

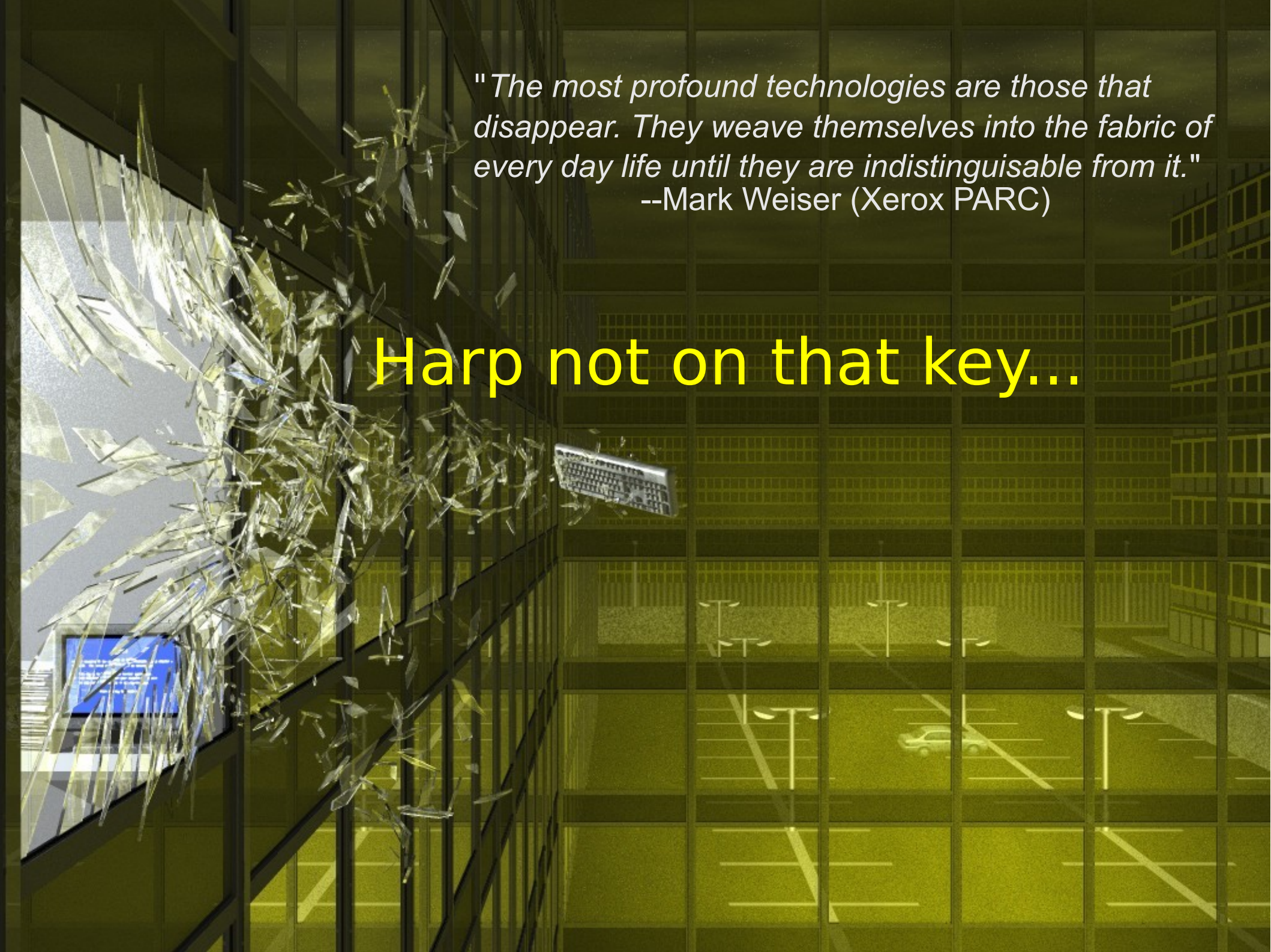
# **Cfengine in a Day**

**Configuration Management for  
Complex Business Problems**

Mark Burgess

© **Cfengine**

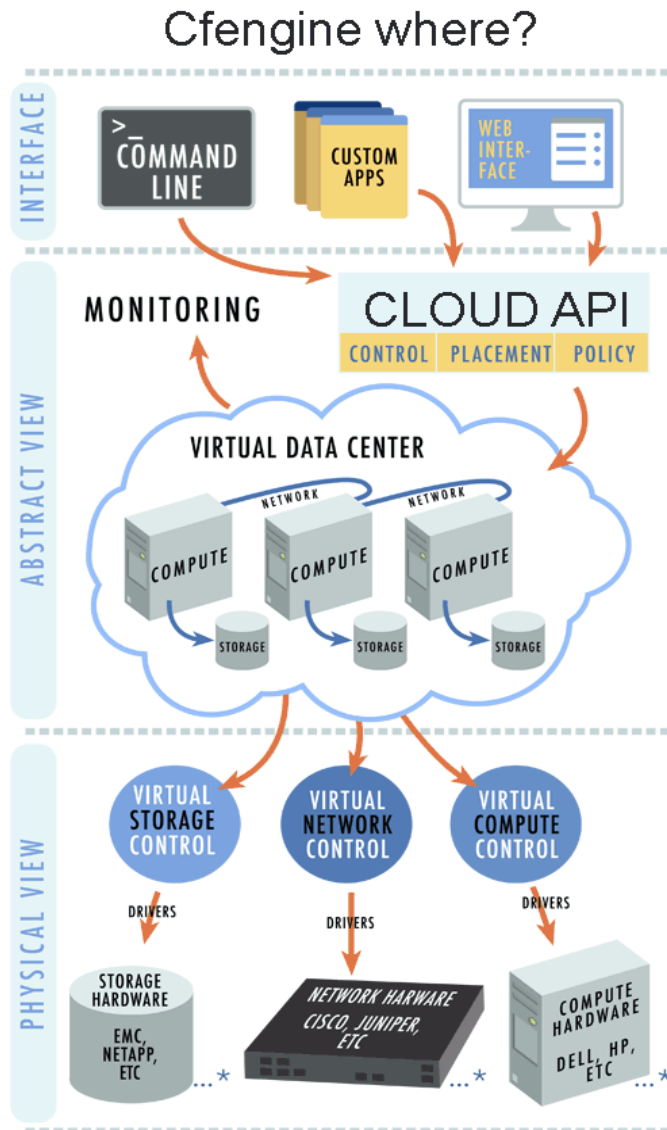




*"The most profound technologies are those that disappear. They weave themselves into the fabric of every day life until they are indistinguishable from it."*  
--Mark Weiser (Xerox PARC)

Harp not on that key...

# Architecture of computing to come



- On demand, disposable computing
- Cfengine is the glue and maintenance at any level to integrate subsystems
- Extensible concept, can adapt to new technologies
- Seamless services



# Risky implementations or quick regular dependability?

- **Rockets**

- Planning
- Overhead
- One chance only
- Major rollout

- **747**

- Less planning
- Reusable
- Change anytime



# Cfengine is...

- An extensive language for manipulating cross-platform environments
- A simple monitoring framework with feedback
- A knowledge resource
- Cfengine is reborn in the Cfengine 3 family
  - Cfengine Community Edition
  - Cfengine Nova Edition
  - Cfengine Constellation
  - Cfengine Galaxy

