CFEngine Essentials

Vertical Sysadmin Training Examples

Aleksey Tsalolikhin, Vertical Sysadmin, Inc.

CFEngine Essentials: Vertical Sysadmin Training Examples

by Aleksey Tsalolikhin

Based on the works of Mark Burgess and CFEngine, Inc.

Chapter 1. About this collection

Vertical Sysadmin, a sysadmin training company and an authorized CFEngine training partner, presents "CFEngine Essentials", a collection of over 200 standalone working examples of using CFEngine 3 intended to get you up to speed faster with the exciting new technology that is CFEngine 3.

This collection:

- Supplements the official documentation, and
- Aids sysadmins in learning CFEngine 3.

This work is based on the materials of Mark Burgess and CFEngine, Inc.

Using this collection

The "CFEngine Essentials" course (formerly known as "Automating System Administration using CFEngine 3") is based on in-class demonstration of the following examples, complete with discussion and exercises.

To encourage and support study of CFEngine 3 for those unable to attend our class, we are making available all the examples and exercises.

Study the course materials in sequence. Check them out from GitHub and list them with the "l.sh" script. Try out and run the examples. Modify them and run them again. Write your own examples. Do the exercises.

Work your way through the materials until you understand it all and have done all the exercises. Make sure to look up unfamiliar terms in the Reference Manual.

Chapter 2. Using git

We keep these examples on GitHub and update them during each class as students ask questions or request additional demonstrations.

With git, you can download the updates right in class.

Installing git

RHEL 5

```
rpm -ihv http://dl.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm

yum install git
```

RHEL 6

```
yum install git
```

Debian

```
apt-get install git
```

Definition: Design Center

The CFEngine Design Center is a curated collection of policies and examples located at

`http://github.com/cfengine/design-center`

The collection comes from CFEngine staff and users and is curated by CFEngine staff.

The Vertical Sysadmin Training Examples are in the "examples" section of the Design Center.

Downloading examples

Download Aleksey's fork of the CFEngine Design Center repo:

```
git clone git://github.com/atsaloli/design-center.git
```

Go to the Vertical Sysadmin Training Examples directory:

```
cd design-center/examples/verticalsysadmin_training_examples
```

Updating examples

Pull in updates:

```
git pull
```

Installing CFEngine

Install the CFEngine package.

Running the examples

All of the examples are shipped as standalone CFEngine 3 files (*.cf) and are runnable:

```
cf-agent -KIb example -f ./example_115.cf
```

-K: "do it now" (over-rides time lock database)

-I: Inform me of any changes made to the system

-b: list of bundles to run (a bundle is a group of CFEngine policies)

-f: input filename (make sure to add the dot slash to specify the current directory)

Example:

```
cf-agent -KIb example -f \
./030-0005_Basic_Examples:_Files,_Processes,_Commands.__Files._Create_a_file.cf
```

Chapter 3. Syntax highlighting

The use of a syntax highlighter is strongly recommended to save time and trouble.

vi

You can install Neil Watson’s CFEngine 3 syntax highlighter using 010-0070-install-vim-plugin.sh, or follow the instructions in "Learning CFEngine 3" or http://www.cfengine.com/cfengine-code-editors

Emacs

See "Learning CFEngine 3" book or cfengine.com/cfengine-code-editors

Chapter 4. Class schedule for on-site training

8 - 4

08:00 AM - 09:30 AM Class

09:30 AM - 10:00 AM Morning break

10:00 AM - 11:30 AM Class

11:30 AM - 12:30 PM Lunch break

12:30 PM - 02:00 PM Class

02:00 PM - 02:30 PM Afternoon break

02:30 PM - 04:00 PM Class

9 - 5

09:00 AM - 10:30 AM Class

10:30 AM - 11:00 AM Morning break

11:00 AM - 12:30 PM Class

12:30 PM - 01:30 PM Lunch break

01:30 PM - 03:00 PM Class

03:00 PM - 03:30 PM Afternoon break

03:30 PM - 05:00 PM Class

10 - 6

10:00 AM - 11:30 AM Class

11:30 AM - 12:00 PM Morning break

12:00 PM - 01:30 PM Class

01:30 PM - 02:30 PM Lunch break

02:30 PM - 04:00 PM Class

04:00 PM - 04:30 PM Afternoon break

04:30 PM - 06:00 PM Class

Chapter 5. Introductory overview

At this point of the course, a brief introductory lecture is given introducing CFEngine and the idea of desired state management.

Chapter 6. Definitions

"Declarative" vs. "Imperative" Programming

> A declarative programming style … is often unfamiliar to newcomers, even if they are experienced programmers in other domains. Most commonly-used programming languages are examples of imperative programming, in which the programmer must describe a specific algorithm or process. Declarative programming instead focuses on describing the particular state or goal be be achieved.
>
> — Mike English *http://spin.atomicobject.com/2012/09/13/from-imperative-to-declarative-system-configuration-with-puppet/*

Examples

Make Me a Sandwich! (Imperative) Spread peanut butter on one slice of bread. Set this slice of bread on a plate, face-up. Spread jelly on another slice of bread. Place this second slice of bread on top of the first, face-down. Bring me the sandwich.

The Sandwich I Desire. (Declarative) There should be a sandwich on a plate in front of me… It should have only peanut butter and jelly between the two slices of bread.

> — Mike English *http://spin.atomicobject.com/2012/09/13/from-imperative-to-declarative-system-configuration-with-puppet/*

Declarative Programming for System Administration

> Declarative programming is a more natural fit for managing system configuration. We want to be talking about whether or not MySQL is installed on this machine or Apache on that machine, not whether yum install mysql-server has been run here or apt-get install apache2 there. It allows us to express intent more clearly in the code. It is also less tedious to write and can even be more portable to different platforms.
>
> — Mike English *http://spin.atomicobject.com/2012/09/13/from-imperative-to-declarative-system-configuration-with-puppet/*

Declarative has a higher Signal to Syntax Ratio

A declarative language allows us to express intent more clearly, to let the intent shine through the syntax of the code. It allows us to have a higher Signal to Syntax ratio.

Promise

A promise is a statement of intention.

Trust is an economic time-saver. If you can’t trust you have to verify, and that is expensive.

To improve trust we make promises. A promise is the documentation of an intention to act or behave in some manner. This is what we need to learn to trust systems.

CFEngine works on a simple notion of promises. Everything in CFEngine can be thought of as a promise to be kept by different resources in the system.

CFEngine manages every intended system outcome as "promises" to be kept.

Promises are always things that can be kept and repaired continuously, on a real time basis, not just once at install-time.

Combining promises with patterns to describe where and when promises should apply is what CFEngine is all about.

Policy

A policy is 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.

The Most Basic Form of a Promise

```
promise_type:

       "promiser"

            attribute1 => value1,
            attribute2 => value2;
```

Example simple promise - create a file

```
files:
```

```
  "/etc/nologin"

      create  => "true",
      comment => "Prevent non-root users from logging in so we can perform maintenance";
```

Basic promise types

files

A promise about a file, including its existence, attributes and contents.

packages

A promise to install (or remove or update or verify) a package.

processes

A promise concerning items in the system process table.

vars

A promise to be a variable, representing a value.

reports

A promise to report a message.

commands

A promise to execute a command.

Example of Promise Type

"files:" indicates the promise type.

```
files:

    "/etc/nologin"

        create  => "true",
        comment => "Prevent non-root users from logging in";
```

Promiser

The promiser is the part of the system that will be affected by the promise.

We are affected by the promises we make.

Example of Promiser

"/etc/nologin" is the promiser (the affected system object).

```
files:

    "/etc/nologin"

        create  => "true",
        comment => "Prevent non-root users from logging in";
```

Promise Body

First, let’s define "body".

Body

> The main part of a book or document, not including the introduction, notes, or appendices (parts added at the end).
>
> — Macmillan Dictionary

Examples of bodies: body of a letter, body of a contract.

The body is where the details are.

Promise body

A promise body is a collection of promise attributes that details and constrains the nature of the promise.

Example of Promise Body

The last three lines constitute the promise body.

```
files:

    "/var/cfengine/i_am_alive"

        create  => "true",
        touch   => "true",
        comment => "Prove CFEngine is running.";
```

Promise Bundle

The promise bundle is one of the basic building blocks of configuration in CFEngine.

A promise bundle is a group of one or more promises.

The bundle allows us to group related promises, and to refer to such groups by name.

We will some examples of promise bundles in the next chapter.

Chapter 7. Basic Examples: Files, Processes, Commands

EXAMPLE 1

Filename: 030-0005_Basic_Examples:_Files,_Processes,_Commands.__Files._Create_a_file.cf

```
bundle agent example {

  files:

      "/tmp/nologin"

               handle => "touch_etc_nologin",
              comment => "Quiesce the system for maintenance",
              create  => "true";

}
```

EXAMPLE 2

Filename: 030-0010_Basic_Examples:_Files,_Processes,_Commands.__Files._Touch_a_file.cf

```
bundle agent example {

  files:

      "/var/cfengine/i_am_alive" -> "Monitoring System"

               handle => "update_heartbeat",
              comment => "Update heartbeat timestamp to confirm CFEngine is running",
              create  => "true",
              touch   => "true";
}
```

EXERCISE - Disable FTP service by creating /etc/ftp.deny

Write a policy promising that /etc/ftp.deny is present, to disable FTP service.

EXAMPLE 3

Filename: 030-0020_Basic_Examples:_Files,_Processes,_Commands.__Processes._No_CUPSd_process._Stop_gracefully.cf

```
bundle agent example {

  processes:

      "cupsd"

               handle => "shutdown_print_services",
              comment => "Shutdown print services on machines
that are not print servers",
         process_stop => "/etc/init.d/cups stop";

}
```

EXAMPLE 4

Filename: 030-0021_Basic_Examples:_Files,_Processes,_Commands.__Processes._No_CUPSd_process._Stop_gracefully_and_Disable_Service.cf

```
bundle agent example {

  processes:

      "cupsd"

               handle => "stop_cupsd",
              comment => "Turn off and disable print services",
         process_stop => "/bin/sh -c '/etc/init.d/cups stop ; /sbin/chkconfig cups off'";

}
```

EXAMPLE 5

Filename: 030-0030_Basic_Examples:_Files,_Processes,_Commands.__Processes._No_IRC_bot_process._Terminate_with_prejudice.cf

```
bundle agent example {

  processes:

      "eggdrop"

               handle => "kill_IRC_bots",
              comment => "We don't want IRC bots on our Web servers",
              signals => { "term", "kill" };

}
```

EXERCISE - Kill a process

Write a policy to signal TERM and then KILL to any process matching "trn". (trn = Threaded Read News, a NetNews client.)

```
Testing it:

cp /bin/sleep ~/trn
~/trn 1000 &
cf-agent  --file ... --bundle ...
```

If you finish before everybody else, please study the Vocabulary Primer and then chapter 1-4 in the Reference Manual.

Writing CFEngine policies

1. State the sysadmin problem
2. Envision the desired end state
3. Translate the desired end state into CFEngine Policy Language

EXAMPLE 6

Filename: 030-0050_Basic_Examples:_Files,_Processes,_Commands.__Commands._date.cf

```
bundle agent example {

  commands:

      "/bin/date"

               handle => "run_date_cmd",
              comment => "Demonstrating running an external command";
}
```

EXAMPLE 7

Filename: 030-0060_Basic_Examples:_Files,_Processes,_Commands.__Commands._echo_hello_world.cf

```
bundle agent example
{

  commands:

      "/bin/echo"  -> "Class"

               handle => "say_hello",
              comment => "Demonstrate a command with arguments",
                 args => "Hello world!";

}
```

EXAMPLE 8

Filename: 030-0065_Basic_Examples:_Files,_Processes,_Commands.__Commands._Relative_path_does_not_work.cf

```
bundle agent example
{
```

```
  commands:

      "echo"
               handle => "insecure_promise",
              comment => "This fails because path to commands must be fully specified for security",
                 args => "Hello world";

}
```

EXAMPLE 9

Filename: 030-0070_Basic_Examples:_Files,_Processes,_Commands.__Commands._Dig_a_deep_hole.cf

```
# demonstrate handling of multi-line output

bundle agent example
{

  commands:

      "/bin/sh -c '/bin/mkdir  -p /usr/local/sbin/a/really/long/path/to; /bin/cp -p /usr/bin/printf /usr/local/sbin/a/really/long/
               handle => "dig_a_deep_hole",
              comment => "Create an executable with a long path name - we'll need
it for the next example";

}
```

EXAMPLE 10

Filename: 030-0075_Basic_Examples:_Files,_Processes,_Commands.__Commands._Quoted_multiline_output.cf

```
# demonstrate handling of multi-line output

bundle agent example
{

  commands:

      "/usr/local/sbin/a/really/long/path/to/printf \"Line 1 of multiline output\nLine 2 of multiline output\nLine 3 of multiline
```

```
             handle => "run_a_command_from_a_deep_hole",
            comment => "Demonstrate how CFEngine truncates path names in command output";
}
```

Chapter 8. Notes on Syntax

A bundle is a group of one or more promises. We will show you two promises in one bundle.

File can hold one or more bundles. We will show you two bundles in one file.

Whitespace does not matter. (Extra whitespace is for humans, not for the parser.)

EXAMPLE 11

Filename: 035-0040_Notes_on_Syntax.__Two_promises_in_one_bundle.cf

```
bundle agent example {

  files:

      "/tmp/hello"

               handle => "make_file_1",
              comment => "Demonstrate a bundle with multiple promises.",
              create  => "true";


  files:

      "/tmp/world"

               handle => "make_file_2",
              comment => "Demonstrate a bundle with multiple promises.",
              create  => "true";

}
```

EXAMPLE 12

Filename: 035-0045_Notes_on_syntax.__Two_promises_in_one_bundle._Condensed.cf

```
bundle agent example {

  files:

      "/tmp/hello" create  => "true";

      "/tmp/world" create  => "true";
}
```

EXAMPLE 13

Filename: 035-0046_Notes_on_Syntax.__Two_bundles_in_one_file.cf

```
bundle agent example1 {

      files: "/tmp/file1" create  => "true";

}

bundle agent example2 {

      files: "/tmp/file2" create  => "true";

}
```

EXAMPLE 14

Filename: 035-0150_Notes_on_Syntax.__Whitespace_and_indentation_do_not_matter.cf

```
# Whitespace/ndentation does not matter, these two bundles are identical.

bundle agent example {

  files:

      "/etc/nologin"

               handle => "touch_etc_nologin",
              comment => "Prepare for system maintenance",
```

```
                create => "true";

}


bundle agent example2 { files: "/etc/nologin" handle => "touch_etc_nologin", comment => "Prepare for system maintenance", create =
```

Chapter 9. Notes on Running CFEngine

EXAMPLE 15

Filename: 037-0010_Notes_on_Running.__Three_passes_through_a_bundle.cf

```
# demonstrate three passes through a bundle by using verbose mode

bundle agent example {

  files:

      "/etc/nologin"

               handle => "touch_etc_nologin",
              comment => "Quiesce the system for maintenance",
              create  => "true";

}
```

Chapter 10. Basic Examples: Classes and Reports

Classes

The Real Basic Form of a Promise

Each promise includes a statement of context, where is that promise applicable, on what class of machines, where and when?

```
promise_type:

   context::

      "promiser"

           attribute1 => value1,
           attribute2 => value2;
```

Example

```
files:

  linux&Sunday::

    "/etc/nologin"

              handle => "disable_nonroot_logins",
             comment => "Take a day off, don't work on Sunday!",
              create => "true";
```

EXAMPLE 16

Filename: 039-0010_Basic_Examples:_Classes_and_Reports.__context.cf

```
bundle agent example {

  files:
```

```
    Sunday&Hr02::

      "/etc/nologin"

               handle => "touch_etc_nologin",
              comment => "Quiesce the system for 2 - 4 A.M. Sunday maintenance",
              create  => "true";

}
```

EXAMPLE 17

Filename: 039-0080_Basic_Examples:_Classes_and_Reports.__hard-class.cf

```
bundle agent example{

  reports:
    linux::
      "I love Linux"

               handle => "report_Linux_love",
              comment => "Check that we are on Linux to demonstrate classes and reports";


  reports:
    WinXP::
      "I love Windows"

               handle => "report_Windows_love",
              comment => "Demonstrate classes";

}
```

EXAMPLE 18

Filename: 039-0085_Basic_Examples:_Classes_and_Reports.__soft-class.cf

```
bundle agent example{
```

```
      # Demonstrate normal ordering and multiple passes through a bundle
  classes:
      "file_exists"
               handle => "create_soft_class",
              comment => "Create a soft class that will be used by reports: promises.",
           expression => fileexists("/tmp/newfile");

  files:
      "/tmp/newfile"

               handle => "create_a_file",
              comment => "Give CFEngine something to do to change system state.",
               create => "true";
  reports:
    file_exists::
      "file /tmp/newfile exists"
               handle => "report_success",
              comment => "Report existence of /tmp/newfile";

  reports:
    !file_exists::
      "file /tmp/newfile does not exist"
               handle => "report_failure",
              comment => "Report lack of existence of /tmp/newfile";
}
```

EXERCISE

Create two files.

Make the existence of the 2nd file conditional on existence of the 1st file.

EXAMPLE 19

Filename: 045-0200_Patterns:_Classes.__Hello_world_report_of_OS_type.cf

```
bundle agent example {
```

```
  reports:

      WinXP:: "Hello world! I am running on a Windows system.";

      linux:: "Hello world! I am running on a Linux system.";

      redhat:: "Hello world! I am running on a redhat Linux system.";
}
```

Chapter 11. Patterns

The CFEngine 3 language can be described by the following equation:

Patterns + Promises = Configuration

We are going to look at some of the patterns in the CFEngine 3 language.

The first of these is "classes".

Class Operators

```
!       negate (not)
&  .     and
| ||     or
( )   groupers
```

EXAMPLE 20

Filename: 045-0202_Patterns:_Classes.__Class_expression_operators.cf

```
bundle agent example {

  reports:

      linux:: "I am running on a linux system.";

      WinXP:: "I am running on a Windows system.";

      !WinXP:: "Thank goodness.";

      WinXP|linux::  "Am I laughing or crying?";
```

```
      WinXP&linux::  "We should never see this report.";

}
```

EXAMPLE 21

Filename: 045-0210_Patterns:_Classes.__Report_day_of_the_week.cf

```
bundle agent example {

      reports: Monday::    "Hello world! I love Mondays!";

      reports: Tuesday::   "Hello world! I love Tuesdays!";

      reports: Wednesday:: "Hello world! I love Wednesdays!";

      reports: Thursday::  "Hello world! I love Thursdays!";

      reports: Friday::    "Hello world! I love Fridays!";

      reports: Saturday::  "Hello world! I love weekends!";

      reports: Sunday::    "Hello world! I love weekends!";
}
```

EXAMPLE 22

Filename: 045-0211_Patterns:_Classes.__Condensed_report_day_of_week.cf

```
bundle agent example {

  reports:

    Monday|Tuesday|Wednesday|Thursday|Friday::
      "Yay!!! I get to work today!"
               handle => "report_a_weekday",
              comment => "Identify work days";

    Saturday|Sunday::
```

```
     "Yay!!! I get to rest today."
              handle => "report_a_rest_day",
             comment => "Identify rest days";

}
```

EXAMPLE 23

Filename: 045-0220_Patterns:_Classes.__OS_and_time_expression.cf

```
bundle agent example {

  reports:

      linux.Hr08:: "Linux system AND we are in the 8th hour.";
      linux.Hr11:: "Linux system AND we are in the 11th hour.";
      linux.Hr12:: "Linux system AND we are in the 12th hour.";

      linux.Hr13:: "Linux system AND we are in the 13th hour.";
      linux.Hr16:: "Linux system AND we are in the 16th hour.";


      linux&Hr22:: "Linux system AND we are in the 22nd hour.";

      linux&Hr20:: "Linux system AND we are in the 20th hour.";
}
```

EXAMPLE 24

Filename: 045-0230_Patterns:_Classes.__Class_expression.OS_and_time.cf

```
bundle agent example {

  reports:

    (redhat&Monday)|(windows&Wednesday)::

      "This promise will execute on Redhat servers on Mondays; or on Windows servers on Wednesdays";
```

```
}
```

EXAMPLE 25

Filename: 045-0240_Patterns:_Classes.__Using_classes_to_determine_role.cf

```
bundle agent example {

  reports:
    ipv4_205_186_156::
      "I am on our public net. I'll be a Web server."
               handle => "configure_webserver";

  reports:
    ipv4_10::
      "I am on our private net. I'll be a database server."
               handle => "configure_db_server";


}
```

EXAMPLE 26

Filename: 045-0250_Patterns:_Classes.__Ensuring_CUPSd_is_running.cf

```
bundle agent example {

  processes:

      "cupsd"

        restart_class => "cups_needs_to_be_started",
              comment => "We DO want print services";


  commands:

    cups_needs_to_be_started::
```

```
      "/etc/init.d/cups"

                args => "start";


}
```

EXAMPLE 27

Filename: 045-0251_Patterns:_Classes.__Ensuring_httpd_is_running.cf

```
bundle agent example {

  processes:
      "httpd"
        restart_class => "start_httpd";

  commands:
    start_httpd::
      "/etc/init.d/httpd start";

}
```

EXAMPLE 28

Filename: 045-0252_Patterns:_Classes.__logical_not.cf

```
bundle agent example {

  reports:

    linux::

      "Yay Linux!";


  reports:

    !linux::
```

```
      "I miss my Linux...";

  reports:

    !windows::

      "Thank goodness it ain't Windows.";
}
```

EXAMPLE 29

Filename: 045-0260_Patterns:_Classes.__Am_I_real_or_virtual.cf

```
bundle agent example {

  classes:
      "i_am_virtual"
               handle => "reality_check",
              comment => "Check if we are running inside a VM",
           expression => regline("(?i).*vmware.*|.*(?i)vbox.*|(?i).*qemu.*", "/proc/scsi/scsi");
      # (?i) adds the "case-insensitive" switch
      # it's the native version of /bin/egrep -i 'vmware|vbox|qemu' /proc/scsi/scsi


  reports:
    i_am_virtual::
      "Running inside a VM";

}
```

EXAMPLE 30

Filename: 045-0260_Patterns:_Classes.__Note_on_what_happens_to_dashes_in_hostnames.cf

```
# set hostname to "my-hostname-has-dashes"

bundle agent example {
```

```
  reports:

    my_hostname_has_dashes::

      "Boo!";

}
```

Chapter 12. Basic Examples: Vars

Variables.

EXAMPLE 31

Filename: 048-0010_Basic_Examples:_Vars.__var.cf

```
bundle agent example {


  vars:
      "my_string"
               handle => "declare_string_var",
              comment => "Demonstrate declaring var type and value",
              string  => "The answer is: ";

      "my_int"
               handle => "declare_int_var",
              comment => "Demonstrate declaring var type and value",
              int     => "42";


  commands:

      "/bin/echo $(my_string)  $(my_int)";
}
```

EXAMPLE 32

Filename: 048-0020_Basic_Examples:_Vars.__typing.cf

```
bundle agent example {

  vars:

      "my_int"
```

```
              handle => "shove_cfengine",
             comment => "Try to assign a real number to an integer variable to see what happens",
                 int => "1.5";

  commands:

      "/bin/echo my_int is $(my_int).";

}
```

EXAMPLE 33

Filename: 048-0030_Basic_Examples:_Vars.__typing_2.cf

```
bundle agent example {

  vars:

      "my_int1"

               handle => "poke_cfengine_with_a_stick",
             comment => "Try to assign a string to an integer variable to see what happens",
                 int => "hello world";


  commands:

      "/bin/echo my int is $(my_int)";

}
```

Chapter 13. Notes on Running

Notes on running CFEngine.

EXAMPLE 34

Filename: 050-0051_Notes_on_Running.__normal_ordering.cf

```
# Demonstrate normal ordering

bundle agent example{

  classes:
      "myclass"
               handle => "create_soft_class",
              comment => "Create a soft class that will be used by reports: promises.",
           expression => fileexists("/tmp/newfile");


  files:
      "/tmp/newfile"

               handle => "create_a_file",
              comment => "Give CFEngine something to do to change system state.",
               create => "true";

  reports:
    myclass::
      "file /tmp/newfile exists";

  reports:
    !myclass::
      "file /tmp/newfile does not exist";
}
```

EXAMPLE 35

Filename: 050-0051_Notes_on_Running.*normal_ordering*create_file_only.cf

```
bundle agent example {

  commands:
      "/bin/echo hello world";


  files:
      "/tmp/newfile"
               create => "true",
              comment => "file created";

}
```

Run the previous example without and with the -K switch (timelock override).

Chapter 14. Knowledge Management

Knowledge Management is one of the key challenges of scale.

