The Ansible Collections Development Cycle

Ansible developers (including community contributors) add new features, fix bugs, and update code in many different repositories. These repositories contain plugins and modules that enable Ansible to execute specific tasks, like adding a user to a particular database or configuring a particular network device. These repositories contain the source code for collections.

Development on collections occurs at the macro and micro levels. Each collection has its own macro development cycle. For more information on the collections development cycle, see :ref:`contributing_maintained_collections`. The micro-level lifecycle of a PR is similar in collections and in `ansible-core`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst`, **line 9**); *backlink*
>
> Unknown interpreted text role "ref".

Macro development: roadmaps, releases, and projects

If you want to follow the conversation about what features will be added to the Ansible package for upcoming releases and what bugs are being fixed, you can watch these resources:

- the :ref:`roadmaps`

  > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst`, **line 20**); *backlink*
  >
  > Unknown interpreted text role "ref".

- the :ref:`Ansible Release Schedule <release_and_maintenance>`

  > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst`, **line 21**); *backlink*
  >
  > Unknown interpreted text role "ref".

- the Ansible Community Working Group .

Micro development: the lifecycle of a PR

If you want to contribute a feature or fix a bug in a collection, you must open a **pull request** ("PR" for short). GitHub provides a great overview of how the pull request process works in general. The ultimate goal of any pull request is to get merged and become part of a collection. Each collection has its own contributor guidelines so please check there for specific details.

Here's an overview of the PR lifecycle:

- :ref:`Contributor opens a PR <collection_quickstart>`

  > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst`, **line 31**); *backlink*
  >
  > Unknown interpreted text role "ref".

- CI runs the test suite
- Developers, maintainers, community review the PR
- Contributor addresses any feedback from reviewers
- Developers, maintainers, community re-review
- PR merged or closed

Making your PR merge-worthy

We do not merge every PR. See :ref:`collection_quickstart` for tips to make your PR useful, attractive, and merge-worthy.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst`, **line 42**); *backlink*
>
> Unknown interpreted text role "ref".

Creating changelog fragments

Changelogs help users and developers keep up with changes to Ansible collections. Many collections build changelogs for each release from fragments. For collections that use this model, you **must** add a changelog fragment to any PR that changes functionality or fixes a bug.

You do not need a changelog fragment for PRs that:

- add new modules and plugins, because Ansible tooling does that automatically;
- contain only documentation changes.

> **Note**
>
> Some collections require a changelog fragment for every pull request. They use the `trivial:` section for entries mentioned above that will be skipped when building a release changelog.

More precisely:

- Every bugfix PR must have a changelog fragment. The only exception are fixes to a change that has not yet been included in a release.
- Every feature PR must have a changelog fragment.
- New modules and plugins (except jinja2 filter and test plugins) must have `versions_added` set correctly, and do not need a changelog fragment. The tooling detects new modules and plugins by their `versions_added` value and announces them in the next release's changelog automatically.
- New jinja2 filter and test plugins, and also new roles and playbooks (for collections) must have a changelog fragment. See

:ref:`changelogs_how_to_format_j2_roles_playbooks` or the antsibull-changelog documentation for such changelog fragments for information on what the fragments should look like.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst, line 65); *backlink***
>
> Unknown interpreted text role "ref".

We build short summary changelogs for minor releases as well as for major releases. If you backport a bugfix, include a changelog fragment with the backport PR.

Creating a changelog fragment

A basic changelog fragment is a `.yaml` or `.yml` file placed in the `changelogs/fragments/` directory. Each file contains a yaml dict with keys like `bugfixes` or `major_changes` followed by a list of changelog entries of bugfixes or features. Each changelog entry is rst embedded inside of the yaml file which means that certain constructs would need to be escaped so they can be interpreted by rst and not by yaml (or escaped for both yaml and rst if you prefer). Each PR **must** use a new fragment file rather than adding to an existing one, so we can trace the change back to the PR that introduced it.

PRs which add a new module or plugin do not necessarily need a changelog fragment. See :ref:`community_changelogs`. Also see :ref:`changelogs_how_to_format` for the precise format changelog fragments should have.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst, line 76); *backlink***
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst, line 76); *backlink***
>
> Unknown interpreted text role "ref".

To create a changelog entry, create a new file with a unique name in the `changelogs/fragments/` directory of the corresponding repository. The file name should include the PR number and a description of the change. It must end with the file extension `.yaml` or `.yml`. For example: `40696-user-backup-shadow-file.yaml`

A single changelog fragment may contain multiple sections but most will only contain one section. The toplevel keys (bugfixes, major_changes, and so on) are defined in the config file for our release note tool. Here are the valid sections and a description of each:

breaking_changes

MUST include changes that break existing playbooks or roles. This includes any change to existing behavior that forces users to update tasks. Breaking changes means the user MUST make a change when they update. Breaking changes MUST only happen in a major release of the collection. Write in present tense and clearly describe the new behavior that the end user must now follow. Displayed in both the changelogs and the :ref:`Porting Guides <porting_guides>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst, line 83); *backlink***
>
> Unknown interpreted text role "ref".

```
breaking_changes:
  - ec2_instance - instance wait for state behavior no longer waits for the instance monitoring status to become OK when launchin
```

major_changes

Major changes to ansible-core or a collection. SHOULD NOT include individual module or plugin changes. MUST include non-breaking changes that impact all or most of a collection (for example, updates to support a new SDK version across the collection). Major changes mean the user can CHOOSE to make a change when they update but do not have to. Could be used to announce an important upcoming EOL or breaking change in a future release. (ideally 6 months in advance, if known. See this example). Write in present tense and describe what is new. Optionally, include a 'Previously..." sentence to help the user identify where old behavior should now change. Displayed in both the changelogs and the :ref:`Porting Guides <porting_guides>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst, line 92); *backlink***
>
> Unknown interpreted text role "ref".

```
major_changes:
  - bitbucket_* modules - client_id is no longer marked as ``no_log=true``. If you relied on its value not showing up in logs and
```