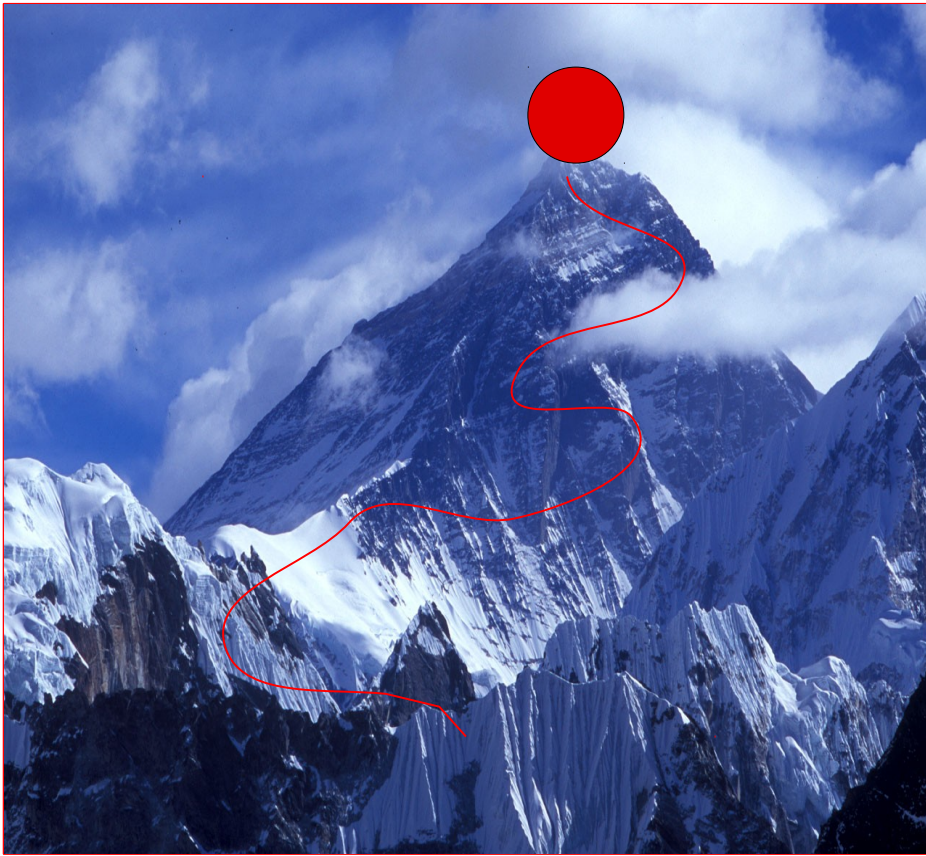


# What's unique about Cfengine?

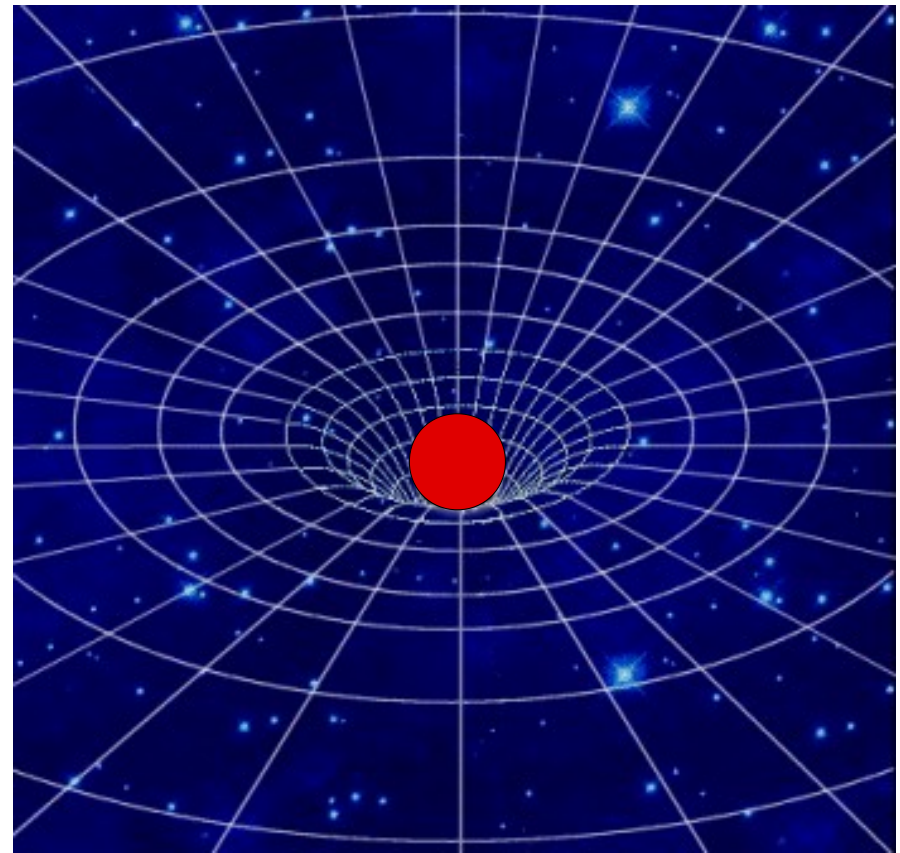
- It's a full maintenance system, not just “roll out”
  - Self-healing (convergent)
  - Self-monitoring
  - Feedback
- Lightweight
- Versatile framework for many tasks – not just package based
- It's based on a model called Promise Theory



# Convergence = self-healing



Baseline and recipe



Convergence to end state

# Solving complex business problems

- Language interface
  - Low level flexible primitives
  - Create multiple interfaces
- Enables multiple overlapping cells
  - Federated management
  - Great freedom to choose
  - “Devops” - complete service application from metal



# Two approaches to starting

- Hit the ground running / get it done / top-down
  - Get started quickly
  - Don't want to think too hard in the beginning
- Psych up slowly / build a model / bottom-up
  - Build a mental model first
  - Strict QA process around the system



# Two aspects to Cfengine

- Study the simple use of Cfengine
  - Install a package from [www.cfengine.com](http://www.cfengine.com) (Tech Corner)
  - Install from source code at [www.cfengine.org](http://www.cfengine.org)
  - Learn about configuration by example
  - Top down
- Study the Cfengine language
  - Jump into a coding approach – syntax, etc
  - Bottom up

# Ultimate goal / benefit

- Learn a method for a scalable way of working
  - Take an issue
  - Turn it into maintainable promises (“guarantees”)
  - Code it in Cfengine
- Keep in mind motivations and constraints:
  - Business integration
  - Reproducible and agile
- **Everything you say becomes a promise**

# Some hands on examples

- After the session you can try “hands-on” examples in the Cfengine source code.
- These get installed from source in:

```
cd /usr/local/share/doc/cfengine
```

```
cf-agent -f ./unit_example.cf
```

```
cf-agent -vf ./unit_example.cf
```





# Installing

- `tar xzf cfengine-3.x.x.tar.gz`
  - `configure && make install`
  - `/usr/local/sbin/cf-key`
  - `cp /usr/local/sbin/cf-* /var/cfengine/bin`
  - `cf-agent`
- 
- `rpm -i cfengine-nova.rpm`
  - `cf-agent --bootstrap --policy-server 123.456.789.xyz`

# What is your perspective?

- Again we can go top-down or bottom-up
  - By issue / service
  - By resource / object class
- How do you see your system?
  - As config files? (/etc/passwd)
  - As packages? (apache2)
  - As services? (DNS)
  - As procedures (patching)
  - As business problems?



# By “service”

- (Show web-server demo)
- Cfengine “bundles” certain promises together, so we can build different abstractions or interfaces to the low-level
- Orion Cloud pack is an example of this
  - Make a bundle for each service
  - Comment in or out, or switch on/off



# By “resource”

- Registry management, DNS demos
- COPBL library
  - A standard library interface
  - Lingua franca for common issues and problems
- The downside of simplifications is that they always break down. But Cfengine allows you to make new ones without delving into the code





# **Gallery of basic promises**

# Examples: the wrapper

```
body common control
{
bundlesequence => { "main"};
inputs => { "cfengine_stdlib.cf" };
}
```

```
bundle agent main
{
# example
}
```

**RED** = cfengine word

**BLUE** = user defined word  
Or COPBL

**BLACK** = user defined data

# Create files / dirs

files:

```
"/home/mark/tmp/test_plain"  
  perms => mog("644", "root", "wheel"),  
  create => "true";  
  
"/home/mark/tmp/test_dir/."  
  perms => mog("644", "root", "wheel"),  
  # will add +x for dirs, see also rxdirs  
  create => "true";
```



# Copy files

files:

```
"/home/mark/tmp/test_plain"
```

```
copy_from => local_cp("$ (sys.workdir)\bin\file");
```

```
"/home/mark/tmp/test_dir"
```

```
copy_from =>  
  secure_cp("$ (sys.workdir)/bin", "serverhost");
```





# Copy directories / trees

files:

```
"/home/mark/tmp/test_dir"
```

```
copy_from => local_cp("${sys.workdir}\bin\."),  
depth_search => recurse("inf");
```

```
"/home/mark/tmp/test_dir"
```

```
copy_from => secure_cp("${sys.workdir}/bin","serverhost"),  
depth_search => recurse("inf");
```



# Editing password/group files

vars:

```
"userset" slist => { "user1", "user2", "user3" };
```

files:

```
" /home/mark/tmp/passwd"
```

```
  edit_line =>
```

```
    set_user_field("mark","7","/set/this/shell");
```

```
" /home/mark/tmp/group"
```

```
  edit_line =>
```

```
    append_user_field("root","4","@(main.userset)");
```



# Editing cron files

methods:

"cron"

```
usebundle => cronjob("/bin/ls", "mark", "*", "5,10");
```

"cron"

```
usebundle => cronjob("/bin/pwd", "mark", "*", "5,10,15");
```



# Disabling and rotating

files:

```
"/home/mark/tmp/test_create"
```

```
  rename => disable;
```

```
"/home/mark/tmp/rotateme"
```

```
  rename => rotate("4");
```



# Hashing for change detection

files:

```
"/home/mark/tmp" -> "me"  
  changes          => detect_all_change,  
  depth_search     => recurse("inf"),  
  action           => background;
```

```
"/home/mark/LapTop/words" -> "you"  
  changes          => detect_all_change,  
  depth_search     => recurse("inf");
```



# Command embedding

commands:

```
Sunday.Hr04.Min05_10.myhost::
```

```
"/usr/bin/update_db";
```

```
any::
```

```
"/etc/mysql/start"
```

```
contain => setuid("mysql");
```



