Adding modules and plugins locally

You can extend Ansible by adding custom modules or plugins. You can create them from scratch or copy existing ones for local use. You can store a local module or plugin on your Ansible control node and share it with your team or organization. You can also share plugins and modules by including them in a collection, then publishing the collection on Ansible Galaxy.

If you are using a local module or plugin but Ansible cannot find it, this page is all you need.

If you want to create a plugin or a module, see :ref:`developing_plugins`, :ref:`developing_modules_general` and :ref:`developing_collections`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst`, **line 12**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst`, **line 12**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst`, **line 12**); ***backlink***

Unknown interpreted text role "ref".

Extending Ansible with local modules and plugins offers shortcuts such as:

- You can copy other people's modules and plugins.
- When writing a new module, you can choose any programming language you like.
- You do not have to clone any repositories.
- You do not have to open a pull request.
- You do not have to add tests (though we recommend that you do!).

Modules and plugins: what is the difference?

If you are looking to add functionality to Ansible, you might wonder whether you need a module or a plugin. Here is a quick overview to help you understand what you need:

- Modules are reusable, standalone scripts that can be used by the Ansible API, the :command:`ansible` command, or the :command:`ansible-playbook` command. Modules provide a defined interface. Each module accepts arguments and returns information to Ansible by printing a JSON string to stdout before exiting. Modules execute on the target system (usually that means on a remote system) in separate processes. Modules are technically plugins, but for historical reasons we do not usually talk about "module plugins".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst`, **line 31**); ***backlink***

  Unknown interpreted text role "command".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 31); *backlink*

Unknown interpreted text role "command".

- :ref:`Plugins <working_with_plugins>` extend Ansible's core functionality and execute on the control node within the `/usr/bin/ansible` process. Plugins offer options and extensions for the core features of Ansible - transforming data, logging output, connecting to inventory, and more.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 32); *backlink*

Unknown interpreted text role "ref".

Adding modules and plugins in collections

You can add modules and plugins by :ref:`creating a collection <developing_collections>`. With a collection, you can use custom modules and plugins in any playbook or role. You can share your collection easily at any time through Ansible Galaxy.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 39); *backlink*

Unknown interpreted text role "ref".

The rest of this page describes other methods of using local, standalone modules or plugins.

Adding a module outside of a collection

You can configure Ansible to load standalone local modules in a specified location or locations and make them available to all playbooks and roles. Alternatively, you can make a non-collection local module available only to specific playbooks or roles.

Adding standalone local modules for all playbooks and roles

To load standalone local modules automatically and make them available to all playbooks and roles, use the :ref:`DEFAULT_MODULE_PATH` configuration setting or the `ANSIBLE_LIBRARY` environment variable. The configuration setting and environment variable take a colon-separated list, similar to `$PATH`. You have two options:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 53); *backlink*

Unknown interpreted text role "ref".

- Add your standalone local module to one of the default configured locations. See the :ref:`DEFAULT_MODULE_PATH` configuration setting for details. Default locations may change without notice.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 55); *backlink*

Unknown interpreted text role "ref".

- Add the location of your standalone local module to an environment variable or configuration:
    - the `ANSIBLE_LIBRARY` environment variable
    - the :ref:`DEFAULT_MODULE_PATH` configuration setting

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 58); *backlink*

Unknown interpreted text role "ref".

To view your current configuration settings for modules:

```
ansible-config dump |grep DEFAULT_MODULE_PATH
```

After you save your module file in one of these locations, Ansible loads it and you can use it in any local task, playbook, or role.

To confirm that `my_local_module` is available:

- type `ansible localhost -m my_local_module` to see the output for that module, or
- type `ansible-doc -t module my_local_module` to see the documentation for that module

> **Note**
>
> Currently, the `ansible-doc` command can parse module documentation only from modules written in Python. If you have a module written in a programming language other than Python, please write the documentation in a Python file adjacent to the module file.

Adding standalone local modules for selected playbooks or a single role

Ansible automatically loads all executable files from certain directories adjacent to your playbook or role as modules. Standalone modules in these locations are available only to the specific playbook, playbooks, or role in the parent directory.

- To use a standalone module only in a selected playbook or playbooks, store the module in a subdirectory called `library` in the directory that contains the playbook or playbooks.
- To use a standalone module only in a single role, store the module in a subdirectory called `library` within that role.

