Inventory plugins

Inventory plugins allow users to point at data sources to compile the inventory of hosts that Ansible uses to target tasks, either using the `-i /path/to/file` and/or `-i 'host1, host2'` command line parameters or from other configuration sources. If necessary, you can :ref:`create custom inventory plugins <developing_inventory_plugins>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)inventory.rst, line 10); *backlink***
>
> Unknown interpreted text role "ref".

Enabling inventory plugins

Most inventory plugins shipped with Ansible are enabled by default or can be used by with the `auto` plugin.

In some circumstances, for example, if the inventory plugin does not use a YAML configuration file, you may need to enable the specific plugin. You can do this by setting `enable_plugins` in your :ref:`ansible.cfg <ansible_configuration_settings>` file in the `[inventory]` section. Modifying this will override the default list of enabled plugins. Here is the default list of enabled plugins that ships with Ansible:

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

```
[inventory]
enable_plugins = host_list, script, auto, yaml, ini, toml
```

If the plugin is in a collection, use the fully qualified name:

```
[inventory]
enable_plugins = namespace.collection_name.inventory_plugin_name
```

If you use a plugin that supports a YAML configuration source, make sure that the name matches the name provided in the `plugin` entry of the inventory source file.

Using inventory plugins

To use an inventory plugin, you must provide an inventory source. Most of the time this is a file containing host information or a YAML configuration file with options for the plugin. You can use the `-i` flag to provide inventory sources or configure a default inventory path.

```
ansible hostname -i inventory_source -m ansible.builtin.ping
```

To start using an inventory plugin with a YAML configuration source, create a file with the accepted filename schema documented for the plugin in question, then add `plugin: plugin_name`. Use the fully qualified name if the plugin is in a collection.

```
# demo.aws_ec2.yml
plugin: amazon.aws.aws_ec2
```

Each plugin should document any naming restrictions. In addition, the YAML config file must end with the extension `yml` or `yaml` to be enabled by default with the `auto` plugin (otherwise, see the section above on enabling plugins).

After providing any required options, you can view the populated inventory with `ansible-inventory -i demo.aws_ec2.yml --graph`:

```
@all:
  |--@aws_ec2:
  |  |--ec2-12-345-678-901.compute-1.amazonaws.com
  |  |--ec2-98-765-432-10.compute-1.amazonaws.com
  |--@ungrouped:
```

If you are using an inventory plugin in a playbook-adjacent collection and want to test your setup with `ansible-inventory`, use the

`--playbook-dir` flag.

Your inventory source might be a directory of inventory configuration files. The constructed inventory plugin only operates on those hosts already in inventory, so you may want the constructed inventory configuration parsed at a particular point (such as last). Ansible parses the directory recursively, alphabetically. You cannot configure the parsing approach, so name your files to make it work predictably. Inventory plugins that extend constructed features directly can work around that restriction by adding constructed options in addition to the inventory plugin options. Otherwise, you can use `-i` with multiple sources to impose a specific order, for example `-i demo.aws_ec2.yml -i clouds.yml -i constructed.yml`.

You can create dynamic groups using host variables with the constructed `keyed_groups` option. The option `groups` can also be used to create groups and `compose` creates and modifies host variables. Here is an aws_ec2 example utilizing constructed features:

```
# demo.aws_ec2.yml
plugin: amazon.aws.aws_ec2
regions:
  - us-east-1
  - us-east-2
keyed_groups:
  # add hosts to tag_Name_value groups for each aws_ec2 host's tags.Name variable
  - key: tags.Name
    prefix: tag_Name_
    separator: ""
groups:
  # add hosts to the group development if any of the dictionary's keys or values is the word 'devel'
  development: "'devel' in (tags|list)"
compose:
  # set the ansible_host variable to connect with the private IP address without changing the hostname
  ansible_host: private_ip_address
```

Now the output of `ansible-inventory -i demo.aws_ec2.yml --graph`:

```
@all:
  |--@aws_ec2:
  |  |--ec2-12-345-678-901.compute-1.amazonaws.com
  |  |--ec2-98-765-432-10.compute-1.amazonaws.com
  |  |--...
  |--@development:
  |  |--ec2-12-345-678-901.compute-1.amazonaws.com
  |  |--ec2-98-765-432-10.compute-1.amazonaws.com
  |--@tag_Name_ECS_Instance:
  |  |--ec2-98-765-432-10.compute-1.amazonaws.com
  |--@tag_Name_Test_Server:
  |  |--ec2-12-345-678-901.compute-1.amazonaws.com
  |--@ungrouped
```

If a host does not have the variables in the configuration above (in other words, `tags.Name`, `tags`, `private_ip_address`), the host will not be added to groups other than those that the inventory plugin creates and the `ansible_host` host variable will not be modified.

Inventory plugins that support caching can use the general settings for the fact cache defined in the `ansible.cfg` file's `[defaults]` section or define inventory-specific settings in the `[inventory]` section. Individual plugins can define plugin-specific cache settings in their config file:

```
# demo.aws_ec2.yml
plugin: amazon.aws.aws_ec2
cache: yes
cache_plugin: ansible.builtin.jsonfile
cache_timeout: 7200
cache_connection: /tmp/aws_inventory
cache_prefix: aws_ec2
```

Here is an example of setting inventory caching with some fact caching defaults for the cache plugin used and the timeout in an `ansible.cfg` file:

```
[defaults]
fact_caching = ansible.builtin.jsonfile
fact_caching_connection = /tmp/ansible_facts
cache_timeout = 3600

[inventory]
cache = yes
cache_connection = /tmp/ansible_inventory
```

Plugin list

You can use `ansible-doc -t inventory -l` to see the list of available plugins. Use `ansible-doc -t inventory <plugin name>` to see plugin-specific documentation and examples.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)inventory.rst, line 143)

Unknown directive type "seealso".

```
.. seealso::

   :ref:`about_playbooks`
       An introduction to playbooks
   :ref:`callback_plugins`
       Callback plugins
   :ref:`connection_plugins`
       Connection plugins
   :ref:`filter_plugins`
       Filter plugins
   :ref:`test_plugins`
       Test plugins
   :ref:`lookup_plugins`
       Lookup plugins
   :ref:`vars_plugins`
       Vars plugins
   `User Mailing List <https://groups.google.com/group/ansible-devel>`_
       Have a question?  Stop by the google group!
   :ref:`communication_irc`
       How to join Ansible chat channels
```