Frequently Asked Questions

Here are some commonly asked questions and their answers.

Where did all the modules go?

In July, 2019, we announced that collections would be the future of Ansible content delivery. A collection is a distribution format for Ansible content that can include playbooks, roles, modules, and plugins. In Ansible 2.9 we added support for collections. In Ansible 2.10 we extracted most modules from the main ansible/ansible repository and placed them in :ref:`collections <list_of_collections>`. Collections may be maintained by the Ansible team, by the Ansible community, or by Ansible partners. The ansible/ansible repository now contains the code for basic features and functions, such as copying module code to managed nodes. This code is also known as `ansible-core` (it was briefly called `ansible-base` for version 2.10).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 13**); ***backlink***

Unknown interpreted text role "ref".

- To learn more about using collections, see :ref:`collections`.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 15**); ***backlink***

  Unknown interpreted text role "ref".

- To learn more about developing collections, see :ref:`developing_collections`.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 16**); ***backlink***

  Unknown interpreted text role "ref".

- To learn more about contributing to existing collections, see the individual collection repository for guidelines, or see :ref:`contributing_maintained_collections` to contribute to one of the Ansible-maintained collections.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 17**); ***backlink***

  Unknown interpreted text role "ref".

Where did this specific module go?

IF you are searching for a specific module, you can check the runtime.yml file, which lists the first destination for each module that we extracted from the main ansible/ansible repository. Some modules have moved again since then. You can also search on Ansible Galaxy or ask on one of our :ref:`chat channels <communication_irc>`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 24**); ***backlink***

Unknown interpreted text role "ref".

How can I set the PATH or any other environment variable for a task or entire play?

Setting environment variables can be done with the *environment* keyword. It can be used at the task or other levels in the play.

```
shell:
  cmd: date
environment:
  LANG=fr_FR.UTF-8
```

```
hosts: servers
environment:
  PATH: "{{ ansible_env.PATH }}:/thingy/bin"
  SOME: value
```

Note

starting in 2.0.1 the setup task from `gather_facts` also inherits the environment directive from the play, you might need to use the `|default` filter to avoid errors if setting this at play level.

How do I handle different machines needing different user accounts or ports to log in with?

Setting inventory variables in the inventory file is the easiest way.

For instance, suppose these hosts have different usernames and ports:

```
[webservers]
asdf.example.com  ansible_port=5000   ansible_user=alice
jkl.example.com   ansible_port=5001   ansible_user=bob
```

You can also dictate the connection type to be used, if you want:

```
[testcluster]
localhost           ansible_connection=local
/path/to/chroot1    ansible_connection=chroot
foo.example.com     ansible_connection=paramiko
```

You may also wish to keep these in group variables instead, or file them in a group_vars/<groupname> file. See the rest of the documentation for more information about how to organize variables.

How do I get ansible to reuse connections, enable Kerberized SSH, or have Ansible pay attention to my local SSH config file?

Switch your default connection type in the configuration file to `ssh`, or use `-c ssh` to use Native OpenSSH for connections instead of the python paramiko library. In Ansible 1.2.1 and later, `ssh` will be used by default if OpenSSH is new enough to support ControlPersist as an option.

Paramiko is great for starting out, but the OpenSSH type offers many advanced options. You will want to run Ansible from a machine new enough to support ControlPersist, if you are using this connection type. You can still manage older clients. If you are using RHEL 6, CentOS 6, SLES 10 or SLES 11 the version of OpenSSH is still a bit old, so consider managing from a Fedora or openSUSE client even though you are managing older nodes, or just use paramiko.

We keep paramiko as the default as if you are first installing Ansible on these enterprise operating systems, it offers a better experience for new users.

How do I configure a jump host to access servers that I have no direct access to?

You can set a `ProxyCommand` in the `ansible_ssh_common_args` inventory variable. Any arguments specified in this variable are added to the sftp/scp/ssh command line when connecting to the relevant host(s). Consider the following inventory group:

```
[gatewayed]
foo ansible_host=192.0.2.1
bar ansible_host=192.0.2.2
```

You can create *group_vars/gatewayed.yml* with the following contents:

```
ansible_ssh_common_args: '-o ProxyCommand="ssh -W %h:%p -q user@gateway.example.com"'
```

Ansible will append these arguments to the command line when trying to connect to any hosts in the group `gatewayed`. (These arguments are used in addition to any `ssh_args` from `ansible.cfg`, so you do not need to repeat global `ControlPersist` settings in `ansible_ssh_common_args`.)