# Kill processes

```
processes:
```

```
  "snmpd"
```

```
    signals => { "term", "kill" };
```

# Restart processes

```
processes:
```

```
    "httpd"
```

```
        restart_class => "lift_off";
```

```
commands:
```

```
    lift_off::
```

```
        "/etc/init.d/apache2 restart";
```





# Check filesystems

storage:

```
"/usr" volume => mycheck( "10%" );
```

# Mount filesystems

storage:

```
"/home/mark/server_home"
```

```
mount => nfs("myserver", "/home/mark");
```



# Software/patch installation

```
packages:
```

```
"apache2"
```

```
package_policy => "add",  
package_method => generic;
```

# PHP-Web service

```
bundle agent app_web_phpapache
{
  vars:
    centos:: "php_pkgs" slist => { "httpd", "php" };
    ubuntu:: "php_pkgs" slist => { "apache2", "php5" };

  packages:
    "$(php_pkgs)"
      comment => "Install Apache webserver with PHP",
      package_policy => "add",
      package_method => generic,
      classes => if_ok("ensure_php_apache_running");

  processes:
    centos.ensure_php_apache_running::
      ".*httpd.*"
      restart_class => "start_httpd";
    ubuntu.ensure_php_apache_running::
      ".*apache2.*"
      restart_class => "start_apache";

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

    start_apache::
      "/etc/init.d/apache2 start";
}
```



# SQL service

```
bundle agent app_db_mysql
{
  vars:
    "mysql_pkgs" slist => { "mysql-server" };

  packages:
    "$(mysql_pkgs)"
      comment => "Prepare MySQL database",
      package_policy => "add",
      package_method => generic,
      classes => if_ok("ensure_mysql_running");

  files:
    ubuntu::
      "/tmp/mysql.sock"
        comment => "Create a temp link to mysql.sock",
        link_from => ln_s("/var/run/mysqld/mysqld.sock");

  processes:
    ubuntu.ensure_mysql_running::
      "/usr/sbin/mysqld.*"
        restart_class => "ubuntu_start_mysql";

  commands:
    ubuntu_start_mysql::
      "/etc/init.d/mysql start";
}
```



# What of Knowledge?

- Hopefully these examples are fairly readable to with basic familiarity
  - What?
  - When?
  - Where?
  - Why?
- But we still don't know *why* we are making these promises
  - Instrument configuration with semantic commentary



# Commentary

```
bundle component name(parameters)
{
what_type:
  where_when::

  # Traditional comment

  "promiser" -> { "promisee1", "promisee2" },
    comment => "The intention ...",
    handle  => "unique_id_label",
    attribute_1 => body_or_value1,
    attribute_2 => body_or_value2;
}
```



# Example – informing operators

processes:

# Comments to expert policy writers

"snmpd"

```
comment => "Make sure SNMP is not active by default",  
handle  => "snmp_kill",  
signals => { "term", "kill" };
```





# Commentary => Knowledge

- The comments appear in log messages
- In Cfengine Nova and upwards they are turned into a relational map
- See relationships, impact analysis
- Understand who are the stakeholders in these promisers
- Tie this knowledge about policy to other institutional documentation (cf-know)



# What does simple mean?

- So far this is artificially simple, as we haven't shown any realistic techniques
  - What happens when you have a large number of these?
- Patterns are the key to “scaling up”
  - Patterns compress information into expressions
- It's a question of Knowledge Management
  - The choice should be yours
  - Cfengine tries to give you flexibility



# What does “pattern” mean?

- Something that models many *actual* cases with a symbolic expression
- Cfengine uses these pattern expressions:
  - **Lists** – enumerated cases
  - **Regular expressions** + search – look and see
  - **Classes** of hosts where promises will apply
- Patterns in Cfengine are generic “networks” not just “hierarchies”, so you can avoid OO-modelling traps.



# Classes - when/where

- Not “OO” classes - actually “sets” of machines with selected properties
- Cfengine evaluates classes each time it wakes up to decide which promises apply “here”

```
promise_type:
```

```
    class_expression::
```

```
        “promiser”
```

```
        attributes => “values”;
```



# Class summary

```
cf3 Defined Classes = (  
  any verbose_mode Sunday Hr08 Morning Min32 Min30_35 Q3  
  Hr08_Q3 Day21 June Yr2009 Lcycle_2 GMT_Hr6 linux atlas  
  undefined_domain 64_bit linux_2_6_27_23_0_1_default  
  x86_64 linux_x86_64 linux_x86_64_2_6_27_23_0_1_default  
  linux_x86_64_2_6_27_23_0_1_default__1 SMP_2009_05_26_17_0  
  2_05_0400 compiled_on_linux_gnu localhost_localdomain  
  localhost net_iface_lo net_iface_wlan0 ipv4_192_168_1_100  
  ipv4_192_168_1 ipv4_192_168 ipv4_192  
  fe80__21c_bfff_fe6e_70ef cfengine_3_0_2a7 cfengine_3_0  
  cfengine_3 SuSE lsb_compliant suse suse_na suse_11_1  
  suse_11 common have_ppkeys )
```

```
Cf3 Negated Classes = ( )
```



# Using classes to make decisions

```
promise_type:

    myclass::

        "promiser_1" ....

    class1|class2::

        "promiser_2" ....

    linux.Hr02|solaris::

        "promiser_3" ....
```



# Custom classes

classes:

```
"myclass" or => { "solaris", "linux" };
```

```
"my_dist" dist => { "10", "20", "40", "30" };
```

```
"summary" expression => classmatch("web.*");
```

```
"some" expression => "linux&Hr02|solaris";
```



# Class promises are kept at runtime

```
cf3 *****
cf3  BUNDLE test
cf3 *****
cf3  =====
cf3  classes in bundle test
cf3  =====
cf3
cf3  +  Private classes augmented:
cf3  +      my_dist
cf3  +      my_dist_10
cf3  +      myclass
cf3
cf3  -  Private classes diminished:
cf3
cf3  ?  Public class context:
cf3  ?      any
cf3  ?      Tuesday
cf3  ?      Hr15
cf3  ?      Min35
cf3  ?      Min35_40
cf3  ?      Q3
```



# Combining classes

- Classes are combined with:
  - NOT: !
  - AND: . (dot) or &
  - OR: | or ||
  - Parentheses ()

```
freebsd.!Hr02||solaris.Hr03
```

```
(freebsd||linux).!mygroup
```



# Regular expressions?

- Many people find regexs hard – very powerful
- Cengine (as of major version 3) uses **Perl Compatible Regular Expressions** throughout
  - Choose from a menu of simple cases you understand
  - Don't get more fancy than your own limit
  - Remember that the point is to make something difficult seem simple, not to show off your skill!
- Come back to this below



# Regular expression examples

- Crash course
  - Dot `.` matches any character, `\.` matches “dot”
  - `[a-z1258]` matches the listed characters
  - `\s` space characters
  - `+` means more than zero
  - `*` means zero or more
- Examples:
  - `.*` match anything
  - `\s+` match at least one whitespace



# One regex example

files:

`"/etc/pass.*"`

`perms => mo("644","root");`

# The “access list” paradigm

- We find it easy to think in terms of lists
- Enumerated data sets, like a “for” loop
- Cfengine expands list variables transparently so that a single expression can represent many similar promises:

`files:`

`“$(list_variable)”`

`perms => mo( "644", "root" );`



# Define a variable

vars:

```
"scalar"  string => "scalar values";
```

```
"listvar"  slist => { "one", "two",  
                      @(otherlist) };
```

```
"otherlist" slist => { "three", "four" }
```

commands:

```
"/bin/echo hello $(listvar);
```



```
#
# apache's default installation
# APACHE_MODULES="access actions alias asis auth autoindex cgi dir imap include log_config mime
negotiation setenvif status userdir"
# your settings
APACHE_MODULES="authz_host actions alias auth_basic authz_groupfile authn_file authz_user autoindex
cgi dir include log_config mime negotiation setenvif status asis dav dav_fs ssl php5 dav_svn authz_default
xyz superduper"
## Type:  string
## Default:  ""
## ServiceRestart: apache2APACHE_USE_CANONICAL_NAME="off"

## Type:  list(Major,Minor,Minimal,ProductOnly,OS,Full)
## Default:  "OS"
## ServiceReload: apache2
#
# How much information the server response header field contains about the server.
# (installed modules, versions, etc.)
# see http://httpd.apache.org/docs-2.0/mod/core.html#servertokens
#
APACHE_SERVERTOKENS="OS"
#
APACHE_EXTENDED_STATUS="off"

## Type:  list(on,off)
## Default:  "off"
## ServiceRestart: apache2
#
# Enable buffered logging
#
APACHE_BUFFERED_LOGS="off"
```



