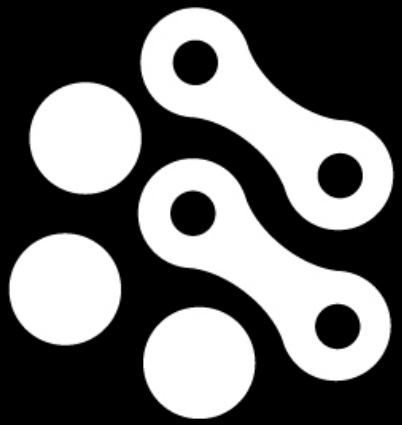
MODELLING INFRASTRUCTURE WITH INVENTORY DATA

HOW DOES ANSIBLE MODEL ITS INVENTORY AND HOW CAN WE MODEL OUR INFRASTRUCTURE?

$ WHOAMI

- {'name': 'Serge van Ginderachter'}
- {'url': 'http://ginsys.eu'}
- {'email': 'serge@ginsys.eu'}
- {'IM': 'serge@vanginderachter.be'}
- {'twitter': '@svg'}
- {'IRC': 'svg'}
- {'Github': 'sergevanginderachter'}



Ginsys

AUTOMATE ALL THINGS

$WORK
INFRASTRUCTURE 'CONSULTANT'

- Serge van Ginderachter
- Started in M$ shops in Small Business Environments (at Belgian scale: 5-150 employees)
- Followed the Jedi path and turned to the bright side of life after 2005
- Ansible since +/- 18 months
- No experience with other cfgmgmt tools
- First upstream commit

```
commit da92ce796b48ec80e3ead1cfe9bcbc71f5fce805
Author Serge van Ginderachter
Date   Wed Oct 10 19:38:30 2012 +0200

  fix missing --limit in docssite examples
```

CURRENT MAJOR PROJECT

- Flemish Government, one of the complex parts of Belgium
- http://www.milieuinfo.be: about environmental things
- Open Source island within the Flemish Government Enterprise
- upcoming stuff: "cloud" built on Ceph storage and Cloudstack
- Ansible since Summer 2012

ANSIBLE INVENTORY

How can this thing be implemented?

```
~/src/ansible/lib/ansible$ ls inventory/
dir.py
expand_hosts.py
group.py
host.py
ini.py
__init__.py
script.py
vars_plugins
```

ANSIBLE.INVENTORY

```
96          elif os.path.exists(host_list):
97              if os.path.isdir(host_list):
98                  # Ensure basedir is inside the directory
99                  self.host_list = os.path.join(self.host_list, "")
100                  self.parser = InventoryDirectory(filename=host_list)
101                  self.groups = self.parser.groups.values()
102              elif utils.is_executable(host_list):
103                  self.parser = InventoryScript(filename=host_list)
104                  self.groups = self.parser.groups.values()
105              else:
106                  self.parser = InventoryParser(filename=host_list)
107                  self.groups = self.parser.groups.values()
108
109              utils.plugins.vars_loader.add_directory(self.basedir(), with_
```

ANSIBLE.INVENTORY.DIR

```
58              if os.path.isdir(fullpath):
59                  parser = InventoryDirectory(filename=fullpath)
60              elif utils.is_executable(fullpath):
61                  parser = InventoryScript(filename=fullpath)
62              else:
63                  parser = InventoryParser(filename=fullpath)
64              self.parsers.append(parser)
```

~/SRC/ANSIBLE/EXAMPLES/HOSTS

```
# This is the default ansible 'hosts' file.

# Ungrouped hosts, specify before any group headers.
green.example.com
blue.example.com
192.168.100.1
192.168.100.10

# A collection of hosts belonging to the 'webservers' group
[webservers]
alpha.example.org
beta.example.org
192.168.1.100
192.168.1.110

# A collection of database servers in the 'dbservers' group
[dbservers]
```

DYNAMIC INVENTORY SCRIPTS

```
{
    "databases"   : {
        "hosts"   : [ "host1.example.com", "host2.example.com" ],
        "vars"    : {
            "a"   : true
        }
    },
    "webservers"  : [ "host2.example.com", "host3.example.com" ],
    "atlanta"     : {
        "hosts"   : [ "host1.example.com", "host4.example.com", "host5.exampl
        "vars"    : {
            "b"   : false
        },
        "children": [ "marietta", "5points" ],
    },

    "marietta"    : [ "host6.example.com" ],
```