> **Warning**
>
> Roles contained in collections cannot contain any modules or other plugins. All plugins in a collection must live in the collection `plugins` directory tree. All plugins in that tree are accessible to all roles in the collection. If you are developing new modules, we recommend distributing them in :ref:`collections <developing_collections>`, not in roles.
>
> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 87); *backlink***
>
> Unknown interpreted text role "ref".

Adding a non-module plugin locally outside of a collection

You can configure Ansible to load standalone local plugins in a specified location or locations and make them available to all playbooks and roles. Alternatively, you can make a standalone local plugin available only to specific playbooks or roles.

> **Note**
>
> Although modules are plugins, the naming patterns for directory names and environment variables that apply to other plugin types do not apply to modules. See :ref:`local_modules`.
>
> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 99); *backlink***
>
> Unknown interpreted text role "ref".

Adding local non-module plugins for all playbooks and roles

To load standalone local plugins automatically and make them available to all playbooks and roles, use the configuration setting or environment variable for the type of plugin you are adding. These configuration settings and environment variables take a colon-separated list, similar to `$PATH`. You have two options:

- Add your local plugin to one of the default configured locations. See :ref:`configuration settings <ansible_configuration_settings>` for details on the correct configuration setting for the plugin type. Default locations may change without notice.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 106); *backlink***
>
> Unknown interpreted text role "ref".

- Add the location of your local plugin to an environment variable or configuration:
    - the relevant `ANSIBLE_plugin_type_PLUGINS` environment variable - for example, `$ANSIBLE_INVENTORY_PLUGINS` or `$ANSIBLE_VARS_PLUGINS`
    - the relevant `plugin_type_PATH` configuration setting, most of which begin with `DEFAULT_` - for example, `DEFAULT_CALLBACK_PLUGIN_PATH` or `DEFAULT_FILTER_PLUGIN_PATH` or `BECOME_PLUGIN_PATH`

To view your current configuration settings for non-module plugins:

```
ansible-config dump |grep plugin_type_PATH
```

After your plugin file is added to one of these locations, Ansible loads it and you can use it in any local module, task, playbook, or role. For more information on environment variables and configuration settings, see :ref:`ansible_configuration_settings`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 117); *backlink***
>
> Unknown interpreted text role "ref".

To confirm that `plugins/plugin_type/my_local_plugin` is available:

- type `ansible-doc -t <plugin_type> my_local_lookup_plugin` to see the documentation for that plugin - for example, `ansible-doc -t lookup my_local_lookup_plugin`

The `ansible-doc` command works for most plugin types, but not for action, filter, or test plugins. See :ref:`ansible-doc` for more details.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 123); *backlink***
>
> Unknown interpreted text role "ref".

Adding standalone local plugins for selected playbooks or a single role

Ansible automatically loads all plugins from certain directories adjacent to your playbook or role, loading each type of plugin separately from a directory named for the type of plugin. Standalone plugins in these locations are available only to the specific playbook, playbooks, or role in the parent directory.

- To use a standalone plugin only in a selected playbook or playbooks, store the plugin in a subdirectory for the correct `plugin_type` (for example, `callback_plugins` or `inventory_plugins`) in the directory that contains the playbooks. These directories must use the `_plugins` suffix. For a full list of plugin types, see :ref:`working_with_plugins`.

    > **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 130); *backlink***
    >
    > Unknown interpreted text role "ref".

- To use a standalone plugin only in a single role, store the plugin in a subdirectory for the correct `plugin_type` (for example, `cache_plugins` or `strategy_plugins`) within that role. When shipped as part of a role, the plugin is available as soon as the role is executed. These directories must use the `_plugins` suffix. For a full list of plugin types, see :ref:`working_with_plugins`.

    > **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 131); *backlink***
    >
    > Unknown interpreted text role "ref".

> **Warning**
>
> Roles contained in collections cannot contain any plugins. All plugins in a collection must live in the collection `plugins` directory tree. All plugins in that tree are accessible to all roles in the collection. If you are developing new plugins, we recommend distributing them in :ref:`collections <developing_collections>`, not in roles.
>
> > **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs]**

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 135); *backlink*

Unknown interpreted text role "ref".

Using `ansible.legacy` to access custom versions of an `ansible.builtin` module

If you need to override one of the `ansible.builtin` modules and are using FQCN, you need to use `ansible.legacy` as part of the fully-qualified collection name (FQCN). For example, if you had your own `copy` module, you would access it as `ansible.legacy.copy`. See :ref:`using_ansible_legacy` for details on how to use custom modules with roles within a collection.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_locally.rst, line 142); *backlink*

Unknown interpreted text role "ref".