CFEngine at AGLT2

Ben Meekhof
ATLAS Great Lakes Tier2
University of Michigan
HEPiX Fall 2014, Oct 13-17

Outline

- Basic CFEngine setup
- History at AGLT2
- Current setup at AGLT2
  - Updating and testing
  - Policy hosts
  - Version Control
  - Monitoring
- Examples
  - Iptables
  - DNS management
  - Other Managed Configs
- Other Notes

Basic CFEngine Structure

- Similar in rough concept to other configuration managers such as puppet
  - Client pulls configuration definition from server
  - Interpreter parses policy code, edits file templates, applies config
  - Some status is produced – dashboard update, email or other capture of output if there are errors or reports

- In CFEngine case, policy is copied and kept locally and continues to be applied even when client cannot reach server

- The most basic "unit" of configuration in CFEngine is called a Promise.
  - Such as…promise to set file permissions
  - Promises are contained in Bundles
- Promises have a body with details which depend on promise type
  - Such as…what are the file permissions? Which attributes are we wanting our promise to effect?

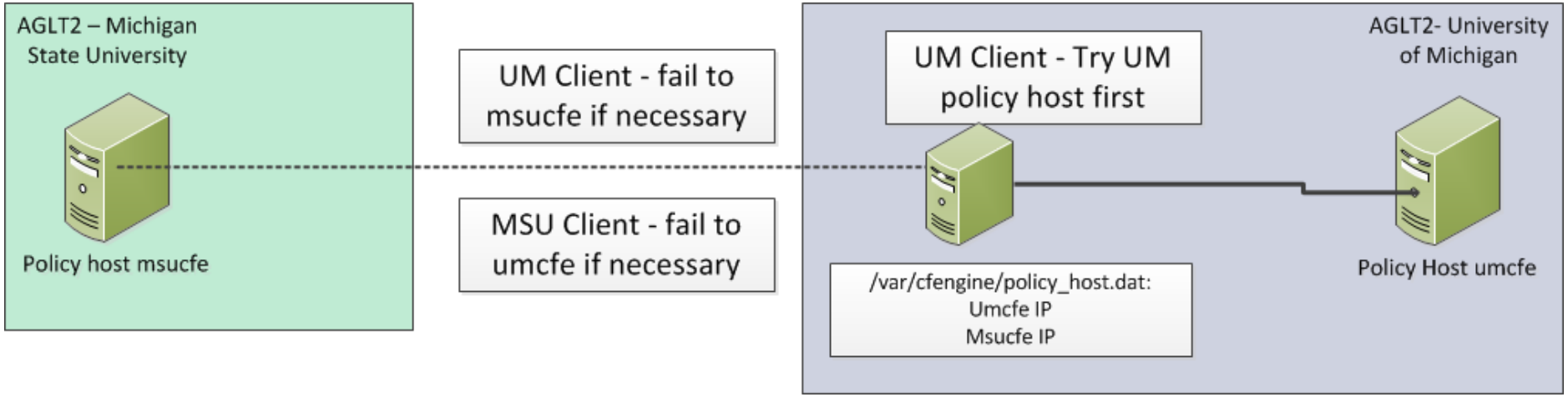
History at AGLT2

- In ~2010 we implemented CFEngine version 2 as our configuration manager. Prior to that we had no or rudimentary config management.
- At that time version 3 was in early releases so we chose to migrate later
  - Also much more familiar with 2 – faster implementation

- Perhaps a year after that (2011) we started implementing CFEngine v3 to manage cluster compute nodes built with rocks
- After some experience, we pushed harder to drop CFE2 nodes and take over needed configuration in CFE3.
  - Ran CFE2 and 3 in parallel for a while
- At this point, today, we don't manage any systems under CFE2.

- Perhaps opposite the correct planning order, this year we've formalized an SVN based workflow for testing and pushing to production.
  - **Easily create new testing/prototyping environments** as SVN branches

Setup at AGLT2



Clients determine which server(s) and what policy to run from /var/cfengine/policy_hosts.dat and policy_path.dat

The policy_path.dat file points at our default production policy stored at /var/cfengine/policy/T2 on the policy hosts (same contents).

