Committers Guidelines

These are the guidelines for people with commit privileges on the repositories in the ansible and ansible-collections GitHub organizations.

Committers of Ansible-core are necessarily Red Hat employees acting as members of the Ansible Core team. Committers of Ansible collections are members of the community or Ansible Engineering. Please read the guidelines before you commit.

These guidelines apply to everyone. At the same time, this is NOT a process document. So just use good judgment. You have been given commit access because we trust your judgment.

That said, use the trust wisely.

If you abuse the trust and break components and builds, and so on, the trust level falls and you may be asked not to commit or you may lose your commit privileges.

Features, high-level design, and roadmap of ansible-core

As a core team member, you are an integral part of the team that develops the :ref:`roadmap <roadmaps>`. Please be engaged, and push for the features and fixes that you want to see. Also keep in mind that Red Hat, as a company, will commit to certain features, fixes, APIs, and so on, for various releases. Red Hat, the company, and the Ansible team must get these changes completed and released as scheduled. Obligations to users, the community, and customers must come first. Because of these commitments, a feature you want to develop yourself may not get into a release if it affects a lot of other parts within Ansible.

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

Any other new features and changes to high level design should go through the proposal process (TBD), to ensure the community and core team have had a chance to review the idea and approve it. The core team has sole responsibility for merging new features based on proposals to Ansible-core.

Features, high-level design, and roadmap of Ansible collections

Collections maintainers define features, high-level design, and roadmap of the collections themselves and are responsible for merging new features to Ansible collections based on proposals discussed with their communities.

Our workflow on GitHub

As a committer, you may already know this, but our workflow forms a lot of our team policies. Please ensure you are aware of the following workflow steps:

- Fork the repository upon which you want to do some work to your own personal repository
- Work on the specific branch upon which you need to commit
- Create a pull request back to the upstream repository and tag the people you would like to review; assign someone as the primary "owner" of your pull request
- Adjust code as necessary based on the comments provided
- Ask someone from the repository committers to do a final review and merge

Addendum to workflow for committers:

The Core Team is aware that this can be a difficult process at times. Sometimes, the team breaks the rules by making direct commits or merging their own pull requests. This section is a set of guidelines. If you are changing a comma in documentation, or making a very minor change, you can use your best judgement. This is another trust thing. The process is critical for any major change, but for little things or getting something done quickly, use your best judgement and make sure people on the team are aware of your work.

Roles on Core

- Core committers: Fine to do pull requests for most things, but we should have a timebox. Hanging pull requests may merge on the judgement of these developers.
- :ref:`Module maintainers <maintainers>`: Module maintainers own specific modules and have indirect commit access through the current module pull request mechanisms.

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

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

Unknown interpreted text role "ref".

- :ref:`Collection maintainers <maintainers>`: Collection maintainers own specific collections and have commit access to them. Each collection can set its own rules for contributions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\[ansible-devel][docs][docsite][rst][community]committer_guidelines.rst, line 50); *backlink*

Unknown interpreted text role "ref".

General rules

Individuals with direct commit access are entrusted with powers that allow them to do a broad variety of things--probably more than we can write down. Rather than rules, treat these as general *guidelines*, individuals with this power are expected to use their best judgement.

- Do NOT
  - Commit directly.
  - Merge your own pull requests. Someone else should have a chance to review and approve the pull request merge. If you are a Core Committer, you have a small amount of leeway here for very minor changes.
  - Forget about alternate environments. Consider the alternatives--yes, people have bad environments, but they are the ones who need us the most.
  - Drag your community team members down. Discuss the technical merits of any pull requests you review. Avoid negativity and personal comments. For more guidance on being a good community member, read our :ref:`code_of_conduct`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\[ansible-devel][docs][docsite][rst][community]committer_guidelines.rst, line 63); *backlink*

Unknown interpreted text role "ref".

  - Forget about the maintenance burden. High-maintenance features may not be worth adding.
  - Break playbooks. Always keep backwards compatibility in mind.
  - Forget to keep it simple. Complexity breeds all kinds of problems.
- Do
  - Squash, avoid merges whenever possible, use GitHub's squash commits or cherry pick if needed (bisect thanks you).
  - Be active. Committers who have no activity on the project (through merges, triage, commits, and so on) will have their permissions suspended.
  - Consider backwards compatibility (goes back to "do not break existing playbooks").
  - Write :ref:`tests<developing_testing>` and be sure that other's pull requests you are reviewing are covered well. Pull requests with tests are looked at with more priority than pull requests without tests that should have them included. While not all changes require tests, be sure to add them for new features, bug fixes, and functionality changes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\[ansible-devel][docs][docsite][rst][community]committer_guidelines.rst, line 73); *backlink*

Unknown interpreted text role "ref".

  - Discuss with other committers, specially when you are unsure of something.
  - Document! If your pull request is a new feature or a change to behavior, make sure you have updated all associated documentation or have notified the right people to do so. It also helps to add the version of `ansible-core` or `collection` against which this documentation is compatible (to avoid confusion between stable and devel docs, for backwards compatibility, and so on).
  - Consider scope, sometimes a fix can be generalized.

- Keep it simple, then things are maintainable, debuggable, and intelligible.

Committers are expected to continue to follow the same community and contribution guidelines followed by the rest of the Ansible community.