EXAMPLE 36

Filename: 070-0151_Knowledge_Management__handle.cf

```
bundle agent example {

  files:

      "/tmp/testfile"

               handle => "create_testfile", # a name for this promise.

      # can be used with depends_on
      # attribute in another promise
      # to document dependency

              create  => "true";
}
```

EXAMPLE 37

Filename: 070-0152_Knowledge_Management__depends_on.cf

```
# Demonstrate how depends_on controls policy flow.
# run in verbose mode.

bundle agent example
{

  files:

      "/tmp/testfile"
           depends_on => { "run_command", "print_report" },
               handle => "create_file",
```

```
              comment => "Demonstrate flow control with depends_on",
               create => "true";


}

bundle agent example2 {

  commands:

      "/adsfbin/echo" -> { "create_file", "Director of Security" }

               handle => "run_command";

  reports:

    linux::

      "Just a report"
               handle => "print_report";
}
```

EXAMPLE 38

Filename: 070-0153_Knowledge_Management__comment.cf

```
# run this in verbose mode and notice the comment

bundle agent example {

  files:

      "/tmp/testfile"


               handle => "mk_file",
              comment => "Create a vital file, needed for XYZ.",
              create  => "true";
}
```

EXAMPLE 39

Filename: 070-0154_Knowledge_Management__comment_with_file_name_and_line_number.cf

```
bundle agent example {

  files:

      "/tmp/testfile"


               handle => "demo_special_variables_in_comment",
              comment => "XYZ needs /tmp/testfile so make it.
Line $(this.promise_linenumber) in $(this.promise_filename)
",
      # this comment will show up in verbose or debug modes

              create  => "true";
}
```

EXAMPLE 40

Filename: 070-0155_Knowledge_Management__promisee.cf

```
bundle agent example {

  files:

      "/etc/httpd/conf/httpd.conf" -> "Web Services team",
      # document stakeholders

              create  => "true";
}
```

Dunbar numbers

Robin Dunbar pointed out that there are limits to human cognition: - We can only have a close relationship to about 5 things. - We can have a working relationship with about 30 things or people. - We can only be acquainted with about 150.

The ‘Dunbar numbers’ are cognitive limits that we have to work around.

http://cfengine.com/markburgess/blog_km.html

Chapter 15. Editing Files

EXAMPLE 41

Filename: 080-0160_Editing_Files.__insert_lines.cf

```
bundle agent example {

  files:

      "/etc/motd"

              handle  => "files_motd",
             comment  => "Create a nice motd",
              create  => "true",
            edit_line => greet_users;
}

#######################################################

bundle edit_line  greet_users {

  insert_lines:

      "Good morning!"
               handle => "greet_user",
              comment => "Happy users = less complaints. Greet the user
politely.";
}
```

EXERCISE

Write a policy that will ensure /etc/motd always contains:

```
Unauthorized use forbidden.
```

If you finish before the rest of the class, please study chapters 1-4 in the Reference Manual.

EXAMPLE 42

Filename: 080-0165_Editing_Files.__delete_lines.cf

```
bundle agent example {

  files:

      "/etc/motd"

               handle => "motd",
              comment => "Create and populate motd",
               create => "true",
            edit_line => proper_greetings;
}


bundle edit_line  proper_greetings {


  delete_lines:

      ".*"
               handle => "empty_entire_file_please",
              comment => "Let's start with a clean slate.";

  insert_lines:

      "Good morning!"
               handle => "greet_user",
              comment => "Greet the user politely.";

}
```

Introducing templates

What are templates? why would we use templates?

What follows is an introduction to template for sysadmins that haven't worked with templates before.

Example 1. Email:

```
Hello __NAME__,

  Please buy our product.

Love,
Company
```

Example 2. Config file:

MASTER_MAP_NAME="auto.master" TIMEOUT=300 BROWSE_MODE="yes" LOGGING="verbose" MAP_OBJECT_CLASS="automountMap" ENTRY_OBJECT_CLASS="automount" MAP_ATTRIBUTE="ou" ENTRY_ATTRIBUTE="cn" VALUE_ATTRIBUTE="automountInformation" USE_MISC_DEVICE="yes" SEARCH_BASE="ou=*SITE*,ou=Sites,dc=VerticalSysadmin,dc=com"

EXAMPLE 43

Filename: 080-0167_Editing_files.__insert_type_file.cf

```
bundle agent example {

  files:

      "/etc/motd"

               handle => "motd",
              comment => "Generate /etc/motd from static template (no expansion)",
            edit_line => insert_stock_motd;
}
```

```
bundle edit_line insert_stock_motd {

  delete_lines:

      ".*";

  insert_lines:

      "/var/cfengine/inputs/templates/motd.txt"

               handle => "motd_contents",
              comment => "insert stock MOTD from static template",
          insert_type => "file";
}
```

EXAMPLE 44

Filename: 080-0168_Editing_Files.__Expand_from_template.cf

```
bundle agent example {

  files:

      "/etc/motd"

            edit_line => InsertFile("/var/cfengine/inputs/templates/motd.txt");
}

bundle edit_line InsertFile(source) {

  insert_lines:

      "$(source)"

          insert_type => "file";
}
```

EXAMPLE 45

Filename: 080-0169_Editing_Files.__Expand_from_template_and_expand_scalars.cf

```
bundle agent example {

  files:

      "/etc/motd"

               handle => "expand_motd_from_template",
              comment => "Demonstrate variable expansion on templated content",
            edit_line => ExpandMeFrom("/var/cfengine/inputs/templates/motd.txt");
}

bundle edit_line ExpandMeFrom(source) {

  delete_lines:
      ".*"
               handle => "empty_entire_file";

  insert_lines:
      "$(source)"
           handle     => "insert_file_from_template_with_var_expansion",
          insert_type => "file",
       expand_scalars => "true";
}
```

Scope of variables

Note: a fully qualified variable consists of the bundle name wherein the variable is defined plus the variable name. Example:

```
bundle agent mybundle { vars: "myvar" string => "myvalue"; }
```

$(myvar) #-- unqualified

$(mybundle.myvar) #-- fully qualified (complete with scope)

EXAMPLE 46

Filename: 080-0171_Editing_Files.__Demo-of-variable-scope.cf

```
bundle agent example {

  vars:

      "var1"

              handle  =>  "declare_var1",
              comment =>  "Declare string variable var1 to demonstrate variable scope",
              string  =>  "Hello World";

}

bundle agent example2 {

  commands:

      "/bin/echo"

              handle  =>  "use_var_1",
             comment  =>  "Use variable declared in another bundle to demonstrate scope",
      #args  =>  "$(var1)";
                 args  =>  "$(example.var1)";

}
```

EXAMPLE 47

Filename: 080-0172b-example.cf

```
bundle agent example {

  vars:

      "first_name"
               string => "Ed";

  files:
```

```
      "/tmp/letter.txt"

            handle    => "write_letter",
            create    => "true",
            edit_line => create_from_template;
}

bundle edit_line create_from_template {

  insert_lines:

      "/var/cfengine/inputs/templates/letter.txt"

               handle => "insert_expanded_template",
          insert_type => "file",
       expand_scalars => "true";
}
```

EXERCISE

Purpose: practice editing file content using a template containing scalar variables.

1. Manually create a template containing:

Hello $(example.first_name),

```
Please buy our product.
```

Love, Company

1. Populate contents of /tmp/letter.txt using the above template and the variable "first_name" defined in a bundle "example".

Gotchas:

- Make sure your bundle name matches the bundle name in the variable in the template.

- Make sure your bundle defines the variable embedded in the template.

If you finish before rest of the group, finish studying the CFEngine Reference Manual chapters 1 -4, and if you finish that, then study the Special Topic guide on Editing File Content.

EXAMPLE 48

Filename: 080-0173_Editing_Files.__Expand_from_template_with_expand_scalars_with_delete_lines.cf

```
bundle agent example {

  files:

      "/etc/motd"

               handle => "my_motd",
              comment => "Create my motd from template -- clear file first",
               create => "true",
            edit_line => ExpandFromTemplate("/var/cfengine/inputs/templates/motd.txt");
}

bundle edit_line ExpandFromTemplate(source) {

  delete_lines:

      ".*";

  insert_lines:

      "$(source)"

         insert_type => "file",
       expand_scalars => "true";
}
```

EXAMPLE 49

Filename: 080-0174_Editing_Files.__add_a_group.cf

```
# Make sure /etc/group contains an entry for
# a "cfengine" group, GID 502

bundle agent example {

  files:

      "/etc/group"

               handle => "group_presence",
              comment => "Ensure CFEngine group is present",
            edit_line => cfengine_group_is_present;

}


bundle edit_line cfengine_group_is_present {

  insert_lines:

      "cfengine:x:502:"

               handle => "cfengine_group_entry",
              comment => "Arbitrary group entry in /etc/group";

}
```

EXAMPLE 50

Filename: 080-0175_Editing_Files.__Removing_the_games_group_from_etc_group_file.cf

```
bundle agent example {

  files:

      "/etc/group"

               handle => "delete_a_group",
              comment => "delete a specified /etc/group entry",
```

```
            edit_line => delete_group("games:x:[0-9]+:");
      # note the parameter
      # you can parameterize bundles
}


bundle edit_line delete_group(group) {

  delete_lines:
      "$(group).*";
}
```

EXAMPLE 51

Filename: 080-2110_Editing_Files.__Configure_autofs_to_use_LDAP.cf

```
bundle agent example {

  vars:

      "site"
               string => readfile( "/etc/site" , "3" );
      # site is a 3 char site code.
      # E.g., mil for Milan, war for Warsaw, etc.

  files:
      "/etc/sysconfig/autofs"
            edit_line => configure("$(site)"),
               create => "true";
}

bundle edit_line configure(site) {

  delete_lines:
      ".*";

  insert_lines:
      "MASTER_MAP_NAME=\"auto.master\"
TIMEOUT=300
```

```
BROWSE_MODE=\"yes\"
LOGGING=\"verbose\"
MAP_OBJECT_CLASS=\"automountMap\"
ENTRY_OBJECT_CLASS=\"automount\"
MAP_ATTRIBUTE=\"ou\"
ENTRY_ATTRIBUTE=\"cn\"
VALUE_ATTRIBUTE=\"automountInformation\"
USE_MISC_DEVICE=\"yes\"
SEARCH_BASE=\"ou=$(site),ou=Sites,dc=VerticalSysadmin,dc=com\"";

}
```

EXAMPLE 52

Filename: 080-2120_Editing_Files.__Configure_autofs._Using_a_template.cf

```
# This file controls the contents of /etc/sysconfig/autofs
# using the template /templates/autofs.tmpl
#
# contents of /templates/autofs.tmpl:
# MASTER_MAP_NAME="auto.master"
# TIMEOUT=300
# BROWSE_MODE="yes"
# LOGGING="verbose"
# MAP_OBJECT_CLASS="automountMap"
# ENTRY_OBJECT_CLASS="automount"
# MAP_ATTRIBUTE="ou"
# ENTRY_ATTRIBUTE="cn"
# VALUE_ATTRIBUTE="automountInformation"
# USE_MISC_DEVICE="yes"
# SEARCH_BASE="ou=$(site),ou=Sites,dc=VerticalSysadmin,dc=com"
#
# and /etc/site which contains, for example, lax

bundle agent example {

  vars:

      "site"
```

```
               string => readfile( "/etc/site" , "3" );

  files:
      "/etc/sysconfig/autofs"
            edit_line => expand_from_autofs_template("$(site)"),
               create => "true";
}


#############################################################

bundle edit_line expand_from_autofs_template(site) {

  delete_lines:
      ".*";

      # /templates/autofs.tmpl

  insert_lines:

      "/var/cfengine/inputs/templates/autofs.txt"
          insert_type => "file",                # insert copy of template
       expand_scalars => "true";             # expand scalars

}

#############################################################
```

Chapter 16. Body Parts: Introductions

body type name {

```
    attribute1 => value1;
    attribute2 => value2;
    ...
    attributeN => valueN;
}
```

EXAMPLE 53

Filename: 110-0380_Body-Parts.__No_world_write_bit.cf

```
bundle agent example {

  files:

      "/tmp/testfile"

              comment => "/tmp/testfile must not be world-writable",
              perms   => not_world_writable;

}


############################################################

body perms not_world_writable
{
               mode   => "o-w";
}
```

EXAMPLE 54

Filename: 110-0381_Body_Parts.__perms.cf

```
bundle agent example {

  files:

      "/tmp/testfile"

                perms => not_world_writable_and_right_group;

}

body perms not_world_writable_and_right_group {

               groups => {"root", "games", "mail" };
               mode   => "o-w";

}
```

EXAMPLE 55

Filename: 110-0382_Body_Parts.__perms.cf

```
bundle agent example {

  files:

      "/tmp/bobsfile"

              create  => "true",
              comment => "Set mode, ownership and group on /tmp/bobsfile",
              perms   => set_file_attributes("777", "bob", "mail");

      "/tmp/susansfile"

              create  => "true",
              comment => "Set mode, ownership and group on /tmp/susansfile",
              perms   => set_file_attributes("000", "susan", "games");
}

#########################################################
```

```
body perms set_file_attributes(mode,owner,group)
{
             mode    => "$(mode)";
             owners  =>  {"$(owner)"};
             groups  =>  {"$(group)"};
}
```

EXAMPLE 56

Filename: 110-0385_Body-Parts.__Remove_a_directory.cf

```
bundle agent example {

  files:

      "/var/logexample/.*"
               handle => "delete_old_logs",
              comment => "Delete files older than specified threshold",
          file_select => days_old("2"),
               delete => tidy;
}

###################################################

body file_select days_old(days)
{
          mtime       => irange(ago(0,0,"$(days)",0,0,0) , now);
          file_result => "!mtime";
}

body delete tidy {

             dirlinks => "delete";  # what to do with Symlinks To Directories
             rmdirs   => "true";    # what to do with Empty Directories

}
```

EXAMPLE 57

Filename: 110-0386_Body-Parts.__Install_a_package.cf

```
bundle agent example {

  packages:

      "php-mysql"

       handle         => "install_php_mysql",
       comment        => "Install a package",
       package_policy => "add", # Ensure that a package is present
       package_method => yum;
}

###############################################################


body common control {

               inputs => { "libraries/cfengine_stdlib.cf"};

}
```

EXAMPLE 58

Filename: 110-0387_Body-Parts.__Remove_a_package.cf

```
bundle agent example {

  packages:

      "php-mysql"

       handle         => "remove_package",
       comment        => "Demonstrate removing a package",
       package_policy => "delete", # Ensure a package is absent
package_architectures => { "x86_64" },
       package_method => yum;
}
```

```
#################################################################



body package_method yum
{
      package_changes => "bulk";
 package_list_command => "/usr/bin/yum --quiet list installed";
package_patch_list_command => "/usr/bin/yum --quiet check-update";

      # Remember to escape special characters like |

package_list_name_regex    => "([^.]+).*";
package_list_version_regex => "[^\s]\s+([^\s]+).*";
package_list_arch_regex    => "[^.]+\.([^\s]+).*";

package_installed_regex => ".*(installed|\s+@).*";
package_name_convention => "$(name).$(arch)";

      # set it to "0" to avoid caching of list during upgrade
package_list_update_command => "/usr/bin/yum --quiet check-update";
package_list_update_ifelapsed => "240";

package_patch_installed_regex => "^\s.*";
package_patch_name_regex    => "([^.]+).*";
package_patch_version_regex => "[^\s]\s+([^\s]+).*";
package_patch_arch_regex    => "[^.]+\.([^\s]+).*";

  package_add_command => "/usr/bin/yum -y install";
package_update_command => "/usr/bin/yum -y update";
package_patch_command => "/usr/bin/yum -y update";
package_delete_command => "/bin/rpm -e --nodeps";
package_verify_command => "/bin/rpm -V";
}
```

EXERCISE

Create executable shell script

Write a CFEngine policy to ensure /usr/local/bin/helloworld exists, has permissions 0755, owner root, group root, and contents:

#!/bin/sh

/bin/echo hello world

EXAMPLE 59

Filename: 110-0388_Body-Parts.__Setting_group_ownership_based_on_OS.cf

```
# Two bundles sharing a body-part that automagically sets
# the correct group ownership based on OS

bundle agent example1 {

  files:

      "/tmp/testfile"
              handle  => "set_file_attributes_automagically",
             comment  => "Set appropriate file attributes everywhere",
              create  => "true",
              perms   => set_mode_700_admin_group_and_specified_user("aleksey");
}


#################################################

bundle agent example2 {

  files:

      "/tmp/testfile2"
              handle  => "set_file_attributes_on_file2",
```

```
             comment  => "Set appropriate file attributes everywhere",
              create  => "true",
              perms   => set_mode_700_admin_group_and_specified_user("rob");
}


################################################


body perms set_mode_700_admin_group_and_specified_user(xyz) {

               mode   => "0700";

               owners => { "$(xyz)" };

      linux::  groups => { "wheel" };
      darwin:: groups => { "admin" };
      sunos::  groups => { "sys" };

}
```

EXAMPLE 60

Filename: 110-0389_Body-Parts.__Replacing_Patterns.cf

```
bundle agent example {

  files:

      "/tmp/data.txt"
            handle    => "turn_dogs_into_cats",
            comment   => "Demonstrate search-and-replace in a file",
            edit_line => transform_dogs_to_cats;

}

bundle edit_line transform_dogs_to_cats {

  replace_patterns:
```

```
      "[Dd]og"

         handle       => "replace_dog_with_cat",
         comment      => "Demonstrate replace_patterns promise",
         replace_with => value("cat");

}

body replace_with value(x)
{
        replace_value => "$(x)";
          occurrences => "all";
}
```

Chapter 17. CFEngine Running Unattended

CFEngine Running Unattended

EXAMPLE 61

Filename: 115-0270_CFEngine_Running_Unattended__demo.cf

```
# Bootstrap CFEngine on your machine to itself

cf-agent --bootstrap -s your.ip.address

cf-agent
```

EXERCISE

Running CFEngine Non-Interactive (as a Service) using promises.cf as input.

1. Put your "motd" exercise file under /var/cfengine/masterfiles/services/
2. Edit /var/cfengine/masterfiles/promises.cf:
   - Add your "motd" exercise file name to "inputs" list
   - Add your "motd" exercise bundle name to "bundlesequence" list
3. Now let’s test it:

```
> /etc/motd               # make a repair necessary
cf-agent -f update.cf -I  # update from "masterfiles" to "inputs"
cf-agent -I               # make sure /etc/motd is repaired
```

4. Remove your /etc/motd and watch CFEngine create /etc/motd within 5 minutes:

```
  watch ls -l /etc/motd 2>/dev/null
```

```
Change cf-execd schedule from 5 min to 1 min.
1. in /var/cfengine/masterfiles/controls/cf_execd.cf, set
     schedule => { "any" };
cf-agent -f update.cf -I
/etc/init.d/cfengine3 restart


Setting up client-server (or, client-server bootstrap)
```

Wipe the slate clean (on both client and server):

```
rpm -e cfengine-community
rm -rf /var/cfengine/

rpm -i cfengine*rpm
```

On the server:

Edit /var/cfengine/masterfiles/def.cf to add client and server IPs to the "acl" list.

On the server, configure the firewall to allow access on cfengine port 5308:

```
/etc/init.d/iptables stop
chkconfig iptables off  # for training only, not production!!
```

Install CFEngine RPM on the client.

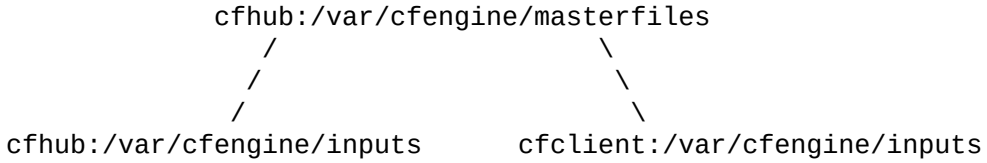
On the server:

```
cf-agent --bootstrap -s 10.1.1.10
cf-agent -f update.cf -IK
```

On the client:

```
cf-agent --bootstrap -s 10.1.1.10
cf-agent -f update.cf -IK
```

Policy Flow Diagram



Chapter 18. CFEngine Grammar: LHS vs RHS

Left-hand side $\Rightarrow$ Right Hand Side

$\Rightarrow$ can be pronounced as "rocket"

It serves to separate LHS from RHS and to differentiate CF3 syntax from CF2, which used the equals sign =

Promise attributes

CFEngine uses many "constraint expressions" as part of the body of a promise. These are attributes of a promise, they detail and constrain the promise.

These take the form:

left-hand-side (cfengine word) $\Rightarrow$ right-hand-side (user defined data).

This can take several forms:

```
cfengine_word => user_defined_body or user_defined_body(parameters)
builtin_function()
"scalar_value" or "$(scalar_variable_name)"
{ "list_element", "list_element2" }
{ @(list_variable_name) }
```

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

EXAMPLE 62

Filename: 120-0420_Cfengine_Grammar:_LHS_vs_RHS.__Example_of_user_defined_body_on_rhs.cf

```
# example of:     cfengine_word => user_defined_body
```

```
bundle agent example
{
  storage:

      "/"

              volume  => my_check_volume;

}


body volume my_check_volume
{
       freespace      => "30%"; # minimum disk space that should be available
       sensible_size  => "100K";        # Minimum size in bytes that should be used
       sensible_count => "10";          # Minimum number of files/directories at top level
}
```

EXAMPLE 63

Filename: 120-0425_Cfengine_Grammar:_LHS_vs_RHS.__Example_of_user_defined_body_on_rhs_WITH_PARAMS.cf

```
# example of:     cfengine_word => user_defined_body(param)

bundle agent example
{
      storage: "/" volume  => my_check_volume("30%", "100K");
      storage: "/var" volume  => my_check_volume("20%", "500K");
}



body volume my_check_volume(min_free_space,size)
{
       freespace      => "$(min_free_space)";   # Min disk space that should be available
       sensible_size  => "$(size)";        # Minimum size in bytes that should be used
       sensible_count => "10";          # Minimum number of files/directories at top level
}
```

EXAMPLE 64

Filename: 120-0430b.cf

```
# Example of
#
#     cfengine_word => builtin_function()
#


bundle agent example {

  classes:

      "cf_agent_is_present"

           expression => fileexists("/var/cfengine/bin/cf-agent");



  commands:

    cf_agent_is_present::

      "/bin/echo CF Agent is present on this system";



  vars:

      "http_reply"

               string => readtcp("localhost","80","GET / HTTP/1.0$(const.r)$(const.n)$(const.r)$(const.n)",500);


  commands:

      "/bin/echo $(http_reply)";
```

```
}
```

EXAMPLE 65

Filename: 120-0430_Cfengine_Grammar:_LHS_vs_RHS.__Example_of_builtin_function_on_rhs.cf

```
# Example of
#     cfengine_word => builtin_function()

bundle agent example {

  vars:

      "http_reply"

               handle => "http_client",
              comment => "Demonstrate a function that returns a string.  Run
'GET / HTTP/1.0' and save the output into var http_reply.",
               string => readtcp("localhost","80","GET / HTTP/1.0$(const.r)$(const.n)$(const.r)$(const.n)",500);


  reports:

    linux::

      "The HTTP reply was:$(const.t)$(http_reply)"

               handle => "display_http_reply",
              comment => "Demonstrate the HTTP reply.";

}
```

EXAMPLE 66

Filename: 120-0430_fix_your_webserver.cf

```
# Example of
#     cfengine_word => builtin_function()
```

```
bundle agent example {

  vars:

      "http_reply"

               handle => "http_client",
              comment => "Demonstrate a function that returns a string.  Run
'GET / HTTP/1.0' and save the output into var http_reply.",
               string => readtcp("localhost","80","GET / HTTP/1.0$(const.r)$(const.n)$(const.r)$(const.n)",500);

  classes:
      "http_ok"
               handle => "check_http_ok",
              comment => "Check that the Web server is returning HTTP OK status code",
           expression => regcmp(".*200 OK.*\n.*","$(http_reply)");

      reports: http_ok:: "HTTP OK";
      reports: !http_ok:: "!!! ATTNETION!!!  Fix your web server!!";

}
```

EXAMPLE 67

Filename: 120-0440_Cfengine_Grammar:_LHS_vs_RHS.__Example_of_scalar_on_rhs.cf

```
# Example of
#      cfengine_word => "quoted scalar"

bundle agent example {

  vars:

      "variable_0"
               handle => "declare_variable_0",
              comment => "RHS is a literal string",
              string  => "String contents...";  # a scalar value

      "variable_1"
```

```
               handle => "declare_variable_1",
              comment => "RHS uses a variable",
              string  => "$(variable_0)";       # a scalar variable
  reports:

    linux::

      "
variable_0: $(variable_0)
variable_1: $(variable_1)
"
               handle => "display_var_values",
              comment => "Display values of both variables";
}
```

EXAMPLE 68

Filename: 120-0450_Cfengine_Grammar:_LHS_vs_RHS.__Example_of_list_on_rhs.cf

```
# Example of
#
#      cfengine_word => { list }    #  (directly and via variable)

body common control {

       bundlesequence => { "example1", "example2" };

}

bundle agent example2 {

  reports:

    linux::

      "Second things second"
               handle => "identify_2nd_bundle",
              comment => "Identify the 2nd bundle to demonstrate bundlesequence";
}
```

```
bundle agent example1 {

  reports:

    linux::

      "First things first"
               handle => "identify_1st_bundle",
              comment => "Identify the 1st bundle to demonstrate bundlesequence";
}
```

EXAMPLE 69

Filename: 120-0450-list_exampleb_var_slist_only.cf

```
# Example of
#
#      cfengine_word => { list }    #  (directly and via variable)

bundle agent example {

  vars:

      "my_slist_0"

               handle => "declare_slist_var",
              comment => "Demonstrate a list on the RHS",
               slist  => {
                           "String contents...",
                           "... are beauutifuuul this time of year"
               };


      "my_slist_1"

               handle => "declare_another_slist_var",
              comment => "Demonstrate a list variable on the RHS",
               slist  => { @(my_slist_0) };
```

```
}
```

Chapter 19. Data Types

CFEngine variables have two meta-types: scalars and lists.