minor_changes

Minor changes to ansible-core, modules, or plugins. This includes new parameters added to modules, or non-breaking behavior changes to existing parameters, such as adding new values to choices[]. Minor changes are enhancements, not bug fixes. Write in present tense.

```
minor_changes:
  - nmcli - adds ``routes6`` and ``route_metric6`` parameters for supporting IPv6 routes (https://github.com/ansible-collections/
```

deprecated_features

Features that have been deprecated and are scheduled for removal in a future release. Write in past tense. Include an alternative, where available, for the feature being deprecated. Displayed in both the changelogs and the :ref:`Porting Guides <porting_guides>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst, line 109); *backlink***
>
> Unknown interpreted text role "ref".

```
deprecated features:
  - mail callback plugin - not specifying ``sender`` is deprecated and will be disallowed in ``community.general`` 6.0.0 (https:/
```

removed_features

Features that were previously deprecated and are now removed. Write in past tense. Include an alternative, where available, for the feature being deprecated. Displayed in both the changelogs and the :ref:`Porting Guides <porting_guides>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)collection_development_process.rst, line 118); *backlink***
>
> Unknown interpreted text role "ref".

```
removed_features:
  - acme_account_facts - the deprecated redirect has been removed. Use ``community.crypto.acme_account_info`` instead (https://gi
```

security_fixes

Fixes that address CVEs or resolve security concerns. MUST use security_fixes for any CVEs. Write in present tense. Include links to CVE information.

```
security_fixes:
  - win_psexec - ensure password is masked in ``psexec_``command return result (https://github.com/ansible-collections/community.
```

bugfixes

Fixes that resolve issues. SHOULD NOT be used for minor enhancements (use `minor_change` instead). Write in past tense to describe the problem and present tense to describe the fix.

```
bugfixes:
  - apt_repository - fix crash caused by  a timeout. The ``cache.update()`` was raising an ``IOError`` because of a timeout in ``
```

known_issues

Known issues that are currently not fixed or will not be fixed. Write in present tense to describe the problem and in imperative tense to describe any available workaround.

```
known_issues:
  - idrac_user - module may error out with the message ``unable to perform the import or export operation`` because there are pen
```

Each changelog entry must contain a link to its issue between parentheses at the end. If there is no corresponding issue, the entry must contain a link to the PR itself.

Most changelog entries are `bugfixes` or `minor_changes`. You can also use `trivial` for any collection that requires a changelog fragment for each pull request. `trivial` changelog fragments are excluded from the changelog output.

Changelog fragment entry format

When writing a changelog entry, use the following format:

```
- scope - description starting with a lowercase letter and ending with a period at the very end. Multiple sentences are allowed (https:/
```

The scope is usually a module or plugin name or group of modules or plugins, for example, `lookup plugins`. While module names can (and should) be mentioned directly (`foo_module`), plugin names should always be followed by the type (`foo inventory plugin`).

For changes that are not really scoped (for example, which affect a whole collection), use the following format:

```
- Description starting with an uppercase letter and ending with a dot at the very end. Multiple sentences are allowed (https://github.co
```

Here are some examples:

```
bugfixes:
  - apt_repository - fix crash caused by ``cache.update()`` raising an ``IOError``
    due to a timeout in ``apt update`` (https://github.com/ansible/ansible/issues/51995).
```

```
minor_changes:
  - lineinfile - add warning when using an empty regexp (https://github.com/ansible/ansible/issues/29443).
```

```
bugfixes:
  - copy - the module was attempting to change the mode of files for
    remote_src=True even if mode was not set as a parameter.  This failed on
    filesystems which do not have permission bits (https://github.com/ansible/ansible/issues/29444).
```

You can find more example changelog fragments in the changelog directory for the community.general development branch.

After you have written the changelog fragment for your PR, commit the file and include it with the pull request.

Changelog fragment entry format for new jinja2 plugins, roles, and playbooks

While new modules and plugins that are not jinja2 filter or test plugins are mentioned automatically in the generated changelog, jinja2 filter and test plugins, roles, and playbooks are not. To make sure they are mentioned, a changelog fragment in a specific format is needed:

```
# A new jinja2 filter plugin:
add plugin.filter:
  - # The following needs to be the name of the filter itself, not of the file
    # the filter is included in!
    name: to_time_unit
    # The description should be in the same format as short_description for
    # other plugins and modules: it should start with an upper-case letter and
    # not have a period at the end.
    description: Converts a time expression to a given unit

# A new jinja2 test plugin:
add plugin.test:
  - # The following needs to be the name of the test itself, not of the file
    # the test is included in!
    name: asn1time
    # The description should be in the same format as short_description for
    # other plugins and modules: it should start with an upper-case letter and
    # not have a period at the end.
    description: Check whether the given string is an ASN.1 time

# A new role:
add object.role:
  - # This should be the short (non-FQCN) name of the role.
    name: nginx
    # The description should be in the same format as short_description for
    # plugins and modules: it should start with an upper-case letter and
    # not have a period at the end.
    description: A nginx installation role

# A new playbook:
add object.playbook:
  - # This should be the short (non-FQCN) name of the playbook.
    name: wipe server
    # The description should be in the same format as short description for
    # plugins and modules: it should start with an upper-case letter and
    # not have a period at the end.
    description: Wipes a server
```