THE ANSIBLE INVENTORY IS NOT A TREE!

```
google
google/gcalendar
google/gcalendar/backend
google/gcalendar/backend/storage1
google/gcalendar/backend/storage2
google/gcalendar/backend/storage3
google/gcalendar/frontend
google/gcalendar/frontend/web1
google/gcalendar/frontend/web2
google/gcalendar/frontend/web3
google/gdrive
google/gdrive/backend
google/gdrive/backend/storage1
google/gdrive/backend/storage2
google/gdrive/backend/storage3
google/gdrive/frontend
google/gdrive/frontend/web1
```

NODES CAN LIVE IN DIFFERENT GROUPS ON DIFFERENT LEVELS

```
google/nginx
google/nginx/web1
google/nginx/web2
google/nginx/web3
google/tomcat
google/tomcat/storage1
google/tomcat/storage2
google/tomcat/storage3
```

INTERNAL ANSIBLE INVENTORY MODEL

- {'web1': {'groups': ['all', 'google', 'gcalendar', 'gdrive', 'gmail', 'frontend']}}
- {'storage2': {'groups': ['all', 'google', 'gcalendar', 'gdrive', 'gmail', 'backend']}}

INTERNAL ANSIBLE INVENTORY MODEL - 2

- {'web1': {'groups': [{'groupname': 'all', 'depth': 0}, {'groupname': 'google', 'depth': 1}, {'groupname': 'gcalendar', 'depth': 2}, {'groupname': 'gdrive', 'depth': 2}, {'groupname': 'gmail', 'depth': 2}, {'groupname': 'frontend', 'depth': 3}]}}
- {'storage2': {'groups': [{'groupname': 'all', 'depth': 0}, {'groupname': 'google', 'depth': 1}, {'groupname': 'gcalendar', 'depth': 2}, {'groupname': 'gdrive', 'depth': 2}, {'groupname': 'gmail', 'depth': 2}, {'groupname': 'frontend', 'depth': 3}]}}

HOW ABOUT INVENTORY VARIABLE PRECEDENCE?

- Ansible docs are very succinct on how inventory variables precede each other
- Because As all things Ansible, KEEP IT SIMPLE, they say.

> *There is only one Empire State Building. One Mona Lisa, etc. Figure out where to define a variable, and do not make it complicated.*

> *Remember: Child groups override parent groups, and hosts always override their groups.*

SO ANSIBLE INVENTORY IS NOT A TREE, BUT VARIABLES PRECEDE IN CHILD GROUPS?

- Child? Parent? Tree?

```
35      def add_child_group(self, group):
36
37          if self == group:
38              raise Exception("cannot add group to itself")
39
40          # do not add if it is already there
41          if not group in self.child_groups:
42              self.child_groups.append(group)
43              group.depth = max([self.depth+1, group.depth])
44              group.parent_groups.append(self)
45              self.clear_hosts_cache()
```

- When we encounter a child group, then "depth" gets a level deeper
- Deeper == more child is more precedence
- That *is* some kind of a tree, no?

THE PROJECT

- mostly a Java shop, very standardized
- {'75% of all virtual machines are tomcat hosts': 'ONE ROLE'}
- every tomcat app is a two node "cluster" behind a loadbalancer; per app one or more healthchecks
- per cluster typically 1 application (context), sometimes more
- 1 big functional application typically == several application clusters
- applications are part of a project, a project is part of an organisation
- every application has three instances in each environment
- ['development', 'testing', 'production']
- some applications communicate with other applications through the loadbalancer

INVENTORY ORGANISATION

```
all inventory
|_ organisation 1
   |_ project 1
      |_ application 1
         |_ dev
            |_ node 1
            |_ node 2
         |_ test
            |_ ..
         |_ prod
            |_ ..
      |_ application 2
            |_ ..
   |_ project 2
      |_ ..
|_ organisation 1
   |_ ..
```

OTHER GROUP TREES (WE HAVE DIFFERENT TREE'S!)

```
|_ development
   |_ organisation1-dev
      |_application1-dev
|_ testing
|_ production
```

OR ALSO

```
tomcat
|_ application1
|_ application2
 <some_other_server_role_besides_tomcat>
|_ application7
|_ application9

</some_other_server_role_besides_tomcat>
```

lowest groups with nodes are at different levels in different tree's

EXAMPLE: A SET OF REVERSE PROXIES:

```
all(0)
. infra(1)
.. rp(2)
... rp-vonet-oe(3)
... rp-vonet-on(3)
... rp-vonet-pr(3)
... rp-innet-oe(3)
... rp-innet-on(3)
... rp-innet-pr(3)
.. rp-vonet(2)
... rp-vonet-on(3)
... rp-vonet-oe(3)
... rp-vonet-pr(3)
.. rp-innet(2)
... rp-innet-on(3)
... rp-innet-oe(3)
... rp-innet-pr(3)
```

