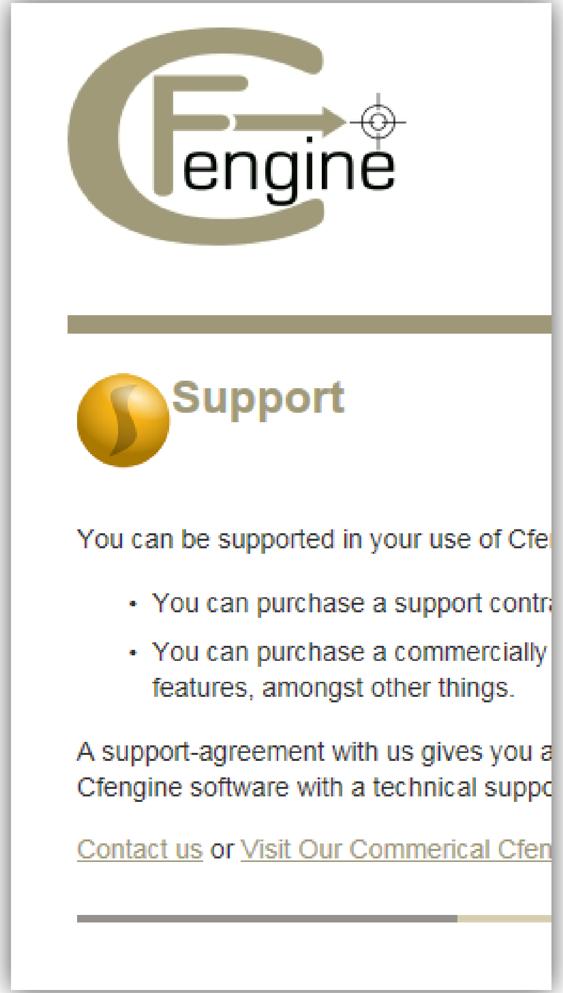
Network Admin Tools

System administrators are normally required to do repetitive tedious tasks that consume significant time and resources. Examples of these redundant tasks include:

- configuring hosts
- creating users
- managing applications, daemons, and services
- monitoring systems for changes
- checking security reports
- monitoring hardware changes

Tools like CFengine allow one administrator the capability to monitor, change, and configure systems (to which he/she has access) throughout the world.

Effective use of CFengine has the potential to reduce the number of additional employees required for a task and decrease the bureaucracy involved with managing systems in a large corporate configuration.



Support

You can be supported in your use of Cfe

- You can purchase a support contr
- You can purchase a commercially features, amongst other things.

A support-agreement with us gives you a Cfengine software with a technical supp

Contact us or Visit Our Commerical Cfen

Popular Network Admin Tools

In addition to CFengine, administrators use a variety of management tools including:

- *Puppet* (http://puppet.reductivelabs.com/):
  A configuration management tool written in Ruby with a client-server model that uses a declarative language to configure clients.

- *LCFG* (http://www.lcfg.org/):
  A client-server configuration management tool that uses XML to define configuration.

- *Bcfg2* (http://trac.mcs.anl.gov/projects/bcfg2):
  A client-server configuration management tool written in Python. It uses specifications and the client responses to configure target hosts.

- *Chef* (http://www.opscode.com/chef/)
  Chef is an open source system tool that provides integrated configuration management to an entire network or infrastructure. To use Chef, you write the code that defines the various parts of your network or infrastructure and then use Chef to apply those settings to your servers.

Suggested Reading
- Puppet
- LCFG
- Bcfg2
- Chef

What Makes CFengine Different?

The first and most attractive feature of CFengine is the price. “Free” is an attractive offer. Even though the community edition is free, it offers numerous services and abilities for system administrators to remotely monitor and configure system components. No longer is one administrator needed at each remote location for general administration.

CFengine allows a centralized administrator to gain access to remote systems and make changes as required. By decreasing the “human factor” in computer science, CFengine increases uptime and system efficiency and reliability.

The other major advantage of CFengine is that it has been in development for 17 years. It is recognized as one of the front runners of remote system administration tools by major corporations and governments all over the world. The product not only saves corporations money because of its price, but it is also recognized as being SOX compliant. Companies that use CFengine get breaks on insurance and other administrative overhead relating to computer science.

