Creating your first collection pull request

This section describes all steps needed to create your first patch and submit a pull request on a collection.

Prepare your environment

> **Note**
>
> These steps assume a Linux work environment with `git` installed.

1. Install and start `docker` or `podman` with the `docker` executable shim. This insures tests run properly isolated and in the exact environments as in CI. The latest `ansible-core` development version also supports the `podman` CLI program.
2. :ref:`Install Ansible or ansible-core <installation_guide>`. You need the `ansible-test` utility which is provided by either of these packages.

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

3. Create the following directories in your home directory:

   ```
   $ mkdir -p ~/ansible_collections/NAMESPACE/COLLECTION_NAME
   ```

   For example, if the collection is `community.mysql`, it would be:

   ```
   $ mkdir -p ~/ansible_collections/community/mysql
   ```

4. Fork the collection repository through the GitHub web interface.
5. Clone the forked repository from your profile to the created path:

   ```
   $ git clone https://github.com/YOURACC/COLLECTION_REPO.git ~/ansible_collections/NAMESPACE/COLLECTION_
   ```

   If you prefer to use the SSH protocol:

   ```
   $ git clone git@github.com:YOURACC/COLLECTION_REPO.git ~/ansible_collections/NAMESPACE/COLLECTION_NAME
   ```

6. Go to your new cloned repository.

   ```
   $ cd ~/ansible_collections/NAMESPACE/COLLECTION_NAME
   ```

7. Ensure you are in the default branch (it is usually `main`):

   ```
   $ git status
   ```

8. Show remotes. There should be the `origin` repository only:

   ```
   $ git remote -v
   ```

9. Add the `upstream` repository:

   ```
   $ git remote add upstream https://github.com/ansible-collections/COLLECTION_REPO.git
   ```

   This is the repository where you forked from.

10. Update your local default branch. Assuming that it is `main`:

    ```
    $ git fetch upstream
    $ git rebase upstream/main
    ```

11. Create a branch for your changes:

    ```
    $ git checkout -b name_of_my_branch
    ```

Change the code

> **Note**
>
> Do NOT mix several bugfixes or features that are not tightly-related in one pull request. Use separate pull requests for different changes.

You should start with writing integration and unit tests if applicable. These can verify the bug exists (prior to your code fix) and verify your code fixed that bug once the tests pass.

> **Note**
>
> If there are any difficulties with writing or running the tests or you are not sure if the case can be covered, you can skip this step. Other contributors can help you with tests later if needed.

> **Note**
>
> Some collections do not have integration tests. In this case, unit tests are required.

All integration tests are stored in `tests/integration/targets` subdirectories. Go to the subdirectory containing the name of the module you are going to change. For example, if you are fixing the `mysql_user` module in the `community.mysql` collection, its tests are in `tests/integration/targets/test_mysql_user/tasks`.

The `main.yml` file holds test tasks and includes other test files. Look for a suitable test file to integrate your tests or create and include a dedicated test file. You can use one of the existing test files as a draft.

When fixing a bug, write a task which reproduces the bug from the issue.

Put the reported case in the tests, then run integration tests with the following command:

```
$ ansible-test integration name_of_test_subdirectory --docker -v
```

For example, if the tests files you changed are stored in `tests/integration/targets/test_mysql_user/`, the command is as follows:

```
$ ansible-test integration test_mysql_user --docker -v
```

You can use the `-vv` or `-vvv` argument, if you need more detailed output.

In the examples above, the default test image is automatically downloaded and used to create and run a test container. Use the default test image for platform independent integration tests such as those for cloud modules.

If you need to run the tests against a specific distribution, see the :ref:`list of supported container images <test_container_images>`. For example:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst, line 135); *backlink***
>
> Unknown interpreted text role "ref".

```
$ ansible-test integration name_of_test_subdirectory --docker fedora35 -v
```

> **Note**
>
> If you are not sure whether you should use the default image for testing or a specific one, skip the entire step - the community can help you later. You can also try to use the collection repository's CI to figure out which containers are used.

If the tests ran successfully, there are usually two possible outcomes:

- If the bug has not appeared and the tests have passed successfully, ask the reporter to provide more details. It may not be a bug or can relate to a particular software version used or specifics of the reporter's local environment configuration.
- The bug has appeared and the tests has failed as expected showing the reported symptoms.

Fix the bug

See :ref:`module_contribution` for some general guidelines about Ansible module development that may help you craft a good code fix for the bug.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst, line 153); *backlink***
>
> Unknown interpreted text role "ref".