Can either modify the .dat, or define a new policy path for temporary testing:

```
cf-agent -DPolicyPath_testing
```

Setup at AGLT2 - details

```
"servercount" int => readstringarray("ps", "$(sys.workdir)/policy_server.dat", "#.*", "[\n]",
2, 100),
comment => "Number of policy servers read from file. Server names are populated into array
'ps' ";

"server" slist => getvalues("ps");

"policydat" comment => "Path to my config on policy server, read from file",
string => readfile("$(sys.workdir)/policy_path.dat",80);

"policyclass" comment => "Class to set policy source on server (over-rides other
definitions)",
slist => classesmatching("PolicyPath_.*");

classes:
"set_policydat" comment => "Have set policy source path from file",
expression => regcmp("/.*",$(policydat));

"set_policyclass" comment => "Check if PolicySource_XXXX was set to indicate alternate policy
path", expression => classmatch("PolicyPath_.*");
}

bundle agent update {
vars:
set_policyclass::
 "csplit" slist => splitstring(nth("policy.policyclass",0),"_",2);

 "policypath" string => concat("$(policy.defpath)/", nth("csplit",1));

set_policydat.!set_policyclass::
  "policypath" string => "$(policy.policydat)";

!set_policydat.!set_policyclass::
  "policypath" string => "$(policy.defpolicy)";
```

ATLAS Great Lakes Tier 2
University of Michigan AGLT2 Michigan State University

Workflow

- Our policy servers are configured to export policy setups from /var/cfengine/policy

- Current production policy is always updated in SVN before pushing to the policy server
  - We're somewhat informal, there's no real validation process except trusting all (5) of us to update working copy before push to server
  - Possible we'll yet consider going to a workflow where all changes have to go through svn/git in combination with a post-commit script to automatically sync policy servers

- When syncing policy to server, the sync script updates a variable to match the current SVN version
  - We always know if dealing with the most current SVN and if it was synced from a modified working dir (svnversion appends "M").

Monitoring

Service search

208 rows omdadmin (admin) 08:59

1 60s | Edit View | Availability | ...

UM,atback1

State	Service	Status detail	Icons	Age	Checked	Perf-O-Meter
OK	CFEngine Installed	OK - CFEngine is installed		2014-08-21 16:39:10	58 sec	
WARN	CFEngine Output	WARN - Last run output:		19 hrs	58 sec	
OK	CFEngine PolicyPath	OK - Policy path is correct: /var/cfengine/policy/T2		2014-08-21 16:39:10	58 sec	
OK	CFEngine Service	OK - cfengine3 service is enabled		2014-08-21 16:39:10	58 sec	

UM,cache

State	Service	Status detail	Icons	Age	Checked	Perf-O-Meter
OK	CFEngine Installed	OK - CFEngine is installed		2014-08-21 16:39:11	23 sec	
OK	CFEngine Output	OK - Good - no output		16 min	23 sec	
OK	CFEngine PolicyPath	OK - Policy path is correct: /var/cfengine/policy/T2		2014-08-21 16:39:11	23 sec	
OK	CFEngine Service	OK - cfengine3 service is enabled		2014-08-21 16:39:11	23 sec	

UM,cache2

State	Service	Status detail	Icons	Age	Checked	Perf-O-Meter
OK	CFEngine Installed	OK - CFEngine is installed		2014-08-21 16:38:56	25 sec	
OK	CFEngine Output	OK - Good - no output		5 hrs	25 sec	
OK	CFEngine PolicyPath	OK - Policy path is correct: /var/cfengine/policy/T2		2014-08-21 16:38:56	25 sec	
OK	CFEngine Service	OK - cfengine3 service is enabled		2014-08-21 16:38:56	25 sec	

UM,cache3

Monitoring

Site alias	UM
Hostname	atback1
Service description	CFEngine Output
Service icons	
Service state	**WARN**
Servicegroups the service is member of	
Service service level	
Service contact groups	all
Service contacts	
Output of check plugin	WARN - Last run output:
Long output of check plugin (multiline)	2014-10-14T09:43:30-0400 error: Can't rename '/etc/ssh/sshd_config.cf-after-edit' to '/etc/ssh/sshd_config' - so promised edits could not be moved into place. (rename: Operation not permitted) 2014-10-14T09:43:30-0400 error: /sshd/files/'/etc/ssh/sshd_config': Unable to save file '/etc/ssh/sshd_config' after editing 2014-10-14T09:43:31-0400 error: /monit/files/'/etc/grid-security/monit.pem': Promised to monitor '/etc/grid-security/monit.pem' for changes, but file does not exist

