Configuration Automation - CFengine

Matt Brown

WAND Network Research Group

The University of Waikato

How will CFengine help me?

- Automates common tasks across large numbers of hosts
- Saves time
- Increases consistency, reduces 'fat finger' mistakes
- Centralised configuration for all network components

What is CFengine?

- cfservd – CFengine server, speaks 'cfengine' on tcp/5308
- cfagent – Configuration agent that does the hard work on each machine
- cfrun – Triggers cfagent on a remote host
- cfexecd – CFengine's own cron replacement (optional)
- cfenvd – Monitors the environment of a host (optional)

How does CFengine work?

- CFengine implements a high-level language which creates a 'policy'
- The CFengine policy defines the desired state
- Actions – consise specification of common sysadmin tasks
  - files, copy, links, editfiles, resolve, defaultroute, groups
- Classes
  - Classes are used to restrict or invoke rules on specific hosts
  - Classes can be compounded using and/or
- Rules – implement specific actions

How does CFengine work?

```
action:
   compound_class::
      rule1
      rule2
   compound_class2::
      rule3
      rule4
action2:
   compound_class3::
      rule5
```

How does CFengine work?

```
copy:
   any::
      $(inputs)/motd dest=/etc/motd owner=root ...
      $(inputs)/issue dest=/etc/issue owner=root ...
   mailservers::
      $(inputs)/exim.conf dest=/etc/exim4/exim.conf
       owner=root mode=0644 define=restart_exim
resolve:
   !nameservers::
      192.0.2.10
      192.0.2.20
```

How does CFengine work?

- When cfagent is invoked a 'run' begins
- A 'run' consists of two stages
  - update – copies latest policy from cfservd on the policy server
    - controlled via update.conf
  - main – executes policy and updates system
    - controlled via cfagent.conf
- Consecutive runs may be required for complex setups
- Each stage defines an action sequence
  - Specifies the actions that will occur and their ordering
- control, import and group/class actions always occur first

How does CFengine work?

```
control:

   any::

      actionsequence = (

         checktimezone          # Set machine to NZT
         resolve                # Setup /etc/resolv.conf
         packages               # Distro package setup
         copy                   # Copy files
         links                  # Setup symlinks
         editfiles              # Edit textfiles
         shellcommands          # Run arbitrary commands
         tidy                   # Tidy up

      )
```

File Locations

- Examples based on Debian
- /var/lib/cfengine2/ - CFengine base directory
  - ppkeys/ - Trusted key storage, based on IP addresses
  - inputs/ - Policy and configuration files
  - outputs/ - cfrun output
  - modules/ - Internal CFengine state
  - state/ - Internal CFengine state
  - rpc_in/ - Internal CFengine state
  - rpp_out/ - Internal CFengine state

Installing and bootstrapping CFengine

- Debian: `apt-get install cfengine2`
- Redhat/Novell: yum install cfengine ?
- Other: http://www.cfengine.org/download.phtml
  - ./configure; make; make install

- inputs/cf.preconf
  - Executed very early in cfagent run
  - Can be used for initial CFengine setup or to recover from disasters
  - Eg. if inputs/update.conf does not exist, wget it from somewhere
  - Setup CFengine keys

Classes

- Crucial component in controlling what happens where
- CFengine automatically defines a number of 'hard' classes

```
mph:~# cfagent -p -v

Defined Classes = ( 10_1_1 10_1_128 10_1_128_3 10_1_129 10_1_1_254
10_1_232 10_1_240 10_1_240_1 10_1_255 10_1_255_254 32_bit Day27 Hr16
Hr16_Q1 January Min05_10 Min09 Q1 Saturday Yr2007 any cfengine_2
cfengine_2_1 cfengine_2_1_20 compiled_on_linux_gnu crc_net_nz
gw_mwp_crc_net_nz i586 ipv4_10 ipv4_10_1 ipv4_10_1_1 ipv4_10_1_128
ipv4_10_1_128_3 ipv4_10_1_129 ipv4_10_1_129_1 ipv4_10_1_1_254
ipv4_10_1_232 ipv4_10_1_232_1 ipv4_10_1_240 ipv4_10_1_240_1
ipv4_10_1_255 ipv4_10_1_255_254 linux linux_2_6_16_2_geode linux_i586
linux_i586_2_6_16_2_geode
linux_i586_2_6_16_2_geode__2_Mon_Oct_30_20_52_07_UTC_2006 mph
mph_crc_net_nz mph_gtw_crc_net_nz mph_mwp_crc_net_nz net_iface_dummy0
net_iface_gtw net_iface_int net_iface_lo net_iface_mwp net_iface_mxx
net_iface_tmp net_nz nz )
```

