**Intro to Playbooks**

**About Playbooks**

Playbooks are a completely different way to use ansible than in adhoc task execution mode, and are particularly powerful.

Simply put, playbooks are the basis for a really simple configuration management and multi-machine deployment system, unlike any that already exist, and one that is very well suited to deploying complex applications.

Playbooks can declare configurations, but they can also orchestrate steps of any manual ordered process, even as different steps must bounce back and forth between sets of machines in particular orders. They can launch tasks synchronously or asynchronously.

While you might run the main /usr/bin/ansible program for ad-hoc tasks, playbooks are more likely to be kept in source control and used to push out your configuration or assure the configurations of your remote systems are in spec.

# Patterns

ansible <pattern\_goes\_here> -m <module\_name> -a <arguments>

Such as:

ansible webservers -m service -a "name=httpd state=restarted"

A pattern usually refers to a set of groups (which are sets of hosts) – in the above case, machines in the “webservers” group.

Anyway, to use Ansible, you’ll first need to know how to tell Ansible which hosts in your inventory to talk to. This is done by designating particular host names or groups of hosts.

The following patterns are equivalent and target all hosts in the inventory:

all

\*

It is also possible to address a specific host or set of hosts by name:

one.example.com

one.example.com:two.example.com

192.0.2.50

192.0.2.\*

The following patterns address one or more groups. Groups separated by a colon indicate an “OR” configuration. This means the host may be in either one group or the other:

webservers

webservers:dbservers

You can exclude groups as well, for instance, all machines must be in the group webservers but not in the group phoenix:

webservers:!phoenix

You can also specify the intersection of two groups. This would mean the hosts must be in the group webservers and the host must also be in the group staging:

webservers:&staging

You can do combinations:

webservers:dbservers:&staging:!phoenix

The above configuration means “all machines in the groups ‘webservers’ and ‘dbservers’ are to be managed if they are in the group ‘staging’ also, but the machines are not to be managed if they are in the group ‘phoenix’ ... whew!

You can also use variables if you want to pass some group specifiers via the “-e” argument to ansible-playbook, but this is uncommonly used:

webservers:!**{{**excluded**}}**:&**{{**required**}}**

You also don’t have to manage by strictly defined groups. Individual host names, IPs and groups, can also be referenced using wildcards

\*.example.com

\*.com

It’s also ok to mix wildcard patterns and groups at the same time:

one\*.com:dbservers

You can select a host or subset of hosts from a group by their position. For example, given the following group:

[webservers]

cobweb

webbing

weber

You can refer to hosts within the group by adding a subscript to the group name:

webservers[0] *# == cobweb*

webservers[-1] *# == weber*

webservers[0:1] *# == webservers[0],webservers[1]*

*# == cobweb,webbing*

webservers[1:] *# == webbing,weber*

Most people don’t specify patterns as regular expressions, but you can. Just start the pattern with a ‘~’:

~(web|db).\*\.example\.com

While we’re jumping a bit ahead, additionally, you can add an exclusion criteria just by supplying the --limit flag to /usr/bin/ansible or /usr/bin/ansible-playbook:

ansible-playbook site.yml --limit datacenter2

And if you want to read the list of hosts from a file, prefix the file name with ‘@’. Since Ansible 1.2:

ansible-playbook site.yml --limit @retry\_hosts.txt