Working with dynamic inventory

If your Ansible inventory fluctuates over time, with hosts spinning up and shutting down in response to business demands, the static inventory solutions described in :ref:`inventory` will not serve your needs. You may need to track hosts from multiple sources: cloud providers, LDAP, Cobbler, and/or enterprise CMDB systems.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]intro_dynamic_inventory.rst`, **line 11**); ***backlink***
>
> Unknown interpreted text role "ref".

Ansible integrates all of these options through a dynamic external inventory system. Ansible supports two ways to connect with external inventory: :ref:`inventory_plugins` and *inventory scripts*.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]intro_dynamic_inventory.rst`, **line 13**); ***backlink***
>
> Unknown interpreted text role "ref".

Inventory plugins take advantage of the most recent updates to the Ansible core code. We recommend plugins over scripts for dynamic inventory. You can :ref:`write your own plugin <developing_inventory>` to connect to additional dynamic inventory sources.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]intro_dynamic_inventory.rst`, **line 15**); ***backlink***
>
> Unknown interpreted text role "ref".

You can still use inventory scripts if you choose. When we implemented inventory plugins, we ensured backwards compatibility through the script inventory plugin. The examples below illustrate how to use inventory scripts.

If you prefer a GUI for handling dynamic inventory, the inventory database on AWX or :ref:`ansible_platform` syncs with all your dynamic inventory sources, provides web and REST access to the results, and offers a graphical inventory editor. With a database record of all of your hosts, you can correlate past event history and see which hosts have had failures on their last playbook runs.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]intro_dynamic_inventory.rst`, **line 19**); ***backlink***
>
> Unknown interpreted text role "ref".

Inventory script example: Cobbler

Ansible integrates seamlessly with Cobbler, a Linux installation server originally written by Michael DeHaan and now led by James Cammarata, who works for Ansible.

While primarily used to kickoff OS installations and manage DHCP and DNS, Cobbler has a generic layer that can represent data for multiple configuration management systems (even at the same time) and serve as a 'lightweight CMDB'.

To tie your Ansible inventory to Cobbler, copy this script to `/etc/ansible` and `chmod +x` the file. Run `cobblerd` any time you use Ansible and use the `-i` command line option (for example, `-i /etc/ansible/cobbler.py`) to communicate with Cobbler using Cobbler's XMLRPC API.

Add a `cobbler.ini` file in `/etc/ansible` so Ansible knows where the Cobbler server is and some cache improvements can be used. For example:

```
[cobbler]

# Set Cobbler's hostname or IP address
host = http://127.0.0.1/cobbler_api

# API calls to Cobbler can be slow. For this reason, we cache the results of an API
# call. Set this to the path you want cache files to be written to. Two files
# will be written to this directory:
#   - ansible-cobbler.cache
#   - ansible-cobbler.index

cache_path = /tmp

# The number of seconds a cache file is considered valid. After this many
# seconds, a new API call will be made, and the cache file will be updated.
```

```
cache_max_age = 900
```

First test the script by running `/etc/ansible/cobbler.py` directly. You should see some JSON data output, but it may not have anything in it just yet.

Let's explore what this does. In Cobbler, assume a scenario somewhat like the following:

```
cobbler profile add --name=webserver --distro=CentOS6-x86_64
cobbler profile edit --name=webserver --mgmt-classes="webserver" --ksmeta="a=2 b=3"
cobbler system edit --name=foo --dns-name="foo.example.com" --mgmt-classes="atlanta" --ksmeta="c=4"
cobbler system edit --name=bar --dns-name="bar.example.com" --mgmt-classes="atlanta" --ksmeta="c=5"
```

In the example above, the system 'foo.example.com' is addressable by ansible directly, but is also addressable when using the group names 'webserver' or 'atlanta'. Since Ansible uses SSH, it contacts system foo over 'foo.example.com', only, never just 'foo'. Similarly, if you tried "ansible foo", it would not find the system... but "ansible 'foo*'" would do, because the system DNS name starts with 'foo'.

The script provides more than host and group info. In addition, as a bonus, when the 'setup' module is run (which happens automatically when using playbooks), the variables 'a', 'b', and 'c' will all be auto-populated in the templates:

```
# file: /srv/motd.j2
Welcome, I am templated with a value of a={{ a }}, b={{ b }}, and c={{ c }}
```

Which could be executed just like this:

```
ansible webserver -m setup
ansible webserver -m template -a "src=/tmp/motd.j2 dest=/etc/motd"
```

> **Note**
>
> The name 'webserver' came from Cobbler, as did the variables for the config file. You can still pass in your own variables like normal in Ansible, but variables from the external inventory script will override any that have the same name.

So, with the template above (`motd.j2`), this results in the following data being written to `/etc/motd` for system 'foo':

```
Welcome, I am templated with a value of a=2, b=3, and c=4
```

And on system 'bar' (bar.example.com):

```
Welcome, I am templated with a value of a=2, b=3, and c=5
```

And technically, though there is no major good reason to do it, this also works:

```
ansible webserver -m ansible.builtin.shell -a "echo {{ a }}"
```

So, in other words, you can use those variables in arguments/actions as well.