Classes

- Policy scripts can define and manipulate classes

```
groups:

    # Based on shell command results

    GeodeCPU = (ReturnsZero(/bin/grep -q “Geode”
      /proc/cpuinfo))


    # Based on inbuilt function results

    mailservers = (IPRange(192.0.2.240/28))


    # Based on other hard classes (eg. hostname)

    Service_quagga = (

        mph

        mfr

    )
```

Classes

- Policy scripts can define classes dynamically

```
control:
    any::
        AddInstallable = ( restart_quagga )
copy:
    Service_quagga::
        $(inputs)/ospfd.conf /etc/quagga/ospfd.conf
            define=restart_quagga
shellcommands:
    restart_quagga::
        "/etc/init.d/quagga restart"
```

Classes

- Classes can also be manipulated at the command line

```
mph:~# cfagent -q -v -Ddist_upgrade
shellcommands:
    dist_upgrade::
        "/usr/bin/apt-get -y dist-upgrade"
```

apt-get only executed when -Ddist_upgrade occurs

```
mph:~# cfagent -q -v -Nservice_quagga
import:
    Service_quagga::
        cf.quagga
```

Prevents cf.quagga from being imported

Variables

- Defined in the control section
- Can be set to explicit values or use a number of special functions

```
control:

  any::

    workdir = ( /var/lib/cfengine2 )

    inputs = ( $(workdir)/inputs )

    host_config = ( $(inputs)/hosts/$(host) )

    kver = (ExecShellResult(/bin/uname -r | cut -f -2 -d-))

    rand = (RandomInt(5,10))

    motd = (ReadFile(/etc/motd,1024))
```

Special Variables

- Can be read to get information or set to control operation
  - $(host) - hostname
  - $(smtpserver) - server to send email through
  - $(sysadmin) - email address to sent reports to
  - $(timezone)
- Most of these can be modified in the control action
- Variables can be referenced with $(var) or ${var}

Authentication

- TCPwrappers style allow/deny specification
- CFengine uses mutual public key based authentication
  - cfservd must trust cfagent key, and cfagent must trust cfservd key
- Each host could be both cfagent and cfservd
  - Typically one host is dedicated as the policy 'server'
- Host key lives in $(cfbasedir)/ppkeys/localhost.{pub,priv}
  - /var/lib/cfengine2/ppkeys/localhost.{pub,priv}
- Generate keys using the cfkey utility

Trust

- CFengine trusts a key when it is found in ppkeys/
  - Lookup is based on remote user and source IP address
- cfservd running as policy server at 192.0.2.1
- cfagent running as root@192.0.2.40 connects to 192.0.2.1
- cfservd at 192.0.2.1 checks:
  - Does $(cfbasedir)/ppkeys/root-192.0.2.40.pub exist?
- cfagent at 192.0.2.40 checks:
  - Does $(cfbasedir)/ppkeys/cfengine-192.0.2.1.pub exist?
- Both sides use challenge/response auth to verify that the remote end holds the corresponding private key

Establishing Trust

- TrustKeysFrom cfservd configuration option

```
TrustKeysFrom = ( 192.0.2.0/24 )
```

- Explicitly trusts the **first** key seen from an IP/user tuple
  - ppkeys/ must be manually updated if a host's key changes

- Out of band trust establishment
  - Manually copy key from client and install in ppkeys/ on the server
  - Pregenerate keys for all clients on the server
    - Client retrieves keys as part of installation process, perhaps using cf.preconf script

Dynamic Addresses

- CFengine can cope with host IP addresses changing
  - With some loss of security
- DynamicAddresses cfservd.conf configuration option

```
DynamicAddress = ( 192.0.2.56/29 )
```

- Keeps a database of trusted keys, allows connection if incoming key has been used before
- Can be combined with TrustKeysFrom to accept any incoming key
- Very useful for installation of CPE type devices

Dynamic Addresses

```
control:
    # Trust dynamic keys from the installation subnet
    TrustKeysFrom = ( 192.0.2.48/29 )
    DynamicAddresses = ( 192.0.2.48/29 )
    # Allow dynamic keys from the CPE network range
    DynamicAddresses = ( 192.0.2.56/29 )
```

- Allows any CPE to connect from 192.0.2.56/29 provided that it has been configured from within 192.0.2.48/29 to initialise the dynamic key database with it's key.