Note that `ssh -W` is available only with OpenSSH 5.4 or later. With older versions, it's necessary to execute `nc %h:%p` or some equivalent command on the bastion host.

With earlier versions of Ansible, it was necessary to configure a suitable `ProxyCommand` for one or more hosts in `~/.ssh/config`, or globally by setting `ssh_args` in `ansible.cfg`.

How do I get Ansible to notice a dead target in a timely manner?

You can add `-o ServerAliveInterval=NumberOfSeconds` in `ssh_args` from `ansible.cfg`. Without this option, SSH and therefore Ansible will wait until the TCP connection times out. Another solution is to add `ServerAliveInterval` into your global SSH configuration. A good value for `ServerAliveInterval` is up to you to decide; keep in mind that `ServerAliveCountMax=3` is the SSH default so any value you set will be tripled before terminating the SSH session.

How do I speed up run of ansible for servers from cloud providers (EC2, openstack,..)?

Don't try to manage a fleet of machines of a cloud provider from your laptop. Rather connect to a management node inside this cloud provider first and run Ansible from there.

How do I handle not having a Python interpreter at /usr/bin/python on a remote machine?

While you can write Ansible modules in any language, most Ansible modules are written in Python, including the ones central to letting Ansible work.

By default, Ansible assumes it can find a :command:`/usr/bin/python` on your remote system that is either Python2, version 2.6 or higher or Python3, 3.5 or higher.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 151); *backlink***
>
> Unknown interpreted text role "command".

Setting the inventory variable `ansible_python_interpreter` on any host will tell Ansible to auto-replace the Python interpreter with that value instead. Thus, you can point to any Python you want on the system if :command:`/usr/bin/python` on your system does not point to a compatible Python interpreter.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 154); *backlink***
>
> Unknown interpreted text role "command".

Some platforms may only have Python 3 installed by default. If it is not installed as :command:`/usr/bin/python`, you will need to configure the path to the interpreter via `ansible_python_interpreter`. Although most core modules will work with Python 3, there may be some special purpose ones which do not or you may encounter a bug in an edge case. As a temporary workaround you can install Python 2 on the managed host and configure Ansible to use that Python via `ansible_python_interpreter`. If there's no mention in the module's documentation that the module requires Python 2, you can also report a bug on our bug tracker so that the incompatibility can be fixed in a future release.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 159);** ***backlink***

Unknown interpreted text role "command".

Do not replace the shebang lines of your python modules. Ansible will do this for you automatically at deploy time.

Also, this works for ANY interpreter, for example ruby: `ansible_ruby_interpreter`, perl: `ansible_perl_interpreter`, and so on, so you can use this for custom modules written in any scripting language and control the interpreter location.

Keep in mind that if you put `env` in your module shebang line (`#!/usr/bin/env <other>`), this facility will be ignored so you will be at the mercy of the remote *$PATH*.

How do I handle the package dependencies required by Ansible package dependencies during Ansible installation ?

While installing Ansible, sometimes you may encounter errors such as *No package 'libffi' found* or *fatal error: Python.h: No such file or directory* These errors are generally caused by the missing packages, which are dependencies of the packages required by Ansible. For example, *libffi* package is dependency of *pynacl* and *paramiko* (Ansible -> paramiko -> pynacl -> libffi).

In order to solve these kinds of dependency issues, you might need to install required packages using the OS native package managers, such as *yum*, *dnf*, or *apt*, or as mentioned in the package installation guide.

Refer to the documentation of the respective package for such dependencies and their installation methods.

Common Platform Issues

What customer platforms does Red Hat support?

A number of them! For a definitive list please see this Knowledge Base article.

Running in a virtualenv

You can install Ansible into a virtualenv on the controller quite simply:

```
$ virtualenv ansible
$ source ./ansible/bin/activate
$ pip install ansible
```

If you want to run under Python 3 instead of Python 2 you may want to change that slightly:

```
$ virtualenv -p python3 ansible
$ source ./ansible/bin/activate
$ pip install ansible
```

If you need to use any libraries which are not available via pip (for instance, SELinux Python bindings on systems such as Red Hat Enterprise Linux or Fedora that have SELinux enabled), then you need to install them into the virtualenv. There are two methods:

- When you create the virtualenv, specify `--system-site-packages` to make use of any libraries installed in the system's Python:

  ```
  $ virtualenv ansible --system-site-packages
  ```

- Copy those files in manually from the system. For instance, for SELinux bindings you might do:

  ```
  $ virtualenv ansible --system-site-packages
  $ cp -r -v /usr/lib64/python3.*/site-packages/selinux/ ./py3-ansible/lib64/python3.*/site-packages/
  $ cp -v /usr/lib64/python3.*/site-packages/*selinux*.so ./py3-ansible/lib64/python3.*/site-packages/
  ```

Running on BSD

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 240)**

Unknown directive type "seealso".

```
.. seealso:: :ref:`working_with_bsd`
```

Running on Solaris

By default, Solaris 10 and earlier run a non-POSIX shell which does not correctly expand the default tmp directory Ansible uses (:file:`~/.ansible/tmp`). If you see module failures on Solaris machines, this is likely the problem. There are several workarounds:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 246);** ***backlink***

Unknown interpreted text role "file".

- You can set `remote_tmp` to a path that will expand correctly with the shell you are using (see the plugin documentation for :ref:`C shell<csh_shell>`, :ref:`fish shell<fish_shell>`, and :ref:`Powershell<powershell_shell>`). For example, in the ansible config file you can set:

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 250);** ***backlink***

  Unknown interpreted text role "ref".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel]`

[docs] [docsite] [rst] [reference_appendices] faq.rst, line 250); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst, line 250); *backlink*

Unknown interpreted text role "ref".

```
remote_tmp=$HOME/.ansible/tmp
```

In Ansible 2.5 and later, you can also set it per-host in inventory like this:

```
solaris1 ansible_remote_tmp=$HOME/.ansible/tmp
```

- You can set :ref:`ansible_shell_executable<ansible_shell_executable>` to the path to a POSIX compatible shell. For instance, many Solaris hosts have a POSIX shell located at :file:`/usr/xpg4/bin/sh` so you can set this in inventory like so:

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst, line 260); *backlink*

  Unknown interpreted text role "ref".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst, line 260); *backlink*

  Unknown interpreted text role "file".

  ```
  solaris1 ansible_shell_executable=/usr/xpg4/bin/sh
  ```

  (bash, ksh, and zsh should also be POSIX compatible if you have any of those installed).

Running on z/OS

There are a few common errors that one might run into when trying to execute Ansible on z/OS as a target.

- Version 2.7.6 of python for z/OS will not work with Ansible because it represents strings internally as EBCDIC.

  To get around this limitation, download and install a later version of python for z/OS (2.7.13 or 3.6.1) that represents strings internally as ASCII. Version 2.7.13 is verified to work.

- When `pipelining = False` in */etc/ansible/ansible.cfg* then Ansible modules are transferred in binary mode via sftp however execution of python fails with

  Error

  SyntaxError: Non-UTF-8 code starting with '\x83' in file /a/user1/.ansible/tmp/ansible-tmp-1548232945.35-274513842609025/AnsiballZ_stat.py on line 1, but no encoding declared; see https://python.org/dev/peps/pep-0263/ for details

  To fix it set `pipelining = True` in */etc/ansible/ansible.cfg*.

- Python interpret cannot be found in default location `/usr/bin/python` on target host.

  Error

  /usr/bin/python: EDC5129I No such file or directory

  To fix this set the path to the python installation in your inventory like so:

  ```
  zos1 ansible_python_interpreter=/usr/lpp/python/python-2017-04-12-py27/python27/bin/python
  ```

- Start of python fails with `The module libpython2.7.so was not found.`

  Error

  EE3501S The module libpython2.7.so was not found.

  On z/OS, you must execute python from gnu bash. If gnu bash is installed at `/usr/lpp/bash`, you can fix this in your inventory by specifying an `ansible_shell_executable`:

  ```
  zos1 ansible_shell_executable=/usr/lpp/bash/bin/bash
  ```

Running under fakeroot

Some issues arise as `fakeroot` does not create a full nor POSIX compliant system by default. It is known that it will not correctly expand the default tmp directory Ansible uses (:file:`~/.ansible/tmp`). If you see module failures, this is likely the problem. The simple workaround is to set `remote_tmp` to a path that will expand correctly (see documentation of the shell plugin you are using for specifics).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst, line 306); *backlink*

Unknown interpreted text role "file".

For example, in the ansible config file (or via environment variable) you can set:

```
remote_tmp=$HOME/.ansible/tmp
```

What is the best way to make content reusable/redistributable?

If you have not done so already, read all about "Roles" in the playbooks documentation. This helps you make playbook content self-contained, and works well with things like git submodules for sharing content with others.

If some of these plugin types look strange to you, see the API documentation for more details about ways Ansible can be extended.

Where does the configuration file live and what can I configure in it?

See :ref:`intro_configuration`.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 333); *backlink***
>
> Unknown interpreted text role "ref".

How do I disable cowsay?

If cowsay is installed, Ansible takes it upon itself to make your day happier when running playbooks. If you decide that you would like to work in a professional cow-free environment, you can either uninstall cowsay, set `nocows=1` in `ansible.cfg`, or set the :envvar:`ANSIBLE_NOCOWS` environment variable:

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 340); *backlink***
>
> Unknown interpreted text role "envvar".

```
export ANSIBLE_NOCOWS=1
```

How do I see a list of all of the ansible_ variables?

Ansible by default gathers "facts" about the machines under management, and these facts can be accessed in playbooks and in templates. To see a list of all of the facts that are available about a machine, you can run the `setup` module as an ad hoc action:

```
ansible -m setup hostname
```

This will print out a dictionary of all of the facts that are available for that particular host. You might want to pipe the output to a pager.This does NOT include inventory variables or internal 'magic' variables. See the next question if you need more than just 'facts'.

How do I see all the inventory variables defined for my host?

By running the following command, you can see inventory variables for a host:

```
ansible-inventory --list --yaml
```

How do I see all the variables specific to my host?

To see all host specific variables, which might include facts and other sources:

```
ansible -m debug -a "var=hostvars['hostname']" localhost
```

Unless you are using a fact cache, you normally need to use a play that gathers facts first, for facts included in the task above.

How do I loop over a list of hosts in a group, inside of a template?

A pretty common pattern is to iterate over a list of hosts inside of a host group, perhaps to populate a template configuration file with a list of servers. To do this, you can just access the "$groups" dictionary in your template, like this:

```
{% for host in groups['db_servers'] %}
    {{ host }}
{% endfor %}
```

If you need to access facts about these hosts, for instance, the IP address of each hostname, you need to make sure that the facts have been populated. For example, make sure you have a play that talks to db_servers:

```
- hosts:  db_servers
  tasks:
    - debug: msg="doesn't matter what you do, just that they were talked to previously."
```

Then you can use the facts inside your template, like this:

```
{% for host in groups['db_servers'] %}
   {{ hostvars[host]['ansible_eth0']['ipv4']['address'] }}
{% endfor %}
```

How do I access a variable name programmatically?

An example may come up where we need to get the ipv4 address of an arbitrary interface, where the interface to be used may be supplied via a role parameter or other input. Variable names can be built by adding strings together using "~", like so:

```
{{ hostvars[inventory_hostname]['ansible_' ~ which_interface]['ipv4']['address'] }}
```

The trick about going through hostvars is necessary because it's a dictionary of the entire namespace of variables. `inventory_hostname` is a magic variable that indicates the current host you are looping over in the host loop.

In the example above, if your interface names have dashes, you must replace them with underscores:

```
{{ hostvars[inventory_hostname]['ansible_' ~ which_interface | replace('_', '-') ]['ipv4']['address'] }}
```

Also see dynamic_variables.

How do I access a group variable?

Technically, you don't, Ansible does not really use groups directly. Groups are labels for host selection and a way to bulk assign variables, they are not a first class entity, Ansible only cares about Hosts and Tasks.

That said, you could just access the variable by selecting a host that is part of that group, see first_host_in_a_group below for an example.

How do I access a variable of the first host in a group?

What happens if we want the ip address of the first webserver in the webservers group? Well, we can do that too. Note that if we are using dynamic inventory, which host is the 'first' may not be consistent, so you wouldn't want to do this unless your inventory is static and predictable. (If you are using AWX or the :ref:`Red Hat Ansible Automation Platform <ansible_platform>`, it will use database order, so this isn't a problem even if you are using cloud based inventory scripts).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 461**); ***backlink***

Unknown interpreted text role "ref".

Anyway, here's the trick:

```
{{ hostvars[groups['webservers'][0]]['ansible_eth0']['ipv4']['address'] }}
```

Notice how we're pulling out the hostname of the first machine of the webservers group. If you are doing this in a template, you could use the Jinja2 '#set' directive to simplify this, or in a playbook, you could also use set_fact:

```
- set_fact: headnode={{ groups['webservers'][0] }}

- debug: msg={{ hostvars[headnode].ansible_eth0.ipv4.address }}
```

Notice how we interchanged the bracket syntax for dots -- that can be done anywhere.

How do I copy files recursively onto a target host?

The `copy` module has a recursive parameter. However, take a look at the `synchronize` module if you want to do something more efficient for a large number of files. The `synchronize` module wraps rsync. See the module index for info on both of these modules.

How do I access shell environment variables?

On controller machine : Access existing variables from controller use the `env` lookup plugin. For example, to access the value of the HOME environment variable on the management machine:

```
---
# ...
  vars:
     local_home: "{{ lookup('env','HOME') }}"
```

On target machines : Environment variables are available via facts in the `ansible_env` variable:

```
{{ ansible_env.HOME }}
```

If you need to set environment variables for TASK execution, see :ref:`playbooks_environment` in the :ref:`Advanced Playbooks <playbooks_special_topics>` section. There are several ways to set environment variables on your target machines. You can use the :ref:`template <template_module>`, :ref:`replace <replace_module>`, or :ref:`lineinfile <lineinfile_module>` modules to introduce environment variables into files. The exact files to edit vary depending on your OS and distribution and local configuration.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 510**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 510**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 510**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 510**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, **line 510**); ***backlink***

Unknown interpreted text role "ref".

How do I generate encrypted passwords for the user module?

Ansible ad hoc command is the easiest option:

```
ansible all -i localhost, -m debug -a "msg={{ 'mypassword' | password_hash('sha512', 'mysecretsalt') }}"
```

The `mkpasswd` utility that is available on most Linux systems is also a great option:

```
mkpasswd --method=sha-512
```

If this utility is not installed on your system (for example, you are using macOS) then you can still easily generate these passwords using Python. First, ensure that the Passlib password hashing library is installed:

```
pip install passlib
```

Once the library is ready, SHA512 password values can then be generated as follows:

```
python -c "from passlib.hash import sha512_crypt; import getpass; print(sha512_crypt.using(rounds=5000).hash(getpass.getpa
```

Use the integrated :ref:`hash_filters` to generate a hashed version of a password. You shouldn't put plaintext passwords in your playbook or host_vars; instead, use :ref:`playbooks_vault` to encrypt sensitive data.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 549); *backlink*
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 549); *backlink*
>
> Unknown interpreted text role "ref".

In OpenBSD, a similar option is available in the base system called `encrypt (1)`

Ansible allows dot notation and array notation for variables. Which notation should I use?

The dot notation comes from Jinja and works fine for variables without special characters. If your variable contains dots (.), colons (:), or dashes (-), if a key begins and ends with two underscores, or if a key uses any of the known public attributes, it is safer to use the array notation. See :ref:`playbooks_variables` for a list of the known public attributes.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 559); *backlink*
>
> Unknown interpreted text role "ref".

```
item[0]['checksum:md5']
item['section']['2.1']
item['region']['Mid-Atlantic']
It is {{ temperature['Celsius']['-3'] }} outside.
```

Also array notation allows for dynamic variable composition, see dynamic_variables.

Another problem with 'dot notation' is that some keys can cause problems because they collide with attributes and methods of python dictionaries.

```
item.update # this breaks if item is a dictionary, as 'update()' is a python method for dictionaries
item['update'] # this works
```

When is it unsafe to bulk-set task arguments from a variable?

You can set all of a task's arguments from a dictionary-typed variable. This technique can be useful in some dynamic execution scenarios. However, it introduces a security risk. We do not recommend it, so Ansible issues a warning when you do something like this:

```
#...
vars:
  usermod_args:
    name: testuser
    state: present
    update_password: always
tasks:
- user: '{{ usermod_args }}'
```

This particular example is safe. However, constructing tasks like this is risky because the parameters and values passed to `usermod_args` could be overwritten by malicious values in the `host facts` on a compromised target machine. To mitigate this risk:

- set bulk variables at a level of precedence greater than `host facts` in the order of precedence found in :ref:`ansible_variable_precedence` (the example above is safe because play vars take precedence over facts)

  > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 607); *backlink*
  >
  > Unknown interpreted text role "ref".

- disable the :ref:`inject_facts_as_vars` configuration setting to prevent fact values from colliding with variables (this will also disable the original warning)

  > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 610); *backlink*
  >
  > Unknown interpreted text role "ref".

Can I get training on Ansible?

Yes! See our services page for information on our services and training offerings. Email info@ansible.com for further details.

We also offer free web-based training classes on a regular basis. See our webinar page for more info on upcoming webinars.

Is there a web interface / REST API / GUI?

Yes! The open-source web interface is Ansible AWX. The supported Red Hat product that makes Ansible even more powerful and easy to use is :ref:`Red Hat Ansible Automation Platform <ansible_platform>`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 631); *backlink*
>
> Unknown interpreted text role "ref".

How do I keep secret data in my playbook?

If you would like to keep secret data in your Ansible content and still share it publicly or keep things in source control, see :ref:`playbooks_vault`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 639); *backlink*
>
> Unknown interpreted text role "ref".

If you have a task that you don't want to show the results or command given to it when using -v (verbose) mode, the following task or playbook attribute can be useful:

```
- name: secret task
  shell: /usr/bin/do_something --value={{ secret_value }}
  no_log: True
```

This can be used to keep verbose output but hide sensitive information from others who would otherwise like to be able to see the output.

The `no_log` attribute can also apply to an entire play:

```
- hosts: all
  no_log: True
```

Though this will make the play somewhat difficult to debug. It's recommended that this be applied to single tasks only, once a playbook is completed. Note that the use of the `no_log` attribute does not prevent data from being shown when debugging Ansible itself via the :envvar:`ANSIBLE_DEBUG` environment variable.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 654); *backlink*
>
> Unknown interpreted text role "envvar".

When should I use {{ }}? Also, how to interpolate variables or dynamic variable names

A steadfast rule is 'always use `{{ }}` except when `when:`'. Conditionals are always run through Jinja2 as to resolve the expression, so `when:`, `failed_when:` and `changed_when:` are always templated and you should avoid adding `{{ }}`.

In most other cases you should always use the brackets, even if previously you could use variables without specifying (like `loop` or `with_` clauses), as this made it hard to distinguish between an undefined variable and a string.

Another rule is 'moustaches don't stack'. We often see this:

```
{{ somevar_{{other_var}} }}
```

The above DOES NOT WORK as you expect, if you need to use a dynamic variable use the following as appropriate:

```
{{ hostvars[inventory_hostname]['somevar_' ~ other_var] }}
```

For 'non host vars' you can use the :ref:`vars lookup<vars_lookup>` plugin:

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 686); *backlink*
>
> Unknown interpreted text role "ref".

```
{{ lookup('vars', 'somevar_' ~ other_var) }}
```

To determine if a keyword requires `{{ }}` or even supports templating, use `ansible-doc -t keyword <name>`, this will return documentation on the keyword including a `template` field with the values `explicit` (requires `{{ }}`), `implicit` (assumes `{{ }}`, so no needed) or `static` (no templating supported, all characters will be interpreted literally)

Why don't you ship ansible in wheel format (or other packaging format) ?

In most cases it has to do with maintainability. There are many ways to ship software and we do not have the resources to release Ansible on every platform. In some cases there are technical issues. For example, our dependencies are not present on Python Wheels.

How do I get the original ansible_host when I delegate a task?

As the documentation states, connection variables are taken from the `delegate_to` host so `ansible_host` is overwritten, but you can still access the original via `hostvars`:

```
original_host: "{{ hostvars[inventory_hostname]['ansible_host'] }}"
```

This works for all overridden connection variables, like `ansible_user`, `ansible_port`, and so on.

How do I fix 'protocol error: filename does not match request' when fetching a file?

Since release `7.9p1` of OpenSSH there is a bug in the SCP client that can trigger this error on the Ansible controller when using SCP as the file transfer mechanism:

```
failed to transfer file to /tmp/ansible/file.txt\r\nprotocol error: filename does not match request
```

In these releases, SCP tries to validate that the path of the file to fetch matches the requested path. The validation fails if the remote filename requires quotes to escape spaces or non-ascii characters in its path. To avoid this error:

- Use SFTP instead of SCP by setting `scp_if_ssh` to `smart` (which tries SFTP first) or to `False`. You can do this in one of four ways:
  - Rely on the default setting, which is `smart` - this works if `scp_if_ssh` is not explicitly set anywhere
  - Set a :ref:`host variable <host_variables>` or :ref:`group variable <group_variables>` in inventory: `ansible_scp_if_ssh: False`

    > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 734); *backlink*
    >
    > Unknown interpreted text role "ref".

    > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 734); *backlink*
    >
    > Unknown interpreted text role "ref".

  - Set an environment variable on your control node: `export ANSIBLE_SCP_IF_SSH=False`
  - Pass an environment variable when you run Ansible: `ANSIBLE_SCP_IF_SSH=smart ansible-playbook`
  - Modify your `ansible.cfg` file: add `scp_if_ssh=False` to the `[ssh_connection]` section
- If you must use SCP, set the `-T` arg to tell the SCP client to ignore path validation. You can do this in one of three ways:
  - Set a :ref:`host variable <host_variables>` or :ref:`group variable <group_variables>`: `ansible_scp_extra_args=-T`,

    > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 739); *backlink*
    >
    > Unknown interpreted text role "ref".

    > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 739); *backlink*
    >
    > Unknown interpreted text role "ref".

  - Export or pass an environment variable: `ANSIBLE_SCP_EXTRA_ARGS=-T`
  - Modify your `ansible.cfg` file: add `scp_extra_args=-T` to the `[ssh_connection]` section

> **Note**
>
> If you see an `invalid argument` error when using `-T`, then your SCP client is not performing filename validation and will not trigger this error.

Does Ansible support multiple factor authentication 2FA/MFA/biometrics/finterprint/usbkey/OTP/...

No, Ansible is designed to execute multiple tasks against multiple targets, minimizing user interaction. As with most automation tools, it is not compatible with interactive security systems designed to handle human interaction. Most of these systems require a secondary prompt per target, which prevents scaling to thousands of targets. They also tend to have very short expiration periods so it requires frequent reauthorization, also an issue with many hosts and/or a long set of tasks.

In such environments we recommend securing around Ansible's execution but still allowing it to use an 'automation user' that does not require such measures. With AWX or the :ref:`Red Hat Ansible Automation Platform <ansible_platform>`, administrators can set up RBAC access to inventory, along with managing credentials and job execution.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 756); *backlink*
>
> Unknown interpreted text role "ref".

The 'validate' option is not enough for my needs, what do I do?

Many Ansible modules that create or update files have a `validate` option that allows you to abort the update if the validation command fails. This uses the temporary file Ansible creates before doing the final update. In many cases this does not work since the validation tools for the specific application require either specific names, multiple files or some other factor that is not present in this simple feature.

For these cases you have to handle the validation and restoration yourself. The following is a simple example of how to do this with block/rescue and backups, which most file based modules also support:

```
- name: update config and backout if validation fails
  block:
     - name: do the actual update, works with copy, lineinfile and any action that allows for `backup`.
       template: src=template.j2 dest=/x/y/z backup=yes moreoptions=stuff
       register: updated

    - name: run validation, this will change a lot as needed. We assume it returns an error when not passing, use `failed_
      shell: run_validation_commmand
      become: yes
      become_user: requiredbyapp
      environment:
        WEIRD_REQUIREMENT: 1
 rescue:
    - name: restore backup file to original, in the hope the previous configuration was working.
      copy:
         remote_src: yes
         dest: /x/y/z
         src: "{{ updated['backup_file'] }}"
 always:
    - name: We choose to always delete backup, but could copy or move, or only delete in rescue.
      file:
         path: "{{ updated['backup_file'] }}"
         state: absent
```

How do I submit a change to the documentation?

Documentation for Ansible is kept in the main project git repository, and complete instructions for contributing can be found in the docs README viewable on GitHub. Thanks!

I don't see my question here

If you have not found an answer to your questions, you can ask on one of our mailing lists or chat channels. For instructions on subscribing to a list or joining a chat channel, see :ref:`communication`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 812); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]faq.rst`, line 814)

Unknown directive type "seealso".

```
.. seealso::

   :ref:`working_with_playbooks`
       An introduction to playbooks
   :ref:`playbooks_best_practices`
       Tips and tricks for playbooks
   `User Mailing List <https://groups.google.com/group/ansible-project>`_
       Have a question?  Stop by the google group!
```