Scalars

- A scalar is a single value.
- Each scalar may have one of three types: string, int or real.
- A scalar variable is represented as $(identifier)

  `or $(bundlename.variablename) (fully qualified name)`

EXAMPLE 70

Filename: 130-510_Data_Types.__Examples_of_scalar_variables.cf

```
bundle agent example {

  vars:

      "my_string"
               handle => "example_string_variable",
              comment => "Demonstrate a string variable",
              string  => "String contents...";

      "my_int"
               handle => "example_integer_variable",
              comment => "Demonstrate an integer variable",
                  int => "42";

      "my_real"
               handle => "example_realnumber_variable",
              comment => "Demonstrate a real-number variable",
              real    => "3.141592654";
```

```
  reports:
    cfengine::
      "
my_string: $(my_string)
my_int:    $(my_int)
my_real:   $(my_real)"
              handle => "display_vars",
             comment => "Display how we use variables";
}
```

Integer Suffixes

Integer values may use suffixes to represent large numbers.

Which is easier to read?

200000 200k

Integer Suffixes

```
'k' = value times 1000.
'm' = value times 1000^2
'g' = value times 1000^3

'K' = value times 1024.
'M' = value times 1024^2
'G' = value times 1024^3

'%' meaning percent, in limited contexts

'inf' = a constant representing an unlimited value.
```

EXAMPLE 71

Filename: 130-530_Data_Types.__Integer_suffixes_demo.cf

```
bundle agent example {

  vars:

      "fourty_two_kilobytes"             int     => "42k";  # 42 x 1000
      "fourty_two_kibibytes"             int     => "42K";  # 42 x 1024
      "infinity"                         int     => "inf";  # infinity
      "twenty_percent"                   int     => "20%";


  reports:
    cfengine_3::
      "42 x 1000 = $(fourty_two_kilobytes)";
      "42 x 1024 = $(fourty_two_kibibytes)";
      "infinity = $(infinity)";
      "twenty_percent = $(twenty_percent)";
}
```

Lists

A list is a collection of scalars (single values).

A list variable is represented as @(identifier) or @(bundlename.identifier)

If you refer to a list variable in scalar context by using $(identifier), cfengine will implicitly loop over the values of @(list).

EXAMPLE 72

Filename: 130-550_Data_Types.__List_variables_and_implicit_looping.cf

```
bundle agent example {

  vars:

      "shopping_list"
              handle => "my_shopping_list",
             comment => "Demonstrate implicit looping over a list",
```

```
             slist   => {
                          "apples",
                          "oranges",
                          "bananas",
                          "grapes",
                          "plantains",
                          "coconuts",
             };

     ###############################################################

  reports:
    linux::
      "buy $(shopping_list)"
               handle => "lets_go_shopping",
              comment => "Fire off the (implicit) loop";
}
```

EXAMPLE 73

Filename: 130-555_Data_Types.__example_of_implicit_looping.removing_unwanted_groups.cf

```
bundle agent example {

  vars:

      "unwanted_groups"
               handle => "blacklist_demo",
              comment => "A practical example of looping over a list?",
                slist => {
                          "games",
                          "mail",
                };
  files:

      "/etc/group"

            edit_line => delete_group("$(unwanted_groups)"),
               handle => "purge_unwanted_groups",
```

```
              comment => "Demonstrate parameterization of an edit_line bundle";

}

#########################################################################

bundle edit_line delete_group(unwanted_group) {

  delete_lines:
      "^$(unwanted_group):.*"
               handle => "purge_a_single_unwanted_group",
              comment => "Remove a specified group";
}
```

EXAMPLE 74

Filename: 130-560_Data_Types.__List_variables._Concatenation_of_slists.cf

```
bundle agent example {

  vars:
      "preface"
               handle => "a_string_var",
              comment => "RHS is a static string",
               string => "Now hear this: ";

      "my_slist"
               handle => "a_list_of_strings",
              comment => "RHS is a list of static strings",
                slist => { "String contents...", "...are great!" };

      "the_sum_of_all_parts"
               handle => "a_longer_list_of_strings",
              comment => "RHS is a list made up of a string var and a list
var",
                slist => { $(preface), @(my_slist) };

  reports:
    linux::
```

```
      " Iterating over @(the_sum_of_all_parts): $(the_sum_of_all_parts)"
               handle => "list_demo",
              comment => "Demonstrate how list variables are handled";

}
```

EXAMPLE 75

Filename: 130-570_Data_Types.__Lists_of_integers.cf

```
bundle agent example {

  vars:

      "my_list"

               handle => "int_list",
              comment => "Demonstrate a list of integers",
                ilist => { "1", "2", "3" };
  reports:

    linux::

      "Iterating over the values in @(my_list):  $(my_list)"
               handle => "demo_int_list",
              comment => "Demonstrate implicit looping over an ilist";
}
```

EXAMPLE 76

Filename: 130-580_Data_Types.__Lists_of_real_numbers.cf

```
bundle agent example {

  vars:

      "my_list"

               handle => "rlist_demo",
```

```
              comment => "Create an rlist",
                rlist => { "1.5", "3.0", "4.5" };


  reports:

    linux::

      "Iterating over the values in @(my_list):  $(my_list)"
               handle => "display_rlist",
              comment => "Demonstrate looping over an rlist";
}
```

EXERCISES

1. Create a list variable containing names of five files to create. For example:

   ```
   /tmp/file1
   /tmp/file2
   /tmp/file3
   /tmp/file4
   /tmp/file5

   Then use a single "files" promise to ensure all
   five files exist.
   ```

1b. For advanced students, add a report if any of the files are over 10 bytes in size.

1. Create a list containing names of processes that should not be running: for example "trn" and "eggdrop"

   ```
   Use a single "processes" promise to ensure these
   processes are not running.
   ```

If you finish before the rest of the class, please study your red CFEngine book.

Chapter 20. Data Structures: Arrays

Arrays are associative (hashes).

They may contain scalars or lists as their elements.

Array variables are written with *[* and *]* brackets:

$(array_name[key_name])

or

$(bundle_name.array_name[key_name])

Example:

Food Prices

- Apple 59c
- Banana 30c
- Oranges 35c

Put the price of Apple into food_prices array:

```
vars:   "food_prices[Apple]"    string =>  "59c";
```

Use the price of Apple:

```
$(food_prices[Apple])
```

EXAMPLE 77

Filename: 140-0615_Data_Structures:_Arrays.__Array_of_strings.cf

```
bundle agent example {
  vars:
      "cfengine_components[cf-monitord]"
               handle => "describe_monitor",
              comment => "Document the cf-monitord component",
               string => "The monitor";

      "cfengine_components[cf-serverd]"
               handle => "describe_server",
              comment => "Document the cf-serverd component",
               string => "The server";

      "cfengine_components[cf-execd]"
               handle => "describe_executor",
              comment => "Document the cf-execd component",
               string => "The executor";
      ##
      "component_names"
               handle => "list_of_hash_keys",
              comment => "Extract the keys from cfengine_components array",
                slist => getindices("cfengine_components");

  reports:
    linux::
      "$(component_names)$(const.t)$(cfengine_components[$(component_names)])"
               handle => "dump_array",
              comment => "Display the array keys and values";


}
```

EXAMPLE 78

Filename: 140-0616_Data_Structures:_Arrays.__Keys_are_case-senSiTiVE.cf

```
bundle agent example {

  vars:
```

```
      "cfengine_components[cf-execd]"
               handle => "describe_executor",
              comment => "Document the cf-execd component",
               string => "The executor";

  reports:

    linux::

      "$(cfengine_components[CF-exEcD])"
               handle => "case_demo",
              comment => "Case maTTerS!";

}
```

See 220-1850_Security.__Configure_sshd,_stub.cf

EXERCISE

Make an array, student_grades.

Populate it with the following data:

```
Key    Value
---    -----
Joe    A
Mary   A
Bob    B
Sue    B
```

Display the contents of the array.

EXAMPLE 79

Filename: 140-0630_Data_Structures:_Arrays.__Array_of_slists.cf

```
bundle agent example {
```

```
    # implicit looping over a slist in an array is broken right now.

    ###########################################################################
    # Configuration section
    ###########################################################################

 vars:
    "config[users]"
             handle => "users_list_in_config_array",
            comment => "Demonstrate how an array value can hold a list",
              slist => { "jim", "jane", "george" };

    "config[packages]"
             handle => "packages_list_in_config_array",
            comment => "Demonstrate how an array value can hold a list",
              slist => { "httpd", "php" };

    ###########################################################################

    "keys"
             handle => "config_array_keys",
            comment => "generate a list containing keys to 'config' array",
              slist => getindices("config");


 reports:

   linux::

    "The value for key $(keys) is: $(config[$(keys)])";

}
```

CFEngine Components

CFEngine 3 consists of a number of components.

Syntax checker and Agent

cf-promises

You can use it to syntax check your policies:

```
cf-promises -f ./your_policy.cf
```

cf-agent

Execute your policies. cf-agent is responsible for maintaining promises about the state of your system. It's the part of cfengine that actually makes changes to the system.

```
cf-agent -f ./your_policy.cf
```

cf-execd

Used to run cf-agent on a regular (and user-configurable) basis.

Tools used in CFEngine inter-node communication

cf-serverd

Used to distribute files to clients requesting them and to listen to network requests for re-running the local cf-agent.

cf-key

Key generation tool – run once on every host to create public/private key pairs for secure communication.

cf-runagent

Remote run agent – used to execute cf-agent on a remote machine. cf-runagent does not keep any promises, but instead is used to ask another machine to do so.

Misc.

Miscellaneous components

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).

cf-know

Knowledge modelling agent – responsible for building and analysing a semantic knowledge network. (Enterprise edition only.)

cf-hub

A data aggregator used as part of the commercial product and not available in the community edition of CFEngine.

CFEngine Package

Run the following to examine what is in the CFEngine package

rpm -q --filesbypkg cfengine-community | less

Chapter 21. CFEngine Standard Library

CFEngine ships with a standard library of promise bodies and bundles dealing with common aspects of system administration.

The CFEngine Standard Library is growing to include all common aspects of system administration.

CFEngine version	Promise bodies	Promise bundles
3.1.5	88	?
3.2.1	99	19
3.3.5	114	29
3.3.8	113	26
3.4.4	124	32

EXAMPLE 80

Filename: 170-1010_COPBL.__Package_add_using_COPBL.cf

```
bundle agent example {

  packages:

      "php-mysql"

               handle => "install_package_php_mysql",
              comment => "Demonstrate installing a package",
       package_policy => "add",
       package_method => yum;
}

########################################

body common control {
       inputs         => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 81

Filename: 170-1020_COPBL.__File_exists_and_is_mode_6_1_2_mog._Without_COPBL.cf

```
bundle agent example {

  files:

      "/tmp/testfile"

              handle  => "set_file_attributes",
              comment => "Demonstrate_setting_file_attributes",
              create  => "true",
              perms   => mog("612","aleksey","cfengine");

}

###############################################################

body perms mog(mode,owner,group)
{
               owners => { "$(owner)", "john", "brian" };
               groups => { "$(group)" };
               mode   => "$(mode)";
}
```

EXAMPLE 82

Filename: 170-1021_COPBL.__File_exists_and_is_mode_6_1_2_mog.cf

```
bundle agent example {

  files:

      "/tmp/testfile"

               handle => "set_file_attributes",
              comment => "/tmp/testfile must be mode 612 for application X to work; it must be owned by user aleksey and group cfe
              create  => "true",
```

```
             perms   => mog("612","aleksey","cfengine");

}

############################################################

body common control {
       inputs         => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 83

Filename: 170-1030_COPBL.__Context_sensitive_file_editing._Set_robs_password.cf

```
bundle agent example {

  files:

      "/etc/shadow"

               handle => "context_sensitive_file_editing_demo",
              comment => "demonstrate context-sensitive file editing capability",
            edit_line => set_user_field("rob",2,"$1$stIAaUZw$ptP75nVkz/EapeuvdWLNC0");

}

#################################################

body common control {
       inputs         => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 84

Filename: 170-1040_COPBL.__Removing_a_file.cf

```
bundle agent example {

  files:
```

```
      "/tmp/testfile.*"

               handle => "demo_removing_files",
              comment => "Demonstrate removing files using body delete tidy",
               delete => tidy;

      # shell equivalent:  rm -r /tmp/testfile*

}


body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 85

Filename: 170-1045b.cf

```
bundle agent example {

  files:

      "/etc/httpd/conf.d/maintenance.conf"

            handle    => "put_website_into_maintenance",
            comment   => "Enable maintenance-mode config block",
            edit_line => uncomment_everything;

}

bundle edit_line uncomment_everything {

  replace_patterns:

      "^#(.*)"
               handle => "uncomment_everything_replace_pattern",
              comment => "If it starts with a hash mark, grab everything after the hash mark, and uncomment it.",
```

```
         replace_with => uncomment;
}


body replace_with uncomment
{
        replace_value => "$(match.1)";
          occurrences => "all";
}
```

EXAMPLE 86

Filename: 170-1045_COPBL.__Commenting_out_file_contents.cf

```
bundle agent example {

  files:

      "/etc/httpd/conf.d/maintenance.conf"

            handle    => "take_website_out_of_maintenance",
            comment   => "Disable maintenance-mode config block",
            edit_line => comment_out_everything;

}

bundle edit_line comment_out_everything {

  replace_patterns:

      "^([^#].*)"
         replace_with => comment("# ");
}

body replace_with comment(c)
{
        replace_value => "$(c) $(match.1)";
          occurrences => "all";
}
```

EXERCISE

Run the following command:

```
date  > /tmp/date.txt
```

Now write a CFEngine policy that will comment out (using #) the contents of that file.

EXAMPLE 87

Filename: 170-1050_COPBL.__Removing_a_file._Remove_centos_httpd_welcome_page.cf

```
bundle agent example {

  files:

      "/etc/httpd/conf.d/welcome.conf"

               handle => "nuke_welcome_conf",
              comment => "Let's keep a low profile and not advertise what software we are running - remove the Welcome page that s
               delete => tidy;
}


body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };

}
```

EXAMPLE 88

Filename: 170-1051_COPBL.__Remove_httpd_welcome_page_by_commenting_out_welcome_conf.cf

```
# welcome.conf is part of the Apache RPM
# to preserve package integrity, comment out this file's contents
# instead of deleting the file
```

```
bundle agent example {

  files:

      "/etc/httpd/conf.d/welcome.conf"

               handle => "comment_out_welcome_dot_conf",
              comment => "Let's keep a low profile and not advertise what software we are running",
            edit_line => comment_out_everything,
              classes => if_repaired("reload_httpd");

  commands:
    reload_httpd::
      "/etc/init.d/httpd"
               handle => "cmd_reload_httpd",
              comment => "Reload httpd configuration",
                 args => "reload";

}

bundle edit_line comment_out_everything {

  replace_patterns:

      "^([^#].*)"

               handle => "comment_out_everything_replace_patterns_promise",
              comment => "If it doesn't start with #, comment it out",
         replace_with => comment("#disabled-by-cfengine# ");

}

body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 89

Filename: 170-1100_COPBL.__classes._if_else.cf

```
bundle agent example {


  files:

      "/tmp/etc/motd"
               handle => "touch_file",
              comment => "Demonstrate body classes if_else",
               create => "true",
              classes => if_else("file_exists","file_missing");

  reports:
    file_exists::
      "All OK"
               handle => "report_OK";

  reports:
    file_missing::
      "WARNING! Unable to create vital file!"
               handle => "report_WARN";

}


body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 90

Filename: 170-1100_COPBL.__classes._persistent.cf

```
bundle agent example {

  files:
      "/tmp/file.txt"
               handle => "persistent_class_demo",
              comment => "Set a persistent class",
               create => "true",
```

```
            classes  => state_repaired("file_fixed");

  reports:
    file_fixed::
      "Persistent class set.  Run in verbose mode to see TTL"
               handle => "report_success",
              comment => "Report if our persistent class persistent_class has been set as expected.";
}

body classes state_repaired(x)
{
     promise_repaired => { "$(x)" };
         persist_time => "10";
}
```

EXAMPLE 91

Filename: 170-1100_COPBL.__comment_lines_matching.cf

```
bundle agent example {

  files:
      "/tmp/scratch"
               handle => "selective_commenting",
              comment => "Remove specific lines",
            edit_line => comment_lines_matching("hello world", "#");

}


body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 92

Filename: 170-1100_COPBL.__contain-silent.cf

```
bundle agent example {

  commands:

      "/bin/date"
               handle => "run_date_cmd",
              comment => "Demonstrate 'body contain silent'",
              contain => silent;


}

body common control {
               inputs => { "libraries/cfengine_stdlib.cf"};
}
```

EXAMPLE 93

Filename: 170-1100_COPBL.__edit_resolv_dot_conf_using_COPBL.cf

```
bundle agent example {

  vars:

      "search_suffix"  string =>  "example.com example2.com";

      "nameservers"    slist  =>  { "8.8.4.4", "8.8.8.8" };

  files:

      "/tmp/resolv.conf"

               handle =>  "edit_resolv_conf",
              comment => "Setup up DNS resolver",
            edit_line =>  resolvconf("$(search_suffix)", "@(example.nameservers)" );
}
```

```
body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 94

Filename: 170-1100_COPBL.__insert_lines.cf

```
bundle agent example {

  files:

      "/tmp/scratch"

                handle => "files_multi_line_insert",
               comment => "Insert multi-line content",
                create => "true",
             edit_line => insert_lines("
hello world
this is line 2
line 3 is great
line 4 is awesome
");
}


body common control {
               inputs => { "libraries/cfengine_stdlib.cf"};
}
```

EXAMPLE 95

Filename: 170-1100_COPBL.__set_variable_values.cf

```
bundle common global {

  vars:
```

```
      "stuff[location]"     string => "Bloomington";
      "stuff[time]"         string => "May-2013";
      "stuff[students]"     string => "11";
      "stuff[lab]"          string => "true";
}

bundle agent example {

  files:

      "/etc/example.conf"
               handle => "populate_config_file_from_array",
              comment => "Demonstrate 'bundle edit_line set_variable_values'",
               create => "true",
            edit_line => set_variable_values("global.stuff");
}


body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 96

Filename: 170-1100_COPBL.__standard_services.cf

```
bundle agent example {

  methods:

    any::

      "Manage www service"

            usebundle => standard_services ("www", "stop");

}

bundle agent standard_services(service,state)
```

```
{
     # DATA,

  vars:

    any::

      "stakeholders[www]" slist => { "www_in", "wwws_in", "www_alt_in" };


    SuSE|suse|debian::

      "startcommand[www]" string => "/etc/init.d/apache2 start";
      "stopcommand[www]"  string => "/etc/init.d/apache2 stop";
      "pattern[www]"      string => ".*apache2.*";


    redhat::

      "startcommand[www]" string => "/etc/init.d/httpd start";
      "stopcommand[www]"  string => "/etc/init.d/httpd stop";
      "pattern[www]"      string => ".*httpd.*";



     # METHODS that implement these ........................................

  classes:

     "start" expression => strcmp("start","$(state)"),
             comment => "Check if to start a service";
     "stop"  expression => strcmp("stop","$(state)"),
             comment => "Check if to stop a service";


  processes:

    start::
```

```
     "$(pattern[$(service)])" ->  { "@(stakeholders[$(service)])" } ,

            comment => "Verify that the service appears in the process table",
      restart_class => "restart_$(service)";

   stop::

     "$(pattern[$(service)])" -> { "@(stakeholders[$(service)])" },

            comment => "Verify that the service does not appear in the process",
       process_stop => "$(stopcommand[$(service)])",
            signals => { "term", "kill"};

 commands:

     "$(startcommand[$(service)])" -> { "@(stakeholders[$(service)])" },

            comment => "Execute command to restart the $(service) service",
         ifvarclass => "restart_$(service)";
}
```

Chapter 22. Patterns + Promises = Configuration

Patterns are a way of compressing information.

The CFEngine 3 language is made of promises and patterns; it’s about using patterns to create concise but powerful promises.

An example of a pattern in CFEngine is a list. You can have a list of things you want, or do not want: for example, a list of packages that should be installed, or processes that should NOT be running.

Implicit looping creates multiple promises that follow the promise pattern.

EXAMPLE 97

Filename: 180-1120_Patterns+Promises=Configuration.__Lists:_Implicit_looping_over_a_list_of_packages.cf

```
bundle agent example {

     ########################################################
     # This is the data section, which describes the desired pattern
     #
     # All you do is add to or edit the list...
     #
     # This is *data-driven* configuration.
     ########################################################
  vars:

     "desired_package"

              handle => "good_packages",
             comment => "list the packages we want",
               slist => {
                          "httpd",
                          "php",
                          "php-mysql",
                          "mysql-server",
               };
```

```
     "unwanted_package"

              handle => "bad_packages",
             comment => "list the packages we do not want",
               slist => {
                          "java",
                          "ruby",
               };

     ########################################################
     # Below here is stock convergent code, forget this...
     ########################################################

  packages:
     "$(desired_package)"
              handle => "add_package",
             comment => "Ensure package is present",
      package_policy => "add",
package_architectures => { "x86_64" },
      package_method => yum;


  packages:
     "$(unwanted_package)"
              handle => "remove_package",
             comment => "Ensure package is absent",
      package_policy => "delete",
package_architectures => { "x86_64" },
      package_method => yum;

}

body common control {
              inputs =>         { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 98

Filename: 180-1130_Patterns+Promises=Configuration.__Lists:_Implicit_looping_over_a_list_of_files.cf

```
body common control {
               inputs =>         { "libraries/cfengine_stdlib.cf" };
}

bundle agent example {

  vars:

      "list_of_files"
               handle => "file_list",
              comment => "Just a file list",
                slist => {
                           "/etc/passwd",
                           "/etc/group",
                };

  files:

      "$(list_of_files)"

               handle => "set_mode_and_ownership",
              comment => "Ensure a list of files is owned by root
and mode 644",
                perms => mo("644","root");
}
```

Regular Expressions

Regular Expressions is another way of writing patterns.

CFEngine supports POSIX and PCRE regular expressions. (PCRE by default.)

EXAMPLE 99

Filename: 180-1190_Patterns+Promises=Configuration.__Regular_expressions._Files.cf

```
bundle agent example {

  files:

      "/etc/pass.*"

               handle => "set_file_perms_on_regex_list_of_files",
              comment => "Files matching /etc/pass.* need to be owned by root and mode 644",
                perms => mo("644","root");

}


body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

Body templates

"A pattern is just a repeated structure. The benefit of seeing patterns is economy: if you can see a pattern, you can take out the commonality, abstract it, and talk about the pattern instead of all the individual cases. This is a Knowledge Management step." cfengine.org

body templates are intended to aid in such abstraction

Classes

This chapter discusses Classes.

EXAMPLE 100

Filename: 180-1340_Patterns +Promises=Configuration.__Classes:_using_classes_to_link_promises._Promise_repair._also_demonstrates_action_logme.cf

```
bundle agent example {

  files:
```

```
      "/etc/ssh/sshd_config"
               handle => "sshd_must_use_protocol_2_only",
              comment => "Make sure SSHD does not use protocol v1; make sure it only uses protocol v2, to increase security",
            edit_line => permit_protocol_2_only,
              classes => if_repaired("sshd_config_file_was_repaired"),
               action => logme("promise $(this.handle)");

  commands:
    sshd_config_file_was_repaired::
      "/etc/init.d/sshd reload"
               handle => "reload_sshd",
              comment => "run sshd init script to reload sshd to pick up new config",
               action => logme("promise $(this.handle)");

}


body action logme(x)
{
           log_string => "$(sys.date) $(x)";

             log_kept => "/var/log/cfengine_keptlog.log";
         log_repaired => "/var/log/cfengine_replog.log";
           log_failed => "/var/log/cfengine_faillog.log";

}



bundle edit_line permit_protocol_2_only {
      delete_lines: ".*Protocol.*1.*";
      insert_lines: "Protocol 2";
}

body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 101

Filename: 180-1350_Patterns+Promises=Configuration.__Classes:_ORing_of_classes_and_fileexists.cf

```
bundle agent example
{

  classes:

      # List form of class expression useful for including functions

      "my_new_class"

               handle => "or_list",
              comment => "Demonstrate list form of class expression useful for including functions",
                 or   => { "linux",
                           "solaris",
                           fileexists("/etc/fstab")
                 };


  reports:

    my_new_class::

      # This will only report Boo! on linux, solaris, or any system
      # on which the file /etc/fstab exists

      "Boo!";
}
```

EXAMPLE 102

Filename: 180-1370_Patterns+Promises=Configuration.__Classes:_Set_a_private_class_based_on_hard_classes_expression.cf

```
bundle agent example {


  classes:

      "good_technology"
```

```
               handle => "good_technology_class",
              comment => "Set a custom class based on built-in classes",
           expression => "linux|solaris";

  reports:
    good_technology::
      "I love good technology"
               handle => "show_respect",
              comment => "Show respect for good technology";

}
```

EXAMPLE 103

Filename: 180-1470_Patterns+Promises=Configuration.__Classes:_Set_a_custom_class_based_on_function_result.cf

```
# - demonstrate setting a custom class using a function

bundle agent example {

  classes:
      "islink"
               handle => "class_islink",
              comment => "Test if /tmp/a is a symbolic link",
           expression => islink("/tmp/a");


  reports:
    islink::
      "/tmp/a is a link";

    !islink::
      "/tmp/a is not a link";
}
```

Example of classes provided by cf-monitord

If you are running cf-monitord, you may also see entropy and anomaly detection classes:

entropy

```
entropy_cfengine_in_low
entropy_cfengine_out_low
entropy_dns_in_low
entropy_dns_out_low
entropy_ftp_in_low
entropy_ftp_out_low
entropy_icmp_in_low
entropy_icmp_out_low
entropy_irc_in_low
entropy_irc_out_low
entropy_misc_in_low
entropy_misc_out_low
entropy_netbiosdgm_in_low
entropy_netbiosdgm_out_low
entropy_netbiosns_in_low
entropy_netbiosns_out_low
entropy_netbiosssn_in_low
entropy_netbiosssn_out_low
entropy_nfsd_in_low
entropy_nfsd_out_low
entropy_smtp_in_low
entropy_smtp_out_low
entropy_tcpack_in_low
entropy_tcpack_out_low
entropy_tcpfin_in_low
entropy_tcpfin_out_low
entropy_tcpsyn_in_low
entropy_tcpsyn_out_low
entropy_udp_in_low
entropy_udp_out_low
entropy_www_in_low
entropy_www_out_low
entropy_wwws_in_low
entropy_wwws_out_low
```

A low entropy value means that most of the events came from only a few (or one) IP addresses. A high entropy value implies that the events were spread over many IP sources.

anomaly

Anomaly detection - example classes:

loadavg_high_ldt - load average higher than usual (based on Leap-Detection Test)

messages_high_dev1 -the current value of the metric is more than 1 standard deviation above the average.

etc.