# APACHE

vars:

```
"add_modules"      slist => {  
    "dav",  
    "dav_fs",  
    "ssl",  
    "php5",  
    "dav_svn"  
};
```

```
"del_modules"      slist => {  
    "php3",  
    "jk",  
    "userdir",  
    "imagemap"  
};
```

column\_edits:

```
"APACHE_MODULES=.*"  
  edit_column => quotedvar("${add_modules}", "append");  
  
"APACHE_MODULES=.*"  
  edit_column => quotedvar("${del_modules}", "delete");
```



# USERS

vars:

```
"users"      slist => readlist(file);  
"tmp"        string => "/tmp/scratch";  
"testing"    string => "/var";
```

files:

```
"$(tmp)"  
  edit_line => ReadPasswds ("$(testing)/masterfiles/passwd",  
                             "@(this.users)");  
  
"$(testing)/etc/passwd"  
  edit_line => SetPasswds ("$(tmp)", "@(this.users)");  
  
"$(testing)/home/$(users)/."  
  create => "true",  
  perms  => userdir ("$(users)");
```



## VMs

```
bundle agent testbundle
```

```
{
```

```
vars:
```

```
    "vmlist" slist => { "host1", "host2",  
                        "host3", "host4",  
                        "host5" };
```

```
methods:
```

```
    "any" usebundle => buildvm( "$(vmlist)" );
```

```
}
```



# Blocks: bundles and bodies

- **Bundles** are collections of promises under a single name – like a “subroutine”
- **Bodies** are **template macros** for simplifying complex sets of promise attributes.
- Declaration:

```
bundle agent myname()  
{  
  files:  
  ..  
    perms => yourname,  
}  

```

```
body perms yourname()  
{  
  mode =>  
}  

```

# “Calling” bundles

- Two ways:
  - Add to bundlesequence (master schedule)
  - Methods promises (more flexible)

```
bundle agent myname()  
{  
  methods:  
    classes::  
    "group" usebundle => yourname("args"),  
}
```

# Linkage...

```
body common control
```

```
{
```

```
bundlesequence => { "myname" };
```

```
}
```

```
bundle agent myname
```

```
{
```

```
methods:
```

```
  classes::
```

```
    "group" usebundle => yourname("args"),
```

```
}
```

# The pattern

Cfengine word		User defined word
What is it?	What's it for?	What's its name?
bundle	agent server	myname
body	perms signals	yourname

# Cfengine Architecture

- Cfengine is an agent based system
- It is composed of multiple binaries
- It admits any kind of network model
  - Uses peer to peer authentication for fully decentralized and devolved management
  - Works on and offline (not dependent on network)
- Core functions are not dependent on 3<sup>rd</sup> party pre-requisites

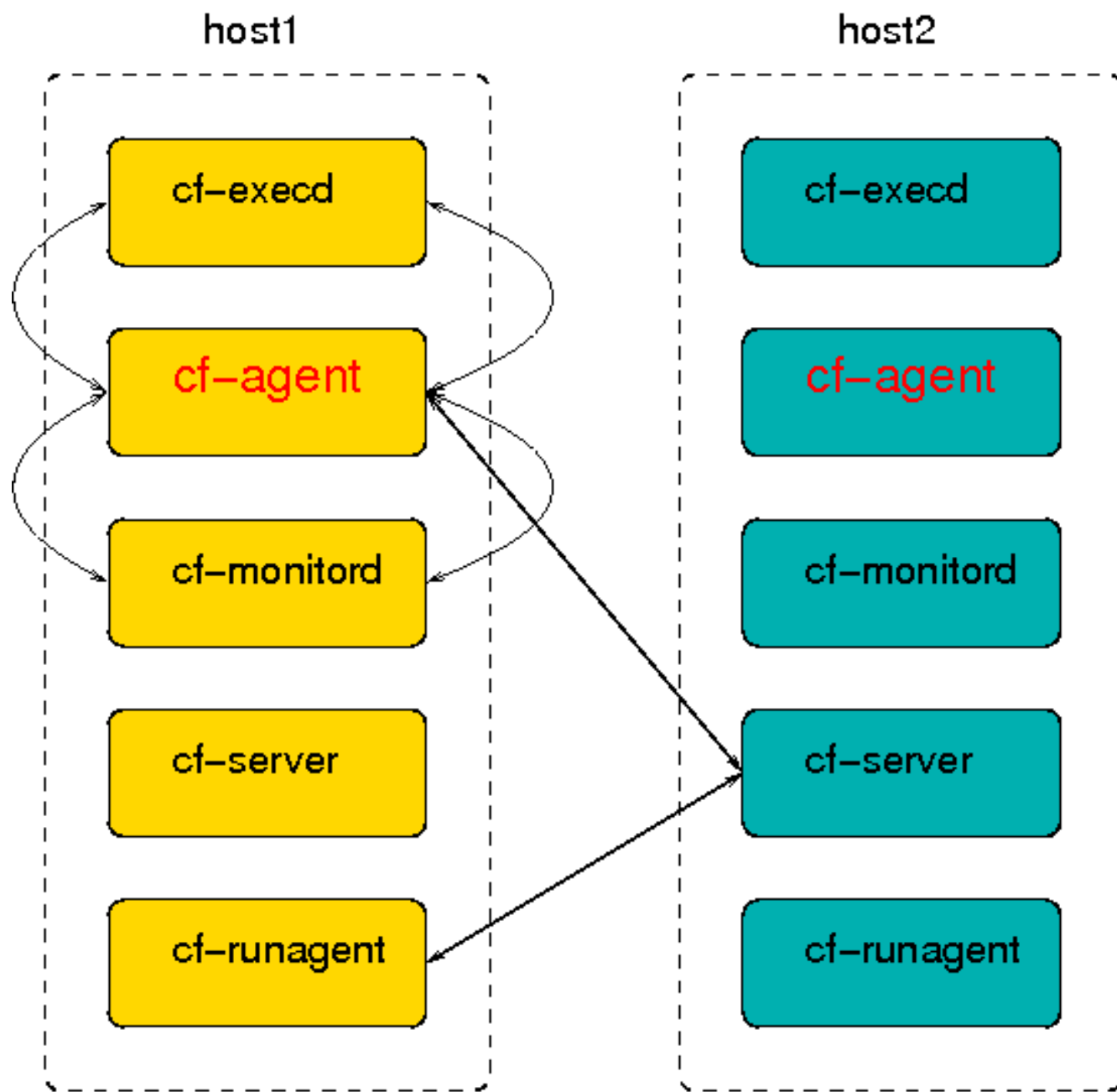


# Cfengine components

- cf-agent (cfagent)
- cf-serverd (cfservd)
- cf-monitord (cfenvd)
- cf-execd (cfexecd)
- cf-promises – promise checker
- cf-report – reporting tool
- cf-know – integrated knowledge management
- Cf-runagent - test