Test your changes

1. Install `flake8` (`pip install flake8`, or install the corresponding package on your operating system).
1. Run `flake8` against a changed file:

```
$ flake8 path/to/changed_file.py
```

This shows unused imports, which is not shown by sanity tests, as well as other common issues. Optionally, you can use the `--max-line-length=160` command-line argument.

2. Run sanity tests:

```
$ ansible-test sanity path/to/changed_file.py --docker -v
```

If they failed, look at the output carefully - it is informative and helps to identify a problem line quickly. Sanity failings usually relate to incorrect code and documentation formatting.

3. Run integration tests:

```
$ ansible-test integration name_of_test_subdirectory --docker -v
```

For example, if the test files you changed are stored in `tests/integration/targets/test_mysql_user/`, the command is:

```
$ ansible-test integration test_mysql_user --docker -v
```

You can use the `-vv` or `-vvv` argument if you need more detailed output.

There are two possible outcomes:

- They have failed. Look at the output of the command. Fix the problem place in the code and run again. Repeat the cycle until the tests pass.
- They have passed. Remember they failed originally? Our congratulations! You have fixed the bug.

In addition to the integration tests, you can also cover your changes with unit tests. This is often required when integration tests are not applicable to the collection.

We use pytest as a testing framework.

Files with unit tests are stored in the `tests/unit/plugins/` directory. If you want to run unit tests, say, for `tests/unit/plugins/test_myclass.py`, the command is:

```
$ ansible-test units tests/unit/plugins/test_myclass.py --docker
```

If you want to run all unit tests available in the collection, run:

```
$ ansible-test units --docker
```

Submit a pull request

1. Commit your changes with an informative but short commit message:

```
$ git add /path/to/changed/file
$ git commit -m "module_name_you_fixed: fix crash when ..."
```

2. Push the branch to `origin` (your fork):

```
$ git push origin name_of_my_branch
```

3. In a browser, navigate to the `upstream` repository (http://github.com/ansible-collections/COLLECTION_REPO).
4. Click the :guilabel:`Pull requests` tab.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst`, line 233); *backlink*
>
> Unknown interpreted text role "guilabel".

GitHub is tracking your fork, so it should see the new branch in it and automatically offer to create a pull request. Sometimes GitHub does not do it, and you should click the :guilabel:`New pull request` button yourself. Then choose :guilabel:`compare across forks` under the :guilabel:`Compare changes` title.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst`, line 235); *backlink*
>
> Unknown interpreted text role "guilabel".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst`, line 235); *backlink*

Unknown interpreted text role "guilabel".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst, line 235); *backlink*

Unknown interpreted text role "guilabel".

5.Choose your repository and the new branch you pushed in the right drop-down list. Confirm.

a. Fill out the pull request template with all information you want to mention.

b. Put `Fixes + link to the issue` in the pull request's description.

c. Put `[WIP] + short description` in the pull request's title. Mention the name of the module/plugin you are modifying at the beginning of the description.

d. Click :guilabel:`Create pull request`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst, line 245); *backlink*

Unknown interpreted text role "guilabel".

6. Add a :ref:`changelog fragment <collection_changelog_fragments>` to the `changelogs/fragments` directory. It will be published in release notes, so users will know about the fix.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst, line 247); *backlink*

Unknown interpreted text role "ref".

a. Run the sanity test for the fragment:

```
$ansible-test sanity changelogs/fragments/ --docker -v
```

b. If the tests passed, commit and push the changes:

```
$ git add changelogs/fragments/myfragment.yml
$ git commit -m "Add changelog fragment"
$ git push origin name_of_my_branch
```

7. Verify the CI tests pass that run automatically on Red Hat infrastructure after every commit.

You will see the CI status in the bottom of your pull request. If they are green and you think that you do not want to add more commits before someone should take a closer look at it, remove `[WIP]` from the title. Mention the issue reporter in a comment and let contributors know that the pull request is "Ready for review".

8. Wait for reviews. You can also ask for review in the `#ansible-community` :ref:`Matrix/Libera.Chat IRC channel <communication_irc>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst, line 268); *backlink*

Unknown interpreted text role "ref".

9. If the pull request looks good to the community, committers will merge it.

For more in-depth details on this process, see the :ref:`Ansible developer guide <developer_guide>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\(ansible-devel)(docs)(docsite)(rst)(community)create_pr_quick_start.rst, line 272); *backlink*

Unknown interpreted text role "ref".