Augeas

A decade of configuration surgery

David Lutterkort
lutter@puppet.com
What's the problem?
Disable PermitRootLogin in sshd_config

$ sed -r 's/PermitRootLogin yes/PermitRootLogin no/'}
Disable PermitRootLogin in sshd_config

```bash
$ sed -r 's/PermitRootLogin  yes/PermitRootLogin  no/
$ sed -r 's/(PermitRootLogin[ \t]+)yes/( PermitRootLogin[ \t]+)no/
```
Disable PermitRootLogin in sshd_config

$ sed -r 's/PermitRootLogin \s+yes/PermitRootLogin \s+no/'

$ sed -r 's/(PermitRootLogin[ \t]+)yes/(\1)no/'

$ sed -r 's/(PermitRootLogin[ \t]+)[a-z]+/(\1)no/'
Disable PermitRootLogin in sshd_config

$ sed -r 's/PermitRootLogin  yes/PermitRootLogin  no/'

$ sed -r 's/(PermitRootLogin[ \t]+)yes/(\1no/'

$ sed -r 's/(PermitRootLogin[ \t]+)[a-z]+/(\1no/'

$ grep PermitRootLogin /etc/ssh/sshd_config

#PermitRootLogin  yes
The problem:
Edit configuration files programmatically
The real problem:
Large number of config file formats
Reading files is easy.
Modifying them is hard.
Whole-file management not always feasible
How does Augeas work?
Handle config files in place and in their native format
Use the same data structure for all files
Preserve 'unimportant' detail and minimize changes
How do you use Augeas?
C library with lots of language bindings
(Ruby, Python, Go, Rust, Lua, Node, Haskell, OCaml, …)
Included in top-shelf config mgmt systems
(Puppet type, augeasproviders, Salt, Mgmt, …)
Getting started: augtool

$ augtool
augtool> help

Admin commands:
  context - change how relative paths are interpreted
  load    - (re)load files under /files
  save    - save all pending changes
  ...

Informational commands:
  errors   - show all errors encountered in processing files
  ...

Getting started: augmatch

(new in 1.10.1)

$ docker pull lutter/augmatch
$ docker run -ti lutter/augmatch
/ # augmatch --help

Usage: augmatch [OPTIONS] FILE
Print the contents of a file as parsed by augeas.

Options:

- `l`, `--lens LENS` use LENS to transform the file
- `m`, `--match EXPR` start printing where nodes match EXPR
- `e`, `--exact` print only exact matches

...
$ cat /etc/exports
/local  207.46.0.0/16(rw,sync)
/home  207.46.0.0/16(rw,root_squash,sync)  192.168.50.2/32(rw,root_squash,sync)
/tmp   207.46.0.0/16(rw,root_squash,sync)
/pub   *(ro,insecure,all_squash)
$ augmatch /etc/exports

dir[1] = /local
dir[1]/client = 207.46.0.0/16
dir[1]/client/option[1] = rw
dir[1]/client/option[2] = sync
dir[2] = /home
dir[2]/client[1] = 207.46.0.0/16
dir[2]/client[1]/option[1] = rw
dir[2]/client[1]/option[3] = sync
...
Getting started: augmatch

$ cat /etc/exports
/local  207.46.0.0/16(rw,sync)
/home  207.46.0.0/16(rw,root_squash,sync)  192.168.50.2/32(rw,root_squash,sync)
/tmp   207.46.0.0/16(rw,root_squash,sync)
/pub   *(ro,insecure,all_squash)
$ cat /etc/exports
/local 207.46.0.0/16(rw,sync)
/home 207.46.0.0/16(rw,root_squash,sync) 192.168.50.2/32(rw,root_squash,sync)
/tmp 207.46.0.0/16(rw,root_squash,sync)
/pub  *(ro,insecure,all_squash)