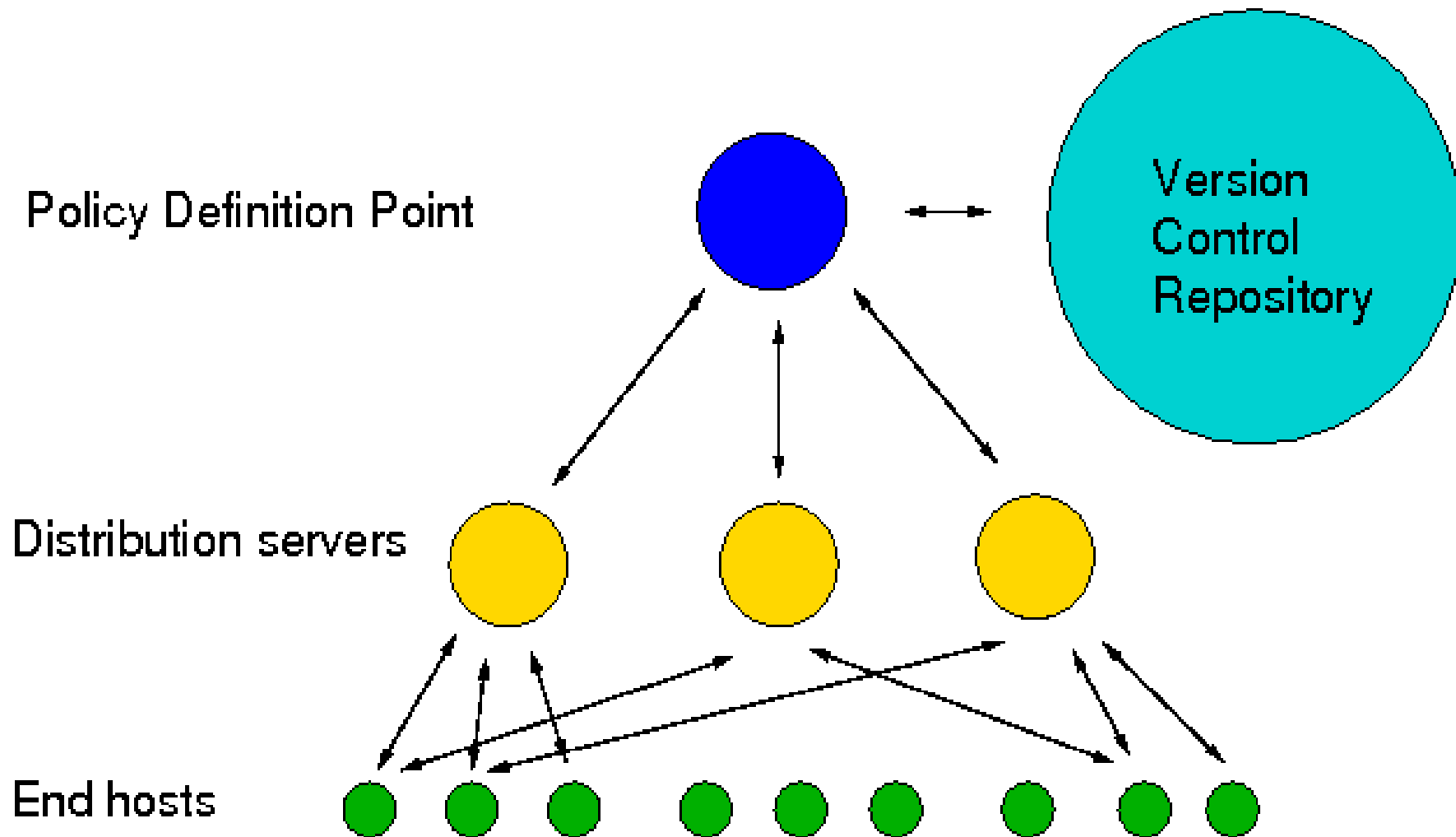


# Policy architecture

- Every machine can be standalone if you want
- Centralization is a useful tool for consistency but a bottleneck for performance
  - Use centralization wisely
  - It is less important than you think
- There is no push in cfengine



# Stages of policy



# Client-Server communications

- SSH-like model
- Not a certificate model
- Not susceptible for SSL bugs/vulnerabilities
- Cfengine Nova
  - Engineered to FIPS 140-2 standards
  - Scales up to a few hundred machines with maximal usage, 5 minute checks and no load balancing
  - Can spread out policy updates more as the numbers grow



# Hands on

## unit\_server\_copy\_localhost.cf



# Server copy -single file example

```
body common control
{
  bundlesequence => { "testbundle" };
  version => "1.2.3";
}

bundle agent testbundle
{
  files:

    "/home/mark/tmp/testcopy"

    copy_from      => remote_cp("/home/mark/src","127.0.0.1"),
    perms           => system,
    depth_search   => recurse("inf");
}

body perms system
{
  mode  => "0644";
}
```



```
body copy_from remote_cp(from,server)
{
source          => "$(from)";
copy_backup     => "true";
purge           => "true";
servers         => { "$(server)", "failover_host" };
}
```

```
body server control
{
allowconnects    => { "127.0.0.1" };
allowallconnects => { "127.0.0.1" };
trustkeysfrom    => { "127.0.0.1" };
}
```

```
bundle server access_rules()
{
access:

    "/home/mark/LapTop"

    admit    => { "127.0.0.1" };
}
```



# Voluntary remote services

- Cannot tell `cf-agent` what to do remotely, except by offering a new config to download if localhost has this as policy
- We can ask `cf-agent` to run its *current policy* immediately with `cf-runagent`
- `cf-agent` can ask another host for a file if it chooses to cooperate





# Public-private keys

- All hosts run `cf-serverd`
  - No danger unless we grant silly access
  - Require SSH-like keys for authentication
- Run `cf-key` on each host
- **Security model:**
  - Supervised key exchange with "trustkey" directive
  - Access control based on IP /key identity



# Example

```
Host# /usr/local/sbin/cf-key
```

```
Making a key pair for cfengine, please wait, this could take a minute
```

```
Writing private key file to /var/cfengine/ppkeys/localhost.priv
```

```
Writing public key file to /var/cfengine/ppkeys/localhost.pub
```



# Setting up cf-serverd

- Server bundles
- Develop access rules
  - For the right to connect
  - For file download access or cf-runagent access
- Must exchange public keys before access can be granted



```
#####  
# Server config  
#####
```

## body server control

```
{  
allowconnects          => { "127.0.0.1" , "::1",  
"10\0\1\.*" };  
allowallconnects       => { "\Q127.0.0.1" , "::1",  
"10\0\1\.*" };  
trustkeysfrom          => { "127\0\0\1" , "::1",  
                           "10\0\1\.*" };  
}  
#####
```

## bundle server access\_rules( )

```
{  
access:  
  
    "/home/mark/LapTop"  
    admit    => { "\Q127.0.0.1" };  
}
```



# Checklist

- Anti-DOS attack rules: allow.\*connects
- Switch on "trust" just long enough to exchange public keys with clients
  - This is a "nuisance", i.e. it's security
  - Like SSH, but not terminal-interactive



# Trouble?

- Unknown service?
  - Register cfengine in /etc/services, port 5308/tcp
- Access denied?
  - Forget domain name declarations?
  - Forget trustkey?
  - Forget access admit/grant rule?
  - Run with -v -d2 to see what's going on....
- Remove server-side trust, just in case
  - Don't want unauthorized hosts to exchange keys & possibly trick us



# Connection procedure

- Client attempts to connect to port 5308
- Server examines IP address of connection and applies rules from
  - allowconnects
  - allowallconnects
  - denyconnects
- If host is allowed to connect, read max 2048 bytes to look for valid hail



# Authentication procedure

- Client sends ID and public key to server
- Server checks whether public key is known
  - If known, host and user are confirmed, go to access control
- If unknown, use **trustkeysfrom** rules to check whether we should accept the client's asserted identity
  - If not in **trustkeysfrom** list, break connection
  - If willing to trust, go to further checks





# Verify ID

- If `skipverify` is set, ignore checks/NAT
- Else check asserted identity by reverse DNS lookup
  - If fails break off
- Check user ID is in `allowusers`
  - If fails break off
- Go to file access control



# Access control rules

- Classes tell us on which host the rule is enforced, NOT to which hosts access is granted
- *Syntax:* /directory names addresses
- admit then deny
- Mapping of privilege **maproot**
- Forced encryption **ifencrypted**
- Symbolic links are not honoured!



# Most specific rule first!

- Access control is evaluated by the rules:
  - First admit rule that matches wins
  - All other admit rules are ignored
  - No admit rule means you're denied!
  - Then look at deny rules (“overrides”)
  - First deny rule that matches wins
  - All other deny rules are ignored
  - No deny rule means you're admitted



# IPv6 issues (old)

- Cfengine tries to use the first address from DNS that responds.
  - If IPv6 is configured (but wrongly) it will fail
- AAAA records in DNS are still messed up
  - .ip6.int (this is deprecated but it works)
  - .ip6.arpa (IETF decided this but it doesn't work yet)



# Scalability considerations

- Avoid client-server communication (local work)
- Or use `cf-execd` splaytimes, or...

```
bundle agent example
```

```
{  
classes:  
  
  "my_result" expression =>  
    splayclass("${sys.host}${sys.ipv4}", "daily");  
  
reports:  
  
  my_result::  
    "Load balanced class activated";  
  
}
```

# Using fail-over servers in copy

```
bundle agent copy
{
  files:

    "/tmp/test1-copy"
      copy_from => cp("/tmp/testfile1", "host");
    "/tmp/test2-copy"
      copy_from => cp("/tmp/testfile2", "host");

}

body copy_from cp(from, server)
{
  source => "$(from)";
  servers => { "$(server)", "failover.example.org" };
}
```