Reference: http://www.iu.hio.no/cfengine/docs/cfengine-Anomalies.pdf

EXAMPLE 104

Filename: 180-1472_Patterns+Promises=Configuration.__Classes:_Report_type_of_weekday._Uses_a_custom_class.cf

```
bundle agent example {

  classes:
      "weekday"
           expression => "Monday|Tuesday|Wednesday|Thursday|Friday";


      "weekend"
           expression => "Saturday|Sunday";

  reports:
    weekday::
      "Today is a weekday.";

}
```

EXAMPLE 105

Filename: 180-1473_Patterns+Promises=Configuration.__Classes:_Report_type_of_weekday._Uses_ifvarclass.cf

```
# report "Hello world!  I love weekends!" on Saturday or Sunday,
# report "Hello world!  I love Mondays|Tuesdays|...|Fridays on a weekday

bundle agent example {

  vars:
      "days"
               handle => "days",
              comment => "Build a list of days to report what day of the week it is.",
                slist => { "Monday",
                           "Tuesday",
                           "Wednesday",
                           "Thursday",
                           "Friday",
                           "Saturday",
                           "Sunday",
                };


  reports:
    linux::
      "Hello world!  I love $(days)s!"
               handle => "report_the_day",
              comment => "Report what day of the week it is";
           ifvarclass => "$(days)";


}
```

EXAMPLE 106

Filename: 180-1474_Patterns+Promises=Configuration.__Classes:_GOTCHA.cf

```
bundle agent example {

  commands:


    linux&Hr08::
```

```
     "/bin/echo Linux system AND we are in the 8th hour.";

     "/bin/echo hello world";   # this promise is NOT in the class "any" !!!
}
```

EXERCISE

1. Set a custom class if the file /tmp/testme exists.
2. Report the presence or absense of the file using "reports" type promises and the class defined in #1 above.
3. Have a "files" type promise create the file "/tmp/testme".

Now, remove /tmp/testme and run your policy and observe and explain what happens.

Classes and Scope

Do classes have scope?

EXAMPLE 107

Filename: 180-1481_Patterns+Promises=Configuration.__Classes:_Scope.cf

```
bundle agent example {

  classes:
      "its_monday"
           expression => "Monday";

  classes:
      "its_wed"
           expression => "Wednesday";
  classes:
      "its_thur"
```

```
           expression => "Thursday";
}


bundle agent example2 {

  reports:
    its_monday::
      "Yay!  I love Mondays!";

  reports:
    its_wed::
      "Yay!  I love Wednesdays!";

  reports:
    its_thur::
      "Yay!  I love Thursdays!";
}
```

EXAMPLE 108

Filename: 180-1482_Patterns+Promises=Configuration.__Classes:_Classes_defined_in_common_bundles_have_global_scope.cf

```
bundle common global_definitions {

  classes:
      "myclass"
           expression => "any";

}


bundle agent example {

  reports:
    myclass::
      "Yay!  myclass is set";

}
```

EXAMPLE 109

Filename: 180-1483_Patterns+Promises=Configuration.__Classes:_if_repaired_creates_global_classes.cf

```
bundle agent example {

  files:
      "/etc/motd"
               create => "true",
              classes => if_repaired("its_wed");

}


bundle agent example2 {

  reports:
    its_wed::
      "Yay!  I love Wednesdays!";

}


body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 110

Filename: 180-1490_Patterns+Promises=Configuration.__Classes._Global_vs_local_classes.cf

```
bundle common global_classes {

      # Classes defined in common bundles are global.
      #
      # They appear in the Defined Classes section at the start of
      # verbose output.
      #
      # Classes defined in all other bundles are local
```

```
  classes:

      "webserver"

          expression  =>  classmatch("web[0-9]+");

}



bundle agent example {
  methods:
      "any" usebundle => example1;
      "any" usebundle => example2;
}

bundle agent example1
{

  classes:
      "local_class"
          expression  =>  classmatch("web[0-9]+");

  reports:
    webserver::
      "bundle 'example1': I am a Web server, and I tested for this using a global class.";

  reports:
    local_class::
      "bundle 'example1': I am a Web server, and I tested for this using a local class.";

}


bundle agent example2
{
```

```
  reports:
    webserver::
      "bundle 'example2': I am a Web server, and I tested for this using a global class.";

  reports:
    local_class::
      "bundle 'example2': I am a Web server, and I tested for this using a local class.";

}
```

EXAMPLE 111

Filename: 180-1491_Patterns+Promises=Configuration.__Classes:_Global_vs_local_classes.local_demo.cf

```
body common control {

       bundlesequence => { "example", "example2" };

}



bundle agent global_classes {

      # Classes defined in common bundles are global.
      #
      # They appear in the Defined Classes section at the start of
      # verbose output.
      #
      # Classes defined in all other bundles are local


  classes:

      "webserver"

          expression  =>  classmatch("web[0-9]+");

}
```

```
bundle agent example
{
  classes:

     "webserver"

         expression  =>  classmatch("web[0-9]+");


  reports:

    webserver::

      "I am a Web server - 1";

}


bundle agent example2
{
  reports:

    webserver::

      "I am a Web server - 2";

}
```

Chapter 23. Methods

There is a special promise type in CFEngine 3 called "methods" that promises to call another promise bundle.

```
methods:
"any"
usebundle => bundle_name;
```

The promiser can be any word, right now it does not matter; the promiser is reserved for future development.

— next slide --

Parameters are optional:

```
methods:
"any"
usebundle => bundle_name("arg1", "arg2");
```

EXAMPLE 112

Filename: 190-1510_Methods.__Simple_example.cf

```
bundle agent example {
  vars:
      "userlist" slist => { "alex", "ben", "charlie", "diana", "rob" };

  methods:
      "any" usebundle => remove_user("$(userlist)");
}
```

```
#############################################

bundle agent remove_user(user) {

  commands:

      "/usr/sbin/userdel $(user)"
              contain => silent;

}


body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };

}
```

EXAMPLE 113

Filename: 190-1511_Methods.__Lock_an_account.cf

```
body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };

}

bundle agent example {

  vars:

      "badusers" slist => {
                            "alex",
                            "ben",
                            "charlie",
                            "diana",
                            "joe"
      };
```

```
      ######################################################

  methods:

      "any" usebundle => lock_user("$(badusers)");

}

#########################################

bundle agent lock_user(user) {

  files:
      "/etc/shadow"

            edit_line => set_user_field("$(user)",2,"!LOCKED");

  files:

      "/etc/passwd"

            edit_line => set_user_field("$(user)",7,"/bin/false");

  files:

      "/etc/sudoers"

            edit_line => delete_lines_matching("^$(user)");

}
```

Methods provide encapsulation of multiple issues

Methods offer powerful ways to encapsulate multiple issues pertaining to a set of parameters.

For example:

Removing a user: 1. userdel 2. sudoers 3. mail spool

EXAMPLE 114

Filename: 190-1521_Methods.*Abstraction_using_methods*Also_demonstrates_setting_defaults_in_variables.cf

```
# Make sure /etc/group contains a "cfengine" group


bundle agent example {

  methods:

      "any"

            usebundle => add_group("example");


}


#######################################################################


bundle agent add_group(groupname) {

  vars:

    any::

      "groupadd_utility"  string =>  "/sbin/groupadd",
               policy =>  "overridable";

      linux:: "groupadd_utility"  string =>  "/usr/sbin/groupadd",
               policy =>  "overridable";

      hpux::  "groupadd_utility"  string =>  "/usr/sbin/addgroup",
               policy =>  "overridable";
```

```
      aix::   "groupadd_utility"  string =>  "/sbin/addgroup",
               policy =>  "overridable";



  commands:

      "$(groupadd_utility)"

                 args => "$(groupname)",
              comment => "Ensure specified group '$(groupname)' group is present.";

}
```

EXAMPLE 115

Filename: 190-1521_Methods.__Abstraction_using_methods.cf

```
# Make sure /etc/group contains a "cfengine" group

bundle agent example {

  methods:

      "any"

               handle => "group_exists",
              comment => "make sure the specified group is always present",
            usebundle => groupadd("cfengine");

}


########################################################################


bundle agent groupadd(groupname) {
```

```
  commands:
      linux:: "/usr/sbin/groupadd" args => "$(groupname)";
      aix::   "/sbin/addgroup" args => "$(groupname)";
      hpux::  "/usr/sbin/addgroup" args => "$(groupname)";

}


bundle agent addgroup(groupname) {

  vars:

      linux:: "groupadd_utility"  string =>  "/usr/sbin/groupadd";
      hpux::  "groupadd_utility"  string =>  "/usr/sbin/addgroup";
      aix::   "groupadd_utility"  string =>  "/sbin/addgroup";


  commands:

      "$(groupadd_utility)"

                args => "$(groupname)",
             comment => "Ensure specified group '$(groupname)' group is present.";

}
```

EXERCISE 1

Practice using "methods" type promises

Write a policy that has two bundles.

The first bundle does something visible (such as a reports type promise that says "bundle1") AND calls the second bundle.

The second bundle reports "bundle2".

What output will you see and in what order? Why? Now run your policy and check.

EXERCISE 2

Now parameterize the 2nd bundle — have the first bundle feed it an argument, and have the 2nd bundle display that argument.

EXERCISE 3

Sysadmin Problem:

/etc/profile should set the ORGANIZATION environment variable when users log in:

```
export ORGANIZATION=MyOrg
```

Policy Writing Exercise:

Write a bundle "etc_profile_contains" that would take an argument and ensure /etc/profile contains the text string specified in the argument.

Demonstrate its use by calling it from another bundle:

```
bundle agent example {
   methods:
     "any"
         usebundle => etc_profile_contains("export ORGANIZATION=MyOrg");
}
```

EXERCISE Methods (re-usable bundles)

Make a bundle called file_contains that takes two arguments: a filename, and a text string. The bundle should ensure that the file specified in the first argument contains the text string specified in the second argument. Example:

```
methods:
 "any" usebundle => file_contains("/etc/profile", "export ORGANIZATION=MyOrg");
 "any" usebundle => file_contains("/etc/motd", "Unauth. use forbidden");
```

EXERCISE Methods

Configuring a web server.

Write a bundle "webserver" that will ensure an Apache httpd package is installed and process is running if its argument is "on":

```
methods:
"any"
usebundle => webserver("on");
```

Then, make sure httpd is not running if its argument is "off".

Hints: - The CFEngine function strcmp() can compare two strings.

Reference: 039-0085_Basic_Examples:_Classes_and_Reports.__soft-class.cf

Chapter 24. Versioning Policy

Here are examples of versioning your policies and integrating CFEngine with a Version Control System.

EXAMPLE 116

Filename: 200-1610_Versioning_Policy.__Version_number._Plain.cf

```
body common control
{
              version => "1.1";
       bundlesequence => { "example" };
}


########################################


bundle agent example
{

  commands:
    linux::
      "/bin/nosuchcommand hello world, i love wednesdays and coffee";
}
```

EXAMPLE 117

Filename: 200-1615_Versioning_Policy.__Embedding_RCS_Id_tag.cf

```
body common control
{
              version => "$Id: 200-1615_Versioning_Policy.__Embedding_RCS_Id_tag.cf,v 1.3 2012/10/24 22:53:25 root Exp root $";
}


########################################
```

```
bundle agent example
{

  commands:
    linux::
      "/bin/nosuchcommand hello world, i love wednesdays and coffee";
}

#comment
```

Chapter 25. File Copying

EXAMPLE 118

Filename: 210-1708_File_Copying.__Local_copy_a_single_file.cf

```
body common control
{
       inputs         => { "libraries/cfengine_stdlib.cf" };
}


########################################


bundle agent example {


  files:

      "/etc/motd"

         copy_from    => local_cp("/var/cfengine/inputs/templates/motd.txt");

}
```

EXERCISE

Copy /var/cfengine/share/CoreBase/*cf to /var/cfengine/inputs

EXERCISE

/root/passwd.bak is a backup (copy) of /etc/passwd

EXAMPLE 119

Filename: 210-1710_File_Copying.__Local_copy_a_directory.cf

```
body common control
{
       inputs         => { "libraries/cfengine_stdlib.cf" };
}


########################################


bundle agent example
{

  vars:

      # A standard location for the source point
      "master_location" string => "/var/cfengine/masterfiles";

  files:

      "/var/cfengine/inputs/."

         comment      => "Update the policy files from the master",
         copy_from    => local_cp("$(master_location)"),
         depth_search => recurse("inf");

      #   /var/cfengine/masterfiles --------->  /var/cfengine/inputs

}
```

EXERCISE

1. Use CFEngine to make /tmp/mirror contain a copy of /usr/local/sbin

(Hint: use a files promise with a copy_from attribute)

1. Now create a new file in /usr/local/sbin and confirm CFEngine will copy it over.

2. Work out how to mirror file removals. (When a file is removed in /usr/local/sbin, it should disappear in /tmp/mirror.)

EXAMPLE 120

Filename: 210-1720_File_Copying.__Remote_copy.cf

```
bundle agent example {

  vars:

      "remote_path" string => "/var/cfengine/masterfiles";
      "remote_server" string => "205.186.156.208";

  files:

      "/var/cfengine/inputs"

               handle => "update_inputs_dir",
              comment => "Pull down latest policy set",
                perms => u_p("600"),
            copy_from => u_remote_cp("$(remote_path)","$(remote_server)"),
         depth_search => u_recurse("inf"),
               action => u_immediate;
}

body perms u_p(p)
{
                mode  => "$(p)";
}

body copy_from u_remote_cp(from,server)
{
          servers     => { "$(server)" };
          source      => "$(from)";
```

```
          compare     => "mtime";
          trustkey    => "true";  # trust the server's public key
}

body depth_search u_recurse(d)
{
               depth => "$(d)";
               xdev  => "true";
}

body action u_immediate
{
            ifelapsed => "0";
}
```

EXAMPLE 121

Filename: 210-1730_File_Copying.__Remote_copy_with_round_robin.cf

```
# use two remote servers, and round-robin between them

bundle agent example
{

  classes:
      "heads"
               handle => "flip_a_coin_class",
              comment => "Generate a class with a 50% probability",
           expression => isgreaterthan(randomint(1,100), 50);


  files:
      "/tmp/test1copy"
            copy_from => round_robin_cp("/var/cfengine/masterfiles/testfile1", "10.1.1.10", "10.1.1.12");
}


body copy_from round_robin_cp(from,server1, server2)
{
```

```
                source => "$(from)";

    heads::
                servers => { "$(server1)", "$(server2)" };

    !heads::
                servers => { "$(server2)", "$(server1)" };
}
```

Chapter 26. Security

Change Detection

In the "computer immunology" research and development phase, Mark added file change detection capability to CFEngine.

EXAMPLE 122

Filename: 220-1820_Security.__Detect_changes_in_etc.cf

```
bundle agent example {

  files:

      "/etc"

         handle       => "etc_tripwire",
         comment      => "Report changes on files in /etc",
         changes      => detect_all_change,
         depth_search => recurse("inf");
}


body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}
```

EXAMPLE 123

Filename: 220-1825_Security.__Detect_changes_in_etc._Uses_classes.cf

```
bundle agent example {
  files:

      "/etc"
```

```
         handle       => "safeguard_files_in_etc",
         comment      => "Keep screaming about changes in /etc",
         changes      => detect_all_change_noupdate,
         depth_search => recurse("inf"),
         classes      => kept_repaired_failed("promise_kept", "promise_repaired", "promise_not_kept)");

reports:

  promise_kept::

        "Kept";

  promise_repaired::
       "Repaired";

   promise_not_kept::

        "not kept";

}




body classes kept_repaired_failed(kept, repaired, failed) {
     promise_kept     => { "$(kept)" };
     promise_repaired => { "$(repaired)" };
     repair_failed    => { "$(failed)" };
     repair_denied    => { "$(failed)" };
     repair_timeout   => { "$(failed)" };
}


body changes detect_all_change_noupdate {
      # This is fierce, and will cost disk cycles
       hash           => "best";
       report_changes => "all";
       update_hashes  => "no";
}
```

```
##

body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
       bundlesequence => { "example" };
}
```

EXAMPLE 124

Filename: 220-1830_Security.__Match_suspicious_process_names.cf

```
bundle agent example {

  vars:

      "suspicious_process_names"
               handle => "process_blacklist",
              comment => "Setup a list of known bad process names",
                slist =>
      {
        "sniff",
        "eggdrop",
        "r00t",
        "^\./",
        "john",
        "crack"
      };
  processes:
      "$(suspicious_process_names)"
               handle => "kill_bad_procs",
              comment => "Kill bad processes on sight",
              signals => { "term", "kill" };
}
```

EXAMPLE 125

Filename: 220-1840_Security.__Check_open_ports.cf

```
bundle agent example {

  vars:
      "listening_ports_and_processes_ideal_scene"
               handle => "expected_tcp_profile",
              comment => "expected network profile (listenting ports)",
               string => "22 sshd 80 httpd 443 httpd 5308 cf-server";

      # end of our expected configuration

  vars:
    centos_5::
      "listening_ports_and_processes"
               handle => "actual_tcp_profile",
              comment => "Our actual network profile",
               string =>
      execresult("/usr/sbin/lsof -i -n -P | \
/bin/grep LISTEN | \
/bin/sed -e 's#*:##' | \
/bin/grep -v 127.0.0.1 | \
/bin/grep -v ::1 | \
/bin/awk '{print $8,$1}' | \
/bin/sort | \
/usr/bin/uniq | \
/bin/sort -n | \
/usr/bin/xargs echo", "useshell");  # this is our
      # actual configuration.
      # we tell CFEngine to use a shell with "useshell"
      # to do a pipeline.

    centos_6::
      "listening_ports_and_processes"
               handle => "actual_tcp_profile",
              comment => "Our actual network profile",
               string =>
      execresult("/usr/sbin/lsof -i -n -P | \
/bin/grep LISTEN | \
/bin/sed -e 's#*:##' | \
/bin/grep -v 127.0.0.1 | \
```

```
/bin/grep -v ::1 | \
/bin/awk '{print $9,$1}' | \
/bin/sort | \
/usr/bin/uniq | \
/bin/sort -n | \
/usr/bin/xargs echo", "useshell");

  classes:
      "reality_does_not_match_ideal_scene"
               handle => "check_profile",
              comment => "Compare desired and actual configuration",
                  not => strcmp (
                                 "$(listening_ports_and_processes)",
                                 "$(listening_ports_and_processes_ideal_scene)"
                  );  # check whether expected configuration matches actual.

  reports:
    reality_does_not_match_ideal_scene::
      "
DANGER!!!
DANGER!!!  Expected open ports and processes:
DANGER!!!  $(listening_ports_and_processes_ideal_scene)
DANGER!!!
DANGER!!!  Actual open ports and processes:
DANGER!!!  $(listening_ports_and_processes)
DANGER!!!
";  # and yell loudly if it does not match.
      # Note:  A "commands" promise could be used in
      # addition to "reports" to send a text message
      # to a sysadmin cell phone, or to feed
      # CRITICAL status to a monitoring system.
}
```

EXAMPLE 126

Filename: 220-1850_Security.__Configure_sshd,_stub.cf

```
bundle agent example {
  vars:
```

```
      "sshd[Protocol]"                                string => "2";
      "sshd[X11Forwarding]"                           string => "yes";
      "sshd[UseDNS]"                                  string => "no";

  methods:
      "any"    usebundle => edit_sshd("example.sshd");
}


bundle agent edit_sshd(params) {

  vars:
      "index" slist => getindices("$(params)");

  reports:
    cfengine::
      "$(index) :  $($(params)[$(index)])";
}
```

Chapter 27. More Examples

EXAMPLE 127

Filename: 230-1910_More_Examples.__WordPress_Diego.cf

```
#Install WordPress:
#       1. Install Infrastructure:
#               1.1. Install httpd and mod_php and PHP MySQL client.
#               1.2. Install MySQL server.
#                       1.2.1. Create WordPress User in MySQL.
#                       1.2.2. Create WordPress Database in MySQL.
#               1.3. Make sure httpd and MySQL servers are running.
#       2. Install the PHP application (WordPress)
#               2.1. Download tarball with the latest version of WordPress PHP application.
#               2.2. Extract it into the httpd document root where it can be run by the Web server.
#               2.3. Create WordPress config file wp-config.php from wp-config-sample.php that's shipped with WordPress.
#               2.4. Tweak wp-config.php to put in the data needed to establish database connection (db name, db username and pass
#

bundle agent example {

  methods:

      "Install WordPress"
            usebundle => wordpress_install;

}

bundle agent wordpress_install
{
  vars:
      "wp_config[DB_NAME]"      string => "wordpress";
      "wp_config[DB_USER]"      string => "wordpress";
      "wp_config[DB_PASSWORD]"  string => "lopsa10linux";
      "wp_config[htmlroot]"     string => "/var/www/html";
      "wp_config[tarfile]"      string => "/root/wordpress-latest.tar.gz";
```

```
      "wp_config[wp_dir]"       string => "$(wp_config[htmlroot])/wordpress";
      "wp_config[wp_config]"    string => "$(wp_config[wp_dir])/wp-config.php";
      "wp_config[wp_cfgsample]" string => "$(wp_config[wp_dir])/wp-config-sample.php";

  methods:

      "Infrastructure"

               handle => "wp_infrastructure",
              comment => "httpd, PHP, MySQL and everything in-between",
            usebundle => wp_infrastructure;

      "Application"

               handle => "wp_application",
              comment => "Install and Configure WordPress PHP app",
            usebundle => wp_application;

}

bundle agent wp_application {

  methods:
      "any" usebundle => wp_tarball_is_present("wordpress_install.wp_config");
      "any" usebundle => wp_tarball_is_unrolled("wordpress_install.wp_config");
      "any" usebundle => wp_config_exists("wordpress_install.wp_config");
      "any" usebundle => wp_is_properly_configured("wordpress_install.wp_config");
}

bundle agent wp_infrastructure {

  methods:
      "any" usebundle => wp_packages_installed("wordpress_install.wp_config");
      "any" usebundle => wp_services_up("wordpress_install.wp_config");
      "any" usebundle => wp_mysql_configuration("wordpress_install.wp_config");
      #"any" usebundle => wp_allow_http_inbound("wordpress_install.wp_config");
}

################################################
```

```
bundle agent wp_packages_installed(params)
{
  vars:
    debian::
      "desired_package" slist => {
                                   "apache2",
                                   "php5",
                                   "php5-mysql",
                                   "mysql-server",
      };
    redhat::
      "desired_package" slist => {
                                   "httpd",
                                   "php",
                                   "php-mysql",
                                   "mysql-server",
      };
  packages:
      "$(desired_package)"
               handle => "install_packages",
              comment => "Install needed packages",
       package_policy => "add",
package_architectures => { "x86_64" },
       package_method => generic,
              classes => if_repaired("packages_added");

  commands:
    packages_added.debian::
      "/usr/sbin/service httpd graceful"
              comment => "Restarting httpd so it can pick up any new modules.";

  commands:
    packages_added.redhat::
      "/sbin/service httpd graceful"
              comment => "Restarting httpd so it can pick up any new modules.";
}