Example – snippets from CRCnet

- /var/lib/cfengine2/inputs/ subdirectories and contents

`cfservd.conf`	`Policy server configuration`
`cfrun.hosts`	`cfrun configuration and host list`
`cfconf/`	`Policy files for CFengine clients`
`shared/`	`Shared input files`
`<distroname>/`	`Distribution specific configs`
`hosts/<hostname>/`	`Host specific configs`

- All configs stored in a subversion repository for version control

Example – cfservd.conf – Policy Server

```
control:

    any::

        domain          = ( crc.net.nz )

        cfrunCommand    = ( "/usr/sbin/cfagent" )

        # Connecting user must be root or cfengine

        AllowUsers      = ( root cfengine )


        AllowConnectionsFrom        = ( 10.1.0.0/16 )

        TrustKeysFrom               = ( 10.1.224.0/24 )

        DynamicAddresses            = ( 10.1.224.0/24 )

        DynamicAddresses            = ( 10.1.247.0/24 )
```

Example – cfservd.conf – Policy Server

```
control:

    any::

        IfElapsed            = ( 0 )

        ExpireAfter          = ( 15 )

        MaxConnections       = ( 40 )

        MultipleConnections = ( true )

        LogAllConnections    = ( true )


grant:

    any::

        /var/lib/cfengine2/inputs  10.1.0.0/16

        /usr/bin/cfagent           127.0.0.1
```

Example – cfrun.hosts – Policy Server

```
domain          = crc.net.nz

outputdir       = /var/lib/cfengine2/outputs

maxchild        = 30    # no more than 30 config pushes at once

access          = root,cfengine


# List of unqualified hostnames that can be pushed to

mph

mfr -D custom_class

mwp:3333        # Non-standard port

...
```

Example – update.conf – Client config

```
control:

    any::

        actionsequence       = ( copy processes tidy )

        domain               = ( crc.net.nz )

        policyhost           = ( cfengine.crc.net.nz )

        workdir              = ( /var/lib/cfengine2 )

        policy_cfinput       = ( $(workdir)/inputs )

        client_cfinput       = ( $(policy_cfinput)/cfconf )

        host_ip              = (ExecShellResult(/sbin/ip addr
          ls dev dummy0 | grep inet | awk '{print $2}' | cut
          -f1 -d'/')

    !PXE_booting::

        BindToInterface      = ( $(host_ip) )
```

Example – update.conf – Client config

```
copy:
    any::
        $(client_cfinput)
            dest=$(workdir)/inputs
            r=inf
            mode=600 owner=root group=root
            exclude=*~ exclude=*.swp ignore.svn
            server=$(policyhost)
            backup=false
            type=sum
```

Example – cfservd.conf – Client config

```
control:

    any::

        domain                  = ( crc.net.nz )

        policyhost              = ( cfengine.crc.net.nz )

        policyip                = ( A.B.C.D )

        cfrunCommand            = ( /usr/sbin/cfagent )

        AllowUsers              = ( root cfengine )

        AllowConnectionsFrom = ( $(policyip) )


grant:

    any::

        /usr/sbin/cfagent    $(policyip)
```

Example – cfagent.conf – Client config

```
control:

    any::

        OutputPrefix          = ( "$(host):" )

        AddInstallable        = ( Distrib_crcnet
          Distrib_crcnetbpc Service_hostapd Service_quagga
          Service_radius )

import:

    any::

        cf.groups

        cf.main

    Distrib_crcnet:  cf.crcnet

    Distrib_crcnetbpc: cf.crcnetbpc

    Service_hostapd: cf.hostapd

    ...
```

Example – cf.groups – Client config

```
groups:
    Distrib_crcnet = ( gtw peon orwell ... )
    Distrib_crcnetbpc = ( mph mfr mwp pir ... )
    Service_hostapd = ( pir mfr )
    Service_radius = ( peon )
    Service_quagga = ( gtw mph mfr mwp ... )
    AssetType_Soekris = ( mph mfr mwp pir )
    AssetType_Atheros = ( mfr mwp pir )

    CPE = ( FileExists(/etc/cpe_version) )
```