Inventory script example: OpenStack

If you use an OpenStack-based cloud, instead of manually maintaining your own inventory file, you can use the `openstack_inventory.py` dynamic inventory to pull information about your compute instances directly from OpenStack.

You can download the latest version of the OpenStack inventory script here.

You can use the inventory script explicitly (by passing the *-i openstack_inventory.py* argument to Ansible) or implicitly (by placing the script at */etc/ansible/hosts*).

Explicit use of OpenStack inventory script

Download the latest version of the OpenStack dynamic inventory script and make it executable.

```
wget https://raw.githubusercontent.com/openstack/ansible-collections-openstack/master/scripts/inventory/openstack_
chmod +x openstack_inventory.py
```

> **Note**
>
> Do not name it *openstack.py*. This name will conflict with imports from openstacksdk.

Source an OpenStack RC file:

```
source openstack.rc
```

> **Note**
>
> An OpenStack RC file contains the environment variables required by the client tools to establish a connection with the cloud provider, such as the authentication URL, user name, password and region name. For more information on how to download, create or source an OpenStack RC file, please refer to Set environment variables using the OpenStack RC file.

You can confirm the file has been successfully sourced by running a simple command, such as *nova list* and ensuring it returns no errors.

> **Note**
>
> The OpenStack command line clients are required to run the *nova list* command. For more information on how to install them, please refer to Install the OpenStack command-line clients.

You can test the OpenStack dynamic inventory script manually to confirm it is working as expected:

```
./openstack_inventory.py --list
```

After a few moments you should see some JSON output with information about your compute instances.

Once you confirm the dynamic inventory script is working as expected, you can tell Ansible to use the *openstack_inventory.py* script as an inventory file, as illustrated below:

```
ansible -i openstack_inventory.py all -m ansible.builtin.ping
```

Implicit use of OpenStack inventory script

Download the latest version of the OpenStack dynamic inventory script, make it executable and copy it to */etc/ansible/hosts*:

```
wget https://raw.githubusercontent.com/openstack/ansible-collections-openstack/master/scripts/inventory/openstack_
chmod +x openstack_inventory.py
sudo cp openstack_inventory.py /etc/ansible/hosts
```

Download the sample configuration file, modify it to suit your needs and copy it to */etc/ansible/openstack.yml*:

```
wget https://raw.githubusercontent.com/openstack/ansible-collections-openstack/master/scripts/inventory/openstack.
vi openstack.yml
sudo cp openstack.yml /etc/ansible/
```

You can test the OpenStack dynamic inventory script manually to confirm it is working as expected:

```
/etc/ansible/hosts --list
```

After a few moments you should see some JSON output with information about your compute instances.

Refreshing the cache

Note that the OpenStack dynamic inventory script will cache results to avoid repeated API calls. To explicitly clear the cache, you can run the openstack_inventory.py (or hosts) script with the `--refresh` parameter:

```
./openstack_inventory.py --refresh --list
```

Other inventory scripts

In Ansible 2.10 and later, inventory scripts moved to their associated collections. Many are now in the community.general scripts/inventory directory. We recommend you use :ref:`inventory_plugins` instead.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]intro_dynamic_inventory.rst, line 204); *backlink***
>
> Unknown interpreted text role "ref".

Using inventory directories and multiple inventory sources

If the location given to `-i` in Ansible is a directory (or as so configured in `ansible.cfg`), Ansible can use multiple inventory sources at the same time. When doing so, it is possible to mix both dynamic and statically managed inventory sources in the same ansible run. Instant hybrid cloud!

In an inventory directory, executable files are treated as dynamic inventory sources and most other files as static sources. Files which end with any of the following are ignored:

```
~, .orig, .bak, .ini, .cfg, .retry, .pyc, .pyo
```

You can replace this list with your own selection by configuring an `inventory_ignore_extensions` list in `ansible.cfg`, or setting the :envvar:`ANSIBLE_INVENTORY_IGNORE` environment variable. The value in either case must be a comma-separated list of patterns, as shown above.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]intro_dynamic_inventory.rst, line 221); *backlink***
>
> Unknown interpreted text role "envvar".

Any `group_vars` and `host_vars` subdirectories in an inventory directory are interpreted as expected, making inventory directories a powerful way to organize different sets of configurations. See :ref:`using_multiple_inventory_sources` for more information.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]intro_dynamic_inventory.rst, line 223); *backlink***
>
> Unknown interpreted text role "ref".

Static groups of dynamic groups

When defining groups of groups in the static inventory file, the child groups must also be defined in the static inventory file, otherwise ansible returns an error. If you want to define a static group of dynamic child groups, define the dynamic groups as empty in the static inventory file. For example:

```
[tag_Name_staging_foo]
```

```
[tag_Name_staging_bar]

[staging:children]
tag_Name_staging_foo
tag_Name_staging_bar
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]intro_dynamic_inventory.rst, line 246)

Unknown directive type "seealso".

```
.. seealso::

   :ref:`intro_inventory`
       All about static inventory files
   `Mailing List <https://groups.google.com/group/ansible-project>`_
       Questions? Help? Ideas?  Stop by the list on Google Groups
   :ref:`communication_irc`
       How to join Ansible chat channels
```