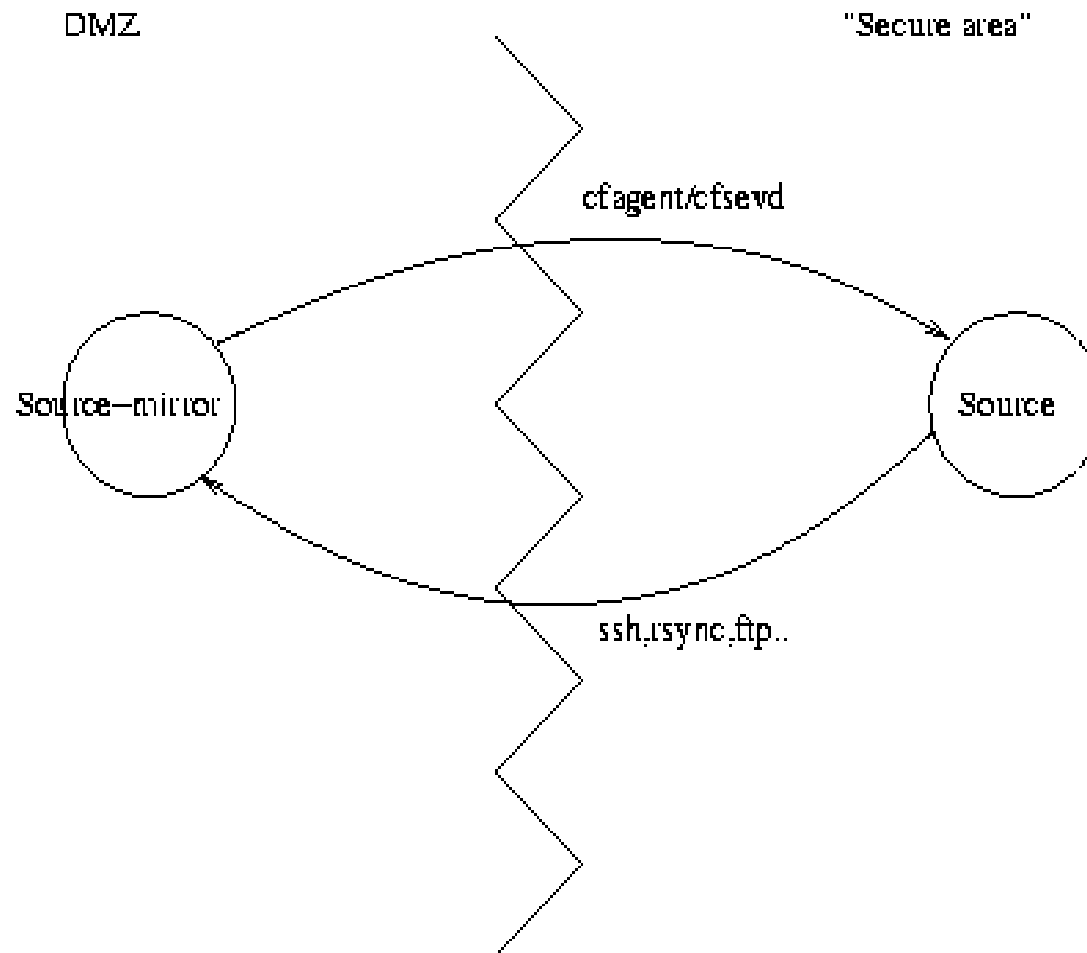
# Using round-robin servers in copy

```
bundle agent copy
{
  classes:
    "flip_a_coin" expression => isgreaterthan(randomint(1,100),50);
  files:
    "/tmp/test1-copy"
      copy_from => flip_cp("/tmp/testfile1",
                           "host1", "host2");
}

body copy_from flip_cp(from,server1, server2)
{
  source => "$(from)";
  flip_a_coin:: servers => { "$(server1)", "$(server2)" };
  !flip_a_coin:: servers => { "$(server2)", "$(server1)" };
}
```

# Dealing with a firewall

- Check out the **FAQ** section of online docs





# Dealing with a firewall

- Check out the FAQ section of online docs
- Several issues
  - NAT problems
    - skipverify,skipidentify
  - Updating config from secure -> DMZ
    - Opening ports for pull
- Need to think models
  - Compare likelihood of bugs in cfengine with that of bugs in firewall...



# Policy servers

- These are the distribution servers
  - May be several for fail-over, scaling
  - Each machine gets a copy of the entire policy and decides which parts apply
- Each machine caches its policy, knows its part in the music, so robust to communication breakdowns
  - Orchestration
- Once machine chosen to be a consistent source



# Extensible interfaces

- So far we used the COPBL = Community Open Promise Body Library
- You can roll your own, and/or contribute back
- Standardization of these interfaces is an industry imperative
- It is Knowledge Management



# Promise bodies

- Let's go through the examples again and see what's going on underneath



# Create files / dirs

files:

```
"/home/mark/tmp/test_plain"  
  perms => mog("644", "root", "wheel"),  
  create => "true";
```

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



# Disabling and rotating

**files:**

```
" /home/mark/tmp/test_create"
```

```
  rename => disable;
```

```
" /home/mark/tmp/rotateme"
```

```
  rename => rotate("4");
```

```
body rename disable
```

```
{
```

```
  disable => "true";
```

```
  disable_suffix => "_blownaway";
```

```
}
```

```
body rename rotate(level)
```

```
{
```

```
  rotate => "$(level)";
```

```
}
```



# Hashing

```
"/home/mark/tmp" -> "me"  
  changes          => detect_all_change,  
  depth_search     => recurse("inf"),  
  action           => background;
```

```
"/home/mark/LapTop/words" -> "you"  
  changes          => detect_all_change,  
  depth_search     => recurse("inf");
```

```
body changes detect_all_change  
{  
  hash          => "md5";  
  report_changes => "content";  
  update        => "yes";  
}
```



# Check filesystems

storage:

```
"/usr" volume => mycheck("10%");
```

```
body volume mycheck(free)
{
check_foreign  => "false";
freespace      => "$(free)";
sensible_size  => "10000";
                # or "10k" (but not "10K")
sensible_count => "2";
}
```





# Mount filesystems

storage:

`"/home/mark/server_home"`

`mount => nfs("myserver", "/home/mark");`

```
body mount nfs(server, source)
{
mount_type => "nfs";
mount_source => "${source}";
mount_server => "${server}";
#mount_options => { "rw" };
edit_fstab => "true";
}
```



# Software/patch installation

packages:

"apache2"

```
package_policy => "add",  
package_method => yum;
```

```
body package_method yum  
  
{  
  package_changes => "bulk";  
  package_list_command => "/usr/bin/yum list installed";  
  package_list_name_regex => "([^.]+).*";  
  package_list_version_regex => "[^\\s]\\s+([/^\\s]+).*";  
  package_list_arch_regex => "[^.]+\\.([/^\\s]+).*";  
  ...  
}
```



# Search and compress files

files:

```
"/home/mark/tmp/testcopy"
```

```
file_select => pdf_files,  
transformer => "/usr/bin/gzip $(this.promiser)",  
depth_search => recurse("inf");
```

```
body file_select pdf_files  
{  
  leaf_name => { ".*\.pdf" , ".*\.fdf" };  
  file_result => "leaf_name";  
}
```



# Search and compress files - 2

files:

```
"/home/mark/tmp/testcopy/*.pdf"
```

transformer =>

```
"/usr/bin/gzip $(this.promiser)";
```



Hands on:

`unit_locate_files_and_compress.cf`



# Back-references in PCRE

- We can extract parts of the matching string with  
( )

```
files:
```

```
"/home/.* /\.ssh/authorized_keys"
```

```
edit_line => add_key( "$ (somekey) " );
```

```
"/home/(.*)/keyfile"
```

```
create      => "true",
```

```
edit_line => AppendIfNoLine( "key_$(match.1)" );
```



# Back-references \$(match.n)

```
bundle edit_line myedit(parameter)
{
  replace_patterns:

  # replace shell comments with C comments

  "#(.*)"          # () make regex backrefs

  replace_with => C_comment,
  select_region => MySection("New section");
}
```

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



# Selecting regions

```
body select_region MySection(x)

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

```
[Section Name]
```

```
# comments
Stuff
```

```
[Next Section]
```

```
more stuff
```





# Selecting HTML regions

```
body select_region MySection(x)