###############################################
```

```
bundle agent wp_services_up(params)
{
  processes:
      "mysqld" restart_class => "start_mysqld";
    redhat::
      "httpd"  restart_class => "start_httpd";
    debian::
      "apache2"  restart_class => "start_httpd";

  commands:
    start_mysqld&debian::
      "/usr/sbin/service mysqld start";

    start_mysqld&redhat::
      "/sbin/service mysqld start";

    start_httpd&redhat::
      "/sbin/service httpd start";

    start_httpd&debian::
      "/usr/sbin/service httpd start";
}

################################################

bundle agent wp_tarball_is_present(params)
{

  classes:
      "wordpress_tarball_is_present"
               handle => "check_for_WP_tarball",
              comment => "check if we already have the WP tarball",
           expression => fileexists("$($(params)[tarfile])");

  commands:
    !wordpress_tarball_is_present::
      "/usr/bin/wget -q --timeout=10 -O $($(params)[tarfile]) http://wordpress.org/latest.tar.gz"
               handle => "download_WP_tarball",
```

```
              classes => if_repaired("we_have_WP_tarball"),
              comment => "Downloading latest version of WordPress.",
              action => logme("promise download_WP_tarball");
}

###############################################

bundle agent wp_tarball_is_unrolled(params)
{

  classes:
      "wordpress_directory_is_present" expression => fileexists("$($(params)[htmlroot])/wordpress/");

  commands:
    we_have_WP_tarball&(!wordpress_directory_is_present)::
      "/bin/tar -C $($(params)[htmlroot]) -xzf $($(params)[tarfile])"
               handle => "extract_tarball",
              depends_on => { "download_WP_tarball" },
              comment => "Unroll wordpress tarball to HTML document root";
}

###############################################

bundle agent wp_mysql_configuration(params)
{

  commands:
      "/usr/bin/mysql -u root -e \"
CREATE DATABASE IF NOT EXISTS $($(params)[DB_NAME]);
GRANT ALL PRIVILEGES ON $($(params)[DB_NAME]).*
TO '$($(params)[DB_USER])'@localhost
IDENTIFIED BY '$($(params)[DB_PASSWORD])';
FLUSH PRIVILEGES;\"
"
               handle => "setup_db",
              comment => "Create DB, DB user, and access credentials";

}
```

```
###############################################

bundle agent wp_config_exists(params)
{

  classes:
      "wordpress_config_file_exists"
               handle => "check_WP_config_file_there",
              comment => "Check if WP config file is present",
           expression => fileexists("$($(params)[wp_config])");

  files:
    !wordpress_config_file_exists::
      "$($(params)[wp_config])"
               handle => "copy_WP_config_file",
              comment => "Copy WP config file from config sample file",
            copy_from => local_cp("$($(params)[wp_cfgsample])"),
                perms => m("a+r");
}

###############################################

bundle agent wp_is_properly_configured(params)
{
  vars:
      "wpparams" slist => getindices("$(params)");

  files:
      "$($(params)[wp_config])"
               handle => "configure_wordpress",
              comment => "Make sure wp-config.php is properly configured",
            edit_line => replace_or_add("define\('$(wpparams)', *'(?!$($(params)[$(wpparams)])).*",
                                        "define('$(wpparams)', '$($(params)[$(wpparams)])');");
}

###############################################

bundle agent wp_allow_http_inbound(params)
{
```

```
  commands:
    iptables_edited::
      "/sbin/service iptables stop"
              comment => "Stopping iptables to allow inbound HTTP connections";
}

body common control
{
               inputs =>          { "libraries/cfengine_stdlib.cf" };

}


body action logme(x)
{
           log_string => "$(sys.date) $(x)";

             log_kept => "/var/log/cfengine_keptlog.log";
         log_repaired => "/var/log/cfengine_replog.log";
           log_failed => "/var/log/cfengine_faillog.log";

}
```

EXAMPLE 128

Filename: 230-1950_More_Examples.__Setting_the_environment_for_a_command.cf

```
body agent control
{
          environment => { "A=123", "B=456", "PGK_PATH=/tmp"};
}

############################################

bundle agent example
{
  commands:
```

```
      "/usr/bin/env"
               handle => "env_cmd",
              comment => "Demonstrate setting up the environment for a command";
}
```

EXAMPLE 129

Filename: 230-1960_More_Examples.__Delete_repo_comments_from_CentOS_repo_and_exclude_postgresql_packages.cf

```
# edit CentOS repo file in /etc/yum.repos.d to exclude
# Postgres packages from downloads/updates (because I want
# to get them from the Postgres.org repo).
#
# Note: I have to strip out the CentOS repo comments otherwise
# due to the nature of the section-aware editing, the comments
# end up in the middle of the previous section.

bundle agent example {

  methods:
      "any"
            usebundle => exclude_postgresql_from_CentOS_repo;

}


bundle agent exclude_postgresql_from_CentOS_repo {

  files:
      "/etc/yum.repos.d/CentOS-Base.repo"
            edit_line => DeleteRepoComments,
               handle => "files__CentOS_Base_repo__strip_repo_comments",
              comment => "Remove CentOS remarks about the repos, they mess up section editing because they are placed outside the

  files:
      "/etc/yum.repos.d/CentOS-Base.repo"
            edit_line => AppendIfNoLine("exclude=postgresql*$(const.n)# Get Postgres packages from PGDG, not CentOS.$(const.n)"),
              comment => "Exclude postgresql packages in CentOS [base] and [update] repos; we'll get them from Postgres Global Dev
```

```
}

#############################################################

bundle edit_line DeleteRepoComments {
  delete_lines:
      "#released updates.*";
      "#packages used/produced in the build but not released";
      "#additional packages that may be useful";
      "#additional packages that extend functionality of existing packages";
      "#contrib - packages by Centos Users";
}

#############################################################

bundle edit_line AppendIfNoLine(parameter) {

  insert_lines:
      "$(parameter)"
        select_region => MyINISection("base");


  insert_lines:
      "$(parameter)"
        select_region => MyINISection("updates");

}

#############################################################



body select_region MyINISection(x)

{
         select_start => "\[$(x)\]";
           select_end => "\[.*\]";

      # This assumes a file format like:
```

```
      #
      # [section 1]
      #
      # lines....
      #
      # [section 2]
      #
      # lines... etc
}
```

EXAMPLE 130

Filename: 230-1970_More_Examples.__Install_php_pecl_module.cf

```
# Install pecl_http PHP module to provide HttpRequest class to our PHP Web app:
#   - run "pecl install pecl_http" and set SELinux type on http.so to textrel_shlib_t
#   - edit /etc/php.ini to tell php to dynamically load http.so


body common control

{
      inputs          => { "libraries/cfengine_stdlib.cf" };
}

#########################################################################


bundle agent php_pecl_http_extension_is_installed_and_integrated {

  files:

      "/etc/php.ini"

            edit_line => tell_php_to_load_http_extension;



  classes:
```

```
      "pecl_http_module_exists" expression => fileexists("/usr/lib64/php/modules/http.so");


  commands:

    !pecl_http_module_exists::

      "/usr/bin/yes ' ' | /usr/bin/pecl install pecl_http && /usr/bin/chcon -t textrel_shlib_t /usr/lib64/php/modules/http.so"  #

              contain => in_shell;  # so we can do a pipeline
}


#####################################################################


bundle edit_line tell_php_to_load_http_extension {
  vars:
      "dynamically_load_http_module" string => "extension=http.so   ; In-house module XYZ requires HttpRequest which is provided b

  insert_lines:
      "$(dynamically_load_http_module)"
             location => in_Dynamic_Extensions_section;
}


#####################################################################


body location in_Dynamic_Extensions_section

{
         before_after => "after";
           first_last => "first";
 select_line_matching => "; Dynamic Extensions ;";
}

# reloading the httpd if php.ini was edited
```

```
# is left as an exercise for the reader
# hint: if_repaired


# TODO: instead of using select_line_matching, use begin and end select region
# to insert the extension=http.so line into /etc/php.ini at the end of instead
# of in the middle of the Dynamic Extensions block so it looks cleaner.
```

Chapter 28. EC2 System Provisioning and Integration Demo

Note: The following was a demo given at CasITConf 2011. Tighter integration with AWS may now exist in CFEngine.

Purpose: proof of concept of multi-node deployment, configuration and integration on Amazon EC2 cloud using CFEngine.

We start on a Ubuntu VM with Amazon EC2 CLI tools installed, courtesy of Florian Drescher of CloudTrainings.com. We configure it with our EC2 credentials.

Then we install CFEngine 3.1.4. Then we run casit_demo.cf to instantiate two servers, "web" and "db", and to install CFEngine 3.1.4 onto them. We then use that CFEngine to install Apache httpd and mod_php and WordPress PHP application on "web" and MySQL server on "db"; and we integrate the two servers. End result: a working instance of WordPress deployed across two servers.

Video: http://www.verticalsysadmin.com/cfengine/casit/

EXAMPLE 131

Filename: 240-2020_EC2_system_provisioning_and_integration.__demo.cf

```
##############################################
bundle common global_vars {


      vars: "desired_servers" slist => {
                                         "web",
                                         "db",
      };
}

##############################################

body common control
{

       bundlesequence => {
                            "global_vars",
                            "no_hosts_known_to_ssh",
```

```
                           servers_provisioned(@{global_vars.desired_servers}),
                           hosts_file_distributed_and_loaded(@{global_vars.desired_servers}),
                           wordpress_installer_distributed_and_run(@{global_vars.desired_servers}),
       };


               inputs =>          { "libraries/cfengine_stdlib.cf" };

}

###############################################


bundle agent no_hosts_known_to_ssh
{
  files:
      "/home/user/.ssh/known_hosts"
               delete => tidy;

      # I don't want to see SSH complaints about keys having changed

}

###############################################


bundle agent servers_provisioned(desired_servers)
{

  classes:
      "$(desired_servers)_up"   expression => fileexists("/home/user/cfengine_ec2/servers/$(desired_servers)");
      "$(desired_servers)_down" not        => fileexists("/home/user/cfengine_ec2/servers/$(desired_servers)");


  reports:
    linux::
      "$(desired_servers) has been provisioned"
           ifvarclass => canonify("$(desired_servers)_up");
```

```
      commands: "/home/user/cfengine_ec2/start_micro_instance.sh $(desired_servers) > /home/user/cfengine_ec2/servers/$(desired_se
           ifvarclass => canonify("$(desired_servers)_down"),
              contain => in_shell;
}


###############################################

bundle agent hosts_file_distributed_and_loaded(desired_servers)
{
      commands: "/usr/bin/scp -o StrictHostKeyChecking=no -i /home/user/ec2/mysshkey_key.pem /etc/hosts ec2-user@$(desired_servers
              contain => in_shell;
}

###############################################

bundle agent wordpress_installer_distributed_and_run(desired_servers)
{
      commands: "/usr/bin/scp -o StrictHostKeyChecking=no -i /home/user/ec2/mysshkey_key.pem /home/user/cfengine_ec2/example102_wo
              contain => in_shell;
}
```

EXAMPLE 132

Filename: 240-2030_EC2_system_provisioning_and_integration.__wordpress_installation.cf

```
#Install WordPress:
#       1. Install Infrastructure:
#               1.1. Install httpd and mod_php and PHP MySQL client.
#               1.2. Install MySQL server.
#                       1.2.1. Secure MySQL
#                       1.2.2. Create WordPress User in MySQL.
#                       1.2.3. Create WordPress Database in MySQL.
#               1.3. Make sure httpd and MySQL servers are running.
#       2. Install the PHP application (WordPress)
#               2.1. Download tarball with the latest version of WordPress PHP application.
#               2.2. Extract it into the httpd document root where it can be run by the Web server.
#               2.3. Create WordPress config file wp-config.php from wp-config-sample.php that's shipped with WordPress.
#               2.4. Tweak wp-config.php to put in the data needed to establish database connection (db name, db username and pass
```

```
#

body common control
{

       bundlesequence => {
                           "wordpress_install",
       };


               inputs =>          { "libraries/cfengine_stdlib.cf" };

}



bundle agent wordpress_install
{
  vars:
      "wp_config[DB_HOST]"           string => "db";
      "wp_config[DB_NAME]"           string => "wordpress";
      "wp_config[DB_USER]"           string => "wordpress";
      "wp_config[DB_PASSWORD]"       string => "L0PSA_2011_Linux";
      "wp_config[DB_ROOT_PASSWORD]"  string => "Linux_2011_L0PSA";
      "wp_config[htmlroot]"          string => "/var/www/html";
      "wp_config[tarfile]"           string => "/root/wordpress-latest.tar.gz";
      "wp_config[wp_dir]"            string => "$(wp_config[htmlroot])/wordpress";
      "wp_config[wp_config]"         string => "$(wp_config[wp_dir])/wp-config.php";
      "wp_config[wp_cfgsample]"      string => "$(wp_config[wp_dir])/wp-config-sample.php";

  methods:

    web::
      "any" usebundle => wp_web_front_end_packages_installed("wordpress_install.wp_config");
      "any" usebundle => wp_web_front_end_services_up("wordpress_install.wp_config");
      "any" usebundle => wp_tarball_is_present("wordpress_install.wp_config");
      "any" usebundle => wp_tarball_is_unrolled("wordpress_install.wp_config");
      "any" usebundle => wp_config_exists("wordpress_install.wp_config");
      "any" usebundle => wp_is_properly_configured("wordpress_install.wp_config");
```

```
      #  "any" usebundle => wp_allow_http_inbound("wordpress_install.wp_config");

    db::
      "any" usebundle => wp_db_back_end_packages_installed("wordpress_install.wp_config");
      "any" usebundle => wp_db_back_end_services_up("wordpress_install.wp_config");
      "any" usebundle => wp_mysql_is_secured("wordpress_install.wp_config");
      "any" usebundle => wp_mysql_configuration("wordpress_install.wp_config");
}


###############################################

bundle agent wp_web_front_end_packages_installed(params)
{
  vars:

    debian::
      "desired_package" slist => {
                                   "apache2",
                                   "php5",
                                   "php5-mysql",
      };

    redhat::
      "desired_package" slist => {
                                   "httpd",
                                   "php",
                                   "php-mysql",
      };
  packages:
      "$(desired_package)"
       package_policy => "add",
       package_method => generic,
              classes => if_repaired("packages_added");

  commands:
    packages_added&&redhat::
      "/sbin/service httpd graceful"
              comment => "Restarting httpd so it can pick up new modules.";
```

```
    packages_added&&debian::
      "/usr/sbin/service apache2 graceful"
              comment => "Restarting httpd so it can pick up new modules.";
}

###############################################
bundle agent wp_db_back_end_packages_installed(params)
{
  vars:

      "desired_package" slist => {
                                  "mysql-server",
      };

  packages:
      "$(desired_package)"
       package_policy => "add",
       package_method => generic,
              classes => if_repaired("packages_added");

}


###############################################

bundle agent wp_web_front_end_services_up(params)
{

  processes:
    redhat::
      "httpd"  restart_class => "start_httpd_redhat";

    ubuntu::
      "apache2"  restart_class => "start_httpd_ubuntu";


  commands:
    start_httpd_redhat::
```

```
      "/sbin/service httpd start";

    start_httpd_ubuntu::
      "/usr/sbin/service apache2 start";

}

############################################

bundle agent wp_db_back_end_services_up(params)
{
  processes:
    redhat::
      "mysqld" restart_class => "start_mysqld_redhat";

    ubuntu::
      "mysqld" restart_class => "start_mysqld_ubuntu";


  commands:
    start_mysqld_redhat::
      "/sbin/service mysqld start";

    start_mysqld_ubuntu::
      "/usr/sbin/service mysqld start";


}

############################################

bundle agent wp_tarball_is_present(params)
{

  classes:
      "wordpress_tarball_is_present" expression => fileexists("$($(params)[tarfile])");

  reports:
    wordpress_tarball_is_present::
```

```
      "WordPress tarball is on disk.";

  commands:
    !wordpress_tarball_is_present::
      "/usr/bin/wget -q -O $($(params)[tarfile]) http://wordpress.org/latest.tar.gz"
             comment => "Downloading latest version of WordPress.";
}

###############################################

bundle agent wp_tarball_is_unrolled(params)
{

  classes:
      "wordpress_directory_is_present" expression => fileexists("$($(params)[htmlroot])/wordpress/");

  reports:
    wordpress_directory_is_present::
      "WordPress directory is present.";

  commands:
    !wordpress_directory_is_present::
      "/bin/tar -C $($(params)[htmlroot]) -xzf $($(params)[tarfile])"
             comment => "Unrolling wordpress tarball to $($(params)[htmlroot]).";
}


###############################################

bundle agent wp_mysql_is_secured(params)
{

      #  remove the test databases and anonymous user created by default and set the MySQL root password
      # at first I tried to use mysql_secure_installation, but it would not take the root password in a
      # pipeline, error: "stty: standard input: Invalid argument"
      # Here is is how I tried to secure the database:
      #commands:
      #  "/usr/bin/printf '\ny\n$($(params)[DB_ROOT_PASSWORD])\n$($(params)[DB_ROOT_PASSWORD])\ny\ny\ny\ny\n' | /usr/bin/mysql_se
      #     contain => in_shell;
```

```
      # instead let's just do what mysql_secure_installation does, so we can do it non-interactively:
      # - remove anonymous users
      # - remove remote root
      # - remove test database
      # - remove privileges on test database
      # - reload privilege tables



  commands:
      "/usr/bin/mysql -u root -e \"
DELETE FROM mysql.user WHERE User='';
DELETE FROM mysql.user WHERE User='root' AND Host!='localhost';
DROP DATABASE test;
DELETE FROM mysql.db WHERE Db='test' OR Db='test\\_%';
FLUSH PRIVILEGES;\"
";

}

###############################################

bundle agent wp_mysql_configuration(params)
{

  commands:
      "/usr/bin/mysql -u root -e \"
CREATE DATABASE IF NOT EXISTS $($(params)[DB_NAME]);
GRANT ALL PRIVILEGES ON $($(params)[DB_NAME]).*
TO '$($(params)[DB_USER])'@'web'
IDENTIFIED BY '$($(params)[DB_PASSWORD])';
FLUSH PRIVILEGES;\"
";

}

###############################################
```

```
bundle agent wp_config_exists(params)
{

  classes:
      "wordpress_config_file_exists"
           expression => fileexists("$($(params)[wp_config])");

  reports:
    wordpress_config_file_exists::
      "WordPress config file $($(params)[wp_config]) is present";

  commands:
    !wordpress_config_file_exists::
      "/bin/cp -p $($(params)[wp_cfgsample]) $($(params)[wp_config])";
}

###############################################

bundle agent wp_is_properly_configured(params)
{
  vars:
      "wpparams" slist => getindices("$(params)");

  files:
      "$($(params)[wp_config])"
           edit_line => replace_or_add("define\('$(wpparams)', *'(?!$($(params)[$(wpparams)])).*",
                                       "define('$(wpparams)', '$($(params)[$(wpparams)])');");
}

###############################################

bundle agent wp_allow_http_inbound(params)
{
  files:
    redhat:: # tested on RHEL only, file location may vary based on Linux distro or OS
      "/etc/sysconfig/iptables"
           edit_line => insert_HTTP_allow_rule_before_the_accept_established_tcp_conns_rule,
             comment => "insert HTTP allow rule into /etc/sysconfig/iptables",
             classes => if_repaired("iptables_edited");
```

```
  commands:
    iptables_edited::
      "/sbin/service iptables restart"
              comment => "Restarting iptables to load new config";
}

bundle edit_line insert_HTTP_allow_rule_before_the_accept_established_tcp_conns_rule(params)
{
  vars:
      "http_rule" string => "-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT";

  insert_lines:
      "$(http_rule)"
             location => before_the_accept_established_tcp_conns_rule;
}

body location before_the_accept_established_tcp_conns_rule
{
         before_after => "before";
           first_last => "first";
 select_line_matching => "^-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT.*";
}

bundle edit_line replace_or_add(pattern,line)
# Diego's.
# Replace a pattern in a file with a single line.
# If the pattern is not found, add the line to the file.
# The pattern must match the whole line (it is automatically
# anchored to the start and end of the line) to avoid
# ambiguity.

{
  replace_patterns:
      "^${pattern}$"
         replace_with => value("${line}"),
              classes => always("replace_done");

  insert_lines:
```

```
    replace_done::
      "${line}";
}


body classes always(x)
# Diego's.
# Define a class no matter what the outcome of the promise is

{
     promise_repaired => { "$(x)" };
         promise_kept => { "$(x)" };
        repair_failed => { "$(x)" };
        repair_denied => { "$(x)" };
       repair_timeout => { "$(x)" };
}




# Todo:
#
#
# MySQL:
# - submit a patch to the MySQL folks to add a non-interactive option to /usr/bin/mysql_secure_installation
# - change the root password using /usr/bin/mysqladmin -u root password 'new-password'
# - I've hardcoded the web server name as 'web', in allowing connects.  make this more flexible.  (how?)
#
# httpd:
# - instead of hardcoding "/var/www/html", derive httpd document root on the fly from httpd config file
#   using Function readstringlist. (If it's Apache, look for DocumentRoot)
```

Chapter 29. Class Examples

EXAMPLE 133

Filename: 250-2050_Class_Examples.__Parsing_readtcp_output_to_set_a_class.cf

```
# this policy runs on an haproxy load balancer
# we check a list of servers (webhosts_list)
# to tst that they are up, and if they are up,
# we make sure our haproxy configuration includes
# them.
#
# This allows us to dynamically integrate new
# Web servers into the round robin.
#
# Reference: https://cfengine.com/forum/read.php?5,19571

bundle agent load_balancer_configured_with_live_webhosts(webhosts_list)
{

  reports:
    load_balancer_hosts::
      "I am a load balancer!!";



      # set variable containing HTTP response from each web server
  vars:

      "my80"
               string => readtcp("$(webhosts_list)","80","GET /index.php
HTTP/1.1$(const.r)$(const.n)Host: $(webhosts_list)$(const.r)$(const.n)$(const.r)$(const.n)",20);


      # set server_ok class if response contains HTTP 200 OK

  classes:
```

```
      "server_ok"
           expression => regcmp(".*200 OK.*\n.*","$(my80)");


      # make sure each live (OK) web server is in the haproxy.conf

  files:

    server_ok&load_balancer_hosts::

      "/etc/haproxy.conf"
            edit_line => append_if_no_line(" server $(webhosts_list)
$(webhosts_list):80 maxconn 32");


}
```

EXAMPLE 134

Filename: 250-2070_Class_Examples.__Set_a_custom_class_based_on_hostname_pattern.cf

```
bundle agent example {

  classes:
      "italy"
           expression => classmatch("^mil.*$");  # Milan

  reports:
    italy::
      "Italy";


}
```

EXAMPLE 135

Filename: 250-2080_Class_Examples.__Set_a_custom_class_based_on_hostname_pattern_and_use_a_bundle_of_promises_based_on_that_class.cf

```
# a simple, all in one file, example of configuring
```

```
# different policies per-country based on hostname naming pattern


bundle common define_global_classes {

      classes: "italy"   expression => classmatch("mil.*");

      classes: "germany" expression => classmatch("berl.*");

}




bundle agent example {

  vars:
      "country"
                slist => { "italy", "germany" };

  methods:
      "any"
           usebundle => "$(country)",
          ifvarclass => "$(country)";
}


bundle agent italy   { commands: "/bin/echo I love Milan";  }
bundle agent germany { commands: "/bin/echo I love Berlin"; }
```

EXAMPLE 136

Filename: 250-2100_Class_Examples.__Persistent_class.cf

```
# FIXME - this example needs work

bundle agent example

{
```

```
  commands:

    ok_but_check_later::

      "/bin/echo YELLOW ALERT (condition \"ok_but_check_later\")";


  commands:

    cannot_repair_promise_DANGER_DANGER::

      "/bin/echo SHIELDS UP, RED ALERT (condition \"cannot_repair_promise_DANGER_DANGER\"";


  commands:

      "/bin/true"

              classes => set_persistent_class_based_on_promise_repair_outcome("ok_but_check_later","cannot_repair_promise_DANGER_D


}


###############################################

body classes set_persistent_class_based_on_promise_repair_outcome(if,else)

# if promise repair succeeded, set a persistent
# class for 10 minutes called "ok_but_check_later";
# else if promise repair failed, set persistent class
# "cannot_repair_promise_DANGER_DANGER".