Required Reading

- Sox Compliant
- Sarbanes-Oxley Act

Benefits of CFengine

- Detect file, content, and process change
- Control file integrity
- Report various changes
- Automatic compliance to defined policy
- Increased chance for SLA compliance
- Always have latest security patches installed
- Always have the right version of the software running
- Start, stop, restart processes

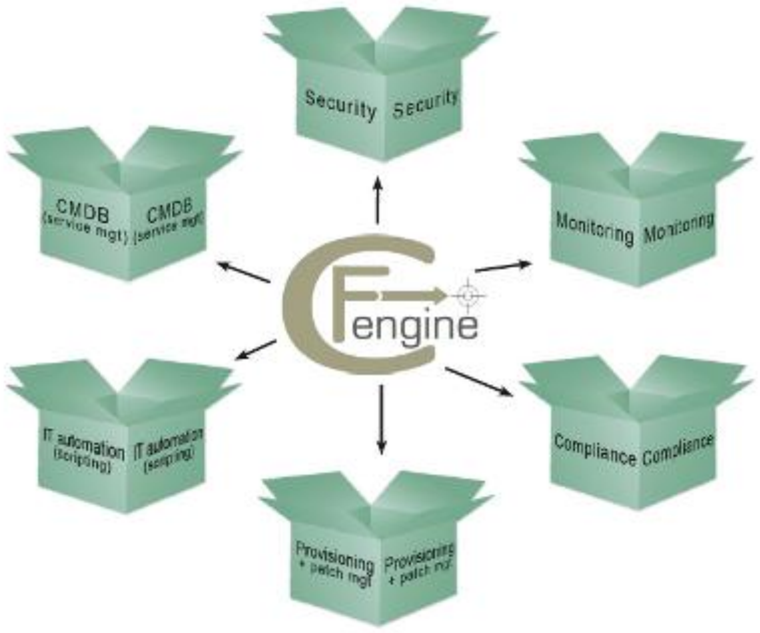


Illustration from: https://CFengine.com/inside/cfv3

Who Use CFengine?

Companies all over the world use CFengine on either one or up to tens of thousands of computers. Some of these companies include fortune 500 companies and the following:

**Logos are trademarked and owned by their respective corporations.*

CFengine Terminology

- **Bundle** - a bundle refers to a collection of promises
- **Promise** – the expression or documentation of an intention to behave or act in a certain way
- **cf-agent** - active agent (responsible for maintaining promises about the state of your system). In CFengine 2 the agent was called cfagent.
- **cf-execd** - scheduler { responsible for running cf-agent on a regular (and user-configurable) basis (in CFengine 2 the scheduler was called cfexecd). EXECUTOR cf-execd keeps the promises made in bundles.
- **cf-know*** - knowledge modelling agent { responsible for building and analyzing a semantic knowledge network. cf-know keeps the promises made in bundles.
- **cf-monitord** - passive monitoring agent (responsible for collecting information about the status of your system, which can be reported upon or used to enforce promises or influence when promises are enforced). In CFengine 2, the passive monitoring agent was known as cfenvd.
- **cf-promises** - Promise validator (used to verify that the promises used by the other components of CFengine are syntactically valid. cf-promises do not execute promises; instead, the syntax checks all promises.
- **cf-runagent** - Remote run agent (used to execute cf-agent on a remote machine. cf-runagent does not keep promises; it is used to ask another machine to do so). In CFengine 2, the remote run agent was called cfrun.

CFengine Terminology (Contd)

- **cf-serverd** - Server used to distribute policy or data files to clients requesting them and used to respond to requests from cf-runagent. In CFengine 2, the remote run agent was called cfservd.
- **cf-report** - Self-knowledge extractor takes data stored in CFengine's embedded databases and converts them to human readable form. Cf-report keeps the promises made in bundles.
- **cf-key** - Key generation tool that runs once on every host to create public/private key pairs for secure communication. In CFengine 2, the key generation tool was called cfkey. cf-key does not keep promises.
- **Libraries** - A library generally refers to collection of standardized CFengine code that can be reused in different scenarios and environments such as bundles of promises, or reusable body-parts.
- **Policy** - a set of intentions about the system, coded as a list of promises. A policy is not a standard, but the result of specific organizational management decisions.
- SOX Compliance (Sarbanes-Oxley Act compliance) - An audited accolade for financial data security required by all companies on the New York stock exchange.
- **Template** - an incomplete piece of CFengine code with blanks to fill-in. It is often a policy fragment that can be re-used in different scenarios. This is often used interchangeably with the term *library*.
- **WORKDIR** - The private work space CFengine uses to write reports and logs.