# List all clients to which we export a directory
$ augmatch --only-value --exact --match dir/client  /etc/exports
207.46.0.0/16
207.46.0.0/16
192.168.50.2/32
207.46.0.0/16
*
$ cat /etc/exports
/local 207.46.0.0/16(rw,sync)
/home 207.46.0.0/16(rw,root_squash,sync)
/tmp 207.46.0.0/16(rw,root_squash,sync)
/pub *(ro,insecure,all_squash)

192.168.50.2/32(rw,root_squash,sync)
Getting started: augmatch

$ cat /etc/exports
/local 207.46.0.0/16(rw,sync)
/home 207.46.0.0/16(rw,root_squash,sync) 192.168.50.2/32(rw,root_squash,sync)
/tmp 207.46.0.0/16(rw,root_squash,sync)
/pub *(ro,insecure,all_squash)

# List all clients to which we export the /home directory
$ augmatch -eom 'dir["/home"]/client ' /etc/exports
207.46.0.0/16
192.168.50.2/32
# Find all directories that are exported to at least one client without having the 'root_squash' option set

$ cat /etc/exports
/local  207.46.0.0/16(rw,sync)
/home  207.46.0.0/16(rw,root_squash,sync)  192.168.50.2/32(rw,sync)
/tmp   207.46.0.0/16(rw,root_squash,sync)
/pub   *(ro,insecure,all_squash)
# Find all directories that are exported to at least one client without having # the 'root_squash' option set

$ cat /etc/exports
/local  207.46.0.0/16(rw,sync)
/home  207.46.0.0/16(rw,root_squash,sync)  192.168.50.2/32(rw,sync)
/tmp   207.46.0.0/16(rw,root_squash,sync)
/pub   *(ro,insecure,all_squash)

$ augmatch -eom 'dir[client[not(option  = "root_squash")]]' /etc/exports
/local
/home
/pub
How can I get involved?
Need to reorganize and expand docs
(Want to move to gitbook)
Other ideas

- Many language bindings could use some love
- Write moar lenses
- Make Augeas work on Windows
A quick tour

This tutorial gives a brief overview of using Augeas, and augtool in particular. It's highly recommended that you follow along on your own; to do so, first download and install Augeas. Then, create a sandbox so that you can safely modify files without affecting your system. The commands below create the sandbox from /etc on your system, but you might want to copy the files from tests/root in the source tarball — all the examples below were run against those files.

```bash
export AUGEAS_ROOT=/tmp/augeas-sandbox
mkdir $AUGEAS_ROOT
sudo chown -R $(id -nu):$(id -ng) $AUGEAS_ROOT
augtool -b
```

The four set commands create four nodes underneath /files/etc/hosts/01, and assign them the values passed as the second argument. The data parsed from a file with full path FILE in the filesystem is stored in the tree underneath /files/FILE; with that the set commands manipulate an entry in the file /etc/hosts. A second hierarchy underneath /augeas/files/FILE contains metadata about the file, such as an indication of any errors encountered when the file was read.

The tree for /etc/hosts puts each host entry into its own subtree, numbered sequentially from 1, so that the first host entry appears under /files/etc/hosts/1, the second under /files/etc/hosts/2, etc. We use the label 01 and put the details of our new host entry underneath /files/etc/hosts/01. This works because reading in a /etc/hosts file will never use labels starting with 0, and tree labels are strings,their numeric value is irrelevant. The order in which host entries are written back to file is determined by the order in which they appear in the tree.

The set command creates non-existant nodes as needed. There is a second command, ins, to create new nodes in the tree that provides more control over where exactly a new node shows up, in particular where in the list of its siblings it appears. Because files are inherently sequential, the order of sibling nodes in the tree matters.

The notation elice[1] and elise[2] tells augtool to set the value for the first and second child of /files/etc/hosts/01. Both nodes are called /files/etc/hosts/01/elise[2]. All paths are relative to /files/etc/hosts/01, so the new host entry can be

https://augeas.net/