SOMETIMES DEPTH IS WRONG!

```
. unison(1)
.. geoserver-dovpub(2)
.. geoserver-dovpub-oe(2)
.. geoserver-dovpub-on(2)
.. geoserver-dovpub-pr(2)
```

[geoserver-dovpub] is a tomcat application, but is *also* member of the [unison] group (to which a unison role maps)

child groups are environment specific subgroups, but they get the the same depth as their parent

PLAYBOOK REMINDER: ANSIBLE LOOPS => WITH_ITEMS

Please "deploy some apps"

- app1
- app2

```
task:
- command: a2ensite {{ item }}
  with_items: applicationslist
```

applicationslist is defined in the inventory

```
applicationslist:
- application1
- application2
```

more loops? nested loops? with_nested, with_subelements, ..

WITHIN AN APPLICATION SERVER

we have several apps and each app can have one or more healthchecks

```
node
|_ subapp1
   |_ healthcheck1
   |_ healthcheck2
|_ subapp1
    ......
```

DATA MODELLING SO FAR: NODES, POOLS AND HEALTHCHECKS

```
publishedapps:
- name:          "web"
  type:         "{{ default_apptype }}"
  port:          8080
  lbport:        "{{ default_lbport }}"
  monitortype:   "{{ default_monitortype }}"
  quorum:        0
  monitors:
  - name:       "{{ default_monitor_appname }}"
    type:        http
    get_path:   "{{ default_get_path }}"
    protocol:   "{{ default_protocol }}"
    get_extra: "{{ default_get_extra }}"
    receive:    "{{ default_receive_string }}"
```

DATA MODELLING SO FAR: NODES, POOLS AND HEALTHCHECKS - CONTINUED

```
(... publishedapps:)
- name: tcp
  type: tcp
  port: 1234
  lbport: 601234
  monitortype: "{{ default_monitortype }}"
  quorum: 0
  monitors:
  - name:         "tcp"
    type:         tcp_half_open
    send:         ""
    receive:      ""
```

CREATE THE CLUSTER APPLICATION MONITORS (HEALTHCHECKS)

snippet of a task:

```
module: ...
  name: >
                      {{  item.0.monitorname |
                          default( 'MON-'
                        ~ item.0.type | default(default_apptype) | upper ~ '-
                        ~ ( '-' ~ item.0.name | default( 'web' ) ) | replace(
                        ~ ( '-' ~ item.1.name | default( 'web' ) ) | replace(
                        ~ '-' ~ zuil
                          ) }}
  send: >
                      {{  'GET ' ~ item.1.get_path ~ ' '
                        ~ item.1.protocol  | default( default_protocol )
                        ~ item.1.get_extra | default( '' )
                        ~ '\\r\\n\\r\\n' }}
  receive:            "{{ item.1.receive | default( default_receive_string )
  port:               "{{ item.1.port | default( 0 ) }}"
```

I will need this monitor name again later, but I can't easily definite and keep it in a variable, as it comes from a loop construct

CREATE THE CLUSTER SERVER POOLS

task snippet where we define the monitors (healthchecks) to be applied to the pool

```
monitors: >
                    {% for mon in item.monitors %}{{
                      '/Common/'
                      ~ item.monitorname |
                        default( 'MON-'
                        ~ item.type | default(default_apptype) | upper ~ '-'
                        ~ ( '-' ~ item.name | default( 'web' ) ) | replace( '
                        ~ ( '-' ~ mon.name  | default( 'web' ) ) | replace( '
                        ~ '-' ~ zuil
                        ) }}{% if not loop.last %},{% endif %}{% endfor %}

with_items: publishedapps
```

sometimes looping within a jinja2 template gives more power, and more possibilities (and more bugs)

MORE LOOPS, MORE DEEPLY NESTED?

- TODO: I need to define proxies based on previous pools
- Each application needs access to a set of other applications.
- == An applications needs the definition of a list of other applictions to get access to
- Then I need to loop though that list,
- and then the list of publishedapps for every application...
- This can possibly go cross-environment...(eg we don't have separate smtp servers per environment, so dev nodes need access to smtp-production)
- Maybe the list of applications can differ per environment?
- ...

THIS IS GETTING TOO COMPLEX...

Or should we NOT automate all things?

MANAGING INVENTORY DATA

updating software versions of applications in different stacks, in different environments

- update default java version: for all dev servers?
- Keep deployed version for each application in each environment seperately
- Nice to have = an inventory tool that keeps track of this
- let's you inherit defaults, but then remember them for particular setup

Q/A ?

THANKS!

https://github.com/sergevanginderachter/revealjs-presentations/tree/cfgmgmtcampeu-ansible-inventory