SANE 2006 – CF Engine 1

15/05/06 11:03

De agenda op pagina 1 is voor de hele dag. Na “hoofdstuk” 6 volgt de lunch.

Hehe... Op de pagina van wie CFEngine gebruiken staat Snow.nl ook genoteerd! <3

CF Engine’s doel = om *configuration* te laten matchen aan *policy*.

CF Engine is made of *convergent* operations -> we know where we want to go and we know where we are. What CF Engine does should always bring you closer to your goal. If there are no steps to be undertaken, CFE shouldn’t do anything. Compare it to your immune system.

cfagent is the main part of CFE which executes many of the actions CFE should undertake.

cfexecd is used as the CFE scheduler and can be run:

- as a replacement for cron (windows)
- from cron
- aside to cron

cfenvd detects anomalies on your system, as compared to the policy set for the system or its original state. For that the daemons has learning capabilities.

cfservd is the file collector. The CFE server pulls files from its clients. You cannot push.

CFE bestaat al bijna veertien jaar! En er zijn maar drie security advisories geweest over deze software! Da’s erg goed...

As part of your policy you'll have an instruction to collect newer versions of the policy. This is the way you update your policies! So instead of telling the system to do something, you’ll ask it to adhere to its current policy. All of this with a keen eye on security.

The only way to hack into a client is by hacking into the CFE server. You should be very strict on securing your CFE server, since its an SPOF. You could also choose to not centralise your infrastructure.

There is no built-in function to download policies from multiple servers to compare them. You –could- build this yourself though.

<u>**Chapter two: installation**</u>

De Darwin Ports versie van CFE slaat z'n binaries op in **/opt/local/sbin**. De local files directory **/var/cfengine** bestaat vreemd genoeg nog niet. Als je met **port contents cfengine** kijkt, zie je dat het **/opt/local/var/cfengine** is. Jammer dat **/var/cfengine** hard-coded in CFE zit. Je moet dus een symlink aanleggen onder OS X.

I tried the example from the sheet "Test it!" and it works :) Output explained ->
field 1 = CFE identifier
field 2 = (blank) name of host running the CFagent
field 3 = summary of shell command run (first 15 characters). You can use a script with a descriptive name or a macro to make this more readable (see sheet "update.conf#2").
field 4 = output of shell command run

CFE, by default, will not allow you to run action more than once a minute. So if you run **cfagent –f ./cfagent.conf** twice within ten seconds, the second run will not provide output. This will prevent you from DoSing yourself. There are naturally also locks implemented to see that CFE is not running currently. Hence there's protection from looping.

CFE actually uses two policies: one to describe the system's policy and one to describe the process for updating the system's policy :) This prevents you from borking up your configuration completely. If they were in one file a mistake would prevent you from re-updating the config.

Anything with two colons (:) in **cfservd.conf** dictates that the following should only be run when a certain condition is true. See sheet "Policyhost:masterfiles".

The control section is used to set variables. The various action sequences defined in the control section define the actions to be undertaken.

You should not configure stuff by hand. CFE is only so smart... If you have a slight permutation of the line you want to enter in file A –already- in file A, then CFE will add the new line as well. Hence you could end up with two entries in cron. So don't edit files that are managed by CFE manually.

The various checks/conditions you can build into your **cfagent.conf** file are actually defined classes. You can get a list of currently defined classes by running **cfagent –v**. All of these are automatically defined by probes built into CFE.

< en we gaan door na de koffie break >

"Examples" sheet:

- The dot is a logical AND
- The pipe is a logical OR
- The class *PasswordClients* is a custom defined class which can be a list of hostnames in a text file or a database. We'll cover custom classes RSN.
- Let's say that the *PasswordClients* rule gets run again and the permissions have changed (yet the contents have not). In that case CFE will reset the permissions to how they should be. It will not re-copy the files, since the checksum is still the same. Copying a file by CTIME though –would- make things different.

Chapter 3: CFEngine rules!

Config files can be made as readable as you like, so they can be used as backup documentation for your systems. With some restrictions you can format them as you like.

Every host reads every config file and then picks out the parts that are relevant to it. So you could put all rules into one huge file, or you could split them across many files that get included for certain classes.

CFEv3 will hopefully remove the need for the ActionSequence variable. Or at least remove the ordering of actions from that variable.

The *force* class is something that can be used to force running certain rules when using cfagent with the –d flag.

< Mark besluit dingen aan te wijzen op het bord door er met een bordenwisser tegen aan te gooien. Totdat hij een stok krijgt aangereikt >

A class can be based on anything that can be measured from the working environment. The presence of a file for example. When defining a class of your own the minus (-) means to exclude the item followed by it.

QUESTION: what happens when a hostname coincides with a hard defined class name?
ANSWER: that's bad design on your side ;p

When defining a selector using parentheses is not required, but it does make things more readable. For now assume that AND gets priority above OR.

In CFE a *spool directory* contains files which are only named after valid users of the system. This is handy when you'd like to cleanup file for users that no longer exist.

Hard classes and variable names are NOT case sensitive. User-defined names are however. TRICKY!

QUESTION: How do you deal with a "standard build" situation?
ANSWER: Ideally you install a standard version of the OS and most of your applications. From there on install CFE and then let CFE perform the standard configuration. The SVN/Webserver case is an example of this.

The *LastSeenExpireAfter* variable can be used for host availability monitoring.

You could also choose to let CFE install your packages for you (standard build), but then you will need to create your own modules for that.

CFE Execution Sequence

cf.preconf is run to verify that basic stuff like a root account and some requisite groups are present on the system.

If you are going to use imported/included config files: use the main config file only to refer to these files. Else you're going to get things mixed up.

You can define new classes interactively when taking a certain action. For example, when installing a new CFE binary you can set New-CFE which will then be used at a later point in time to restart the CFE daemons.

Questions

Yes, you can use subnet notation when defining network addresses: 192.168.0.1/24.

Currently CFE is already parallelizable, but I don't think it'll become –more- parallel (but I hope so) when we remove the ordering from the ActionSequence.

Yes, you can call one action multiple times within one config file. This may seem smart at times, but you could also try and rethink your setup.

No, there is no default policy for cleaning up its backup repository. But you can include this in your own policy.

The website contains no example configs since everyone seems to be too affraid to show what they're running. Christian Pierce has written a style guide though and the Wiki contains some contributions. There are also a bunch of companies setting up consulting services revolving around CFE.