Example – cf.main – Client config

```
control:

    any::

        domain          = ( crc.net.nz )

        smtpserver      = ( smtp.crc.net.nz )

        sysadm          = ( matt@crc.net.nz )

        timezone        = ( NZST )


        policyhost      = ( cfengine.crc.net.nz )

        policyip        = ( A.B.C.D )


        ChecksumUpdates = ( on )

        AddInstallable  = ( restart_cfengine )
```

Example – cf.main – Client config

```
control:

    any::

        workdir                 = ( /var/lib/cfengine2 )

        inputs                  = ( $(workdir)/inputs )

        host_config             = ( $(inputs)/hosts/$(host) )

        shared_config           = ( $(inputs)/shared )


        DefaultPkgMgr           = ( dpkg )

        DPKGInstallCommand      =

          ( “/usr/bin/apt-get -y install %s” )
```

Example – cf.main – Client config

```
control:

    any::

        actionsequence = (

            checktimezone

            resolve

            packages

            copy

            links

            editfiles

            shellcommands

            files

            tidy

            processes )
```

Example – cf.main – Client config

```
resolve:

    any::

        10.1.0.1

        10.1.0.2


copy:

    any::

        $(shared_config)/cf_defaults
          dest=/etc/default/cfengine2 mode=0644 owner=root
          group=root server=$(policyhost) type=sum
          define=restart_cfengine

    !CPE::

        # Copy Host SSH keys onto the client
```

Example – cf.main – Client config

```
files:

    any::

        /usr/bin mode=o-w checksum=md5 r=inf action=fixplain
         ifelapsed=60

        /usr/sbin mode=o-w checksum=md5 r=inf action=fixplain
         ifelapsed=60

        ...


tidy:

    any::

        /tmp pattern=* exclude=.* age=3 r=inf ifelapsed=60

        /var/tmp pattern=* exclude=.* age=3 r=inf
         ifelapsed=60
```

Example – cf.main – Client config

```
shellcommands:
    restart_cfengine::
        "/etc/init.d/cfengine2 restart"


processes:
    any.!PXE_booting::
        "cfservd" restart "/etc/init.d/cfengine2 start"
```

Example – cf.quagga – Client config

```
control:

    any::

        AddInstallable       = ( restart_quagga )

    restart_quagga:

        autodefine           = ( /etc/quagga/*.conf )

files:

    any::

        /var/run/quagga r=0 mode=0755 owner=root group=root
         action=fixdirs ifelapsed=60

        /etc/quagga/* r=0 mode=0640 owner=root group=root
         action=fixplain inform=true syslog=true ifelapsed=60

    !Distrib_crcnetbpc::

        /var/run/quagga/*.vty r=0 mode=0770 owner=quagga
         group=quaggavty action=fixall ...
```

Example – cf.quagga – Client config

```
packages:

    any::

        quagga pkgmfr=dpkg action=install ifelapsed=120

copy:

    Distrib_crcnetbpc::

        $(host_config)/ospfd.conf dest=/etc/quagga/ospfd.conf
         mode=0644 type=sum owner=root group=root encrypt=yes

        ...

    !Distrib_crcnet::

        $(host_config)/ospfd.conf dest=/etc/quagga/ospfd.conf
         mode=0664 type=sum owner=quagga group=quaggavty
         encrypt=yes

        ...
```

Example – cf.quagga – Client config

```
shellcommands:

    any::

    restart_quagga::

        “/etc/init.d/quagga restart”
```

Example – cf.radius – Client config

```
control:

    any::

        AddInstallable = ( restart_freeradius )

packages:

    any::

        freeradius pkgmgr=dpkg version=1.1.3-1crcnet4 cmp=eq
         action=install ifelapsed=120

copy:

    any::

        $(host_config)/radius/ dest=/etc/freeradius/
         mode=0640 recurse=2 owner=root group=freerad
         ignore=.svn define=restart_freeradius backup=false

        ...
```

WAND Network Research Group

Department of Computer Science

The University of Waikato

Private Bag 3105

Hamilton, New Zealand

www.crc.net.nz

www.wand.net.nz

www.waikato.ac.nz

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato