Lookup plugins

- Enabling lookup plugins
- Using lookup plugins
- Forcing lookups to return lists: `query` and `wantlist=True`
- Plugin list

Lookup plugins are an Ansible-specific extension to the Jinja2 templating language. You can use lookup plugins to access data from outside sources (files, databases, key/value stores, APIs, and other services) within your playbooks. Like all :ref:`templating <playbooks_templating>`, lookups execute and are evaluated on the Ansible control machine. Ansible makes the data returned by a lookup plugin available using the standard templating system. You can use lookup plugins to load variables or templates with information from external sources. You can :ref:`create custom lookup plugins <developing_lookup_plugins>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 10); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 10); *backlink*

Unknown interpreted text role "ref".

Note

- Lookups are executed with a working directory relative to the role or play, as opposed to local tasks, which are executed relative the executed script.
- Pass `wantlist=True` to lookups to use in Jinja2 template "for" loops.
- By default, lookup return values are marked as unsafe for security reasons. If you trust the outside source your lookup accesses, pass `allow_unsafe=True` to allow Jinja2 templates to evaluate lookup values.

Warning

- Some lookups pass arguments to a shell. When using variables from a remote/untrusted source, use the *|quote* filter to ensure safe usage.

Enabling lookup plugins

Ansible enables all lookup plugins it can find. You can activate a custom lookup by either dropping it into a `lookup_plugins` directory adjacent to your play, inside the `plugins/lookup/` directory of a collection you have installed, inside a standalone role, or in one of the lookup directory sources configured in :ref:`ansible.cfg <ansible_configuration_settings>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 27); *backlink*

Unknown interpreted text role "ref".

Using lookup plugins

You can use lookup plugins anywhere you can use templating in Ansible: in a play, in variables file, or in a Jinja2 template for the :ref:`template <template_module>` module. For more information on using lookup plugins, see :ref:`playbooks_lookups`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 35); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 35); *backlink*

Unknown interpreted text role "ref".

```
vars:
  file_contents: "{{ lookup('file', 'path/to/file.txt') }}"
```

Lookups are an integral part of loops. Wherever you see `with_`, the part after the underscore is the name of a lookup. For this reason, most lookups output lists and take lists as input; for example, `with_items` uses the :ref:`items <items_lookup>` lookup:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 42); *backlink*

Unknown interpreted text role "ref".

```
tasks:
  - name: count to 3
    debug: msg={{ item }}
    with_items: [1, 2, 3]
```

You can combine lookups with :ref:`filters <playbooks_filters>`, :ref:`tests <playbooks_tests>` and even each other to do some complex data generation and manipulation. For example:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 49); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 49); *backlink*

Unknown interpreted text role "ref".

```
tasks:
  - name: valid but useless and over complicated chained lookups and filters
    debug: msg="find the answer here:\n{{ lookup('url', 'https://google.com/search/?q=' + item|urlencode)|join(' ') }}"
    with_nested:
      - "{{ lookup('consul_kv', 'bcs/' + lookup('file', '/the/question') + ', host=localhost, port=2000')|shuffle }}"
```

```
    - "{{ lookup('sequence', 'end=42 start=2 step=2')|map('log', 4)|list) }}"
    - ['a', 'c', 'd', 'c']
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 59)

Unknown directive type "versionadded".

```
.. versionadded:: 2.6
```

You can control how errors behave in all lookup plugins by setting `errors` to `ignore`, `warn`, or `strict`. The default setting is `strict`, which causes the task to fail if the lookup returns an error. For example:

To ignore lookup errors:

```
- name: if this file does not exist, I do not care .. file plugin itself warns anyway ...
  debug: msg="{{ lookup('file', '/nosuchfile', errors='ignore') }}"
```

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 68)

Cannot analyze code. No Pygments lexer found for "ansible-output".

```
.. code-block:: ansible-output

    [WARNING]: Unable to find '/nosuchfile' in expected paths (use -vvvvv to see paths)

    ok: [localhost] => {
        "msg": ""
    }
```

To get a warning instead of a failure:

```
- name: if this file does not exist, let me know, but continue
  debug: msg="{{ lookup('file', '/nosuchfile', errors='warn') }}"
```

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 82)

Cannot analyze code. No Pygments lexer found for "ansible-output".

```
.. code-block:: ansible-output

    [WARNING]: Unable to find '/nosuchfile' in expected paths (use -vvvvv to see paths)

    [WARNING]: An unhandled exception occurred while running the lookup plugin 'file'. Error was a <class 'ansible.errors.AnsibleE

    ok: [localhost] => {
        "msg": ""
    }
```

To get a fatal error (the default):

```
- name: if this file does not exist, FAIL (this is the default)
  debug: msg="{{ lookup('file', '/nosuchfile', errors='strict') }}"
```

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 98)

Cannot analyze code. No Pygments lexer found for "ansible-output".

```
.. code-block:: ansible-output

    [WARNING]: Unable to find '/nosuchfile' in expected paths (use -vvvvv to see paths)

    fatal: [localhost]: FAILED! => {"msg": "An unhandled exception occurred while running the lookup plugin 'file'. Error was a <c
```

Forcing lookups to return lists: `query` and `wantlist=True`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 110)

Unknown directive type "versionadded".

```
.. versionadded:: 2.5
```

In Ansible 2.5, a new Jinja2 function called `query` was added for invoking lookup plugins. The difference between `lookup` and `query` is largely that `query` will always return a list. The default behavior of `lookup` is to return a string of comma separated values. `lookup` can be explicitly configured to return a list using `wantlist=True`.

This feature provides an easier and more consistent interface for interacting with the new `loop` keyword, while maintaining backwards compatibility with other uses of `lookup`.

The following examples are equivalent:

```
lookup('dict', dict_variable, wantlist=True)

query('dict', dict_variable)
```

As demonstrated above, the behavior of `wantlist=True` is implicit when using `query`.

Additionally, `q` was introduced as a shortform of `query`:

```
q('dict', dict_variable)
```

Plugin list

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\plugins\(ansible-devel)(docs)(docsite)(rst)(plugins)lookup.rst, line 142)

Unknown directive type "seealso".

```
.. seealso::
```

```
:ref:`about_playbooks`
    An introduction to playbooks
:ref:`inventory_plugins`
    Ansible inventory plugins
:ref:`callback_plugins`
    Ansible callback plugins
:ref:`filter_plugins`
    Jinja2 filter plugins
:ref:`test_plugins`
    Jinja2 test 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
```