What is a Promise?

The term “promise” is another interesting and unique concept to CFengine. In today’s world, we spend a lot of time adjusting to changes, especially in the IT world. Instead of managing changes with CFengine, the philosophy is to “promise” something will be done and will be done correctly.

The promise concept is similar to promising your mother or spouse you will take out the trash. In most cases, it can be assumed the job will be completed in a proper manner and if anything goes wrong, there is a plan to “clean up the mess” or return to the previous state (where you were prior to the promise being implemented).

The CFengine software manages every intended system outcome as `promises' to be kept. A CFengine promise corresponds roughly to a rule in other software products, but importantly, promises are always tasks that can be kept and repaired continuously, on a real time basis, not just once at install-time.

Suggested Review

- CFengine Documentation

System Management

There are four commonly cited phases in managing systems, summarized as follows:

- Build
- Deploy
- Manage
- Audit

These separate phases originate with a model of system management based on transactional changes. CFengine's conception of management is somewhat different, as transaction processing is not a good model for system management, but we can use this template to see how CFengine works differently.

Build - A system is based on a number of decisions and resources that need to be `built' before they can be implemented. Building the trusted foundations of a system is the key to guiding its development. You don't need to decide every detail, just enough to build trust and predictability into your system.

In CFengine, what you build is a template of proposed promises for the machines in an organization such that, if the machines all make and keep these promises, the system will function seamlessly as planned. This is how it works in a human organization, and this is how it works for computers too.

Deploy, manage, and audit will be discussed on the next page.

Content taken from:
http://www.cfengine.org/manuals/cf3-tutorial.html

System Management (Contd)

Deploy - Deploying really means implementing the policy that was already decided. In transaction systems, one tries to push out changes one by one, hence ‘deploying’ the decision. In CFengine, you simply publish your policy (in CFengine parlance these are ‘promise proposals’) and the machines see the new proposals and can adjust accordingly. Each machine runs an agent that is capable of implementing policies and maintaining them over time without further assistance.

Manage - Once a decision is made, unplanned events will occur. Such incidents traditionally set off alarms and humans rush to make new transactions to repair them. In CFengine, the autonomous agent manages the system, and you only have to deal with rare events that cannot be dealt with automatically.

Audit - In traditional configuration systems, the outcome is far from clear after a one-shot transaction, so administrators usually audit the system to determine what actually happened. In CFengine, changes are not just initiated once but are also locally audited and maintained. Decision outcomes are assured by design in CFengine and maintained automatically, so the main worry is managing conflicting intentions. Users can sit back and examine regular reports of compliance generated by the agents, without having to arrange for new ‘roll out’ transactions