{
     promise_repaired => { "$(if)" };
        repair_failed => { "$(else)" };
         persist_time => "10"; # in minutes
}
```

```
#############################################################################

# Here is an example of how you might use a persistent class
#
# 1. You detect a rootkit from some filesearch and you want
# to delete it immediately, but the danger is maybe not over.
#
# 2. You define persistent class for the next hour DEFCON1
# which activates repeated scans of the filesystem looking
# for trouble as you suspect you might be under attack.
```

Introduction

CFEngine has some special variables.

You can see the whole list in section "Special Variables" in the reference manual, but here is a taste of them.

Chapter 30. Special Variables

EXAMPLE 137

Filename: 260-2420_Special_Variables.__Const.cf

```
bundle agent example {

  reports:
    cfengine::
      "
one$(const.r)two
";

  !any::
      "A carriage return character is $(const.r)The carriage has returned!!! the king is back.";
      "A report with a$(const.t)tab in it";

      "The value of variable named $(const.dollar)(const.dollar) is $(const.dollar)";

      "The value of variable named \$(const.dollar) is $(const.dollar)";
      # backslash does not work to stop interpolation of the variable

      "A newline with either $(const.n) or with $(const.endl) is ok";
      "But a string with \n in it does not have a newline!";



}
```

Special variables with scope "edit"

Special variables with scope "edit" allow you to access information about editing promises during their execution.

edit.filename

edit.filename
: Points to the filename of the file currently making an edit promise.

EXAMPLE 138

Filename: 260-2440_Special_Variables.__Edit.cf

```
# INPUT
# Put a few text files in /tmp (ending in .txt), and put
# the line "hello world" in one of them.
#
# CFEngine will report which file contains the line "hello world".
#

#############################################################

bundle agent example

{
  files:

      "/tmp/.*.txt"
            handle => "cfengine_grep_dash_l",
            comment => "Return files matching given string",
            edit_line => grep_dash_l("hello world");
}

#############################################################



bundle edit_line grep_dash_l(regex)
{
  classes:

      "ok" expression => regline("$(regex)","$(edit.filename)");
```

```
  reports:

    ok::

      "File $(edit.filename) has a line with \"$(regex)\" in it";

}
```

EXAMPLE 139

Filename: 260-2460_Special_Variables.__Match._While_searching_for_files.cf

```
bundle agent example
{

      # INPUT
      # Please create the following files before running this example:
      # /tmp/cf2_test1
      # /tmp/cf3_test2

  files:

      "/tmp/(cf[23])_(.*)"
            edit_line => report_match_variables("$(match.0) $(match.1) $(match.2)");

}

bundle edit_line report_match_variables(data) {

  reports:
    cfengine::

      "$(data)";

      # OUTPUT
      # You should see:
      # R: /tmp/cf2_test1 cf2 test1
      # R: /tmp/cf3_test2 cf3 test2
```

```
}
```

EXAMPLE 140

Filename: 260-2470_Special_Variables.__Match._While_editing_a_file.cf

```
#   INPUT
#
#   File /tmp/cf3_test containing a Unix shell style comment:
#
#   one
#   two
#   three
#   # four
#   five
#   six


############################################################

bundle agent example

{

  files:

      "/tmp/cf3_test"

            create    => "true",
            edit_line => replace_shell_comments_with_C_comments;
}

############################################################

bundle edit_line replace_shell_comments_with_C_comments
{

  replace_patterns:
```

```
      "#(.*)"

         replace_with => C_comment;

}


#############################################################

body replace_with C_comment

{
        replace_value => "/* $(match.1) */"; # backreference 0
          occurrences => "all";  # first, last all
}

#############################################################
#
#   OUTPUT
#
#   File /tmp/cf3_test should now have a C style comment:
#
#   one
#   two
#   three
#   /*  four */
#   five
#   six
```

EXAMPLE 141

Filename: 260-2480_Special_Variables.__Mon._Report_environmental_conditions.cf

```
# report environmental conditions

bundle agent example {

  reports:
```

```
    linux::
      "
Metric     CurVal, AvgVal, StDev
cfengine_in      ${mon.value_cfengine_in},  ${mon.av_cfengine_in},  ${mon.dev_cfengine_in}
cfengine_out     ${mon.value_cfengine_out},  ${mon.av_cfengine_out},  ${mon.dev_cfengine_out}
cpu              ${mon.value_cpu},  ${mon.av_cpu},  ${mon.dev_cpu}
cpu0             ${mon.value_cpu0},  ${mon.av_cpu0},  ${mon.dev_cpu0}
cpu1             ${mon.value_cpu1},  ${mon.av_cpu1},  ${mon.dev_cpu1}
cpu2             ${mon.value_cpu2},  ${mon.av_cpu2},  ${mon.dev_cpu2}
cpu3             ${mon.value_cpu3},  ${mon.av_cpu3},  ${mon.dev_cpu3}
diskfree         ${mon.value_diskfree},  ${mon.av_diskfree},  ${mon.dev_diskfree}
dns_in           ${mon.value_dns_in},  ${mon.av_dns_in},  ${mon.dev_dns_in}
dns_out          ${mon.value_dns_out},  ${mon.av_dns_out},  ${mon.dev_dns_out}
ftp_in           ${mon.value_ftp_in},  ${mon.av_ftp_in},  ${mon.dev_ftp_in}
ftp_out          ${mon.value_ftp_out},  ${mon.av_ftp_out},  ${mon.dev_ftp_out}
icmp_in          ${mon.value_icmp_in},  ${mon.av_icmp_in},  ${mon.dev_icmp_in}
icmp_out         ${mon.value_icmp_out},  ${mon.av_icmp_out},  ${mon.dev_icmp_out}
irc_in           ${mon.value_irc_in},  ${mon.av_irc_in},  ${mon.dev_irc_in}
irc_out          ${mon.value_irc_out},  ${mon.av_irc_out},  ${mon.dev_irc_out}
loadavg          ${mon.value_loadavg},  ${mon.av_loadavg},  ${mon.dev_loadavg}
messages         ${mon.value_messages},  ${mon.av_messages},  ${mon.dev_messages}
netbiosdgm_in    ${mon.value_netbiosdgm_in},  ${mon.av_netbiosdgm_in},  ${mon.dev_netbiosdgm_in}
netbiosdgm_out   ${mon.value_netbiosdgm_out},  ${mon.av_netbiosdgm_out},  ${mon.dev_netbiosdgm_out}
netbiosns_in     ${mon.value_netbiosns_in},  ${mon.av_netbiosns_in},  ${mon.dev_netbiosns_in}
netbiosns_out    ${mon.value_netbiosns_out},  ${mon.av_netbiosns_out},  ${mon.dev_netbiosns_out}
netbiosssn_in    ${mon.value_netbiosssn_in},  ${mon.av_netbiosssn_in},  ${mon.dev_netbiosssn_in}
netbiosssn_out   ${mon.value_netbiosssn_out},  ${mon.av_netbiosssn_out},  ${mon.dev_netbiosssn_out}
nfsd_in          ${mon.value_nfsd_in},  ${mon.av_nfsd_in},  ${mon.dev_nfsd_in}
nfsd_out         ${mon.value_nfsd_out},  ${mon.av_nfsd_out},  ${mon.dev_nfsd_out}
otherprocs       ${mon.value_otherprocs},  ${mon.av_otherprocs},  ${mon.dev_otherprocs}
rootprocs        ${mon.value_rootprocs},  ${mon.av_rootprocs},  ${mon.dev_rootprocs}
smtp_in          ${mon.value_smtp_in},  ${mon.av_smtp_in},  ${mon.dev_smtp_in}
smtp_out         ${mon.value_smtp_out},  ${mon.av_smtp_out},  ${mon.dev_smtp_out}
ssh_in           $(mon.value_ssh_in),  ${mon.av_ssh_in},  ${mon.dev_ssh_in}
ssh_out          ${mon.value_ssh_out},  ${mon.av_ssh_out},  ${mon.dev_ssh_out}
syslog           ${mon.value_syslog},  ${mon.av_syslog},  ${mon.dev_syslog}
tcpack_in        ${mon.value_tcpack_in},  ${mon.av_tcpack_in},  ${mon.dev_tcpack_in}
tcpack_out       ${mon.value_tcpack_out},  ${mon.av_tcpack_out},  ${mon.dev_tcpack_out}
tcpfin_in        ${mon.value_tcpfin_in},  ${mon.av_tcpfin_in},  ${mon.dev_tcpfin_in}
```

```
tcpfin_out       ${mon.value_tcpfin_out},  ${mon.av_tcpfin_out},  ${mon.dev_tcpfin_out}
tcpmisc_in       ${mon.value_tcpmisc_in},  ${mon.av_tcpmisc_in},  ${mon.dev_tcpmisc_in}
tcpmisc_out      ${mon.value_tcpmisc_out},  ${mon.av_tcpmisc_out},  ${mon.dev_tcpmisc_out}
tcpsyn_in        ${mon.value_tcpsyn_in},  ${mon.av_tcpsyn_in},  ${mon.dev_tcpsyn_in}
tcpsyn_out       ${mon.value_tcpsyn_out},  ${mon.av_tcpsyn_out},  ${mon.dev_tcpsyn_out}
temp0            ${mon.value_temp0},  ${mon.av_temp0},  ${mon.dev_temp0}
temp1            ${mon.value_temp1},  ${mon.av_temp1},  ${mon.dev_temp1}
temp2            ${mon.value_temp2},  ${mon.av_temp2},  ${mon.dev_temp2}
temp3            ${mon.value_temp3},  ${mon.av_temp3},  ${mon.dev_temp3}
udp_in           ${mon.value_udp_in},  ${mon.av_udp_in},  ${mon.dev_udp_in}
udp_out          ${mon.value_udp_out},  ${mon.av_udp_out},  ${mon.dev_udp_out}
users            ${mon.value_users},  ${mon.av_users},  ${mon.dev_users}
webaccess        ${mon.value_webaccess},  ${mon.av_webaccess},  ${mon.dev_webaccess}
weberrors        ${mon.value_weberrors},  ${mon.av_weberrors},  ${mon.dev_weberrors}
www_in           ${mon.value_www_in},  ${mon.av_www_in},  ${mon.dev_www_in}
www_out          ${mon.value_www_out},  ${mon.av_www_out},  ${mon.dev_www_out}
wwws_in          ${mon.value_wwws_in},  ${mon.av_wwws_in},  ${mon.dev_wwws_in}
wwws_out         ${mon.value_wwws_out},  ${mon.av_wwws_out},  ${mon.dev_wwws_out}
";
}
```

EXAMPLE 142

Filename: 260-2490_Special_Variables.__Mon._React_to_environmental_conditions.cf

```
# report environmental conditions

bundle agent example {

  reports:
    linux::
      "Percent CPU utilization         ${mon.value_cpu}";
      "Percent CPU0 utilization         ${mon.value_cpu0}";
      "Percent CPU1 utilization         ${mon.value_cpu1}";

  classes:
      "CPUoverload"
           expression =>  isgreaterthan("$(mon.value_cpu)","80");
```

```
  reports:
    CPUoverload::
      "CPU utilization is over threshold!!!";

}
```

EXAMPLE 143

Filename: 260-2500_Special_Variables.__Sys._Report_sys_variables.cf

```
bundle agent example {

  reports:

    cfengine_3::

      "sys.arch: $(sys.arch)
sys.cdate: $(sys.cdate)
sys.cf_agent: $(sys.cf_agent)
sys.cf_execd: $(sys.cf_execd)
sys.cf_hub: $(sys.cf_hub)
sys.cf_key: $(sys.cf_key)
sys.cf_know: $(sys.cf_know)
sys.cf_monitord: $(sys.cf_monitord)
sys.cf_promises: $(sys.cf_promises)
sys.cf_report: $(sys.cf_report)
sys.cf_runagent: $(sys.cf_runagent)
sys.cf_serverd: $(sys.cf_serverd)
sys.cf_twin: $(sys.cf_twin)
sys.cf_version: $(sys.cf_version)
sys.class: $(sys.class)
sys.date: $(sys.date)
sys.domain: $(sys.domain)
sys.expires: $(sys.expires)
sys.exports: $(sys.exports)
sys.fqhost: $(sys.fqhost)
sys.fstab: $(sys.fstab)
sys.host: $(sys.host)
sys.interface: $(sys.interface)
```

```
sys.ipv4: $(sys.ipv4)
sys.ipv4[interface_name]: $(sys.ipv4[interface_name])
sys.ipv4_1[interface_name]: $(sys.ipv4_1[interface_name])
sys.ipv4_2[interface_name]: $(sys.ipv4_2[interface_name])
sys.ipv4_3[interface_name]: $(sys.ipv4_3[interface_name])
sys.key_digest: $(sys.key_digest)
sys.license_owner: $(sys.license_owner)
sys.licenses_granted: $(sys.licenses_granted)
sys.licenses_installtime: $(sys.licenses_installtime)
sys.long_arch: $(sys.long_arch)
sys.maildir: $(sys.maildir)
sys.nova_version: $(sys.nova_version)
sys.os: $(sys.os)
sys.ostype: $(sys.ostype)
sys.policy_hub: $(sys.policy_hub)
sys.release: $(sys.release)
sys.resolv: $(sys.resolv)
sys.uqhost: $(sys.uqhost)
sys.version: $(sys.version)
sys.windir: $(sys.windir)
sys.winprogdir: $(sys.winprogdir)
sys.winprogdir86: $(sys.winprogdir86)
sys.winsysdir: $(sys.winsysdir)
sys.workdir: $(sys.workdir)
";

}


# OUTPUT on my system, myhost.example.com
#    R: sys.arch: x86_64
#    sys.cdate: Sun_May_15_11_25_03_2011
#    sys.cf_agent: "/var/cfengine/bin/cf-agent"
#    sys.cf_execd: "/var/cfengine/bin/cf-execd"
#    sys.cf_hub: "/var/cfengine/bin/cf-hub"
#    sys.cf_key: "/var/cfengine/bin/cf-key"
#    sys.cf_know: "/var/cfengine/bin/cf-know"
#    sys.cf_monitord: "/var/cfengine/bin/cf-monitord"
#    sys.cf_promises: "/var/cfengine/bin/cf-promises"
```

```
#    sys.cf_report: "/var/cfengine/bin/cf-report"
#    sys.cf_runagent: "/var/cfengine/bin/cf-runagent"
#    sys.cf_serverd: "/var/cfengine/bin/cf-serverd"
#    sys.cf_twin: "/var/cfengine/bin/cf-agent"
#    sys.cf_version: 3.1.5
#    sys.class: linux
#    sys.date: Sun May 15 11:25:03 2011
#    sys.domain: example.com
#    sys.expires:
#    sys.exports: /etc/exports
#    sys.fqhost: myhost.example.com
#    sys.fstab: /etc/fstab
#    sys.host: myhost.example.com
#    sys.interface: venet0
#    sys.ipv4: 127.0.0.1
#    sys.ipv4[interface_name]: $(sys.ipv4[interface_name])
#    sys.ipv4_1[interface_name]: $(sys.ipv4_1[interface_name])
#    sys.ipv4_2[interface_name]: $(sys.ipv4_2[interface_name])
#    sys.ipv4_3[interface_name]: $(sys.ipv4_3[interface_name])
#    sys.key_digest: MD5=c4348f13c55363743ba5544a7808dff5
#    sys.license_owner: $(sys.license_owner)
#    sys.licenses_granted: $(sys.licenses_granted)
#    sys.licenses_installtime: $(sys.licenses_installtime)
#    sys.long_arch: linux_x86_64_2_6_18_028stab070_4__1_SMP_Tue_Aug_17_18_32_47_MSD_2010
#    sys.maildir: /var/spool/mail
#    sys.nova_version: $(sys.nova_version)
#    sys.os: linux
#    sys.ostype: linux_x86_64
#    sys.policy_hub: $(sys.policy_hub)
#    sys.release: 2.6.18-028stab070.4
#    sys.resolv: /etc/resolv.conf
#    sys.uqhost: myhost
#    sys.version: #1 SMP Tue Aug 17 18:32:47 MSD 2010
#    sys.windir: /dev/null
#    sys.winprogdir: /dev/null
#    sys.winprogdir86: /dev/null
#    sys.winsysdir: /dev/null
#    sys.workdir: /var/cfengine
```

EXAMPLE 144

Filename: 260-2510_Special_Variables.__This_promise_filename.cf

```
# let's say this file is called 00181_Special_Variables__this_promise_filename.cf

bundle agent example
{
  reports:
    linux::

      "$(this.promise_filename)";
}

# Here is what cf-agent would output:
#
# myhost# cf-agent -b example -f ./00181_Special_Variables__this_promise_filename.cf -KI
# >> Using command line specified bundlesequence
# R: hello world
# R: ./00181_Special_Variables__this_promise_filename.cf
# myhost#
```

EXAMPLE 145

Filename: 260-2520_Special_Variables.__This_promise_linenumber.cf

```
# let's say this file is called 00182_Special_Variables__this_promise_linenumber.cf

bundle agent example
{
  reports:
    cfengine::

      "$(this.promise_linenumber)";
      "$(this.promise_linenumber)";
}

# Here is what you'd see running cf-agent:
#
```

```
# myhost# cf-agent -b example -f ./00182_Special_Variables__this_promise_linenumber.cf -KI
# >> Using command line specified bundlesequence
# R: 7
# R: 8
# myhost#
```

EXAMPLE 146

Filename: 260-2525_Special_Variables.__This_promiser.transformer.simple.cf

```
bundle agent example {

  files:

      "/var/log/.*.conf"

              handle => "compress_old_files",
             comment => "Compress files more than 2 days old",
         depth_search => recurse("inf"),
         transformer => "/bin/gzip $(this.promiser)";
}
```

EXAMPLE 147

Filename: 260-2530_Special_Variables.__This_promiser._Find_world_writable.cf

```
bundle agent example {

  files:

      "/etc/.*"

          file_select => world_writeable,
          transformer => "/bin/echo DETECTED WORLD WRITABLE FILE $(this.promiser)";
}


body file_select world_writeable
{
```

```
          search_mode => { "o+w" };
          file_result => "mode";

}
```

EXAMPLE 148

Filename: 260-2540_Special_Variables.__This_promiser._Compress_pdf_files.cf

```
#######################################################
#
# Find and compress PDF files
#
#######################################################

bundle agent example

{
  files:

      "/tmp/pdfs"

          file_select => pdf_files,
          transformer => "/usr/bin/gzip $(this.promiser)",
         depth_search => recurse("inf");

}

#############################################

body file_select pdf_files

{
             leaf_name => { ".*.pdf" , ".*.fdf" };  # leaf_name = regex to match on
      # the file NAME (ignoring the
      # full directory path)
          file_result => "leaf_name";
}
```

```
#############################################

body depth_search recurse(d)

{
               depth => "$(d)";
}

# Given the following files:
#
# /tmp/pdfs/a.pdf
# /tmp/pdfs/b.txt
# /tmp/pdfs/c.pdf
# /tmp/pdfs/d.doc,
#
# Generates the following output (with -I switch):
#
# Transforming: /usr/bin/gzip /tmp/pdfs/a.pdf
# -> Transformer /tmp/pdfs/a.pdf => /usr/bin/gzip /tmp/pdfs/a.pdf seemed to work ok
# Transforming: /usr/bin/gzip /tmp/pdfs/c.pdf
# -> Transformer /tmp/pdfs/c.pdf => /usr/bin/gzip /tmp/pdfs/c.pdf seemed to work ok
#
# and leaves the following files:
#
# /tmp/pdfs/c.pdf.gz
# /tmp/pdfs/b.txt
# /tmp/pdfs/d.doc
# /tmp/pdfs/a.pdf.gz
```

EXAMPLE 149

Filename: 260-2550_Special_Variables.__This_promiser._Find_world_writable_files_but_not_symlinks.cf

```
bundle agent example {

  files:

      "/etc/.*"
```

```
          file_select => world_writeable_but_not_a_symlink,
          transformer => "/bin/echo DETECTED WORLD WRITABLE NON-SYMLINK FILE $(this.promiser)";
}


body file_select world_writeable_but_not_a_symlink
{
          search_mode => { "o+w" };
           file_types => { "symlink" };
          file_result => "mode.!file_types";

}
```

EXAMPLE 150

Filename: 260-2560_Special_Variables.__This_promiser._In_commands_promises.cf

```
# Note: this does not work in 3.1.5; fixed in version 3.2.0.
# Broken again in 3.2.1


bundle agent example {

  commands:

      "/bin/echo $(this.promiser)";

}
```

Chapter 31. File Selection

EXAMPLE 151

Filename: 270-2710_File_Selection.__Select_by_mode.cf

```
##############################################

bundle agent example

{

  files:

      "/tmp/test_from/."

          file_select => mode_777,
          transformer => "/bin/gzip $(this.promiser)",
         depth_search => recurse("inf");

}

##############################################

body file_select mode_777

{
          search_mode => { "777" };
          file_result => "mode";
}

##############################################

body depth_search recurse(d)

{
               depth => "$(d)";
```

```
}
```

EXAMPLE 152

Filename: 270-2720_File_Selection.__Select_files_more_than_N_days_old.cf

```
# The following policy selects files modified over a year ago
#
# Literally, it selects files whose mtime is between 1 year old and 100 years old.


bundle agent example

{
  files:

      "/tmp/test_from"

          file_select => modified_over_a_year_ago,
          transformer => "/bin/echo DETECTED $(this.promiser) MATCHING FILTER",
         depth_search => recurse("inf");

}

############################################

body file_select modified_over_a_year_ago

{
                mtime => irange(ago(100,0,0,0,0,0),ago(1,0,0,0,0,0));   # modified more than a year ago
      # ago (Years, Months, Days, Hours, Minutes, Seconds)
      # example: 20 minutes and 30 seconds ago:  ago(0,0,0,0,20,30);

          file_result => "mtime";
}

############################################

body depth_search recurse(d)
```

```
{
               depth => "$(d)";
}
```

EXAMPLE 153

Filename: 270-2730_File_Selection.__Select_by_several_things.cf

```
bundle agent example {

  files:

      "/var/logexample/."

               handle => "remove_world_writable_files",
              comment => "if you make a file world-writable, I WILL delete it",
          file_select => compound_filter,
         depth_search => recurse("inf"),
              delete  => tidy;
}

body common control {

               inputs => {"/var/cfengine/masterfiles/libraries/cfengine_stdlib.cf"};
}



body file_select compound_filter

{

          search_mode => { "777" };
            leaf_name => { ".*\.pdf" , ".*\.fdf" };  # leaf_name = regex to match


          file_result => "leaf_name&mode";   # this is a class expression
}
```

EXAMPLE 154

Filename: 270-2730_File_Selection.__Select_files_more_than_N_days_old._More_elegant.cf

```
# The following policy selects files modified over a year ago
#
# More elegant version, courtesy of Dan Klein.


bundle agent example

{
  files:

      "/tmp/test_from"

          file_select => modified_over_a_year_ago,
          transformer => "/bin/echo DETECTED $(this.promiser) MATCHING FILTER",
      #depth_search => recurse("inf");
         depth_search => recurse("inf");

}

############################################

body file_select modified_over_a_year_ago

{
               mtime => irange(ago(1,0,0,0,0,0),now); # modified between a year ago and now
          file_result => "!mtime";
}

############################################

body depth_search recurse(d)

{
               depth => "$(d)";
}
```

EXAMPLE 155

Filename: 270-2740_File_Selection.__Compress_old_files.cf

```
#########################################################
#
# Searching for permissions
#
#########################################################


###############################################

bundle agent example

{
  files:

      "/tmp/test_from"

          file_select => days_old("1"),
          transformer => "/bin/gzip $(this.promiser)",
         depth_search => recurse("inf");

}

###############################################

body depth_search recurse(d)

{
               depth => "$(d)";
}

#########################################

body file_select days_old(days)
{
          mtime       => irange(0,ago(0,0,"$(days)",0,0,0));
```

```
          file_result => "mtime";
}
```

EXAMPLE 156

Filename: 270-2750_File_Selection.__Compress_old_pdf_files.cf

```
#########################################################
#
# Searching for permissions
#
#########################################################


###############################################

bundle agent example

{
  files:

      "/tmp/test_from"

          file_select => compound_filter,
          transformer => "/bin/gzip $(this.promiser)",
         depth_search => recurse("inf");

}

###############################################

body file_select compound_filter

{

            leaf_name => { ".*\.pdf" , ".*\.fdf" };  # leaf_name = regex to match
                mtime => irange(ago(1,0,0,0,0,0),now);   # modified within 1 year
      # the above automatically define classes
      # only if the right hand side matches
```

```
      # file being examined

          file_result => "leaf_name.(!mtime)";   # this is a class expression
}

#############################################

body depth_search recurse(d)

{
               depth => "$(d)";
}
```

Chapter 32. Process Selection

EXAMPLE 157

Filename: 280-2810_Process_Selection.__Kill_based_on_process_owner_username.cf

```
# Kill all processes belonging to user "victim".
# For the demonstration, in another window, run:
#    useradd victim && su - victim
# You will see cf-agent kill victim's session.
#
# You can dry-run this with cf-agent --dry-run.


bundle agent example {

  processes:

      ".*"

     process_select   => by_process_owner("victim"),
              signals => { "term", "kill" };

}

############################################################

body process_select by_process_owner(username) {

        process_owner => { "$(username)" };
       process_result => "process_owner";

}
```

EXAMPLE 158

Filename: 280-2820_Process_Selection.__Select_by_vsize.cf

```
# kill all processes over a certain vsize (total Virtual Memory size in kb)

bundle agent example {

  processes:

      ".*"

      process_select  => vsize_exceeds("30000"),
              signals => { "term", "kill" };

}

############################################################

body process_select vsize_exceeds(vsize_limit) {

                vsize => irange("$(vsize_limit)","inf"); # vsize is over
      # $(vsize_limit)
       process_result => "vsize";
}
```

EXAMPLE 159

Filename: 280-2830_Process_Selection.__Select_by_process_owner,command,_and_vsize.cf

```
# Scenario: you have a memory leak in your Web app
# that causes "bloat" in httpd processes.
#
# kill all apache httpd processes over a certain vsize
# (vsize = total Virtual Memory size in kb)

bundle agent example {

  processes:

      ".*"

      process_select  => vsize_exceeds("apache", "/usr/sbin/httpd.*", "30000"),
```

```
              signals => { "term", "kill" };

}

#############################################################

body process_select vsize_exceeds(process_owner, process_command, vsize_limit) {

        process_owner => { "apache" };  # username of process owner

              command => "$(process_command)";  # username of process owner

                vsize => irange("$(vsize_limit)","inf"); # vsize is over
      # $(vsize_limit)

       process_result => "process_owner&command&vsize";  # class expression
}
```

EXAMPLE 160

Filename: 280-2840_Process_Selection.__Graceful_restart_of_bloated_apache_httpd.cf

```
# Scenario: you have a memory leak in your Web app
# that causes "bloat" in httpd processes.
#
# Issue a graceful restart command to the httpd
# if any apache httpd processes exceed vsize limit
# (vsize = total Virtual Memory size in kb).

bundle agent example {

  processes:

      ".*"

      process_select  => vsize_exceeds("apache", "httpd", "30000"),
        process_count => set_class("big_apache_httpd_procs");

  commands:
```

```
    big_apache_httpd_procs::
      "/usr/sbin/httpd -k graceful";


}

#############################################################

body process_select vsize_exceeds(process_owner, command, vsize_limit) {

        process_owner => { "apache" };  # username of process owner

              command => "/usr/sbin/httpd.*";  # username of process owner

                vsize => irange("$(vsize_limit)","inf"); # vsize is over
      # $(vsize_limit)
       process_result => "process_owner&command&vsize";
}

#############################################################

body process_count set_class(classname)

{
          match_range => "1,inf"; # Integer range for acceptable number of matches for this process
      # (In this case, one or more processes


      in_range_define => { "$(classname)" }; # List of classes to define if the matches are in range.

