Inventory

Topics

Ansible works against multiple systems in your infrastructure at the same time. It does this by selecting portions of systems listed in Ansible's inventory file, which defaults to being saved in the location /etc/ansible/hosts.

Not only is this inventory configurable, but you can also use multiple inventory files at the same time (explained below) and also pull inventory from dynamic or cloud sources, as described in *Dynamic Inventory*.

Hosts and Groups

The format for /etc/ansible/hosts is an INI-like format and looks like this:

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

The things in brackets are group names, which are used in classifying systems and deciding what systems you are controlling at what times and for what purpose.

It is ok to put systems in more than one group, for instance a server could be both a webserver and a dbserver. If you do, note that variables will come from all of the groups they are a member of, and variable precedence is detailed in a later chapter.

If you have hosts that run on non-standard SSH ports you can put the port number after the hostname with a colon. Ports listed in your SSH config file won't be used with the paramiko connection but will be used with the openssh

connection.

To make things explicit, it is suggested that you set them if things are not running on the default port:

```
badwolf.example.com:5309
```

Suppose you have just static IPs and want to set up some aliases that don't live in your host file, or you are connecting through tunnels. You can do things like this:

```
jumper ansible_ssh_port=5555 ansible_ssh_host=192.168.1.50
```

In the above example, trying to ansible against the host alias "jumper" (which may not even be a real hostname) will contact 192.168.1.50 on port 5555. Note that this is using a feature of the inventory file to define some special variables. Generally speaking this is not the best way to define variables that describe your system policy, but we'll share suggestions on doing this later. We're just getting started.

Adding a lot of hosts? If you have a lot of hosts following similar patterns you can do this rather than listing each hostname:

```
[webservers]
www[01:50].example.com
```

For numeric patterns, leading zeros can be included or removed, as desired. Ranges are inclusive. You can also define alphabetic ranges:

```
[databases]
db-[a:f].example.com
```

You can also select the connection type and user on a per host basis:

```
[targets]

localhost              ansible_connection=local
other1.example.com     ansible_connection=ssh        ansible_ssh_user=mpdehaan
other2.example.com     ansible_connection=ssh        ansible_ssh_user=mdehaan
```

As mentioned above, setting these in the inventory file is only a shorthand, and we'll discuss how to store them in individual files in the 'host_vars' directory a bit later on.

Host Variables

As alluded to above, it is easy to assign variables to hosts that will be used later in playbooks:

```
[atlanta]
host1 http_port=80 maxRequestsPerChild=808
host2 http_port=303 maxRequestsPerChild=909
```

Group Variables

Variables can also be applied to an entire group at once:

```
[atlanta]
host1
```

```
host2

[atlanta:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com
```

Groups of Groups, and Group Variables

It is also possible to make groups of groups and assign variables to groups. These variables can be used by /usr/bin/ansible-playbook, but not /usr/bin/ansible:

```
[atlanta]
host1
host2

[raleigh]
host2
host3

[southeast:children]
atlanta
raleigh

[southeast:vars]
some_server=foo.southeast.example.com
halon_system_timeout=30
self_destruct_countdown=60
escape_pods=2

[usa:children]
southeast
northeast
southwest
northwest
```

If you need to store lists or hash data, or prefer to keep host and group specific

variables separate from the inventory file, see the next section.

Splitting Out Host and Group Specific Data

The preferred practice in Ansible is actually not to store variables in the main inventory file.

In addition to storing variables directly in the INI file, host and group variables can be stored in individual files relative to the inventory file.

These variable files are in YAML format. See *YAML Syntax* if you are new to YAML.

Assuming the inventory file path is:

```
/etc/ansible/hosts
```

If the host is named ‘foosball’, and in groups ‘raleigh’ and ‘webservers’, variables in YAML files at the following locations will be made available to the host:

```
/etc/ansible/group_vars/raleigh
/etc/ansible/group_vars/webservers
/etc/ansible/host_vars/foosball
```

For instance, suppose you have hosts grouped by datacenter, and each datacenter uses some different servers. The data in the groupfile ‘/etc/ansible/group_vars/raleigh’ for the ‘raleigh’ group might look like:

```
---
ntp_server: acme.example.org
database_server: storage.example.org
```

It is ok if these files do not exist, as this is an optional feature.

As an advanced use-case, you can create *directories* named after your groups or hosts, and Ansible will read all the files in these directories. An example with the 'raleigh' group:

```
/etc/ansible/group_vars/raleigh/db_settings
/etc/ansible/group_vars/raleigh/cluster_settings
```

All hosts that are in the 'raleigh' group will have the variables defined in these files available to them. This can be very useful to keep your variables organized when a single file starts to be too big, or when you want to use *Ansible Vault* on a part of a group's variables. Note that this only works on Ansible 1.4 or later.

Tip: In Ansible 1.2 or later the group_vars/ and host_vars/ directories can exist in either the playbook directory OR the inventory directory. If both paths exist, variables in the playbook directory will override variables set in the inventory directory.

Tip: Keeping your inventory file and variables in a git repo (or other version control) is an excellent way to track changes to your inventory and host variables.

List of Behavioral Inventory Parameters

As alluded to above, setting the following variables controls how ansible interacts with remote hosts. Some we have already mentioned:

```
ansible_ssh_host
```

```
  The name of the host to connect to, if different from the alias you wish to give to it.
ansible_ssh_port
  The ssh port number, if not 22
ansible_ssh_user
  The default ssh user name to use.
ansible_ssh_pass
  The ssh password to use (this is insecure, we strongly recommend using --ask-pass or SSH keys)
ansible_sudo
  The boolean to decide if sudo should be used for this host. Defaults to false.
ansible_sudo_pass
  The sudo password to use (this is insecure, we strongly recommend using --ask-sudo-pass)
ansible_sudo_exe (new in version 1.8)
  The sudo command path.
ansible_connection
  Connection type of the host. Candidates are local, ssh or paramiko.  The default is paramiko before Ansib
ansible_ssh_private_key_file
  Private key file used by ssh.  Useful if using multiple keys and you don't want to use SSH agent.
ansible_shell_type
  The shell type of the target system. By default commands are formatted using 'sh'-style syntax by default
ansible_python_interpreter
  The target host python path. This is useful for systems with more
  than one Python or not located at "/usr/bin/python" such as \*BSD, or where /usr/bin/python
  is not a 2.X series Python.  We do not use the "/usr/bin/env" mechanism as that requires the remote user'
  path to be set right and also assumes the "python" executable is named python, where the executable might
  be named something like "python26".
ansible\_\*\_interpreter
  Works for anything such as ruby or perl and works just like ansible_python_interpreter.
  This replaces shebang of modules which will run on that host.
```

Examples from a host file:

```
some_host         ansible_ssh_port=2222     ansible_ssh_user=manager
aws_host          ansible_ssh_private_key_file=/home/example/.ssh/aws.pem
freebsd_host      ansible_python_interpreter=/usr/local/bin/python
ruby_module_host  ansible_ruby_interpreter=/usr/bin/ruby.1.9.3
```

See also

Dynamic Inventory
Pulling inventory from dynamic sources, such as cloud providers

Introduction To Ad-Hoc Commands
Examples of basic commands

Playbooks
Learning ansible's configuration management language

Mailing List
Questions? Help? Ideas? Stop by the list on Google Groups

irc.freenode.net
#ansible IRC chat channel

Previous Next