Content taken from:
http://www.cfengine.org/manuals/cf3-tutorial.html

System Management Cycle

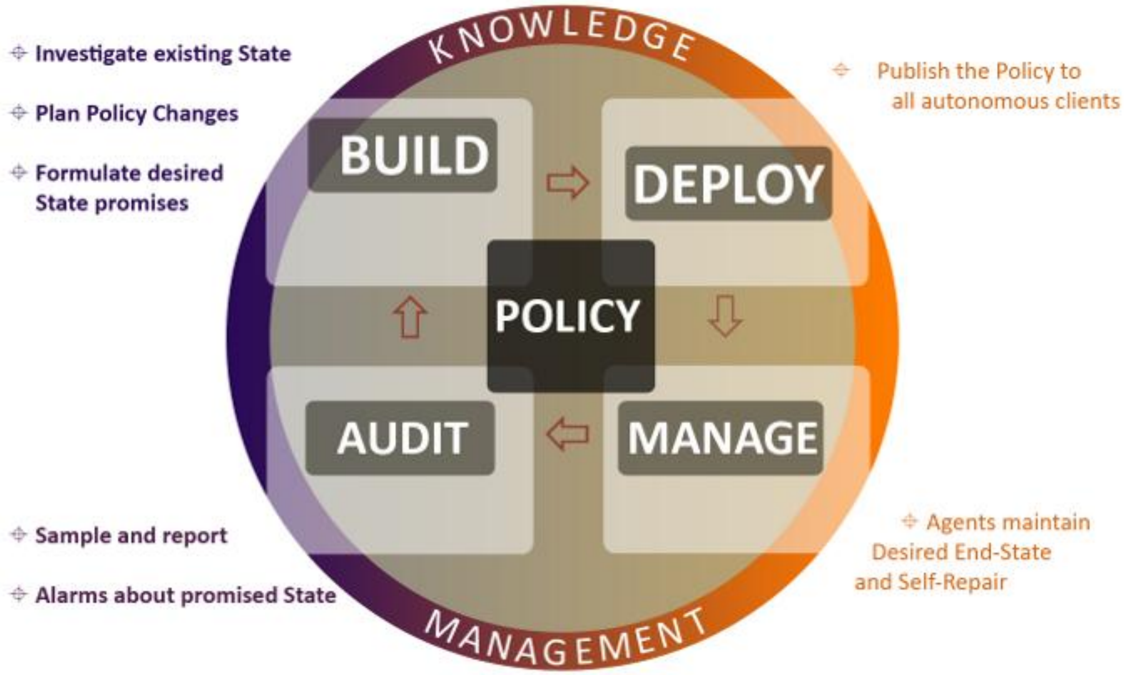

KNOWLEDGE
Investigate existing State
Plan Policy Changes
Formulate desired State promises
Publish the Policy to all autonomous clients
BUILD
DEPLOY
POLICY
AUDIT
MANAGE
Sample and report
Alarms about promised State
Agents maintain Desired End-State and Self-Repair
MANAGEMENT

Image taken from www.cfengine.org

CFengine's system management cycle is built around:

- ❖ Build
- ❖ Deploy
- ❖ Manage
- ❖ Audit

These four functions are centered around the various policies in place in an organization.

CFengine Installation (Dependencies)

In order to install CFengine, you should first ensure that the following packages are installed:

- OpenSSL Open source Secure Sockets Layer for encryption.
- BerkeleyDB (version 3.2 or later) Light-weight flat-file database system.

In addition...It is recommended to make the Perl Compatible Regular Expression (PCRE) library available as this is a significant improvement over the more standard POSIX libraries.

In order to run CFengine on Windows machines, you need to install the basic Cygwin DLL from: http://www.cygwin.com

Additional functionality becomes available if other libraries are present, e.g. OpenLDAP, client libraries for MySQL and PostgreSQL, etc. It is possible to run CFengine without these, but related functionality will be missing. Students should make sure that all of these items are installed for the various lab activities included with this lesson plan.

Recommended Links:

- OpenSSL
- BerkeleyDB
- Cygwin

CFengine Installation

Most popular Linux based systems have package support available that includes a package manager for both CFengine2 and CFengine 3 at this time. If your particular distribution does not have package management support, you can enter the following at the command line:

```
tar zxf CFengine-x.x.x.tar.gz
cd CFengine-x.x.x
./configure
make
make install
```

These commands will install binaries in **/usr/local/sbin**.
(Since this location is not necessarily a local file system on all hosts, users are encouraged to keep local copies of the binaries on each host, inside the CFengine trusted work directory.)

From the root command line in Debian based systems, you can also use the following command:

```
apt-get install cfengine3
```

Select **PLAY** below to review installing CFengine.

View Video VideoLesson7InstallCfengine3(C4L7S20).mp4

Key Authentication

CFengine agents authenticate with a server via key exchange.

The cf-key binary will create a public and private key pair. This is done for every server and client. For two hosts to authenticate, each must have a copy of the other's public key file. This exchange is normally done manually, but CFengine may be configured to do this one time only.

Please refer to the reference manual for more information.

Important Directories

Directories	Descriptions
/var/CFengine/bin	CFengine binaries
/var/CFengine/inputs	Main configuration files
/var/CFengine/ppkeys	Storage for authentication keys
/var/cf-masterfiles	The master files, on the server, that agents will request from the server
/var/cf-failsafe	A backup of important CFengine files to allow for automatic recovery

CFengine files are normally located in **/var/CFengine**. CFengine will create some directories automatically in this location. The two important ones that must be created by hand are **/var/CFengine/bin** and **/var/CFengine/inputs**. The bin directory contains the binary components listed earlier. This location allows CFengine to be more self-contained and fault tolerant. For example, the traditional location of **/usr/local/bin** is not always a local file system and therefore less reliable.

The inputs directory contains all of the configuration files that CFengine will use to maintain itself and the client hosts. The majority of work with CFengine will involve files located here. The mandatory files are **failsafe.cf**, **update.cf** and **promises.cf**.