}
```

EXAMPLE 161

Filename: 280-2860_Process_Selection.__Select_by_several_things.cf

```
#############################################################
#
# Simple test processes
```

```
#
############################################################


bundle agent example

{
  processes:

      ".*"

      process_count   => anyprocs,
      process_select  => proc_finder;


  reports:

    any_procs::

      "Found processes in range";

    in_range::
      "Found no processes in range";

}

############################################################

body process_select proc_finder

{

              command => "vim .*"; # (Anchored) regular expression matching the command/cmd field of a process

          stime_range => irange(ago(1,0,0,0,0,0),ago(0,0,0,0,0,10)); # Processes started between 1 year and 10 seconds ago

        process_owner => { "root" }; # List of regexes matching the user of a process

       process_result => "stime&command&process_owner";
```

```
      # pid => irange("1","10");   # Range of integers matching the process id of a process
      # ttime_range => irange(0,accumulated(1,0,0,0,0,0));
      #    # ttime_range is the range of integers matching the total elapsed time of a process
      #    # in this case, between 0 and 1 year
}

#############################################################

body process_count anyprocs

{
          match_range => "0,0"; # Integer range for acceptable number of matches for this process
      # (In this case, one or more processes


  out_of_range_define => { "any_procs" }; # List of classes to define if the matches are out of range

      in_range_define => { "in_range" }; # List of classes to define if the matches are in range.
      # We should never have a process that has a count of 0.

}
```

EXAMPLE 162

Filename: 280-2865_Process_Selection.__Select_by_stime.cf

```
bundle agent example

{
  processes:

      ".*sleep.*"

      process_select  => newborns,
              signals => { "term" };
```

```
}

#############################################################

body process_select newborns

{


          stime_range => irange(ago(0,0,0,1,0,0), now); # Processes started between 1 hour and 1 second ago
       process_result => "stime";
}
```

Chapter 33. Special Notes

EXAMPLE 163

Filename: 290-2910_Special_Notes.__Iteration_over_a_global_list._Using_parameterization.cf

```
# Scalar references to *local* list variables imply iteration.
# To iterate over a global list variable, map the global list
# into the local context, or supply it to the bundle as a
# parameter.
#
# Example of mapping it into the local context

body common control {
       bundlesequence => { runme(@(g.myusers)) };  # note lack of
      # " symbols
}

bundle common g
{
      vars: "myusers"  slist => { "joe", "mary", "ann" };

}


bundle agent runme(x)

{

  reports:
    linux::
      "$(x)";
}
```

EXAMPLE 164

Filename: 290-2920_Special_Notes.__Iteration_over_a_global_list._Direct_method.cf

```
# Scalar references to *local* list variables imply iteration.
# To iterate over a global list variable, map the global list
# into the local context.  There are two ways to do it, this
# is the direct method.
#
# Reference:
# http://www.cfengine.org/manuals/cf3-reference.html#Variable-expansion-in-cfengine-3



bundle common g
{
      vars: "myusers"  slist => { "joe", "mary", "ann" };

}


bundle agent example

{
      vars: "mylist" slist => { @(g.myusers) };

  reports:
    linux::
      "$(mylist)";
}
```

There is no limit to the length of lists or arrays, but there is a limit to the size of variable-expanded strings (scalars). The final result of any single variable expansion is limited to about 4k.

/usr/local/share/doc/cfengine/ contains over 220 examples (originally unit tests).

They don’t all work, but most do.

Potentially useful in learning CFEngine.

Orion Cloud Pack - library for EC2

Essential Files

promises.cf Main configuration file. update.cf Update configuration. failsafe.cf Failsafe configuration. cfengine_stdlib.cf CFEngine standard library.

Maintenance Examples

change_mgt.cf Implement security tripwire on files/directories. ensure_ownership.cf Home directory ownership and permission maintenance. fix_broken_software.cf Package installation and permission correction. garbage_collection.cf Log rotation and removal. harden_xinetd.cf Disable xinetd services specified. iptables.cf Secure system with sysctl.conf and iptables. name_resolution.cf Edit /etc/resolv.conf to the specified DNS servers

System Setup Examples

c_cpp_env.cf Set up C programming environment. db_mysql.cf Install and run MySQL db_postgresql.cf Install and run PostgreSQL db_sqllite.cf Install and run SQLlite jboss_server.cf Prepare JAVA environment and run JBOSS. nagios.cf Setup NAGIOS monitoring node. nginx_perlcgi.cf Setup NGINX webserver perlCGI. nginx_phpcgi.cf Setup NGINX webserver phpCGI. ntp.cf Setup NTP server and clients. perl_env.cf PERL programming language install. php_webserver.cf Setup a PHP webserver. python_env.cf PYTHON programming install. ruby_env.cf Setup ruby on rails environment. sshd_conf.cf Ensure sshd config is correct. tomcat_server.cf Setup a tomcat server. varnish.cf Set up Varnish web accelerator

always specify the class , or else you may inadvertently inherit the class specification from an earlier promise

EXAMPLE 165

Filename: 290-2961_Special_Notes.__Be_careful_with_class.cf

```
bundle agent example {
  commands:
    customclass::
      "/bin/echo customclass is set";
      "/bin/echo this is always true";
}

# run cf-agent on this policy with and without -Dcustomclass
```

EXAMPLE 166

Filename: 290-2965_Special_Notes.__No_safeguard_for_syntactically_correct_but_insane_policy.cf

```
# a mutually exclusive configuration

body common control {
               inputs => { "libraries/cfengine_stdlib.cf",
               };

}

#############################################################


bundle agent example {

  files:

      "/tmp/plug"
               delete => tidy;

  files:

      "/tmp/plug"
               create => "true";
}
```

Chapter 34. Packages

EXAMPLE 167

Filename: 300-0010_Packages.__complain_if_a_package_version_is_too_new.cf

```
body common control {

              inputs  =>  { "libraries/cfengine_stdlib.cf" };

}



bundle agent example {
  packages:

      "lsof"

       package_method => yum,
       package_policy => "add", # ensure package is installed
       package_select => "<=",
      package_version => "4.78-2",
              classes => if_else("we_have_4_78_2_or_lesser","we_have_greater_than_4_78_2"),
              comment => "Ensure our version of lsof is not greater than 4.78-2.  Let's say 4.78-3 and above in
troduce a new feature that is not compatible with our in-house application.";

  reports:
    we_have_greater_than_4_78_2::
      "your lsof version is > 4.78-2";
    we_have_4_78_2_or_lesser::
      "your lsof version is <= 4.78-2";


}
```

Package versions are of data type string, not number! Thus numeric comparison, while it can be attempted, is fraught with peril and frustration.

Which version is newer:

1.2.3f 1.2.3-4 1.2.3-hotpotato

See http://semver.org/ for a proposal for a meaningful versioning standard.

EXAMPLE 168

Filename: 300-0020_Packages.__install_packages_from_local_filesystem_based_repository.cf

```
body common control {
               inputs => {"libraries/cfengine_stdlib.cf"};
}

bundle agent example {

  packages:
      "epel-release"
       package_policy => "add",
      package_version => "5-4",
package_architectures => { "noarch" },
       package_method => rpm_filebased("/repo/RPMs");

}


body package_method rpm_filebased(path)
# Contributed by Aleksey Tsalolikhin. Written on 29-Feb-2012.
# Based on yum_rpm body in COPBL by Trond Hasle Amundsen.
# Intended to install packages from local package repository.
# You must specify the path to the local package repository as the argument.


{
package_file_repositories => { "$(path)" };
      # the above is an addition to Trond's yum_rpm body

package_add_command    => "/bin/rpm -ihv ";
```

```
      # The above is a change from Trond's yum_rpm body, this makes the commands rpm only.
      # The reason I changed the install command from yum to rpm is yum will be default
      # refuse to install the epel-release RPM as it does not have the EPEL GPG key,
      # but rpm goes ahead and installs the epel-release RPM and the EPEL GPG key.

package_name_convention => "$(name)-$(version).$(arch).rpm";
      # The above is a change from Tron's yum_rpm body. When package_file_repositories is in play,
      # package_name_convention has to match the file name, not the package name, per the
      # CFEngine 3 Reference Manual

      # The rest is unchanged from Trond's yum_rpm body
      package_changes => "bulk";
 package_list_command => "/bin/rpm -qa --qf '%{name} %{version}-%{release} %{arch}\n'";

package_list_name_regex    => "^(\S+?)\s\S+?\s\S+$";
package_list_version_regex => "^\S+?\s(\S+?)\s\S+$";
package_list_arch_regex    => "^\S+?\s\S+?\s(\S+)$";

package_installed_regex => ".*";


package_delete_command => "/bin/rpm -e --allmatches";
package_verify_command => "/bin/rpm -V";
}


# Example output from running this policy:
# linux# cf-agent -f ./MISC_install_packages_from_local_filesystem_based_repository.cf  -b example -KI
# >> Using command line specified bundlesequence
#
# Q:rpm -ihv  "/repo/RPM ...:Preparing...                ##################################################
# Q:rpm -ihv  "/repo/RPM ...:epel-release                ##################################################
#
# Q:rpm -ihv  "/repo/RPM ...:
# linux#
```

EXAMPLE 169

Filename: 300-0030_Packages.__install_RPM_from_a_local_directory.cf

```
body common control {
       bundlesequence => { "commands__install_PGDG_yum_repo_RPM" };
}


bundle agent commands__install_PGDG_yum_repo_RPM {

  packages:

      "pgdg-centos"

       package_policy => "add",
       package_method => yum_filebased;

}


body package_method yum_filebased
{
package_file_repositories => { "/repo" };    # A list of machine-local directories to search for packages

      package_changes => "bulk";
 package_list_command => "/usr/bin/yum list installed";

      # Remember to escape special characters like |

package_list_name_regex    => "([^.]+).*";
package_list_version_regex => "[^\s]\s+([^\s]+).*";
package_list_arch_regex    => "[^.]+\.([^\s]+).*";

package_installed_regex => ".*installed.*";
package_name_convention => "$(name).$(arch)";

  package_add_command => "/usr/bin/yum -y install";
package_delete_command => "/bin/rpm -e";
package_verify_command => "/bin/rpm -V";
}
```

Chapter 35. Classes (advanced)

EXAMPLE 170

Filename: 310-0010_Classes.__setting_multiple_classes_as_a_result_of_a_single_promise.cf

```
body common control {
               inputs => { "cfengine_stdlib.cf" };
}

bundle agent example {

  vars:
      "slist_of_classes" slist => { "class1", "class2" };

  files:
      "/etc/httpd/conf/httpd.conf"
            edit_line => insert_lines("#test comment"),
             classes  => if_repaired_set_these_classes("@(example.slist_of_classes)");

  reports:
      class1:: "class1";
      class2:: "class2";

}

body classes if_repaired_set_these_classes(list)
{
     promise_repaired => { "@(list)" };
}
```

EXAMPLE 171

Filename: 310-0020_Classes.__Return_codes.cf

```
bundle agent example {
```

```
  commands:

      "/bin/false"

              classes => cmd_kept("1","ok");

  reports:
    ok::

      "Command completed successfully";

}




body classes cmd_kept(code,class)
{
 repaired_returncodes => { "$(code)" };
     promise_repaired => { "$(class)" };
}
```

EXAMPLE 172

Filename: 310-0021_Classes.__returncodes.cf

```
# customize CFEngine's idea of promise kept, returned or failed
# based on command's return code.
#
# For this demo, add an account "joe" and then use
# userdel to remove it.
#
# Run "chattr +i /etc/passwd" after adding the "joe" account
# to induce a failure to remove Joe.
#

bundle agent example {

  commands:
```

```
     "/usr/sbin/userdel"
                args => "joe",
             comment => "We don't want joe on our systems ever again.",
             classes => customized_for_userdel;

  reports:

     joe_removed::   "Joe was removed";
     no_joe::        "Don't worry, there is no account for Joe.";
     (!joe_removed)&(!no_joe)::  "!!!!  Joe is still here.  !!!!";
}

body classes customized_for_userdel {

     kept_returncodes => { "6" };   # user was not present in the file
 repaired_returncodes => { "0" }; # user was successfully removed
     promise_repaired => { "joe_removed" };
         promise_kept => { "no_joe" };
}
```

EXAMPLE 173

Filename: 310-0030_Classes.__Canonifying_variables_to_use_them_as_class_names.cf

```
bundle agent example {

  vars:
     "myarray[loc@t!on]" string  => "Bloomington";
     "myarray[t!me###]"     string  => "first week of April";


     "index" slist =>  getindices("myarray");


     "cindex[$(index)]" string => canonify("$(index)");


  reports:
```

```
    cfengine::

      "Original keys: $(index)";
      "Canonified keys: $(cindex[$(index)])";
}
#############################################################
# Data Structures:

Array myarray
```

loc@t [mailto:loc@t]!on,Bloomington t!me###,first week of April

Chapter 36. List index

loc@t [mailto:loc@t]!on t!me###

Chapter 37. Array cindex

loc@t [mailto:loc@t]!on,loc_t_on t!me###,t_me___

```
EXAMPLE 174

Filename: 310-0040_Classes.__ifvarclass.cf
```

bundle agent example {

```
vars:  "fruit"  string =>  "banana";

reports:
```

linux
    "I love bananas for breakfast"

```
   ifvarclass => "$(fruit)";
```

}

```
EXAMPLE 175

Filename: 310-0050_Classes.__non_persistent_class.cf
```

body common control {

```
inputs => { "libraries/cfengine_stdlib.cf" };
```

}

bundle agent example {

```
files:
    "/tmp/file.txt"
             create => "true",
          edit_line => insert_lines("hello world 1234"),
```

```
            classes => if_repaired("promise_repaired");

reports:
  promise_repaired::
    "soft class is set";

}
```

```
Regular Expressions
```

```
#!/usr/bin/perl



$record =
"James Alexander Richard Smith";

if ( $record =~ /^(.*?) (.*) (.*)$/ ) {

  print "First name: $1\n";
  print "Middle name(s): $2\n";
  print "Last name: $3\n";
}
```

EXAMPLE 176

Filename: 320-0020_Regular_Expressions.__commentinging_out_file_content.cf

```
body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };

}


bundle agent example {

  files:
```

```
      "/tmp/testfile"

            edit_line => comment_out_everything;
}


bundle edit_line comment_out_everything {

  replace_patterns:

      "^([^#].*)"

         replace_with => comment("#");

}
```

EXAMPLE 177

Filename: 320-0030_Regular_Expressions.__replace_patterns.cf

```
body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}

bundle agent example {

  files:

      "/tmp/data.txt"
            edit_line => change_dogs_to_cats;

}


bundle edit_line change_dogs_to_cats {

  replace_patterns:

      "dog"
```

```
        replace_with => value("cat");
}
```

Let’s say you want to write a regex that will match any string that does NOT contain the string "hello world". Use:

^((?!hello world).)*$

This is explained in http://stackoverflow.com/questions/406230/regular-expression-to-match-string-not-containing-a-word

Chapter 38. Functions

EXAMPLE 178

Filename: 330-0010_Functions.__putting_command_output_into_variable.cf

```
bundle agent example {

  vars:
      "my_result"
               string => execresult("/bin/ls /tmp", "noshell");

  reports:
    linux::
      "Variable is $(my_result)";

}
```

EXAMPLE 179

Filename: 330-0020_Functions.__function_countlinesmatching.cf

```
bundle agent example {

  vars:
      "no" int => countlinesmatching("^cfengine:.*","/etc/group");

  commands:
      "/bin/echo"
                 args => "Found $(no) lines matching";
}
```

EXAMPLE 180

Filename: 330-0030_Functions.__functions_canonify.cf

```
bundle agent example {
```

```
  vars:

      "canonified_text"

              string  => canonify("hello!@#$%world");

  reports:

    cfengine::

      "$(canonified_text)";

}
```

EXAMPLE 181

Filename: 330-0040_Functions.__get_info_from_env_variable.cf

```
bundle agent example
{
  vars:

      "myvar" string => getenv("USER","20");


  classes:

      "isdefined" not => strcmp("$(myvar)","");

  reports:
    isdefined::
      "I am running as user $(myvar)";

}
```

Chapter 39. Commands

EXAMPLE 182

Filename: 340-0010_Commands.__ifelapsed.cf

```
# do not use -K switch when running this example!!
#
# Run it in verbose mode and grep the output for "elapsed"

bundle agent example {

  commands:

      "/bin/echo /bin/cycle_shield_frequencies.sh"

               action => every_2_minutes;
}

body action every_2_minutes
{
            ifelapsed => "2";  # in minutes
}
```

EXAMPLE 183

Filename: 340-0020_Commands.__Getting_shell_to_interpolate_a_shell_variable_requires_useshell_attribute.cf

```
body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}


bundle agent example {

  commands:
```

```
      "/bin/echo"
                 args => " \"hello $(const.dollar)USER $(const.t)adfs\"",
               contain => in_shell;

}
```

EXAMPLE 184

Filename: 340-0030_Commands.__contain-preview.cf

```
bundle agent example {

  commands:

      "/bin/touch /tmp/test2"

               contain => preview;

}

body contain preview {

               preview => "true";

}
```

Chapter 40. Linking Promises with Classes

EXAMPLE 185

Filename: 350-0010_Linking_Promises_with_Classes.__if_repaired.cf

```
body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}

bundle agent example {

  files:
      "/tmp/httpd.conf"
               create => "true",
             edit_line => insert_lines("ServerName localhost"),
               classes => if_repaired("httpd_restart_needed");


  commands:
    httpd_restart_needed::
      "/etc/init.d/httpd reload";


}
```

EXAMPLE 186

Filename: 350-0020_Linking_Promises_with_Classes.__if_repaired._stop_cups_and_complain.cf

```
body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };

}
```

```
bundle agent example {

  processes:

      "cupsd"

         process_stop => "/etc/init.d/cups stop",
              comment => "We don't want print services on our Web servers.",
              classes => if_repaired("complain_loudly_about_cups");

  commands:
    complain_loudly_about_cups::
      "/bin/echo send up a flare about CUPS";
}
```

EXAMPLE 187

Filename: 350-0030_Linking_Promises_with_Classes.__Verbose_logging_of_repairs.cf

```
bundle agent example

{

  files:

      "/tmp/newfile3"

               handle => "newfile3_exists",
               create => "true",
               action => log("Created very important file $(this.promiser)");
}

body action log(message)
{
          log_string => "$(sys.date) In $(this.promise_filename) on line $(this.promise_linenumber), in promise $(this.handle):
         log_repaired => "stdout";
}
```

```
body agent control {

               syslog => "true";

}
```

EXAMPLE 188

Filename: 350-0040_Linking_Promises_with_Classes.__edit_crontab_and_HUP_crond.cf

```
body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}


bundle agent example {

  files:

      "/var/spool/cron/root"

            edit_line => cf_execd_entry_is_present,
               create => "true",
              classes => if_repaired("restart_crond");

  processes:
    restart_crond::
      "crond"
              signals => { "hup" };


}


bundle edit_line cf_execd_entry_is_present {

  insert_lines:
      "5,10,15,20,25,30,35,40,45,50,55 * * * * /var/cfengine/bin/cf-execd -F";
}
```

Chapter 41. Dynamic Bundlesequence

EXAMPLE 189

Filename: 370-0005_Dynamic_Bundlesequence.__Activate_a_class_if_it_is_appropriate_for_my_context.cf

```
# I want to target a promise to a certain group of servers.
# However I want to abstract the elements of that group from
# the promises that target that group, so that when I add an
# element to that group, I only need to update *one* promise,
# the one enumerating that group.
#
# The following policy will report "I am a webserver" if its
# hostname is listed in "webservers" slist.

bundle common global_vars {

  vars:
      "webservers"
                slist => { "web01", "web02", "web03" };

}


bundle common global_classes {

  classes:

      "webfarm"
           expression => reglist("@(global_vars.webservers)", escape("$(sys.host)"));

}

bundle agent example {

  reports:
```

```
    webfarm::

      "I am a web server";

}
```

EXAMPLE 190

Filename: 370-0050_Dynamic_Bundlesequence.__Jeff_Blaine._dynamic_bundlesequence_with_parameterized_mybundle.cf

```
# by Jeff Blaine

# Changes the order of NTP servers in ntp.conf based on site (using class)
# if we're in site x, site X servers should be first; if we're in site y,
# site y servers should be first.

# Example run:
#    [cfengine00  practical_examples]# cf-agent -KI -f ./MISC_dynamic_bundlesequence_with_parameterized_mybundle.cf   -Dsite_x
#     -> Edited file /etc/ntp.conf
#    [cfengine00  practical_examples]# cat /etc/ntp.conf
#    fudge   127.127.1.0 stratum 10
#    server  127.127.1.0
#    # will be destroyed by CFEngine, so don't do that.
#    # This file is configured by CFEngine.  Manual edits to this file
#    server ntp-sitex.our.org
#    server ntp-sitey.our.org
#    restrict -4 default kod notrap nomodify nopeer noquery
#    restrict -6 default kod notrap nomodify nopeer noquery
#    [cfengine00  practical_examples]# cf-agent -KI -f ./MISC_dynamic_bundlesequence_with_parameterized_mybundle.cf   -Dsite_y
#     -> Edited file /etc/ntp.conf
#    [cfengine00  practical_examples]# cat /etc/ntp.conf
#    fudge   127.127.1.0 stratum 10
#    server  127.127.1.0
#    # will be destroyed by CFEngine, so don't do that.
#    # This file is configured by CFEngine.  Manual edits to this file
#    server ntp-sitey.our.org
#    server ntp-sitex.our.org
#    restrict -4 default kod notrap nomodify nopeer noquery
#    restrict -6 default kod notrap nomodify nopeer noquery
```

```
#    [cfengine00  practical_examples]#
#



bundle common g
{
      # Define NTP servers in a specific order per site.