- Clicking on the Warn link brings up the actual output relayed by the plugin
  - Someone marked sshd_config immutable to protect from management
- Writing check_mk plugins is well documented, I just followed the example and wrote a very simple plugin in one day.
  - Agent plugin is bash that does some simple service checks and timestamp comparisons to determine if previous cf-agent run had output
  - On check_mk server side, python script to parse agent plugin output and return status code
- http://mathias-kettner.com/checkmk_devel_agentbased.html

Example - iptables

Cfengine policy file iptables.cf:

```
"netumich" slist => { "141.211.0.0/16", "141.213.0.0/16‟ };

CobblerServers::
 "tcp[80]" slist => { @(fw.netumich) };

CFE_PS::
  "tcp[5308]" slist => { @(fw.netaglt2), @(fw.netpublic) };

any::
  "tcp_ports" slist => getindices("tcp");
  "udp_ports" slist => getindices("udp");
```

Cfengine file template for /etc/sysconfig/iptables:

```
[%CFEngine BEGIN %]
-A INPUT -d -s $(iptables.tcp_ports) $(iptables.tcp[$(iptables.tcp_ports)]) -j ACCEPT
[%CFEngine END %]

[%CFEngine BEGIN %]
-A INPUT -d -s $(iptables.udp_ports) $(iptables.udp[$(iptables.udp_ports)]) -j ACCEPT
[%CFEngine END %]
```

Examples - DNS

```
UM::
"mastersite" string => "msu",
comment => "Site to slave zones from (key in 'zones' array)";

"soa" string => "dns.local",
comment => "Zone SOA used in zone definition files";

MSU::
"mastersite" string => "um";
"soa" string => "msuinfo.msulocal";

DNS_SERVERS::

"serial" string => execresult("/bin/date +%Y%m%d%H%M", "noshell");

"zones[um]" slist => {
        "local",
        "1.1.10.in-addr.arpa",
        "0.10.10.in-addr.arpa‟ };
```

Examples - DNS

```
files:
"/var/named/zones/$(zones[$(g.sitename)]).domain"
perms => mog("0640","root","named"),
classes => if_repaired("reload_named"),
copy_from =>
secure_cp("$(stash)/$(zones[$(g.sitename)]).domain","@(policy.server)");

"$(stage)/soa.include.domain.tmpl"
create => "true",
classes => if_repaired("reload_named"),
copy_from => secure_cp("$(stash)/soa.include.domain.tmpl","@(policy.server)");

"/var/named/zones/soa.include.domain"
create => "true",
perms => mog("0640","root","named"),
ifvarclass => "reload_named",
edit_template => "$(stage)/soa.include.domain.tmpl";
```

soa.include.domain.tmpl:

```
$TTL 3D
@ IN SOA $(named.soa). root.$(named.soa). ( $(named.serial)
```

Blah, blah, NS, MX, etc

Other Managed Services

- Users/groups – we wrote a small shell script to merge a master copy of passwd/group with local machine versions
  - CFE 3.6 has a “user” datatype which we haven’t explored
- Yum repositories – cfengine policy sets up repos appropriate to system type, populates baseurl with local and remote mirrors, and sets per-repo excludes
- FusionInventory client installation (system inventory reporter).
- There really is no service we don’t eventually end up managing in CFEngine.
  - Our build system, Cobbler, does a very minimal installation before handing it off to CFEngine on first boot. Don’t want 2 places to manage.

Other Notes

- Earlier CFEngine versions (bf 3.4?) wouldn't iterate associative arrays correctly
  - Made the iptables example given impossible to work
  - Has been fixed definitely in 3.5 (maybe 3.4?)

- In general CFEngine data containers can be non-intuitive to deal with
  - Example: Policy => "free" allows appending to a list. But only if it's already defined. But don't define it as empty or "cf_null" because you'll be surprised how literally those values are taken.

- Version 3.6 introduces a native JSON variable type. This should help the situation considerably and open up many integration paths.

- **Range** is one example of an ENC that can be tied to CFEngine http://syslog.me/2013/11/18/external-node-classification-the-cfengine-way/

- **Linkedin's Redis** tool was developed to provide visibility into system state populated by CFEngine. See https://github.com/linkedin/sysops-api

Questions?

Ask anytime – bmeekhof@umich.edu