{
select_start => "<$(x)>";
select_end   => "</$(x)>";
}
```

```
<ul>
<li> Item 1
<li> Item 2
</ul>

<ol>
```

Hands on:

unit\_edit\_comment\_lines.cf  
unit\_edit\_sectioned\_file.cf



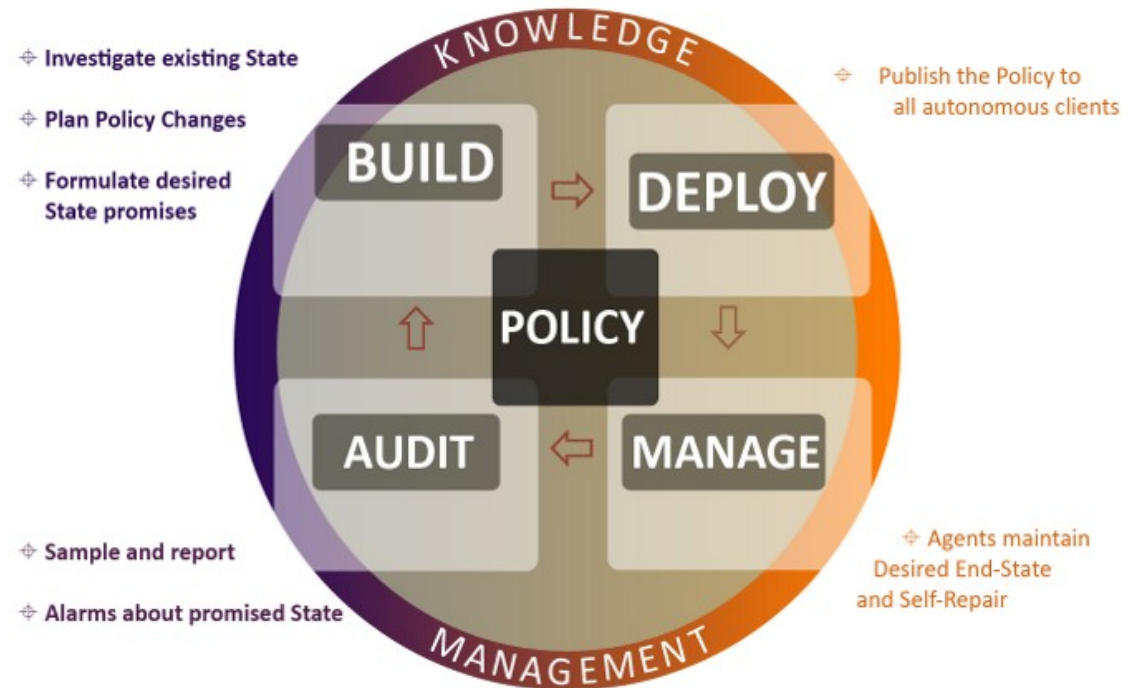
# Going forward with Cfengine

- Road map for Cfengine

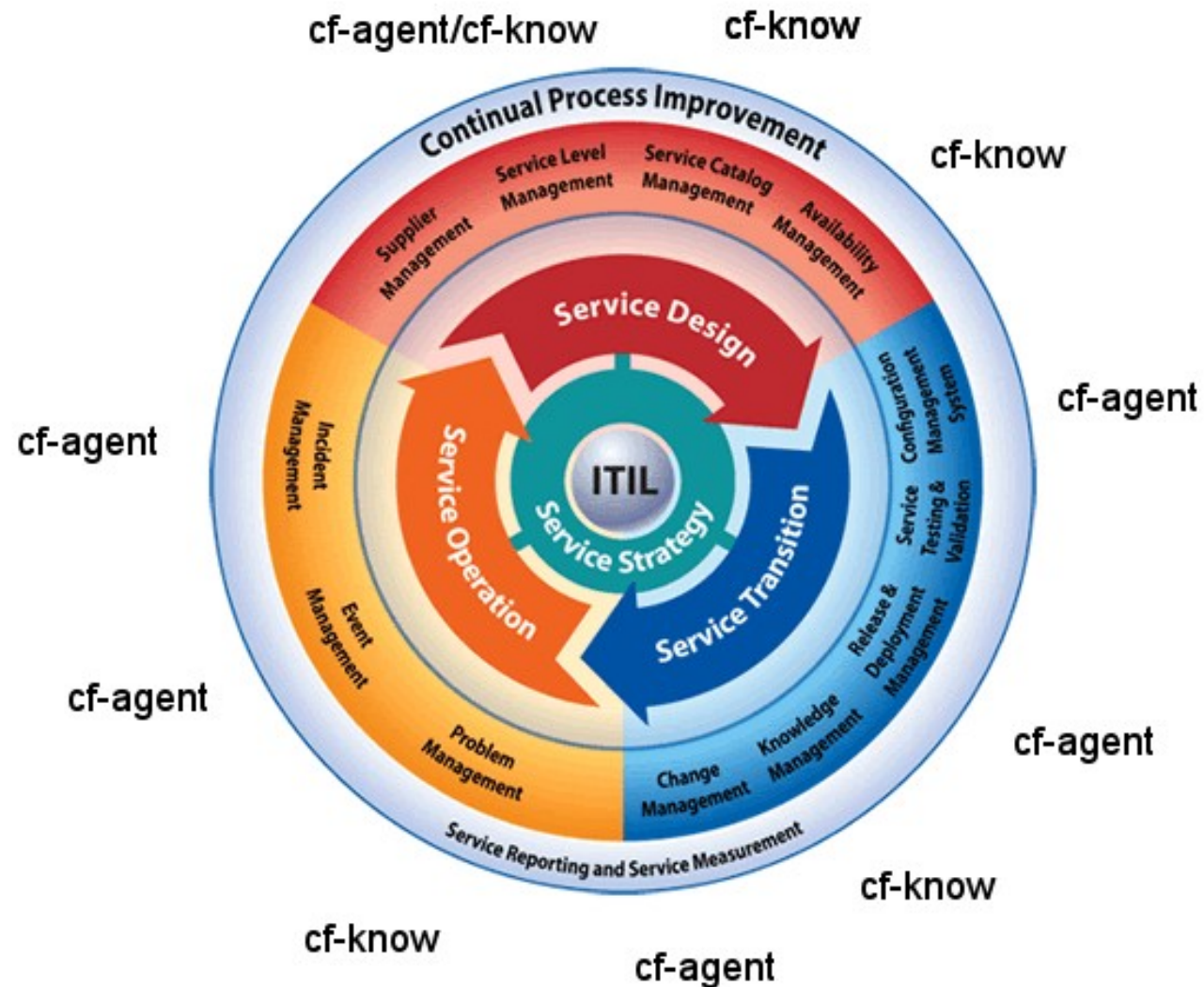
- Nova
- Constellation
- Galaxy

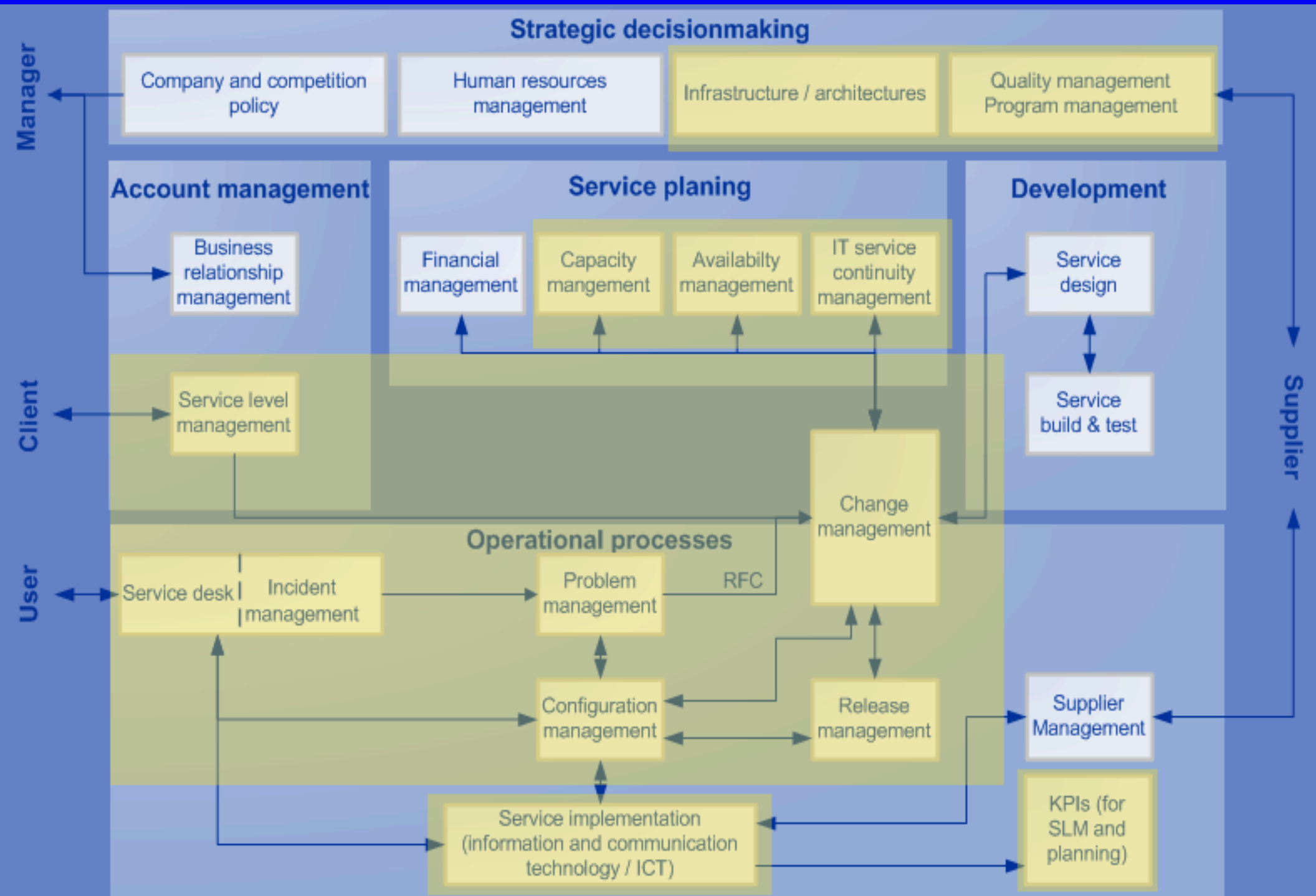
- Designed for

- Scaling
- Integrated knowledge
- Business integration – compliance etc



# Service Management





# Now does Nova work?

- Adds simpler get started procedure (turnkey)
  - Based on a default model “out of the box”
- Adds automatic auto-analysis and knowledge map generation
- Extended reporting
- Windows support. VM support, DB support, multi-node orchestration, GUI, LDAP integration, Zenoss integration....
- Designed to be simple to upgrade both to new Nova and Constellation



## Company Policies

- Configuration Management
- PCI-DSS

[webserver\\_process.cf](#) [edit](#)  
[webserver\\_pack.cf](#) [edit](#)  
[webserver\\_IP.cf](#) [edit](#)  
[hardening\\_all.cf](#) [edit](#)

## Online help - Copernicus

Search

[Examples](#)  
[Reference manual](#)  
[Commands](#)  
[Best-practices](#)  
[Special Topics](#)

Click to  
view all

Click to  
view all

[New](#) | [Dry run](#) | [Save](#) | [Publish](#) | [Cancel](#) | [Options](#)

webserver\_process.cf

hardening\_all.cf

webserver\_pack.cf

[Maximize](#)