      # You could define everything here in "main" and change the references
      # there in "methods:" if you wanted to.

  vars:

    site_x::
      "ntpservers" slist => {
                              "ntp-sitex.our.org",
                              "ntp-sitey.our.org",
      };

    site_y::
      "ntpservers" slist => {
                              "ntp-sitey.our.org",
                              "ntp-sitex.our.org",
      };
}

bundle agent main
{
  methods:

    site_x::
      "site_x" usebundle =>
      system_ntpclient_configure(@(g.ntpservers));

    site_y::
      "site_y" usebundle =>
      system_ntpclient_configure(@(g.ntpservers));
}
```

```
body common control
{
       bundlesequence => {
                           "main"
       };

      # Building a per-site inputs list is beyond the scope of this
      # example.
               inputs => {
                           "libraries/cfengine_stdlib.cf",
                           "ntp.cf"
               };
}
```

EXAMPLE 191

Filename: 370-0050_Dynamic_Bundlesequence.__Jeff_Blaine.ntp.cf

```
#Author: Jeff Blaine

#
# Given a list of servers, establish a basic NTP configuration file
# containing that list of servers as well as a set of "restrict"
# lines based on OS (some OSes don't support all modern options
# to the restrict directive).
#
# EXERCISE: augment the following (with new bundle(s) as needed)
#           to ensure that the appropriate NTP package(s) are
#           installed on the host, per OS.  Make this bundle
#           below depend on the package being installed first.
#
# EXERCISE: augment the following to ensure that the NTP client
#           is running.  This new logic should depend on the client
#           package(s) being installed per the exercise above.
#
bundle agent system_ntpclient_configure(servers)
{
  vars:
```

```
    solaris::

      "configfile" string => "/etc/inet/ntp.conf";

      # SunOS5.10 (at least) does not support 'kod' or '-6' like Linux
      "restrictlines" slist => {
                                "restrict default notrap nomodify nopeer noquery",
      };

    redhat|centos::

      "configfile" string => "/etc/ntp.conf";

      "restrictlines" slist => {
                                "restrict -4 default kod notrap nomodify nopeer noquery",
                                "restrict -6 default kod notrap nomodify nopeer noquery",
      };

  files:

    redhat|centos|solaris::

      "$(configfile)"
            edit_line =>
      ntpclient_config_edit(@(system_ntpclient_configure.servers),
                            @(system_ntpclient_configure.restrictlines));
}

bundle edit_line ntpclient_config_edit(servers, restrictlines)
{
  delete_lines:
      ".*";

  insert_lines:
      # Add our static content first (4 lines).
      "# This file is configured by CFEngine.  Manual edits to this file
# will be destroyed by CFEngine, so don't do that.
server  127.127.1.0
```

```
fudge   127.127.1.0 stratum 10"
          insert_type => "preserve_block",
             location => start;

      # ^^^^ There is a bug in 3.2.0 (at least) that will cause the
      # above promise definition to not keep the proper order of lines.
      # Just be aware.  In this case, it just makes a silly looking file
      # that still functions properly as far as NTP is concerned.

      # Add our NTP servers, one per line
      "server $(servers)";

      # Add our restrict rules, one per line
      "$(restrictlines)";
}
```

Chapter 42. Databases

Demonstrate CFEngine integration with PostgreSQL.

EXAMPLE 192

Filename: 380-0020_Databases.__db_demo.cf

```
# Demonstration of CFEngine's databases promises.
# First, install and configure a PostgreSQL database
# cluster and create an database.
# Then use "databases" type promises to set up and
# maintain the schema of 3 tables.
#
# Note: package_list_update_ifelapsed should be set to 0
# for demoes.
#
# Demoes: - self-heal from database cluster shutdown
#         - self-heal from dropping a table
#         - self-heal from dropping a table column
#         - self-heal from changes to pg_hba.conf




body common control {

              version => "1.1 21-Oct-2011";

   host_licenses_paid => "10";

               inputs => { "libraries/cfengine_stdlib.cf" };

       bundlesequence =>        {
                                  "db_cluster_is_installed",
```

```
                                    "pg_hba_conf_trusts_local_users",

                                    "db_cluster_is_running",

                                    "database_exists",

                                    "schema_exists_and_is_correct",
       };

}

###############################################

bundle agent db_cluster_is_installed {

  packages:


      "postgresql-server"

       package_policy => "add",
       package_method => yum,
              classes => if_repaired("start_postgres");

      "postgresql"
       package_policy => "add",
       package_method => yum;


  commands:

    start_postgres::

      "/sbin/service postgresql start";


}

###############################################
```

```
bundle agent pg_hba_conf_trusts_local_users {


  files:
      "/var/lib/pgsql/data/pg_hba.conf"

      # this is a regular comment

            edit_line => trust_local_users,
              comment => "Allow root to access the DB cluster so CFEngine can set up the database and table schema",
      # the above was a Knowledge Management comment
              classes => if_repaired("reload_postgres");

  commands:

    reload_postgres::

      "/sbin/service postgresql reload";


}


################################################


bundle agent db_cluster_is_running {


  processes:

      "postgres"

        restart_class => "start_postgres";


  commands:
```

```
    start_postgres::

      "/sbin/service postgresql start";

}



#################################################

bundle agent database_exists {

  commands:

      "/usr/bin/createdb -U postgres conference >/dev/null 2>/dev/null"
              contain => in_shell;

}


#################################################


bundle agent schema_exists_and_is_correct {

  vars:
      "create_and_verify"
                slist => { "create", "verify" };


  databases:


      "conference/speakers"

   database_operation => "$(create_and_verify)",
        database_type => "sql",
     database_columns => {
```

```
                            "speaker_name,varchar,50",
                            "speaker_bio,varchar,600",
                            "speaker_affiliation,varchar,50",
     },
      database_server => demo_postgres_server;


      "conference/rooms"

   database_operation => "$(create_and_verify)",
        database_type => "sql",
     database_columns => {
                            "room_name,varchar,256",
                            "room_number_of_seats,integer",
     },
      database_server => demo_postgres_server;


      "conference/talks"

   database_operation => "$(create_and_verify)",
        database_type => "sql",
     database_columns => {
                            "speaker_name,varchar,256",
                            "room_name,varchar,256",
                            "start_time,date",
     },
      database_server => demo_postgres_server;


}


###############################################


body database_server demo_postgres_server {

      db_server_owner => "postgres";
```

```
   db_server_password => "";

       db_server_host => "localhost";

       db_server_type => "postgres";

db_server_connection_db => "postgres";

}

################################################

bundle edit_line trust_local_users {

      delete_lines: ".*";

      insert_lines: "
# !!! This file is under CFEngine control.  Do not edit
# it directly or your changes may be overwritten.
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
local   all         all                               trust
";

}
```

EXAMPLE 193

Filename: 380-0030_Databases.__Create_DB_Users.cf

```
body common control {

       bundlesequence => { "create_db_users" };
               inputs => { "libraries/cfengine_stdlib.cf" };

}
```

```
#############################################################################################

bundle common db {

  vars:

      "db_users"

              slist   => splitstring(execresult ("/usr/bin/psql -AXqt -c 'select usename from pg_user' -U postgres", "noshell"), "
              comment => "List of DB users";


      "createuser_defaults"
               string => " -U postgres  --no-createdb --no-createrole --no-superuser ",
              comment => "Default arguments we'll use with /usr/bin/createuser to create regular unprivileged PostgreSQL accounts"

}

#############################################################################################


bundle agent create_db_users {

  classes:

      "postgres_node"

           expression => returnszero ("/usr/bin/pgrep postmaster >/dev/null", "useshell"),
              comment => "Identify if this node is running postgres.";


  methods:

    postgres_node::

      "any"
            usebundle => create_pg_user("nagios", "$(db.createuser_defaults)"),
              comment => "Every node that runs postgres should have pg user nagios for monitoring using check-postgres.pl plugin"
```

```
    specialcase1::

      "any"
            usebundle => create_pg_user("superuser1", " -U postgres --superuser "),
              comment => "Create db superuser superuser1";

    specialcase2::

      "any"
            usebundle => create_pg_user("regularuser1", "$(db.createuser_defaults)"),
              comment => "Application X requires regularuser1";


}


#########################################################################################

bundle agent create_pg_user(username,args) {


  classes:

      "$(username)_exists"

           expression => reglist("@(db.db_users)","$(username)"),
              comment => "Check if username already exists in the database.";


  commands:


      "/usr/bin/createuser $(args)  $(username)"
              contain => in_shell_and_silent,
           ifvarclass => "!$(username)_exists",
              comment => "Create PostgreSQL user $(username) with createuser args $(args)";

}
```

```
#######################################################################################################
```

EXAMPLE 194

Filename: 385-0010_COPBL.__modified_set_variable_values.cf

```
# this file contains a modified set_variable_values bundle.
# the main difference is you won't get lines like
# "name =value2" if you start with "name = value1".
# Instead you get lines like "name=value2".

bundle common global {

  vars:

      "stuff[Location]" string => "Chicago";
      "stuff[Time]"     string => "Monday, April 2nd";
}



bundle agent example {

  files:

      "/tmp/example"
              create => "true",
           edit_line => set_variable_values("global.stuff");
}


bundle edit_line set_variable_values(v)

{
  vars:

      "index" slist => getindices("$(v)");

  field_edits:
```

```
     # match a line starting like the key *BLANK SPACE* = something

     "\s*$(index)\s+=.*"

          edit_field => col("=","1","$(index)","set"),
             comment => "Edit name=value definition, if there is whitespace after the name to eliminate said whitespace otherwise

     # match a line starting like the key = something

     "\s*$(index)=.*"

          edit_field => col("=","2","$($(v)[$(index)])","set"),
             comment => "Edit name=value definition to set the value.  Incidentally, this gets rid of any whitespace after the ed

  insert_lines:

     "$(index)=$($(v)[$(index)])",

             comment => "Insert name=value definition";
}

body edit_field col(split,col,newval,method)
{
   field_separator    => "$(split)";
   select_field       => "$(col)";
   value_separator    => ",";
   field_value        => "$(newval)";
   field_operation    => "$(method)";
   extend_fields      => "true";
   allow_blank_fields => "true";
}

body replace_with value(x)
{
       replace_value => "$(x)";
         occurrences => "all";
}
```

Chapter 43. Misc.

EXAMPLE 195

Filename: 390-0010_Processes.__Restart_a_process_if_it_is_running_or_start_it_if_it_is_not_running.cf

```
bundle agent example

{
  processes:

      ".*"

      process_count   => anyprocs,
      process_select  => proc_finder;


  commands:

    process_running::

      "/bin/echo restart command";

    process_not_running::
      "/bin/echo start command";

}

############################################################

body process_select proc_finder

{

              command => "sendmail: .*"; # (Anchored) regular expression matching the command/cmd field of a process

       process_result => "command";
```

```
}

############################################################

body process_count anyprocs

{
          match_range => "0,0"; # Integer range for acceptable number of matches for this process


  out_of_range_define => { "process_running" }; # List of classes to define if the matches are out of range

      in_range_define => { "process_not_running" }; # List of classes to define if the matches are in range.


}
```

Chapter 44. Files

EXAMPLE 196

Filename: 400-0010_disable_and_rename.cf

```
body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };

}


bundle agent example {

  files:
      "/bin/chown"
               rename => disable_for_good;
}


body rename disable_for_good
{
              disable => "true";
         disable_mode => "000";
}
```

EXAMPLE 197

Filename: 400-0010_Files.___edit_multiple_files.cf

```
bundle agent example {

      # Demonstrate using regex to edit multiple files

  files:
```

```
      "/tmp/etc/.*.conf"

            edit_line => has_my_name_in_it,
             pathtype => "regex",
              comment => "Every *.conf file in /etc/ should have my name in it.";

}


bundle edit_line has_my_name_in_it {

      insert_lines: "# This file belongs to by Aleksey Tsalolikhin.";

}
```

EXAMPLE 198

Filename: 400-0010_Files.__input_type._preserving_order_while_editing_a_file.cf

```
body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}


bundle agent example {

  files:

      "/tmp/testfile"

              create  => "true",
            edit_line => proper_greetings;
}


#######################################################


bundle edit_line  proper_greetings {
```

```
  insert_lines:

      "Good Evening"
             location => after("Good Day");

}
```

EXAMPLE 199

Filename: 400-0010_Files.__owners.cf

```
bundle agent example {

  files:

      "/tmp/testfile"

              comment => "/tmp/testfile must be mode 612 for application X to work; it must be owned by user aleksey and group cf
              create  => "true",
              perms   => proper_owner("aleksey");

}

#############################################################

body perms proper_owner(user)
{
               owners => { "$(user)", "rob", "user2" };
}
```

EXAMPLE 200

Filename: 400-0010_Files.__perms_groups.cf

```
bundle agent example {

  files:
```

```
      "/tmp/testfile"

                perms => acceptable_groups;
}

body perms acceptable_groups {

               groups => {"root", "games", "mail" };

}
```

EXAMPLE 201

Filename: 400-0010_Files.__rename.cf

```
body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };

}


bundle agent example {

  files:
      "/bin/chown"
               rename => to("/bin/CHOWN");
}
```

EXAMPLE 202

Filename: 400-0010_Files.__Repository.cf

```
body common control {
               inputs => { "libraries/cfengine_stdlib.cf" };
}

bundle agent example {
```

```
  files:

      "/tmp/file.txt"

               create => "true",
            edit_line => insert_lines("$(sys.date)"), # guarantees an edit
          edit_defaults => timestamp,
             repository => "/var/cfengine/repository",
                comment => "Save all history of edits to this important file.";
}

body edit_defaults timestamp
{
          edit_backup => "timestamp";
}

#
# Example output:
#
# [root@web01 verticalsysadmin_training_examples]# cf-agent -f ./MISC__files__repository.cf -b example -KI
#  >> Using command line specified bundlesequence
#  -> Edited file /tmp/file.txt
# Moved /tmp/file.txt_1333404966_Mon_Apr__2_17_16_07_2012.cf-before-edit to repository location /var/cfengine/repository/_tmp_file
# [root@web01 verticalsysadmin_training_examples]# cf-agent -f ./MISC__files__repository.cf -b example -KI
#  >> Using command line specified bundlesequence
#  -> Edited file /tmp/file.txt
# Moved /tmp/file.txt_1333404969_Mon_Apr__2_17_16_10_2012.cf-before-edit to repository location /var/cfengine/repository/_tmp_file
# [root@web01 verticalsysadmin_training_examples]#
```

Chapter 45. PXEboot Kickstart Server

EXAMPLE 203

Filename: 410-0010_pxeboot_kickstart_server.cf

```
# configure a system to be a pxeboot kickstart server
# and to serve CentOS 5.7 i386.  configure kickstart
# config file to bootstrap CFEngine onto the new system:
# download and install CFEngine RPM, and download,
# install and execute CFEngine policy set.

# assumes that contents of CentOS 5.7 i386 installation DVD
# are in the Apache document root, /var/www/html/centos-5.7-i386
# (TODO - make a promise to mirror CentOS to this directory
# as per http://drcs.ca/blog/how-to-mirror-centos-5-and-use-it-as-a-local-yum-repository/ )
#
#
# assumes the pxeboot/kickstart server address is 192.168.1.1
#
# WARNING: lowers the firewall instead of poking holes for UDP 67 and 69 (bootp and tftp)
# (TODO: poke holes for UDP 67 and 69 instead of lowering firewall)
#
# Assumes CFEngine RPM cfengine-community-3.2.1-1.el5.i386.rpm is in /var/www/html
# (this needs to be done manually as cfengine.com requires login to access RPMs)
#
# Assumes CFengine policy files are in the httpd document root,
# cfengine_inputs.tar

body common control {

               inputs => { "libraries/cfengine_stdlib.cf" };

       bundlesequence =>        {
                                  "packages",
                                  "enable_services",
                                  "configure_dhcpd_config_file",
```

```
                                "run_pxe_commands_to_setup_pxeboot",
                                "start_services",
                                "configure_firewall_to_allow_bootp_and_tftp",
                                # not really.  i just turn off the firewall.
                                # be warned.
                                "configure_kickstart_file",
       };
}

bundle agent packages {


  vars:
      "desired_packages"
      ##########################################################
      #              START OF PACKAGE LIST                     #

                slist =>
      {
        "system-config-netboot",
        "httpd",
        "xinetd",
        "tftp",
        "dhcp",
      };

      #              END OF PACKAGE LIST                       #
      ##########################################################

  packages:

      "$(desired_packages)"

       package_method => yum,
       package_policy => "add";


}
```

```
bundle agent enable_services {

      # make sure services are configured to start at boot

  commands:
      "/sbin/chkconfig xinetd on";
      "/sbin/chkconfig tftp on";
      "/sbin/chkconfig httpd on";
      "/sbin/chkconfig dhcpd on";

}



bundle agent configure_dhcpd_config_file {

  files:
      "/etc/dhcpd.conf"
               create => "true",
             edit_line => my_dhcpd_config;
}

bundle edit_line my_dhcpd_config {

      delete_lines: ".*";

  insert_lines:

      "allow booting;
allow bootp;
class \"pxeclients\" {match if substring(option vendor-class-identifier, 0, 9) = \"PXEClient\"; next-server 192.168.1.1; filename
ddns-update-style ad-hoc;

subnet 192.168.0.0 netmask 255.255.0.0 {
range 192.168.1.2 192.168.1.254;
}
"
          insert_type => "preserve_block";
```

```
}


bundle agent start_services {

      # make sure services are configured to start at boot

  commands:
      "/etc/init.d/httpd start";
      "/etc/init.d/xinetd start";
      "/etc/init.d/dhcpd start";
}

bundle agent configure_firewall_to_allow_bootp_and_tftp {
      # this bundle should edit iptables to allow UDP 67 and 69
  commands:
      "/etc/init.d/iptables stop";  # quick and dirty, not safe
}


bundle agent configure_kickstart_file {

  files:
      "/var/www/html/centos-5.7-i386.ks"

               create => "true",
            edit_line => my_kickstart_file;
}

bundle edit_line my_kickstart_file {


      delete_lines: ".*";

  insert_lines:
      "
cmdline
install
url --url http://192.168.1.1/centos-5.7-i386
```

```
lang en_US.UTF-8
keyboard us
clearpart --all
autopart
network --device eth0 --bootproto dhcp --hostname newborn
rootpw cfengine
firewall --enabled --port=22:tcp --port=22:tcp
authconfig --enableshadow --enablemd5
selinux --disabled
timezone --utc America/Los_Angeles
bootloader --location=mbr --driveorder=hda --append=\"rhgb quiet\"
reboot

%packages
@core
@base
device-mapper-multipath
-sysreport


%post
echo Downloading CFEngine RPM
wget http://192.168.1.1/cfengine-community-3.2.1-1.el5.i386.rpm
echo
echo

echo Downloading CFEngine inputs tar-ball
wget http://192.168.1.1/cfengine_inputs.tar
echo
echo

echo Installing CFEngine RPM
rpm -ihv cfengine-community-3.2.1-1.el5.i386.rpm
echo
echo

echo Removing the masterfiles that were shipped with 3.2.1
echo We provide our own policy set.
rm -f /var/cfengine/masterfiles/*
```

```
echo
echo

echo Extracting CFEngine policies
mkdir /var/cfengine/inputs >/dev/null 2>/dev/null
tar -C /var/cfengine/inputs -xvf cfengine_inputs.tar
echo
echo


echo Running CFEngine for the first time
/usr/local/sbin/cf-agent -I
"
          insert_type => "preserve_block";
}

bundle agent run_pxe_commands_to_setup_pxeboot {

  vars:
      "exec_result" string => execresult("/usr/sbin/pxeos -l", "noshell");

  classes:
      "centos_is_installed"
           expression => regcmp("centos-5.7-i386","$(exec_result)");

  commands:
    !centos_is_installed::
      "/usr/sbin/pxeos -a -i centos-5.7-i386 -p HTTP -D 0 -s 192.168.1.1 -L /centos-5.7-i386 -K 'http://192.168.1.1/centos-5.7-i38
      "/usr/sbin/pxeboot -a -O centos-5.7-i386  -K 'http://192.168.1.1/centos-5.7-i386.ks' -r 1000 192.168";

}
```

Chapter 46. Using cf-monitord

EXAMPLE 204

Filename: 800-0010_Monitor.__Example_of_using_monitoring.cf

```
# report environmental conditions

bundle agent example {

  vars:
      "threshold" int => "50";

      ##########################################3
  classes:

      "CPU_load_high"
           expression => isgreaterthan("$(mon.value_cpu)","$(threshold)");

  reports:
    CPU_load_high::
      "!!!!! CPU LOAD IS OVER THRESHOLD OF $(threshold) percent !!!! ";

}
```

Chapter 47. EXERCISES

1. Write a policy to signal TERM and then KILL to any process matching "trn".

   ```
   Testing it:

   cp /bin/sleep ~/trn
   ~/trn 1000 &
   cf-agent -f ...
   ```

1b. Make a list of processes you don’t want to run (let’s say "trn" and "eggdrop") and put this list into an slist variable. Write a promise to signal *term* and then *kill* to any process in that list.

EXERCISE 2

Write a policy to create /tmp/myname.txt and put your name in it.

EXERCISE 3

Manually create a template: echo *Hello, $(mybundle.myname). The time is $(sys.date).* > /tmp/file.dat

Note: a fully qualified variable consists of the bundle name wherein the variable is defined plus the variable name. Example:

```
bundle agent mybundle { vars: "myvar" string => "myvalue"; }
```

The fully qualified variable name is $(mybundle.myvar).

Now write a policy to populate contents of /tmp/file.txt using this template file, /tmp/file.dat.

Make sure your bundle defines the variable embedded in the template, and that your bundle name matches the bundle name embedded in the template.

Your policy should use an edit_lines bundle containing an insert_lines promise with the following attributes:

```
insert_type => "file",
expand_scalars => "true";
```

If you finish before rest of the group, finish studying the CFEngine Reference Manual chapters 1 -4, and if you finish that, then study the Special Topic guide on Editing File Content.

EXERCISE 4. Classes.

Set a custom class based on the existance of a file. For example:

```
classes:

"file_exists"

expression  =>  fileexists("/etc/site_id") ;
```

Then write another promise that is conditional on the above class.

Run it when the file exists, and when it does not, and observe how CFEngine dynamically configures your server.

EXERCISE 5. File Editing

Write a policy to create /etc/motd as follows: It should **always** say "Unauthorized use forbidden."

EXERCISE 5b. File Editing and Classes

/etc/motd should **always** say "Unauthorized use forbidden."

On weekends, it should also say "Go home, it’s the weekend".

Test by defining "Saturday" class on the command line:

```
cf-agent -D Saturday  --file ... --bundle ...
```

EXERCISE 6. Running CFEngine Non-Interactively (as a Service)

Demonstrating of CFEngine running in non-interactive mode, using /var/cfengine/inputs/promises.cf as its input:

EXERCISE 6a.

Set a variable, and then display its value in a report.

EXERCISE 6b.

Report the current time using:

```
a) output from /bin/date (captured using execresult() function)

b) built-in special variable $(sys.date)
```

EXERCISE 6c. File Editing - replace_patterns - uses CFEngine Standard Library

Create (manually) a data file:

```
/tmp/data.txt

line 1
line 2
line 3
```

Use cf-agent to replace "line 2" with "line two".

EXERCISE 6d. File Editing - replace_patterns - uses CFEngine Standard Library

Manually create a template /var/cfengine/masterfiles/templates/motd.dat:

```
This system is property of __ORGANIZATION__.
Unauthorized use forbidden.
CFEngine maintains this system.
CFEngine last ran on $(sys.date).
```

Write a CFEngine policy to generate /etc/motd from /var/cfengine/inputs/templates/motd.dat as follows:

- Replace *ORGANIZATION* with the name of your organization.
- Expand the special variable $(sys.date).

Use all of the following promise types:

delete_lines insert_lines replace_patterns

EXERCISE 7. Integrate your motd policy from exercise 6b into the default cfengine policy in masterfiles so that it propagates to all servers.

EXERCISE 8. Reporting when CFEngine makes a change to your system

Write a promise that logs when it is repaired to /var/log/cfengine/repairs.log

Reference: Special Topics guide on Reporting.

EXERCISE 11 - Precision File Editing

Insert the following three lines (and you can keep them in order, as a single block, using insert_lines attribute insert_type ⇒ "preserve_block";) into /etc/profile BEFORE the HOSTNAME=… line. (Hint: look at the "location" attribute of insert_lines)

if [-x /bin/custom] then /bin/custom fi

EXERCISE 15 - Using Classes to time Command Execution

Problem: All practice machines should be shutdown at end of class at 17:00

Desired state: The command "/sbin/shutdown -h now" is running when we are in the 17th hour of the day, so the system shuts down cleanly and on time.

EXERCISE 16 - CFEngine Standard Library

Given a file /tmp/file.txt:

apples oranges

Use the CFEngine Standard Library to comment out "oranges" and append "bananas", resulting in:

apples # oranges bananas

Hint: use the following: - bundle edit_line insert_lines - bundle edit_line comment_lines_matching

EXERCISE 17 - CFEngine Standard Library - run a command as a specific user

Run the command /usr/bin/id as user "nobody".

Hint: use "body contain setuid".

EXERCISE 18 - CFEngine Standard Library - linking promises

Problem: Increase security by ensuring sshd is running with "PermitRootLogin no".

How does the system look like in the correct configuration:

1. Make sure /etc/ssh/sshd_config contains the line "PermitRootLogin no"

2. Make sure sshd is running using this configuration

How to code it in CFEngine:

1. a files promise to edit sshd_config

2. a commands promise to restart sshd to reload the new config

Exercise: use the "body classes if_repaired" to link 1 and 2 above to make sure 2 happens.

EXERCISE 19 - Bonus Points - Restart sshd if process start time of sshd predates the modification timestamp of /etc/ssh/sshd_config (Process selection is demonstrated in **Process_Selection** in verticalsysadmin_training_examples)

EXERCISE 20 - Write a CFEngine policy to install and configure a Wiki web service.

Chapter 48. After-class survey

Your opinion is important. Could you please let us know your thoughts on the training you just received?

After Professional Training

For students who received professional training:

Please fill out the survey at

`https://www.surveymonkey.com/s/VSAdmin`

After self-study of the Materials only

For students who bought the course materials only:

`Please email your feedback to aleksey@verticalsysadmin.com:`

1. What did you like best about the materials?
2. What could be improved about the materials?
3. Would you like information about on-site training?