Creating collections

- Naming your collection
- Creating a new collection
- Creating a collection from a custom template
- Creating collections with ansible-creator

To create a collection:

1. Create a `new collection<creating_new_collections>`, optionally using a custom `collection template<creating_collection_from_custom_template>`, with the `ansible-galaxy collection init` command.
2. Add modules and other content to the collection.
3. Build the collection into a collection artifact with `ansible-galaxy collection build<building_collections>`.
4. Publish the collection artifact to Galaxy with `ansible-galaxy collection publish<publishing_collections>`.

A user can then install your collection on their systems.

Naming your collection

Collection names consist of a namespace and a name, separated by a period (`.`). Both namespace and name should be valid Python identifiers. This means that they should consist of ASCII letters, digits, and underscores.

> Usually namespaces and names use lower-case letters, digits, and underscores, but no upper-case letters.

You should check Ansible Galaxy's namespace list to ensure that your desired namespace is not already registered. If you use a namespace or full collection name that conflicts with another collection on Galaxy, then the `ansible-galaxy collection install` can install the wrong collection.

There are a few special namespaces:

ansible

> The ansible namespace is owned by Red Hat and reserved for official Ansible collections. Two special members are the synthetic `ansible.builtin` and `ansible.legacy` collections. These cannot be found on Ansible Galaxy, but are built-in into ansible-core.

community

> The community namespace is owned by the Ansible community. Collections from this namespace generally live in the GitHub ansible-collection organization. If you want to create a collection in this namespace, `request<request_coll_repo>` it on the forum.

local

> The local namespace does not contain any collection on Ansible Galaxy, and the intention is that this will never change. You can use the `local` namespace for collections that are locally on your machine or locally in your Git repositories, without having to fear collisions with actually existing collections on Ansible Galaxy.

To request a new namespace in Galaxy, do the following:

1. Log in to the Ansible forum.
2. Submit a namespace request.

The Red Hat Community and Partner Engineering team will be notified of your namespace request and create it for you.

Creating a new collection

Create your collection skeleton in a path that includes `ansible_collections`, for example collections/ansible_collections/.

To start a new collection, run the following command in your collections directory:

```
ansible_collections#> ansible-galaxy collection init my_namespace.my_collection
```

> Both the namespace and collection names use the same strict set of requirements. Both are limited to alphanumeric characters and underscores, must have a minimum length of three characters, and cannot start with an underscore.

It will create the structure `[my_namespace]/[my_collection]/[collection skeleton]`.

Hint

If Git is used for version control, the corresponding repository should be initialized in the collection directory.

Once the collection exists, you can populate the directories with the content you want inside the collection. See ansible-collections GitHub Org to get a better idea of what you can place inside a collection.

Reference: the `ansible-galaxy collection` command

Currently the `ansible-galaxy collection` command implements the following sub commands:

- `init`: Create a basic collection based on the default template included with Ansible or your own template.
- `build`: Create a collection artifact that can be uploaded to Galaxy or your own repository.
- `publish`: Publish a built collection artifact to Galaxy.
- `install`: Install one or more collections.

To learn more about the `ansible-galaxy` command-line tool, see the `ansible-galaxy` man page.

Creating a collection from a custom template

The built-in collection template is a simple example of a collection that works with `ansible-core`, but if you want to simplify your development process you may want to create a custom collection template to pass to `ansible-galaxy collection init`.

A collection skeleton is a directory that looks like a collection directory but any `.j2` files (excluding those in `templates/` and `roles/*/templates/`) will be templated by `ansible-galaxy collection init`. The skeleton's `galaxy.yml.j2` file should use the variables `namespace` and `collection_name` which are derived from `ansible-galaxy init namespace.collection_name`, and will populate the metadata in

the initialized collection's `galaxy.yml` file. There are a few additional variables available by default (for example, `version` is `1.0.0`), and these can be supplemented/overridden using `--extra-vars`.

An example `galaxy.yml.j2` file that accepts an optional dictionary variable `dependencies` could look like this:

```
namespace: {{ namespace }}
name: {{ collection_name }}
version: {{ (version|quote) is version('0.0.0', operator='gt',
version_type='semver')|ternary(version, undef('version must be a valid semantic version
greater than 0.0.0')) }}
dependencies: {{ dependencies | default({}, true) }}
```

To initialize a collection using the new template, pass the path to the skeleton with `ansible-galaxy collection init`:

```
ansible_collections#> ansible-galaxy collection init --collection-skeleton
/path/to/my/namespace/skeleton --extra-vars "@my_vars_file.json" my_namespace.my_collection
```

You can configure the collection skeleton using `collection_skeleton` in `ansible.cfg`. To ignore files and directories in the collection skeleton, use the `collection_skeleton_ignore` option in `ansible.cfg` .

```
[galaxy]
collection_skeleton = /path/to/collection_skeleton
collection_skeleton_ignore = ^.git$,^.*/.git_keep$,^\./CLAUDE\.md$
```

The `collection_skeleton_ignore` option is a list of regular expressions to match files and directories to ignore. The regex are matched against the relative path of the file or directory from the skeleton directory.

> Before `ansible-core` 2.17, collection skeleton templating is limited to the few hardcoded variables including `namespace`, `collection_name`, and `version`.

> The default collection skeleton uses an internal filter `comment_ify` that isn't accessibly to `--collection-skeleton`. Use `ansible-doc -t filter|test --list` to see available plugins.

Creating collections with ansible-creator

ansible-creator is designed to quickly scaffold an Ansible collection project.

> The Ansible Development Tools package offers a convenient way to install `ansible-creator` along with a curated set of tools for developing automation content.

After installing `ansible-creator` you can initialize a project in one of the following ways:

- Use the `init` subcommand.
- Use `ansible-creator` with the Ansible extension in Visual Studio Code.

`collections`
Learn how to install and use collections.

`collection_structure`
Directories and files included in the collection skeleton

Ansible Development Tools (ADT)
Python package of tools to create and test Ansible content.

`Communication<communication>`
Got questions? Need help? Want to share your ideas? Visit the Ansible communication guide

---

□□□ #2
□ □□□ □□□ 2026-10-31 14:33:37 UTC
□ □□□ □□□ 2026-10-31 14:33:38 UTC