```
1 # Literals
2 1234
3 0.0e101
4 .123
5 0b01010011100
6 0o01234567
7 0x0987654321abcdef
8 # Error Literals
9 .0b000
10 0.0e
11 0e
12
13 # String Literals
14 'For\''
15 "God\''"
16 ""so loved
17 the world""
18 ''that he gave
19 his only begotten\'' ''
20 'that whosoever believeth \
21 in him'
```

mission status "may be seen from"

[arcitect view](#)  
[manager view](#)  
[operator view](#)

About:

"Overview of IT operations" (Text)

Output Dry Run (webserver\_process.cf):

&gt;&gt;

Syntax checks(webserver\_process.cf):

&gt;&gt;

Cfengine Mission Portal - knowledge bank - Opera

Menu Cfengine Mission ...

Secure cfengine.com/demo/knowledge.php

Search with Google

CFENGINE MISSION PORTAL knowledge bank

SUMMARY STATUS PLANNING KNOWLEDGE

Knowledge bank

Copernicus local cluster view

```
graph LR; MS((mission status)) --- MV((manager view)); MS --- OV((operator view)); MS --- SV((security view)); MS --- AV((architect view)); MS --- SO((system outputs)); MV --- BV(businessvalue); MV --- SL(server lifecycle); MV --- MI(management issues); MV --- CR(compliance report); OV --- LSR(lastseen report); OV --- SI(software installed.); OV --- F(faults); OV --- SR(setuid report); OV --- PNK(promises not kept ..); OV --- PR(promise report); OV --- C(contexts); SV --- PI(patches installed ..); SV --- PNK; AV --- C; SO --- C
```

References to 'mission status' in the context of 'system\_knowledge'

Also mentioned in contexts of: [mission\\_portal\\_views](#) any  
mission\_status:: **network host status**: (URL)  
mission\_status:: **portal**: (URL)  
mission\_status:: **weakest hosts**: (URL)  
mission\_status.system\_outputs:: *"Messages sent by cfengine to the operators as a matter of policy."* (description)  
mission\_status.host\_portal:: *"Portal to access mission status, monitored host data"* (description)  
mission\_status.system\_reports:: *"Reports from around the system"* (description)  
mission\_status.system\_policy:: *"Local policy description with comments and dependencies"* (description)  
system\_knowledge.mission\_status:: *"Overview of IT operations"* (description)

Insight, leads and perspectives:

mission status "may be seen from"  
[security view](#)  
[operator view](#)  
[manager view](#)  
[architect view](#)

View (100%)





## CFENGINE MISSION PORTAL host eternity.iu.hio.no

Status : host

SUMMARY

STATUS

PLANNING

KNOWLEDGE

Select host:

eternity.iu.hio.no

Select host

## Host Details (discovered)

Alias: eternity.iu.hio.no  
OS class: linux\_x86\_64  
Release: 2.6.22.19-0.4-default  
Flavour: SuSE\_10  
Last IP-address: 128.39.89.233  
Last data: Sun Mar 6 10:35:34 2011  
ID:  
SHA=8863e92018e2651280be79d88b2cd1222386338eac8ac6298e1b136b4b4c2b19

## Status (measured)

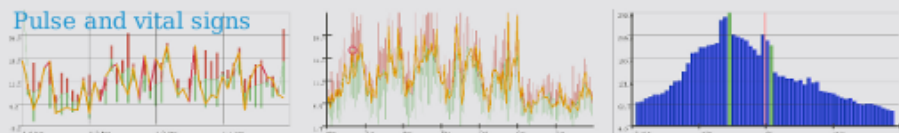
Average Load: 23.28%  
Average Free  
Disk: 19.04%  
Average network  
speed: 0.01  $\Delta$  51.00 bytes/s

## Analysis

Week	Day	Hour	Perf	Chng	Coms	Anom
98.5	95.9	89.4	100.0	100.0	80.0	98.7



## Pulse and vital signs



## Generate report

Simple search string:

Bundle profile

commit

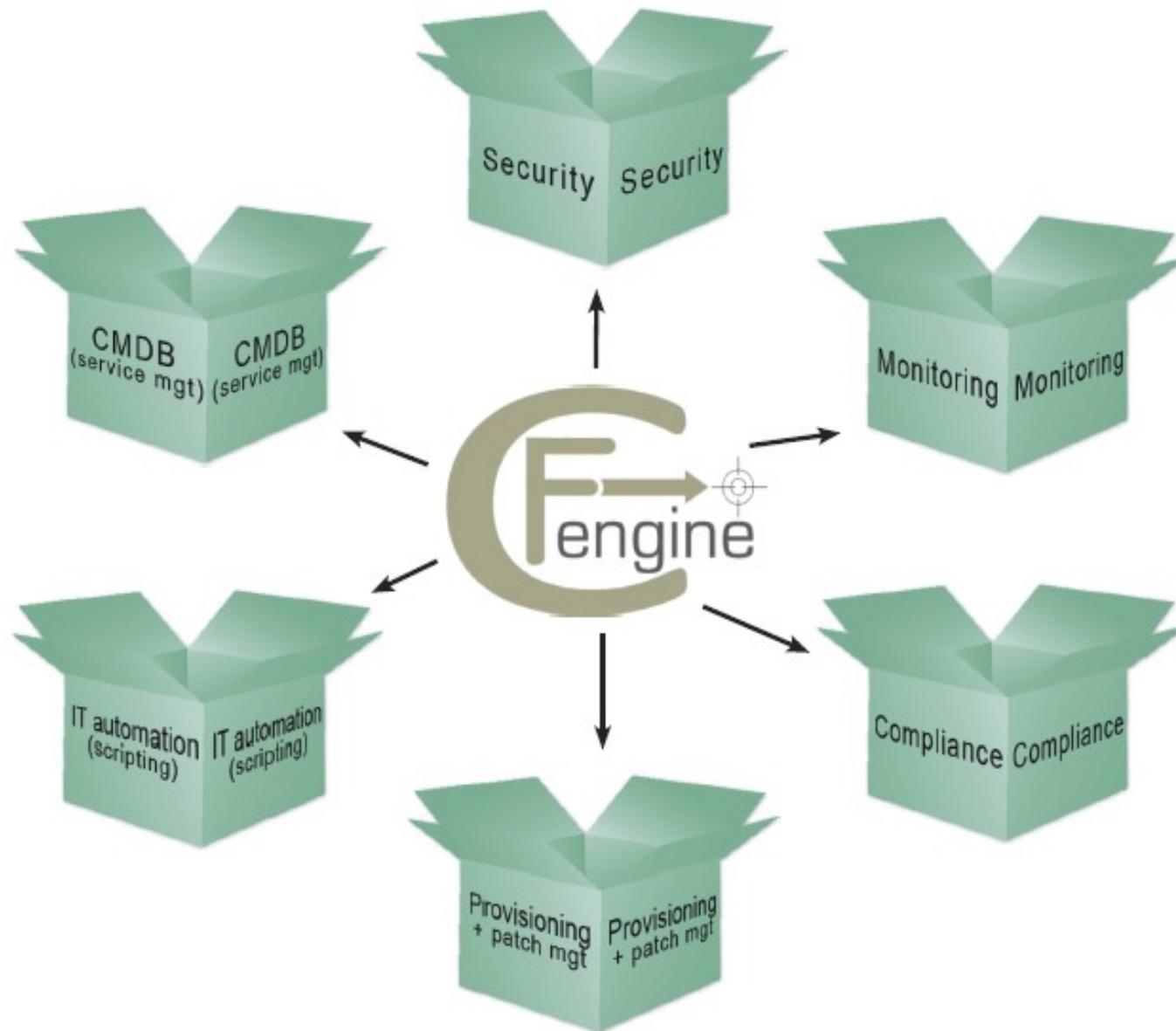
## Monitored jobs

Total number under surveillance:5

On this host:

View (100%)

# Finally: how do *you* view cfengine?



# Generic promise syntax

```
what_promise:
```

```
when_where::
```

```
    "what_object" -> { "stakeholder" },
```

```
        how_attribute_1 => how_value1,
```

```
        how_attribute_2 => how_value2;
```

**Thank you!**

Advance to Cfengine Nova

[contact@cfengine.com](mailto:contact@cfengine.com)