Important Files

Files	Descriptions
promises.cf	This is the main configuration file. The agent will automatically start with this file.
update.cf	This is a simplified file whose purpose is to ensure the agent is configured properly so that it can do its job.
failsafe.cf	This file is run by the agent if other files are missing or contain errors. This gives the agent the ability to recover from failure.
cf-server.cf	This file configures the CFengine server. It can be named anything but choosing this name is logical.
cf-execd.cf	This file will configure the CFengine executor. Like cfserver.cf, this file could be named something else.
cfbackup.cf	This makes a local backup of CFengine to ensure the agent can recover from serious data loss.
crontabs.cf	This manages host crontables.
library.cf	This contains a collection of reusable code similar to a subroutine library.

CFengine Syntax

The syntax of CFengine files is relatively simple and somewhat Perl-like. However, CFengine tends to be more sensitive to white space:

- Sections are contained within brackets
- Commas separate parts of the same action
- Actions are ended with a semicolon
- Body part lines end with semicolons
- Variables are identified by $ and usually contained in brackets to separate them from surrounding text.
- Most user defined information is contained within double quotations
- Comments begin with # or can be included in the promise so that CFengine will print them during a run (comment => "My comment").

Syntax, Identifiers, & Names

The CFengine 3 language has a few simple rules:

- CFengine built-in words, and identifiers of your choosing (the names of variables, bundles, body templates and classes) may only contain the usual alphanumeric and underscore characters (a-z, A-Z, 0-9, and underscore (_).
- All other `literal' data must be quoted.
- Declarations of promise bundles in the form:

```
Bundle agent-type identifier
{
...
}
```

- Declarations of promise body-parts in the form:

```
body constraint_type template_identifier
{
...
}
```

matching and expanding on a reference inside a promise of the form ‘constraint_type => template_identifier’.

Continued . . .

Syntax, Identifiers, & Names

Continued from previous . . .

The CFengine 3 language has a few simple rules:

- CFengine uses many `constraint expressions' as part of the body of a promise. These take the form: left-hand-side (cfengine word) '=>' right-hand-side (user defined data). This can take several forms:

```
cfengine_word => user_defined_template(parameters)
                 user_defined_template
                 builtin_function()
                 "quoted literal scalar"
                 { list }
```

In each of these cases, the right hand side is a user choice.

Select **PLAY** below to review CFengine's code sample.

View Video
VideoLesson7CodeSample
(C4L7S26).mp4

Hello World Promise

Most programmers use the famous Hello World script to “program” their first language example. CFengine also has a *hello world* example:

```
body common control
{
        bundlesequence => { "test" };
}
bundle agent test
{
        reports:

                Yr2011::

                        "Hello world";

}
```

Reference
- CFengine Manual

Hello World Explanation

If you try to process the "hello world" program using the **cf-promises** command, you will see output similar to this:

```
atlas$ ~/portable/CFengine3/trunk/src/cf-promises -r -f
./unit_null_config.cf
```

```
Summarizing promises as text to ./unit_null_config.cf.txt
Summarizing promises as html to ./unit_null_config.cf.html
```

The `-r' option produces a report. Examine the files produced:

```
cat ./unit_null_config.cf.txt
firefox ./unit_null_config.cf.html
```

You also will see a summary of how CFengine interprets the files, either in HTML or text. By default, the CFengine components also dump a debugging file, e.g. `promise_output_agent.html', `promise_output_agent.txt' with an expanded view.

Select **PLAY** below to review CFengine's hello world program.

View Video VideoLesson7HelloWorld(C4L7S28).mp4

Log Files

promise_summary.log
A time-stamped log of the percentage fraction of promises kept after each run.

cf3.HOSTNAME.runlog
A time-stamped log of when each lock was released. This shows the last time each individual promise was verified.

cf_value.log
A time stamped log of the business value estimated from the execution of the automation system.

Lesson Summary

The lesson on CFengine was created to introduce Linux Administrators to a powerful remote system administrative tool. CFengine is a widely accepted and tested tool used by hundreds of large corporations around the world. CFengine has been developed and used for over 17 years and has a proven track record of being easily implemented and configured in the real world.

In this lesson, you were introduced to the basic elements involved with maintaining a Linux infrastructure using the CFengine tool. Specifically, you:

- Reviewed CFengine terminology
- Reviewed the installation processes of CFengine
- Reviewed log files of interest
- Reviewed default file locations
- Researched and explained a script
- Researched and implemented a script to add users
- Researched automated reporting using CFengine

Students who develop and demonstrate a basic understanding of CFengine and other remote administrative processes will maintain a competitive advantage in the world of IT and technology.